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

function Get-LocalAdminAccount
{
    Param([Parameter(Mandatory = $True)][string]$State)

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

    Write-Host "Getting Local Admin Account: " $State

    if($State = "Template")
    {
        $global:LocalAdminAccount = "Administrator"
        $global:LocalAdminAccountPassword = "cH3r0k33"

    }
    elseif($State = "Configured")
    {
        $global:LocalAdminAccount = "Administrator"
        $global:LocalAdminAccountPassword = "cH3r0k33"
    }
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

### Function: Set HostName & VMName ++$GPID
### -----------------------------------------------
function Create-ECI.EMI.Automation.VMName
{
    Param (
    [Parameter(Mandatory = $True)][string]$GPID,
    [Parameter(Mandatory = $True)][string]$HostName
    )

    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray

    ### GPID + HostName
    ###-------------------
    $global:VMName = $GPID + "_" + $HostName

    ### HostName + GPID
    ###-------------------
    #$global:VMName = $HostName + "_" + $GPID

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


### Function: Set VMGuest Execution Policy
### -----------------------------------------------
function Configure-ECI.EMI.Automation.ExecutionPolicyonVMGuest
{
    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

    $ExecutionPolicyonVMGuest =
    {
        if ($(Get-ExecutionPolicy) -ne "Bypass")
        {
            Write-Host "`nSetting Execution Policy on VM Guest to Bypass:"
            Set-ExecutionPolicy ByPass -Scope LocalMachine

```

```
67              #REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v
                ConsentPromptBehaviorAdmin /t REG_DWORD /d 0 /f
68              #Start-Process powershell -ArgumentList '-noprofile -Command
                Set-ExecutionPolicy ByPass -Scope LocalMachine' -verb RunAs
69              #REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v
                ConsentPromptBehaviorAdmin /t REG_DWORD /d 5 /f
70          }
71          else
72          {
73              Write-Host "`nExecution Policy on VM Guest already set to Bypass"
74          }
75      }
76
77      #Write-Host "vCenter_Account          : " $vCenter_Account     -ForegroundColor Gray
78      #Write-Host "vCenter_Password         : " $vCenter_Password    -ForegroundColor Gray
79      #Write-Host "Creds.LocalAdminName     : " $Creds.LocalAdminName   -ForegroundColor
        Gray
80      #Write-Host "Creds.LocalAdminPassword : " $Creds.LocalAdminPassword
        -ForegroundColor Gray
81
82      Write-Host "INVOKING: $((Get-PSCallStack)[0].Command) on $VMName" -ForegroundColor
        Cyan
83
84      $Invoke = Invoke-VMScript -VM $VMName -ScriptText $ExecutionPolicyonVMGuest
        -ScriptType Powershell -GuestUser $Creds.LocalAdminName -GuestPassword
        $Creds.LocalAdminPassword
85      #$Invoke = Invoke-VMScript -VM $VMName -ScriptText $ExecutionPolicyonVMGuest
        -ScriptType Powershell -HostUser $vCenter_Account -HostPassword $vCenter_Password
86      #$Invoke = Invoke-VMScript -VM $VMName -ScriptText $ExecutionPolicyonVMGuest
        -ScriptType Powershell -HostUser $vCenter_Account -HostPassword $vCenter_Password
        -GuestUser Administrator -GuestPassword cH3r0k33
87
88      #if($Invoke.ExitCode -ne 0)
89      #{
90      #    $Abort = $True
91      #    Write-Host "ABORT ERROR: Aborting Script Execution" -ForegroundColor Red
92      #    #Send-Alert
93      #    Exit
94      #}
95
96      Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
97  }
98
99  function Install-ECI.EMI.Automation.ECIModulesonVMGuest
100 {
101     Param(
102         [Parameter(Mandatory = $True)][string]$Env,
103         [Parameter(Mandatory = $True)][string]$Environment
104     )
105
106     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
107
108     $Source = "\\eciscripts.file.core.windows.net\clientimplementation\" + $Environment
        +"\ECI.Modules." + $Env
109     $Destination = "C:\Program Files\WindowsPowerShell\Modules\"
110
111     Write-Host "Source      : " (Get-Item $Source)        -ForegroundColor Cyan
112     Write-Host "Destination : " $Destination             -ForegroundColor Cyan
113
114     #(Get-Item $Source) | Copy-ECI.EMI.VM.GuestFile -LocalToGuest -VMName $VMName
        -Destination $Destination
115
116     $Modules = Get-ChildItem $Source
117     foreach($Module in $Modules)
118     {
119         Write-Host "Installing Module: " $Module -ForegroundColor White
120         #Copy-ECI.EMI.VM.GuestFile -LocalToGuest -Source $Module.FullName -VMName
            $VMName  -Destination $Destination
```

```powershell
    }
    Write-Host "Please Wait... This may take a minute." -ForegroundColor DarkGray

   ###----------------------
    ### Guest state Retry Loop
    ###----------------------
    $Retries             = 4
    $RetryCounter        = 0
    $RetryTime           = 15
    $RetryTimeIncrement = $RetryTime
    $Success             = $False

    while($Success -ne $True)
    {
        try
        {
            ### Copy all files at once to avoid multiple logins.
            Get-Item $Source | Copy-VMGuestFile -LocalToGuest -Destination $Destination
            -VM $VMName -Force -Confirm:$false -GuestUser $Creds.LocalAdminName
            -GuestPassword $Creds.LocalAdminPassword
            #Copy-ECI.EMI.VM.GuestFile -LocalToGuest -Source $Module.FullName -VMName
            $VMName  -Destination $Destination

            $Success = $True
            Write-Host "$FunctionName - Succeded: " $Success -ForegroundColor Green
        }
        catch
        {
            if($RetryCounter -ge $Retries)
            {
                Throw "ECI.THROW.TERMINATING.ERROR: ERROR Copying VMGuest Files:! "
            }
            else
            {
                ### Retry x Times
                ###--------------------
                $RetryCounter++

                ### Write ECI Error Log
                ###---------------------------------
                Write-Error -Message ("ECI.ERROR.Exception.Message: " +
                $global:Error[0].Exception.Message) -ErrorAction Continue
                -ErrorVariable ECIError
                if(-NOT(Test-Path -Path $ECIErrorLogFile)) {(New-Item -ItemType file
                -Path $ECIErrorLogFile -Force | Out-Null)}
                $ECIError | Out-File -FilePath $ECIErrorLogFile -Append -Force

                ### Error Handling Action
                ###---------------------------------
                Start-ECI.EMI.Automation.Sleep -Message "Retry Invoke-VMScript." -t
                $RetryTime

                ### Restart VM Tools
                ###--------------------
                if($RetryCounter -eq ($Retries - 1))
                {
                    Write-Host "Bailout Reached: Retry Counter..." $RetryCounter
                    -ForegroundColor Magenta
                    Restart-ECI.EMI.VM.VMTools -VMName $VMName
                }
                $RetryTime = $RetryTime + $RetryTimeIncrement
            }
        }
    }

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

```powershell
function Install-ECI.EMI.Automation.ECIModulesonVMGuest-ORIGINAL
{
    Param(
        [Parameter(Mandatory = $True)][string]$Env,
        [Parameter(Mandatory = $True)][string]$Environment
    )

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

    $Source = "\\eciscripts.file.core.windows.net\clientimplementation\" + $Environment
    +"\ECI.Modules." + $Env
    $Destination = "C:\Program Files\WindowsPowerShell\Modules\"

    Write-Host "Source      : " (Get-Item $Source)        -ForegroundColor Cyan
    Write-Host "Destination : " $Destination             -ForegroundColor Cyan

    #(Get-Item $Source) | Copy-ECI.EMI.VM.GuestFile -LocalToGuest -VMName $VMName
    -Destination $Destination

    $Modules = Get-ChildItem $Source
    foreach($Module in $Modules)
    {
        Write-Host "Installing Module: " $Module -ForegroundColor White
        #Copy-ECI.EMI.VM.GuestFile -LocalToGuest -Source $Module.FullName -VMName
        $VMName  -Destination $Destination
    }
    Write-Host "Please Wait... This may take a minute." -ForegroundColor DarkGray


    ### Copy ECI ModuleFolders
    ###-----------------------
    $RetryCount = 3
    try
    {
        ### Copy all files at once to avoid multiple logins.
        Get-Item $Source | Copy-VMGuestFile -LocalToGuest -Destination $Destination -VM
        $VMName -Force -Confirm:$false -GuestUser $Creds.LocalAdminName -GuestPassword
        $Creds.LocalAdminPassword
        #Copy-ECI.EMI.VM.GuestFile -LocalToGuest -Source $Module.FullName -VMName
        $VMName  -Destination $Destination
    }
    catch
    {
        Write-Host "ERROR Copying VMGuest Files: " $Error[0] -ForegroundColor Red

        for ($i=1; $i -le $RetryCount; $i++)
        {
            Start-ECI.EMI.Automation.Sleep -t 30
            Write-Warning  "VMTools Not Responding. Retrying...." -WarningAction Continue
            Install-ECI.EMI.Automation.ECIModulesonVMGuest -Env $Env -Environment
            $Environment
        }

        Write-ECI.ErrorStack
    }

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

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

    $ScriptText =
    {
        $ECIModulePath = "C:\Program Files\WindowsPowerShell\Modules\ECI.Modules*"

```

```
242             try
243             {
244                 Remove-Item -Path $ECIModulePath -Recurse -Force -Confirm:$false
245             }
246             catch
247             {
248                 Write-ECI.ErrorStack
249             }
250         }
251
252         Write-Host "INVOKING: $((Get-PSCallStack)[0].Command) on $VMName" -ForegroundColor
            Cyan
253
254     #Write-Host "Creds.LocalAdminName     : " $Creds.LocalAdminName -ForegroundColor Magenta
255     #Write-Host "Creds.LocalAdminPassword : " $Creds.LocalAdminPassword -ForegroundColor
        Magenta
256     #Write-Host "vCenter_Account          : " $vCenter_Account -ForegroundColor Magenta
257     #Write-Host "vCenter_Password         : " $vCenter_Password -ForegroundColor Magenta
258
259         $Invoke = Invoke-VMScript    -ScriptText $ScriptText -VM $VMName  -ScriptType
            Powershell -GuestUser $Creds.LocalAdminName -GuestPassword
            $Creds.LocalAdminPassword -HostUser $vCenter_Account -HostPassword $vCenter_Password
260
261         #if($Invoke.ExitCode -ne 0)
262         #{
263         #    $Abort = $True
264         #    Write-Host "ABORT ERROR: Aborting Script Execution" -ForegroundColor Red
265         #    #Send-Alert
266         #    Exit
267         #}
268         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
269     }
270
271     function Set-ECI.EMI.Automation.LocalAdminAccount
272     {
273         Param(
274         [Parameter(Mandatory = $False)][switch]$Template,
275         [Parameter(Mandatory = $False)][switch]$ECI
276         )
277
278         $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
279
280         [hashtable]$global:Creds = @{}
281
282         if($Template -eq $True)
283         {
284             $DataSetName = "TemplateLocalAdmin"
285             $ConnectionString =  $DevOps_DBConnectionString
286             $Query = "SELECT AdminPassword FROM definitionVMOSCustomizationSpec WHERE
                ServerRole = '$ServerRole' AND BuildVersion = '$BuildVersion'"
287             Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
                $ConnectionString -Query $Query
288
289             $Creds.CredentialState    = "Template-Creds"
290             $Creds.LocalAdminName     = "Administrator"
291             $Creds.LocalAdminPassword =  $AdminPassword
292         }
293         elseif($ECI -eq $True)
294         {
295
296             $DataSetName = "ECILocalAdmin"
297             $ConnectionString =  $DevOps_DBConnectionString
298             $Query = "SELECT ECILocalAdminName,ECILocalAdminPassword FROM
                definitionOSParameters WHERE ServerRole = '$ServerRole' AND BuildVersion =
                '$BuildVersion'"
299             Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
                $ConnectionString -Query $Query
300
```

```
301             $Creds.CredentialState    = "ECI-Creds"
302             $Creds.LocalAdminName     = $ECILocalAdminName
303             $Creds.LocalAdminPassword = $ECILocalAdminPassword
304         }
305
306         Write-Host "Creds State    : " $Creds.CredentialState -ForegroundColor Cyan
307         Write-Host "LocalAdminName : " $Creds.LocalAdminName -ForegroundColor Cyan
308
309         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
310         Return $Creds
311     }
312
313     ###
    ================================================================================
    ==========
314     ### Configure Desired State        <<<----- CMDLET
315     ###
    ================================================================================
    ==========
316     ###
317     function Configure-DesiredState
318     {
319         [CmdletBinding()]
320         Param(
321         [Parameter(Mandatory = $True)][int]$ServerID,
322         [Parameter(Mandatory = $True)][string]$HostName,
323         [Parameter(Mandatory = $True)][string]$FunctionName,
324         [Parameter(Mandatory = $True)][string]$PropertyName,
325         [Parameter(Mandatory = $True)][string]$DesiredState,
326         [Parameter(Mandatory = $True)][scriptblock]$GetCurrentState,
327         [Parameter(Mandatory = $True)][scriptblock]$SetDesiredState,
328         [Parameter(Mandatory =
        $True)][ValidateSet("Report","Configure")][string]$ConfigurationMode,
329         [Parameter(Mandatory = $True)][ValidateSet($True, $False)][string]$AbortTrigger
330         )
331
332         ###================================================
333         ### GET CURRENT CONFIGURATION-STATE:
334         ###================================================
335         function Get-CurrentState
336         {
337             try
338             {
339                 Invoke-Command $GetCurrentState
340             }
341             catch
342             {
343                 Write-ECI.ErrorStack
344             }
345         }
346
347         ###================================================
348         ### COMPARE CURRENT/DESIRED-STATE:
349         ###================================================
350         function Compare-DesiredState
351         {
352             $global:Compare = ($CurrentState -eq $DesiredState)
353             #Write-ConfigReport `n('- ' * 10)`n "COMPARE: $Compare" `n('- ' * 10)`n
            "FUNCTION: $FunctionName" `n "CURRENTSTATE: $CurrentState" `n "DESIREDSTATE:
            $DesiredState"
354             Write-ConfigReport `r`n('- ' * 20)`r`n"COMPARE          : $Compare"`r`n('- ' *
            20)`r`n"FUNCTION         : $FunctionName"`r`n"CURRENTSTATE     :
            $CurrentState"`r`n"DESIREDSTATE     : $DesiredState"`r`n ('- ' * 20)`r`n
355
356             if($Compare -eq $True)
357             {
358                 ### CURRENT STATE = DESIRED STATE: True
359                 ### ------------------------------------
360                 Write-ConfigReport `r`n"The Current-State Matches the Desired-State."
```

```
        }
        elseif($Compare -eq $False)
        {
            ### CURRENT STATE = DESIRED STATE: False
            ### ------------------------------------
            Write-ConfigReport `r`n"The Current-State Does Not Match the Desired-State."
        }
    }

    ###================================================
    ### SET DESIRED-STATE:
    ###================================================
    function Set-DesiredState
    {
        Write-ConfigReport `r`n('- ' * 20)`r`n"SET DESIRED STATE: $DesiredState"`r`n('-
        ' * 20)`r`n"FUNCTION          : $FunctionName"`r`n"DESIREDSTATE     :
        $DesiredState"`r`n('- ' * 20)`r`n

        foreach ($State in $DesiredState)
        {
            if($ConfigurationMode -eq "Configure")
            {
                Write-ConfigReport `r`n "CONFIG MODE: Setting Desired State."`r`n
                [ScriptBlock]$DesiredStateConfiguration = {Invoke-Command
                $SetDesiredState}

                try
                {
                    Invoke-Command $DesiredStateConfiguration
                }
                catch
                {
                    Write-ECI.ErrorStack
                }
            }
            elseif($ConfigurationMode -eq "Report")
            {
                Write-ConfigReport `r`n "REPORT MODE: Reporting Data Only!" `r`n
            }
        }
    }

    ###================================================
    ### VERIFY DESIRED-STATE:
    ###================================================
    function Verify-DesiredState
    {
        foreach ($State in $DesiredState)
        {
            ### VERIFY: Current State
            ### ------------------------------------------------------
            Get-CurrentState
            $global:VerifyState = $CurrentState

            ### COMPARE: Verify State - Desired State
            ### ------------------------------------------------------
            $global:Verify = ($VerifyState -eq $DesiredState) ### <-- True/False
            Write-ConfigReport `r`n('- ' * 20)`r`n"VERIFY             : $Verify"  `r`n('-
            ' * 20)`r`n"VERIFYSTATE       : $VerifyState"`r`n"DESIREDSTATE     :
            $DesiredState"`r`n('- ' * 20)`r`n

            ### VERIFY = TRUE
            ### ------------------------------------------------------
            if($Verify -eq $True)
            {
                Write-Host `r`n"The Current-State Matches the Desired-State."

                ### ABORT = FALSE
                ### --------------------
```

```
                $global:Abort = $False
            }

            ###  VERIFY = FALSE
            ### ------------------------------------------------
            elseif($Verify -eq $False)
            {
                ### Increment Verify Error Counter
                $global:VerifyErrorCount ++
                $global:VerifyErrorFunction = $FunctionName
                $global:VerifyErrorDesiredState = $DesiredState

                Write-Host `r`n("-" * 50)`r`n"The Current-State Does Not Match the
                Desired-State."`r`n("-" * 50)`r`n  -ForegroundColor Yellow


                if ($AbortTrigger -eq $True)
                {
                    ### ABORT TRIGGER = TRUE
                    ### --------------------
                    $global:Abort = $True
                    $VerifyColor = "Red"

                    #Throw-AbortError
                }
                elseif($AbortTrigger -eq $False)
                {
                    ### ABORT TRIGGER = FALSE
                    ### --------------------
                    $global:Abort = $False
                    $VerifyColor = "Yellow"
                }

                Write-Host "VerifyErrorCount        : " $VerifyErrorCount
                -ForegroundColor $VerifyColor
                Write-Host "VerifyErrorFunction     : " $VerifyErrorFunction
                -ForegroundColor $VerifyColor
                Write-Host "VerifyErrorDesiredState : " $VerifyErrorDesiredState
                -ForegroundColor $VerifyColor
                Write-Host "VerifyErrorVerifyState  : " $VerifyErrorVerifyState
                -ForegroundColor $VerifyColor

                if ($Abort -eq $True)
                {
                    Write-Host "ServerID      : " $ServerID     -ForegroundColor Red
                    Write-Host "HostName      : " $HostName     -ForegroundColor Red
                    Write-Host "VMName        : " $VMName       -ForegroundColor Red
                    Write-Host "FunctionName  : " $FunctionName -ForegroundColor Red
                    Write-Host "Verify        : " $Verify       -ForegroundColor Red
                    Write-Host "AbortTrigger  : " $AbortTrigger -ForegroundColor Red
                    Write-Host "Abort         : " $Abort        -ForegroundColor Red
                    Write-Host "Error: " $Error[0]
                }
            }
        }
    }

    ###============================================
    ### REPORT DESIRED-STATE:
    ###============================================
    function Report-DesiredState
    {
         ### Update Config Log
        ###-----------------------------------
        Write-Host `r`n"UPDATING SERVER CONFIGLOG RECORD: " -ForegroundColor DarkCyan
        Write-Host "ServerID      : " $ServerID      -ForegroundColor DarkCyan
        Write-Host "HostName      : " $HostName      -ForegroundColor DarkCyan
        Write-Host "FunctionName  : " $FunctionName  -ForegroundColor DarkCyan
        Write-Host "Verify        : " $Verify        -ForegroundColor DarkCyan
```

```
489         Write-Host "Abort         : " $Abort         -ForegroundColor DarkCyan
490
491         $Params = @{
492             ServerID      = $ServerID
493             HostName      = $HostName
494             FunctionName  = $FunctionName
495             PropertyName  = $PropertyName
496             CurrentState  = $CurrentState
497             Verify        = $Verify
498             Abort         = $Abort
499         }
500         Write-ECI.ConfigLog @Params
501
502         ### Update Desired State
503         ###-----------------------------------
504         Write-Host `r`n"UPDATING SERVER DESIRED STATE RECORD:" -ForegroundColor DarkCyan
505         Write-Host "ServerID      : " $ServerID       -ForegroundColor DarkCyan
506         Write-Host "HostName      : " $HostName       -ForegroundColor DarkCyan
507         Write-Host "PropertyName  : " $PropertyName   -ForegroundColor DarkCyan
508         Write-Host "CurrentState  : " $CurrentState   -ForegroundColor DarkCyan
509         Write-Host "DesiredState  : " $DesiredState   -ForegroundColor DarkCyan
510         Write-Host "Verify        : " $Verify         -ForegroundColor DarkCyan
511         Write-Host "Abort         : " $Abort          -ForegroundColor DarkCyan
512
513         $Params = @{
514             ServerID      = $ServerID
515             HostName      = $HostName
516             PropertyName  = $PropertyName
517             CurrentState  = $CurrentState
518             DesiredState  = $DesiredState
519             Verify        = $Verify
520             Abort         = $Abort
521         }
522         Write-ECI.DesiredState @Params
523
524         ### Update Current State
525         ###-----------------------------------
526         Write-Host `r`n"UPDATING SERVER CURRENT STATE RECORD:" -ForegroundColor DarkCyan
527         Write-Host "ServerID      : " $ServerID       -ForegroundColor DarkCyan
528         Write-Host "HostName      : " $HostName       -ForegroundColor DarkCyan
529         Write-Host "PropertyName  : " $PropertyName   -ForegroundColor DarkCyan
530         Write-Host "CurrentState  : " $CurrentState   -ForegroundColor DarkCyan
531
532         $Params = @{
533             ServerID      = $ServerID
534             HostName      = $HostName
535             PropertyName  = $PropertyName
536             CurrentState  = $CurrentState
537         }
538         Write-ECI.CurrentState @Params
539
540
541 ### DOES THIS PART EVER EXECUTE??????????????????????????????????????????
542         ### IF ABORT = TRUE
543         ### --------------------
544         if($Abort -eq $True)
545         {
546             ### ABORT = TRUE: Throw Abort Error
547             ###
                --------------------------------------------------------------------------------
                --------
548             Write-ConfigReport "ABORTTRIGGER: " $AbortTrigger "`t`tABORT: " $Abort
549
550             Write-Host "ABORT ERROR:  " "ServerID: " $ServerID "HostName: " $HostName
                "VMName: " $VMName "FunctionName: " $FunctionName "Verify: " $Verify
                "AbortTrigger: " $AbortTrigger "Abort: " $Abort
551             #Throw-AbortError -ServerID $ServerID -HostName $HostName -VMName $VMName
                -FunctionName $FunctionName -Verify $Verify  -PropertyName $PropertyName
                -AbortTrigger $AbortTrigger -Abort $Abort
```

```
552         }
553     }
554
555     ###============================================
556     ###--------------------------------------------
557     ### CONFIGURE DESIRED-STATE
558     ###--------------------------------------------
559     ###============================================
560     &{
561         BEGIN
562         {
563         }
564
565         PROCESS
566         {
567             Get-CurrentState
568             Compare-DesiredState
569             if($Compare -eq $False){Set-DesiredState}
570             Verify-DesiredState
571             Report-DesiredState
572         }
573
574         END
575         {
576             $CurrentState = $Null
577             $DesiredState = $Null
578             $VerifyState  = $Null
579             $Verify       = $Null
580             $Abort        = $Null
581         }
582     }
583 }
584
585 ###
586 ###
    ================================================================================================
    ==========
587
588 function List-AllParameters-deleteme #<-- deleteme?????
589 {
590     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
591
592     ### Server  Parameters
593     ###-------------------------------------------------------------
594     Write-Host `r`n('-' * 50)`n "Server Parameters  : "
    -ForegroundColor Gray
595     Write-Host "ServerID                    : " $ServerID
    -ForegroundColor Gray
596
597     ### Server Request Parameters
598     ###-------------------------------------------------------------
599     Write-Host `r`n('-' * 50)`n "ServerRequest Parameters  : "
    -ForegroundColor Gray
600     Write-Host "RequestID                   : " $RequestID
    -ForegroundColor Gray
601     Write-Host "RequestDateTime             : " $RequestDateTime
    -ForegroundColor Gray
602     Write-Host "HostName                    : " $HostName
    -ForegroundColor Gray
603     Write-Host "ServerRole                  : " $ServerRole
    -ForegroundColor Gray
604     Write-Host "IPv4Address                 : " $IPv4Address
    -ForegroundColor Gray
605     Write-Host "SubnetMask                  : " $SubnetMask
    -ForegroundColor Gray
606     Write-Host "DefaultGateway              : " $DefaultGateway
    -ForegroundColor Gray
607     Write-Host "InstanceLocation            : " $InstanceLocation
```

```
        -ForegroundColor Gray
608     Write-Host "BackupRecovery             : " $BackupRecovery
        -ForegroundColor Gray
609     Write-Host "DisasterRecovery           : " $DisasterRecovery
        -ForegroundColor Gray
610
611     ### VM Parameters
612     ###-------------------------------------------------------------------
613     Write-Host `r`n('-' * 50)`n "VM Parameters             : " -ForegroundColor Gray
614     Write-Host "VMParameterID              : " $VMParameterID -ForegroundColor Gray
615     Write-Host "vCPUCount                  : " $vCPUCount -ForegroundColor Gray
616     Write-Host "vMemorySizeGB              : " $vMemorySizeGB -ForegroundColor Gray
617     Write-Host "OSVolumeGB                 : " $OSVolumeGB -ForegroundColor Gray
618     Write-Host "SwapVolumeGB               : " $SwapVolumeGB -ForegroundColor Gray
619     Write-Host "DataVolumeGB               : " $DataVolumeGB -ForegroundColor Gray
620     Write-Host "LogVolumeGB                : " $LogVolumeGB -ForegroundColor Gray
621     Write-Host "SysVolumeGB                : " $SysVolumeGB -ForegroundColor Gray
622
623     ### OS Parameters
624     ###-------------------------------------------------------------------
625     Write-Host `r`n('-' * 50)`n "OS Parameters               : "  -ForegroundColor Gray
626     Write-Host "NetworkInterfacename       : " $NetworkInterfacename -ForegroundColor
        Gray
627     Write-Host "LocalAdministrator         : " $LocalAdministrator -ForegroundColor Gray
628     Write-Host "CDROMLetter                : " $CDROMLetter -ForegroundColor Gray
629     Write-Host "IPv6Preference             : " $IPv6Preference -ForegroundColor Gray
630     Write-Host "WindowsFirewallPreference  : " $WindowsFirewallPreference
        -ForegroundColor Gray
631     Write-Host "IEESCPreference            : " $InternetExplorerESCPreference
        -ForegroundColor Gray
632     Write-Host "RemoteDesktopPreference    : " $RemoteDesktopPreference
        -ForegroundColor Gray
633     Write-Host "RDPResetrictionsPreference : " $RDPResetrictionsPreference
        -ForegroundColor Gray
634     Write-Host "SwapFileBufferSizeMB       : " $SwapFileBufferSizeMB -ForegroundColor
        Gray
635     Write-Host "SwapFileLocation           : " $SwapFileLocation -ForegroundColor Gray
636     Write-Host "SwapFileMemoryThreshholdGB : " $SwapFileMemoryThreshholdGB
        -ForegroundColor Gray
637     Write-Host "SwapFileMultiplier         : " $SwapFileMultiplier -ForegroundColor Gray
638     Write-Host  `r`n('-' * 50)`r`n -ForegroundColor Gray
639 }
640
641 ### Import VMWare Modules
642 ### -----------------------------------------------
643 function Import-ECI.EMI.Automation.VMWareModules
644 {
645     Param(
646         [Parameter(Mandatory = $True)][string]$Env,
647         [Parameter(Mandatory = $True)][string]$Environment,
648         [Parameter(Mandatory = $True)][string]$ModuleName,
649         [Parameter(Mandatory = $True)][version]$ModuleVersion
650     )
651
652     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
653
654     ### Modify PSModulePath
655     ###--------------------------
656     $global:VMModulesPath = "\\eciscripts.file.core.windows.net\clientimplementation\"
        + $Environment + "\Vendor.Modules." + $Env + "\VMWare\PowerCLI.10.1.0\"
657     $env:PSModulePath = $env:PSModulePath + ";" +  $VMModulesPath
658     #$ModuleName = $VMModulesPath + $VMModulesName
659
660
661
662     if((Get-Module -ListAvailable -Name $ModuleName) -ne $Null)
663     {
664         Write-Host "Importing VNWare Modules: " $ModuleName  $ModuleVersion
```

```
            -ForegroundColor Gray

            try
            {
                ### Uninstall Existing Modules
                ###---------------------------
                $VMModules = Get-Module -Name $ModuleName ; if($VMModules) {$VMModules |
                Remove-Module}

                ### Import Modules
                ###---------------------------
                Get-Module -ListAvailable -Name $ModuleName | Import-Module -Force #
                -MinimumVersion $ModuleVersion
                import-module -Name vmware.vimautomation.vds -Force
            }
            catch
            {
                Write-ECI.ErrorStack
            }
        }
        else
        {
             Write-Host "Modules not available! Check the env:PSModulePath." (Get-Module
             -ListAvailable $VMModules) -ForegroundColor Red
        }

        Set-PowerCLIConfiguration -Scope User -ParticipateInCEIP $false
        -InvalidCertificateAction ignore -Confirm:$false

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

    ### ------------------------------------------------
    function Get-ECI.EMI.Automation.vCenter
    {
        Param([Parameter(Mandatory = $True)][string]$InstanceLocation)

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

        switch ( $InstanceLocation )
        {
            "Lab" { $vCenter = "ecilab-bosvcsa01.ecilab.corp" }
            "BOS" { $vCenter = "bosvc.eci.cloud"      }
            "QTS" { $vCenter = "cloud-qtsvc.eci.corp" }
            "SAC" { $vCenter = "sacvc.eci.cloud"      }
            "LHC" { $vCenter = "lhcvc.eci.cloud"      }
            "LD5" { $vCenter = "ld5vc.eci.cloud"      }
            "HK"  { $vCenter = "hkvc.eci.cloud"       }
            "SG"  { $vCenter = "sgvc.eci.cloud"       }
        }


        Write-Host "Instance Location  : " $InstanceLocation -ForegroundColor Cyan
        Write-Host "vCenter            : " $vCenter -ForegroundColor Cyan

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

    ### ------------------------------------------------
    function Connect-ECI.EMI.Automation.VIServer
    {
        Param(
            [Parameter(Mandatory = $False)][string]$InstanceLocation,
            [Parameter(Mandatory = $False)][string]$vCenter,
            [Parameter(Mandatory = $False)][string]$User,
```

```powershell
        [Parameter(Mandatory = $False)][string]$Password
    )

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

    if($User){$vCenter_Account = $User}
    if($Password){$vCenter_Password = $Password}

    ### Show/Hide VMWare Module Progress Bar
    $global:ProgressPreference = "Continue" ### Continue/SilentlyContinue

    ### Hide Certificate Warning Message
    Set-PowerCLIConfiguration -Scope User -ParticipateInCEIP $false
    -InvalidCertificateAction ignore -Confirm:$false | Out-Null

    if($InstanceLocation)
    {
        ### Connect to vCenter
        ###--------------------------------
        if($global:DefaultVIServers.Name -eq (Get-ECI.EMI.Automation.vCenter
        -InstanceLocation $InstanceLocation))
        {
            Write-Host "Using Current VI Server Session : " $global:DefaultVIServers
            -ForegroundColor Cyan
            #Disconnect-VIServer -Server $global:DefaultVIServers -confirm:$false
        }
        elseif($global:DefaultVIServers.Name -ne (Get-ECI.EMI.Automation.vCenter
        -InstanceLocation $InstanceLocation))
        {
            write-host "Connecting to InstanceLocation : " $InstanceLocation
            -ForegroundColor Cyan
            $vCenter = Get-ECI.EMI.Automation.vCenter -InstanceLocation $InstanceLocation
            $global:VISession = Connect-VIServer -Server $vCenter -User
            $vCenter_Account -Password $vCenter_Password
        }
    }

    if($vCenter)
    {
        ### Connect to vCenter
        ###--------------------------------
        if($global:DefaultVIServers.Name -eq $vCenter)
        {
            Write-Host "Using Current VI Server Session : " $global:DefaultVIServers
            -ForegroundColor Cyan
            #Disconnect-VIServer -Server $global:DefaultVIServers -confirm:$false
        }
        elseif($global:DefaultVIServers.Name -ne $vCenter)
        {
            write-host "Connecting to vCenterName : " $vCenter -ForegroundColor Cyan
            $global:VISession = Connect-VIServer -Server $vCenter -User
            $vCenter_Account -Password $vCenter_Password
        }

    }

    ### Get  InstanceUUID
    ###--------------------------------
    $global:vCenterUUID = $VISession.InstanceUuid
    Write-Host "vCenterUUID                              : " $vCenterUUID -ForegroundColor
    DarkCyan

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

### Delete Server Logs
### ------------------------------------------------
```

```powershell
function Delete-ECI.EMI.Automation.ServerLogs
{
    Param([Parameter(Mandatory = $True)][string]$HostName)

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

    ### Delete Log Path
    #$VMLogPath = $AutomationLogPath + "\" + $HostName + "\"
    $VMLogPath = $AutomationLogPath + "\" + $HostName + "\temp\"
    Write-Host "Deleting Server Logs:" $VMLogPath -ForegroundColor DarkCyan

    if(Test-Path -Path $VMLogPath)
    {
        Remove-Item -Path $VMLogPath -Include * -Recurse -Force -Confirm:$false
    }
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

function Copy-ECI.EMI.Automation.VMLogsfromGuest    #<--- Consolidatec
{
    Param(
    [Parameter(Mandatory = $True)][string]$HostName,
    [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

    #VMLogs to Local
    $VMLogDestination = ($AutomationLogPath + "\" + $HostName)
    $GuestLogs = ($AutomationLogPath + "\" + $HostName)

    #if(-NOT(Test-Path -Path $VMLogDestination)) {(New-Item -ItemType directory -Path $VMLogDestination -Force | Out-Null)}
    Copy-ECI.EMI.VM.GuestFile -GuestToLocal -VM $VMName -Source $GuestLogs -Destination $VMLogDestination

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

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

    ### Set Folders

#working
#{

    $VMLogDestination = $AutomationLogPath + "\" + $HostName + "\"
    $GuestLogSource = $AutomationLogPath + "\" + $HostName #+ "\*.*"

    #foreach($Log in $GuestLogSource)
    #{

    #}

    if(-NOT(Test-Path -Path $VMLogDestination)) {(New-Item -ItemType directory -Path $VMLogDestination -Force | Out-Null)}
    write-host "Copying Log Files from Guest..."        -ForegroundColor Cyan
    write-host "GUEST LOG SOURCE :" $GuestLogSource     -ForegroundColor DarkCyan
    write-host "LOG DESTINATION  :" $VMLogDestination   -ForegroundColor DarkCyan

    #Copy-VMGuestFile -Source $GuestLogSource -Destination $VMLogDestination -VM $VMName -GuestToLocal -GuestUser $Creds.LocalAdminName -GuestPassword $Creds.LocalAdminPassword

```

```
848   ###<--- Use Cmdlet!!!!!!!!!!!!!!!
849   Write-Host "USE Cmdlet" -ForegroundColor Magenta
850       #Copy-VMGuestFile -Source $GuestLogSource -Destination $VMLogDestination -VM
          $VMName -GuestToLocal -GuestUser $Creds.LocalAdminName -GuestPassword
          $Creds.LocalAdminPassword
851       $GuestLogSource | Copy-VMGuestFile -Destination $VMLogDestination -VM $VMName
          -GuestToLocal -GuestUser $Creds.LocalAdminName -GuestPassword
          $Creds.LocalAdminPassword
852   #Copy-ECI.EMI.VM.GuestFile -GuestToLocal -VM $VMName -Source $GuestLogSource
      -Destination $VMLogDestination
853
854       Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
855   }
856
857   ### Write Server Logs to SQL
858   function Write-ECI.EMI.Automation.VMLogstoSQL
859   {
860       $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
          "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
861
862       Write-Host "Writing All Logs to SQL . . . " -ForegroundColor Cyan
863       Write-ConfigLog-SQL
864       Write-DesiredState-SQL
865       Write-CurrentState-SQL
866       Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
867   }
868
869
870   ### Write Server Config Log
871   ### --------------------------------------------------
872   function Write-ECI.ConfigLog
873   {
874       Param(
875       [Parameter(Mandatory = $True)][string]$ServerID,
876       [Parameter(Mandatory = $True)][string]$HostName,
877       [Parameter(Mandatory = $True)][string]$FunctionName,
878       [Parameter(Mandatory = $True)][string]$PropertyName,
879       [Parameter(Mandatory = $True)][string]$CurrentState,
880       [Parameter(Mandatory = $True)][string]$Verify,
881       [Parameter(Mandatory = $True)][string]$Abort
882       )
883
884       $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
          "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
885
886       ### Create Log File
887       ###--------------------------
888       $VMLogName = "ConfigLog"
889       #$VMLogFile = $AutomationLogPath + "\" + $HostName + "\" + $VMLogName + "_" +
          $HostName  + ".txt"
890       $VMLogFile = $AutomationLogPath + "\" + $HostName + "\temp\" + $VMLogName + "_" +
          $HostName  + ".txt"
891       if(-NOT(Test-Path -Path $VMLogFile)) {(New-Item -ItemType file -Path $VMLogFile
          -Force | Out-Null)}
892
893       Set-Variable -Name $VMLogName -Value $VMLogFile -Option AllScope -Scope global -Force
894
895       ## Write Log to Guest
896       ###--------------------------
897       $VMLogEntry = "ServerID=" + $ServerID + "," + "HostName=" + $HostName + "," +
          "FunctionName=" + $FunctionName + "," +  "PropertyName=" + $PropertyName + "," +
          "CurrentState=" + $CurrentState + "," +  "Verify=" + $Verify + "," +  "Abort=" +
          $Abort
898
899       Write-Host "LOG: $VMLogName LOGFILE : $VMLogFile" -ForegroundColor DarkGray
900       Write-Host "LOG: $VMLogName ENTRY   : $VMLogEntry" -ForegroundColor DarkGray
901       $VMLogEntry | Out-File -FilePath $VMLogFile -Force -Append
902
903       Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
```

```
}
function Write-ConfigLog-Old
{
    Param(
    [Parameter(Mandatory = $True)][string]$ServerID,
    [Parameter(Mandatory = $True)][string]$HostName,
    [Parameter(Mandatory = $True)][string]$FunctionName,
    [Parameter(Mandatory = $True)][string]$PropertyName,
    [Parameter(Mandatory = $True)][string]$CurrentState,
    [Parameter(Mandatory = $True)][string]$Verify,
    [Parameter(Mandatory = $True)][string]$Abort
    )

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

    ### Create Log File
    ###---------------------------
    $LogName = "ConfigLog"
    $LogFile = $AutomationLogPath + "\" + $HostName + "\" + $VMLogName + "_" +
    $HostName  + ".txt"
    #if(-NOT(Test-Path -Path $VMLogFile)) {(New-Item -ItemType file -Path $VMLogFile
    -Force | Out-Null)}

    Set-Variable -Name $VMLogName -Value $VMLogFile -Option AllScope -Scope global -Force

    ## Write Log to Guest
    ###---------------------------
    $LogEntry = "ServerID=" + $ServerID + "," + "HostName=" + $HostName + "," +
    "FunctionName=" + $FunctionName + "," +  "PropertyName=" + $PropertyName + "," +
    "CurrentState=" + $CurrentState + "," +  "Verify=" + $Verify + "," +  "Abort=" +
    $Abort

    Write-Host "LOG LOGFILE : $LogFile" -ForegroundColor DarkGray
    Write-Host "LOG ENTRY   : $LogEntry" -ForegroundColor DarkGray
    $LogEntry | Out-File -FilePath $LogFile -Force -Append

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

### Write Server Config Logs to SQL
### ------------------------------------------------
function Write-ConfigLog-SQL
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray

    ### Set Log Name
    ###---------------------------
    $VMLogName = "ConfigLog"
    #$VMLogPath = $AutomationLogPath + "\" + $HostName + "\"
    $VMLogPath = $AutomationLogPath + "\" + $HostName + "\temp\"
    $ConnectionString  = $DevOps_DBConnectionString

    ### Open Database Connection
    $Connection = New-Object System.Data.SQLClient.SQLConnection
    $Connection.ConnectionString = $ConnectionString
    $Connection.Open()
    ### Insert Row
    $cmd = New-Object System.Data.SqlClient.SqlCommand
    $cmd.Connection = $Connection

    ### Import Log File
    ###----------------------
    $VMLastLog = Get-ChildItem -Path ($VMLogPath) | Where-Object {($_ -like $VMLogName
    + "*")} | Sort-Object LastAccessTime -Descending | Select-Object -First 1
    $VMLastLogFile = $VMLogPath + "\" + $VMLastLog
    $VMLastLogFile = Get-Content -Path $VMLastLogFile

```

```
965         foreach ($Record in $VMLastLogFile)
966         {
967             $Keys = $Null
968             $Values = $Null
969 
970             foreach($Column in ($Record.split(",")))
971             {
972               $Key = $Column.split("=")[0]
973               $Value =  "'" + $Column.split("=")[1] + "'"
974               $Keys = $Keys + $Key + ","
975               $Values = $Values + $Value + ","
976             }
977             $Keys = $Keys.Substring(0,$Keys.Length-1)
978             $Values = $Values.Substring(0,$Values.Length-1)
979 
980             $Query = "INSERT INTO ServerConfigLog($Keys) VALUES ($Values)"
981 
982             ### Show Results
983             ###------------------------
984             Write-Host "SQLQuery: " $Query -ForegroundColor DarkCyan
985 
986             $cmd.CommandText = $Query
987             $cmd.ExecuteNonQuery() #| Out-Null
988         }
989         ### Close
990         $connection.Close()
991 
992         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
993     }
994 
995     ### Write Server Desired State
996     ### -----------------------------------------------
997     function Write-ECI.DesiredState
998     {
999         Param(
1000        [Parameter(Mandatory = $True)][string]$ServerID,
1001        [Parameter(Mandatory = $True)][string]$HostName,
1002        [Parameter(Mandatory = $True)][string]$PropertyName,
1003        [Parameter(Mandatory = $True)][string]$CurrentState,
1004        [Parameter(Mandatory = $True)][string]$DesiredState,
1005        [Parameter(Mandatory = $True)][string]$Verify,
1006        [Parameter(Mandatory = $True)][string]$Abort
1007        )
1008 
1009        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
           "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1010 
1011        ### Set Log Name
1012        ###--------------------------
1013        $VMLogName = "DesiredStateLog"
1014        #$VMLogFile = $AutomationLogPath + "\" + $HostName + "\" + $VMLogName + "_" +
           $HostName  + ".txt"
1015        $VMLogFile = $AutomationLogPath + "\" + $HostName + "\temp\" + $VMLogName + "_" +
           $HostName  + ".txt"
1016        if(-NOT(Test-Path -Path $VMLogFile)) {(New-Item -ItemType file -Path $VMLogFile
           -Force | Out-Null)}
1017 
1018        Set-Variable -Name $VMLogName -Value $VMLogFile -Option AllScope -Scope global -Force
1019 
1020        ## Write Log to Guest
1021        ###--------------------------
1022        $VMLogEntry =   "ServerID=" + $ServerID + "," + "HostName=" + $HostName + "," +
           "PropertyName=" + $PropertyName + "," +  "CurrentState=" + $CurrentState + "," +
           "DesiredState=" + $DesiredState + "," +  "Verify=" + $Verify + "," +  "Abort=" +
           $Abort
1023 
1024        Write-Host "LOG: $VMLogName LOGFILE : $VMLogFile" -ForegroundColor DarkGray
1025        Write-Host "LOG: $VMLogName ENTRY   : $VMLogEntry" -ForegroundColor DarkGray
1026        $VMLogEntry | Out-File -FilePath $VMLogFile -Force -Append
```

```
1027
1028        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1029    }
1030    function Write-DesiredState-old
1031    {
1032        Param(
1033        [Parameter(Mandatory = $True)][string]$ServerID,
1034        [Parameter(Mandatory = $True)][string]$HostName,
1035        [Parameter(Mandatory = $True)][string]$PropertyName,
1036        [Parameter(Mandatory = $True)][string]$CurrentState,
1037        [Parameter(Mandatory = $True)][string]$DesiredState,
1038        [Parameter(Mandatory = $True)][string]$Verify,
1039        [Parameter(Mandatory = $True)][string]$Abort,
1040        [Parameter(Mandatory = $False)][switch]$RuninGuest
1041        )
1042
1043        ### Set Log Name
1044        ###---------------------------
1045        $VMLogName = "DesiredStateLog"
1046        $VMLogPath = $AutomationLogPath + "\" + $HostName + "\"
1047
1048        ### Set Log File Name
1049        ###---------------------------
1050        $VMLogFile = $VMLogPath + $VMLogName + "_" + $HostName  + ".txt" #+ "_" + (Get-Date
          -format "MM-dd-yyyy_hh_mm_ss") + ".txt"
1051        if(-NOT(Test-Path -Path $VMLogFile)) {(New-Item -ItemType file -Path $VMLogFile
          -Force | Out-Null)}
1052
1053        Set-Variable -Name $VMLogName -Value $VMLogFile -Option AllScope -Scope global -Force
1054
1055        $VMLogEntry =   "ServerID=" + $ServerID + "," + "HostName=" + $HostName + "," +
          "PropertyName=" + $PropertyName + "," +  "CurrentState=" + $CurrentState + "," +
          "DesiredState=" + $DesiredState + "," +  "Verify=" + $Verify + "," +  "Abort=" +
          $Abort
1056        $VMLogEntry | Out-File -FilePath $VMLogFile -Force -Append
1057    }
1058
1059    ### Write Server Desired State to SQL
1060    ### -----------------------------------------------
1061    function Write-DesiredState-SQL
1062    {
1063        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
          "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1064
1065        ### Set Log Name
1066        ###---------------------------
1067        $VMLogName = "DesiredState"
1068        #$VMLogPath = $AutomationLogPath + "\" + $HostName + "\"
1069        $VMLogPath = $AutomationLogPath + "\" + $HostName + "\temp\"
1070        $ConnectionString  = $DevOps_DBConnectionString
1071
1072        ### Open Database Connection
1073        $Connection = New-Object System.Data.SQLClient.SQLConnection
1074        $Connection.ConnectionString = $ConnectionString
1075        $Connection.Open()
1076        ### Insert Row
1077        $cmd = New-Object System.Data.SqlClient.SqlCommand
1078        $cmd.Connection = $Connection
1079
1080        ### Import Log File
1081        ###----------------------
1082        $VMLastLog = Get-ChildItem -Path ($VMLogPath) | Where-Object {($_ -like $VMLogName
          + "*")} | Sort-Object LastAccessTime -Descending | Select-Object -First 1
1083        $VMLastLogFile = $VMLogPath + "\" + $VMLastLog
1084
1085        $VMLastLogFile = Get-Content -Path $VMLastLogFile
1086
1087        foreach ($Record in $VMLastLogFile)
1088        {
```

```
1089            $Keys = $Null
1090            $Values = $Null
1091
1092            foreach($Column in ($Record.split(",")))
1093            {
1094              $Key = $Column.split("=")[0]
1095              $Value =  "'" + $Column.split("=")[1] + "'"
1096              $Keys = $Keys + $Key + ","
1097              $Values = $Values + $Value + ","
1098            }
1099            $Keys = $Keys.Substring(0,$Keys.Length-1)
1100            $Values = $Values.Substring(0,$Values.Length-1)
1101
1102            $Query = "INSERT INTO ServerDesiredState($Keys) VALUES ($Values)"
1103
1104            ### Show Results
1105            ###------------------------
1106            write-host "SQLQuery: " $Query -ForegroundColor DarkCyan
1107
1108            $cmd.CommandText = $Query
1109            $cmd.ExecuteNonQuery() #| Out-Null
1110        }
1111        ### Close
1112        $connection.Close()
1113    }
1114
1115    ### Write Server Curren State
1116    ### ---------------------------------------------
1117    function Write-ECI.CurrentState
1118    {
1119        Param(
1120        [Parameter(Mandatory = $True)][string]$ServerID,
1121        [Parameter(Mandatory = $True)][string]$HostName,
1122        [Parameter(Mandatory = $True)][string]$PropertyName,
1123        [Parameter(Mandatory = $True)][string]$CurrentState
1124        )
1125
1126        ### Set Log Name
1127        ###--------------------------
1128        $VMLogName = "CurrentStateLog"
1129        #$VMLogFile = $AutomationLogPath + "\" + $HostName + "\" + $VMLogName + "_" +
          $HostName  + ".txt"
1130        $VMLogFile = $AutomationLogPath + "\" + $HostName + "\temp\" + $VMLogName + "_" +
          $HostName  + ".txt"
1131        if(-NOT(Test-Path -Path $VMLogFile)) {(New-Item -ItemType file -Path $VMLogFile
          -Force | Out-Null)}
1132
1133        Set-Variable -Name $VMLogName -Value $VMLogFile -Option AllScope -Scope global -Force
1134
1135        ## Write Log to Guest
1136        ###--------------------------
1137        $VMLogEntry =   "ServerID=" + $ServerID + "," + "HostName=" + $HostName + "," +
          "PropertyName=" + $PropertyName + "," +  "CurrentState=" + $CurrentState
1138
1139        Write-Host "LOG: $VMLogName LOGFILE : $VMLogFile" -ForegroundColor DarkGray
1140        Write-Host "LOG: $VMLogName ENTRY   : $VMLogEntry" -ForegroundColor DarkGray
1141        $VMLogEntry | Out-File -FilePath $VMLogFile -Force -Append
1142
1143        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1144    }
1145    function Write-CurrentState-old
1146    {
1147        Param(
1148        [Parameter(Mandatory = $True)][string]$ServerID,
1149        [Parameter(Mandatory = $True)][string]$HostName,
1150        [Parameter(Mandatory = $True)][string]$PropertyName,
1151        [Parameter(Mandatory = $True)][string]$CurrentState
1152        )
1153
```

```
1154        ### Set Log Name
1155        ###---------------------------
1156        $VMLogName = "CurrentStateLog"
1157        $VMLogPath = $AutomationLogPath + "\" + $HostName + "\"
1158
1159        ### Set Log File Name
1160        ###---------------------------
1161        $VMLogFile = $VMLogPath + $VMLogName + "_" + $HostName + ".txt" #+ "_" + (Get-Date
            -format "MM-dd-yyyy_hh_mm_ss") + ".txt"
1162        if(-NOT(Test-Path -Path $VMLogFile)) {(New-Item -ItemType file -Path $VMLogFile
            -Force | Out-Null)}
1163
1164        Set-Variable -Name $VMLogName -Value $VMLogFile -Option AllScope -Scope global -Force
1165
1166        ### Export Log Entry
1167        ###---------------------------
1168        $VMLogEntry =   "ServerID=" + $ServerID + "," + "HostName=" + $HostName + "," +
            "PropertyName=" + $PropertyName + "," +  "CurrentState=" + $CurrentState
1169        $VMLogEntry | Out-File -FilePath $VMLogFile -Force -Append
1170    }
1171
1172    ### Write Server Current State to SQL
1173    ### -----------------------------------------------------
1174    function Write-CurrentState-SQL
1175    {
1176        #Param([Parameter(Mandatory = $True)][string]$ServerID)
1177
1178        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1179
1180        ### Set Log Name
1181        ###---------------------------
1182        $VMLogName = "CurrentStateLog"
1183        #$VMLogPath = $AutomationLogPath + "\" + $HostName  + "\"
1184        $VMLogPath = $AutomationLogPath + "\" + $HostName + "\temp\"
1185        $ConnectionString = $DevOps_DBConnectionString
1186
1187        ### Open Database Connection
1188        $Connection = New-Object System.Data.SQLClient.SQLConnection
1189        $Connection.ConnectionString = $ConnectionString
1190        $Connection.Open()
1191        ### Insert Row
1192        $cmd = New-Object System.Data.SqlClient.SqlCommand
1193        $cmd.Connection = $Connection
1194
1195        ### Import Log File
1196        ###----------------------
1197        $VMLastLog = Get-ChildItem -Path ($VMLogPath) | Where-Object {($_ -like $VMLogName
            + "*")} | Sort-Object LastAccessTime -Descending | Select-Object -First 1
1198        $VMLastLogFile = $VMLogPath + "\" + $VMLastLog
1199
1200        $VMLastLogFile = Get-Content -Path $VMLastLogFile
1201
1202        foreach ($Record in $VMLastLogFile)
1203        {
1204            #$Keys = $Null
1205            #$Values = $Null
1206
1207            $ServerID     = ($Record.split(","))[0].split("=")[1]
1208            $HostName     = ($Record.split(","))[1].split("=")[1]
1209            $PropertyName = ($Record.split(","))[2].split("=")[1]
1210            $CurrentState = ($Record.split(","))[3].split("=")[1]
1211
1212            $CurrentStateDateTime = "'" + (Get-Date -Format "yyyy-MM-dd HH:mm:ss") + "'"
1213
1214            if($CurrentState -eq "False"){$CurrentState = 0}
1215            if($CurrentState -eq "True"){$CurrentState = 1}
1216            else {$CurrentState = "'" + $CurrentState + "'"}
1217
```

```
1218            ### FORMAT: UPDATE tblTable SET column1 = value1, column2 = value2...., columnN
                = valueN WHERE [condition];
1219
1220            $Query = "UPDATE ServerCurrentState SET $PropertyName = $CurrentState,
                CurrentStateDateTime =  $CurrentStateDateTime WHERE ServerID = $ServerID"
1221
1222            ### Show Results
1223            ###------------------------
1224            write-host "SQLQuery: " $Query -ForegroundColor DarkCyan
1225
1226            $cmd.CommandText = $Query
1227            $cmd.ExecuteNonQuery() #| Out-Null
1228        }
1229        ### Close
1230        $connection.Close()
1231    }
1232
1233    ###                                                    <--- why????
1234    ### ------------------------------------------------
1235    function Write-ConfigReport
1236    {
1237        Param(
1238        [Parameter(Mandatory = $True, Position = 0)] [string]$Message,
1239        [Parameter(Mandatory = $False, Position = 1)] [string]$String,
1240        [Parameter(Mandatory = $False, Position = 2)] [string]$String2,
1241        [Parameter(Mandatory = $False, Position = 3)] [string]$String3,
1242        [Parameter(Mandatory = $False, Position = 4)] [string]$String4,
1243        [Parameter(Mandatory = $False, Position = 5)] [string]$String5,
1244        [Parameter(Mandatory = $False, Position = 6)] [string]$String6,
1245        [Parameter(Mandatory = $False, Position = 7)] [string]$String7,
1246        [Parameter(Mandatory = $False, Position = 8)] [string]$String8,
1247        [Parameter(Mandatory = $False, Position = 9)] [string]$String9,
1248        [Parameter(Mandatory = $False, Position = 10)] [string]$String10
1249        )
1250            function Start-ConfigReport
1251        {
1252            ### Create Timestamp
1253            $VMLogName = "ConfigReport"
1254            $TimeStamp  = Get-Date -format "MM-dd-yyyy_hh_mm_ss"
1255
1256            ### Create Log Folder
1257            $global:ConfigReportPath = $AutomationLogPath + "\" + $HostName + "\"
1258            $global:ConfigReportPath = $AutomationLogPath + "\" + $HostName + "\temp\"
1259            if(-NOT(Test-Path -Path $ConfigReportPath)) {(New-Item -ItemType directory
                -Path $ConfigReportPath -Force | Out-Null) }
1260
1261            ### Create Log File
1262            #$global:ConfigReportFile = $ConfigReportPath + "ConfigReport" + "_" +
                $TimeStamp + ".log"
1263            $global:ConfigReportFile = $ConfigReportPath + "ConfigReport" + "_" + $HostName
                + ".log"
1264
1265            if(-NOT(Test-Path -Path $ConfigReportFile)) {(New-Item -ItemType file -Path
                $ConfigReportFile -Force | Out-Null) }
1266        }
1267
1268        if (((Get-Variable 'ConfigReportFile' -Scope Global -ErrorAction 'Ignore')) -eq
        $Null) #if (-NOT($LogFile))
1269        {
1270            Start-ConfigReport
1271        }
1272
1273        ### Write the Message to the Config Report.
1274        $Message = $Message + $String + $String2 + $String3 + $String4 + $String5 + $String6
1275        Write-Host $Message
1276        $Message | Out-File -filepath $ConfigReportFile -append   # Write the Log File Emtry
1277    }
1278
1279    ### Server Build Tag     #<---------------------------------Rewrite or
```

```
    Delete????????????
1280 ### ------------------------------------------------
1281 function Write-ServerBuildTag
1282 {
1283     Write-Host "Writing ServerBuildTag:"`r`n("-" * 50)
1284
1285     ### Create Server Build Tag
1286     $ServerBuildTagPath = $AutomationLogPath + "ServerBuildTag_" + $VMName #+ "_" +
        $(get-date -f MM-dd-yyyy_HH_mm_ss) + ".txt"
1287     $ServerBuildTag = @()
1288     # Build Hash Table for Reports
1289     #------------------------------
1290
1291     $hash = [ordered]@{
1292         AD_OU                   = $ClientOU
1293         AD_UserPrincipalName    = $ADAccount.UserPrincipalName
1294         #EX_MailBox_EmailAddress = $MailBox.EmailAddresses
1295         EX_MailBox_Name         = $MailBox.Name
1296         EX_DistinguishedName    = $MailBox.DistinguishedName
1297         EX_Client_UPN           = $ClientUPN       = $ClientUPN
1298     }
1299     $PSObject = New-Object PSObject -Property $hash
1300     $ReportData += $PSObject
1301     $script:ReportData  = $ReportData
1302
1303
1304         $ServerBuildTag += (Get-ECI.EMI.Automation.ServerRecord-SQL -ServerID $ServerID)
1305
1306         Get-ECI.EMI.Automation.ServerCurrentState-SQL -ServerID $ServerID
1307         Get-ECI.EMI.Automation.ServerDesiredState-SQL -ServerID $ServerID
1308         Get-ECI.EMI.Automation.ServerConfigLog-SQL    -ServerID $ServerID
1309
1310
1311     $ServerBuildTag     = [ordered]@{
1312         VMGuestName     = (Get-WmiObject Win32_ComputerSystem).Name
1313         VMGuestOS       = [environment]::OSVersion.Version
1314         VMGuestDomain   = (Get-WmiObject Win32_ComputerSystem).Domain
1315         BuildDate       = $(get-date)
1316         Engineer        = "CBrennan - (cbrennan@eci.com)"
1317     }
1318     $ServerBuildTag | Out-File -FilePath $ServerBuildTagPath -Force
1319
1320     ### Get Module Meta Data
1321     $Modules = Get-Module | Where-Object {($_.Name -like "ECI.Core.*")}
1322     foreach($Module in $Modules)
1323     {
1324         $ModuleData = [ordered]@{
1325             Module        = $Module
1326             ModuleVersion = $Module.Version
1327         }
1328         $ModuleData | Out-File -FilePath $ServerBuildTagPath -Append
1329
1330         Write-Host "Module: " $Module
1331         Write-Host "ModuleVersion: " $Module.Version
1332     }
1333
1334 }
1335
1336 ### Open SQL Server Connection - SQL
1337 ### ------------------------------------------------
1338 function Open-SQLConnection
1339 {
1340     [OutputType([bool])]
1341
        Param([Parameter(Mandatory=$true,ValueFromPipelineByPropertyName=$true,Position=0)]$C
        onnectionString)
1342
1343     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
```

```
1344
1345        try
1346        {
1347            $Connection = New-Object System.Data.SqlClient.SqlConnection $ConnectionString;
1348            $Connection.Open();
1349            $Connection.Close();
1350
1351            Write-Host "SQL Connectivity Results: "
1352            Write-Host "ConnectionString: " $ConnectionString
1353            Return $True;
1354        }
1355        catch
1356        {
1357            Write-Host "SQL Connectivity Results: "
1358            Write-Host "ConnectionString: " $ConnectionString
1359            Return $False;
1360            $Abort = $True
1361            Invoke-AbortError
1362        }
1363    }
1364
1365    ###
    ================================================================================================
    ==========
1366    ### Get SQL Data - Get-ECI.EMI.Automation.SQLData  <---- CMDLET
1367    ###
    ================================================================================================
    ==========
1368    function Get-ECI.EMI.Automation.SQLData
1369    {
1370        [CmdletBinding(SupportsPaging = $true)]
1371
1372        PARAM(
1373            [Parameter(Mandatory = $True)] [string]$DataSetName,
1374            [Parameter(Mandatory = $True)] [string]$ConnectionString,
1375            [Parameter(Mandatory = $True)] [string]$Query,
1376            [Parameter(Mandatory = $False)] [switch]$Quiet
1377        )
1378
1379        BEGIN
1380        {
1381            ### Database Connection Object
1382            $Connection = New-Object System.Data.SQLClient.SQLConnection
1383            $Connection.ConnectionString = $ConnectionString
1384            $Connection.Open()
1385
1386            ### SQL Query Command Object
1387            $Command = New-Object System.Data.SQLClient.SQLCommand
1388            $Command.Connection = $Connection
1389            $Command.CommandText = $Query
1390            $Reader = $Command.ExecuteReader()
1391        }
1392
1393        PROCESS
1394        {
1395            ### Datatable Object
1396            $Datatable = New-Object System.Data.DataTable
1397            $Datatable.Load($Reader)
1398
1399            ### Get Columns from the  Datatable
1400            ###-------------------------------
1401            #Write-Host "Datatable.Rows.Count: " $Datatable.Rows.Count -ForegroundColor
                DarkGray
1402
1403            if($Datatable.Rows.Count -gt 0)
1404            {
1405                $Columns = $Datatable | Get-Member -MemberType Property,NoteProperty |
                    ForEach-Object {$_.Name} | Sort-Object -Property Name
1406            }
```

```
1407            elseif($Datatable.Rows.Count -eq 0)
1408            {
1409                Write-Host $DataSetName ": No Records Found Matching Query!"
                    -ForegroundColor Red
1410            }
1411
1412            #$global:Parameters = @()
1413            $PSObject = New-Object PSObject
1414
1415            foreach($Column in $Columns)
1416            {
1417                ### .Trim() all Values
1418                [string]$Value = $Datatable.$Column
1419                $Value = $Value.Trim()
1420
1421                ### Create Variables from Datatable
1422                #Set-Variable -Name $Column -Value $Datatable.$Column -Scope Global
1423                Set-Variable -Name $Column -Value $Value -Scope Global
1424
1425                #$PSObject | Add-Member -MemberTypeNoteProperty -Name $Column -Value
                    $Datatable.$Column
1426                $PSObject | Add-Member -type NoteProperty -Name $Column -Value
                    $Datatable.$Column
1427
1428                ### Build Parameter Set from Datatable
1429                #$Parameters += Get-Variable -Name $Column
1430
1431                ### Build All Parameters Set
1432                #$global:AllParameters += Get-Variable -Name $Column
1433            }
1434
1435            ### Name Dataset name = PSObject
1436            Write-Host "Getting DataSet from SQL: " $DataSetName -foregroundcolor DarkCyan
1437            Set-Variable -Name $DataSetName -Value $PSObject -Scope global
1438
1439            ### Display Parameters (or Not)
1440            if(-NOT($Quiet))
1441            {
1442                $Datatable | FL
1443                #$Parameters | FT
1444            }
1445        }
1446
1447        END
1448        {
1449            ### Close Database Connection
1450            $Connection.Close()
1451        }
1452    }
1453
1454
1455    ### Get System Config - SQL
1456    ### --------------------------------------------------
1457    function Get-ECI.EMI.Automation.SystemConfig
1458    {
1459        param(
1460        [Parameter(Mandatory = $True,Position=0)][string]$Env,
1461        [Parameter(Mandatory = $True,Position=0)][string]$DevOps_ConnectionString
1462        )
1463
1464        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1465
1466        $DataSetName = "SystemConfig"
1467        $ConnectionString = $DevOps_ConnectionString
1468        $Query = "SELECT * FROM SystemConfig WHERE Env = '$Env'"
1469
1470        Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
            $ConnectionString -Query $Query
```

```
1471
1472
1473        #$global:DevOps_ConnectionString = $DevOps_ConnectionString
1474        #Get-Variable -Name Portal_DBConnectionString
1475
1476    <#
1477        switch ($Env)
1478        {
1479            "Dev"   { $global:Portal_DBConnectionString = $DevPortal_DBConnectionString   }
1480            "Stage" { $global:Portal_DBConnectionString = $StagePortal_DBConnectionString }
1481            "Prod"  { $global:Portal_DBConnectionString = $ProdPortal_DBConnectionString  }
1482        }
1483
1484        switch ($Env)
1485        {
1486            "Dev"   { $global:vCenter_Account = $DevvCenter_Account   }
1487            "Stage" { $global:vCenter_Account = $StagevCenter_Account }
1488            "Prod"  { $global:vCenter_Account = $ProdvCenter_Account  }
1489        }
1490
1491        switch ($Env)
1492        {
1493            "Dev"   { $global:vCenter_Password = $DevvCenter_Password   }
1494            "Stage" { $global:vCenter_Password = $StagevCenter_Password }
1495            "Prod"  { $global:vCenter_Password = $ProdvCenter_Password  }
1496        }
1497    #>
1498
1499        Write-Host "DevOps_DBConnectionString  : " $DevOps_DBConnectionString
1500        Write-Host "Portal_DBConnectionString  : " $Portal_DBConnectionString
1501        Write-Host "vCenter_Account            : " $vCenter_Account
1502        Write-Host "vCenter_Password           : " (ConvertTo-SecureString
        $vCenter_Password -AsPlainText -Force)
1503        #$vCenter_Password
1504
1505
1506        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1507    }
1508
1509    ### Get Build Version - SQL
1510    ### ------------------------------------------------
1511    function Get-ECI.EMI.Automation.BuildVersion
1512    {
1513        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1514
1515        $DataSetName = "ServerBuildVersion"
1516        $ConnectionString =  $DevOps_DBConnectionString
1517        $Query = "SELECT * FROM definitionServerBuildVersions WHERE Production = '$True'"
1518
1519        Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
        $ConnectionString -Query $Query
1520
1521        Write-Host "ServerBuildVersion.BuildVersion: " $ServerBuildVersion.BuildVersion
1522
1523        $global:BuildVersion = $ServerBuildVersion.BuildVersion
1524
1525        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1526        Return $BuildVersion
1527    }
1528
1529    ### Get Server Request Parameters - SQL
1530    ### ------------------------------------------------
1531    function Get-ECI.EMI.Automation.ServerRequest
1532    {
1533        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1534
1535        $DataSetName = "ServerRequest"
```

```
1536        $ConnectionString =  $Portal_DBConnectionString
1537        $Query = "SELECT * FROM ServerRequest WHERE RequestID = '$RequestID'"
1538
1539        Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
            $ConnectionString -Query $Query
1540
1541        Write-Host "ServerRequest - RequestId  : $RequestId"     -ForegroundColor Cyan
1542        Write-Host "ServerRequest - GPID       : $GPID"          -ForegroundColor Cyan
1543        Write-Host "ServerRequest - CWID       : $CWID"          -ForegroundColor Cyan
1544        Write-Host "ServerRequest - HostName   : $HostName"`n`n  -ForegroundColor Cyan
1545
1546        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1547        Return $global:RequestId
1548    }
1549
1550    ### Get Server Role - SQL
1551    ### --------------------------------------------------
1552    function Get-ECI.EMI.Automation.ServerRole
1553    {
1554        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1555
1556        $DataSetName = "ServerRoleData"
1557        $ConnectionString = $DevOps_DBConnectionString
1558        $Query = "SELECT * FROM definitionServerRoles WHERE ServerRole =
            '$RequestServerRole' AND BuildVersion = '$BuildVersion'"
1559
1560        Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
            $ConnectionString -Query $Query
1561
1562        $global:ServerRole = $ServerRole
1563        Write-Host "ServerRole: " $ServerRole -ForegroundColor Cyan
1564        Return $ServerRole
1565    }
1566
1567    ### Get VMWare Template - SQL
1568    ### --------------------------------------------------
1569    function Get-ECI.EMI.Automation.VMWareTemplate
1570    {
1571        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1572
1573        $DataSetName = "VMWareTemplate"
1574        $ConnectionString = $DevOps_DBConnectionString
1575        $Query = "SELECT * FROM definitionVMTemplates WHERE ServerRole = '$ServerRole' AND
            BuildVersion = '$BuildVersion'"
1576
1577        Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
            $ConnectionString -Query $Query
1578
1579        $global:VMTemplateName = $VMTemplateName
1580        Write-Host "VMTemplateName: " $VMTemplateName -ForegroundColor Cyan
1581
1582        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1583        Return $global:VMTemplateName
1584    }
1585
1586    ### Get VMWare OSCustomizationSpec - SQL
1587    ### --------------------------------------------------
1588    function Get-ECI.EMI.Automation.OSCustomizationSpec-encrypt
1589    {
1590        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1591
1592        $DataSetName = "VMWareOSCustomizationSpec"
1593        $ConnectionString = $DevOps_DBConnectionString
1594        $Connection = New-Object System.Data.SQLClient.SQLConnection
1595        $Connection.ConnectionString = $ConnectionString
1596        $Connection.Open()
```

```
1597        $Command = New-Object System.Data.SQLClient.SQLCommand
1598        $Command.Connection = $Connection
1599        $Command.CommandText = $Query
1600        $Reader = $Command.ExecuteReader()
1601        $DataTable = New-Object System.Data.DataTable
1602        $DataTable.Load($Reader)
1603
1604        $DataTable
1605
1606        #Write-host "DecryptedPassword: " $DecryptedPassword -ForegroundColor Magenta
1607
1608        Write-Host "OSCustomizationSpecName: " $OSCustomizationSpecName -ForegroundColor Cyan
1609        Return $global:OSCustomizationSpecName
1610
1611        $Connection.Close()
1612    }
1613
1614    function Get-ECI.OSCustomizationSpec-original
1615    {
1616        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1617
1618        $DataSetName = "VMWareOSCustomizationSpec"
1619        $ConnectionString = $DevOps_DBConnectionString
1620
1621        #OPEN SYMMETRIC KEY SQLSymmetricKey  DECRYPTION BY CERTIFICATE
            SelfSignedCertificate;
1622        #SELECT FirstName, LastName,LoginID,UserPassword,EncryptedPassword,
            CONVERT(varchar, DecryptByKey(EncryptedPassword)) AS 'DecryptedPassword'  FROM
            UserDetails;
1623
1624        ### Decrypt AdminPassword
1625        $Query = "SELECT *, CONVERT(varchar, DecryptByKey(AdminPassword)) AS
            'AdminPassword' FROM definitionVMOSCustomizationSpec WHERE ServerRole =
            '$ServerRole' AND BuildVersion = '$BuildVersion'"
1626
1627        Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
            $ConnectionString -Query $Query
1628
1629        Return $global:OSCustomizationSpecName
1630    }
1631
1632    function Get-ECI.EMI.Automation.OSCustomizationSpec
1633    {
1634        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1635
1636        $DataSetName = "VMWareOSCustomizationSpec"
1637        $ConnectionString = $DevOps_DBConnectionString
1638        $Query = "OPEN SYMMETRIC KEY SQLSymmetricKey  DECRYPTION BY CERTIFICATE
            SelfSignedCertificate; SELECT *, CONVERT(varchar, DecryptByKey(EncryptedPassword))
            AS 'DecryptedPassword' FROM definitionVMOSCustomizationSpec WHERE ServerRole =
            '$ServerRole' AND BuildVersion = '$BuildVersion'"
1639
1640        Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
            $ConnectionString -Query $Query
1641
1642        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1643        Return $global:OSCustomizationSpecName
1644    }
1645
1646    ### Get VM Parameters - SQL
1647    ### -------------------------------------------------
1648    function Get-ECI.EMI.Automation.VMParameters
1649    {
1650        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1651
1652        $DataSetName = "VMParameters"
```

```
1653        $ConnectionString = $DevOps_DBConnectionString
1654        $Query = "SELECT * FROM definitionVMParameters WHERE serverRole = '$ServerRole' AND
            BuildVersion = '$BuildVersion'"
1655
1656        Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
            $ConnectionString -Query $Query
1657
1658        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1659    }
1660
1661    ### Get OS Parameters - SQL
1662    ### ------------------------------------------------
1663    function Get-ECI.EMI.Automation.OSParameters
1664    {
1665        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1666
1667        $DataSetName = "OSParameters"
1668        $ConnectionString = $DevOps_DBConnectionString
1669        $Query = "SELECT * FROM definitionOSParameters WHERE ServerRole = '$ServerRole' AND
            BuildVersion = '$BuildVersion'"
1670
1671        Get-ECI.EMI.Automation.SQLData -DataSetName $DataSetName -ConnectionString
            $ConnectionString -Query $Query
1672
1673        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1674    }
1675
1676    ### Check Server Record - SQL                              <--- Set Configuration
        Mode!!!!!!!!!!
1677    ### ------------------------------------------------
1678    function Check-ECI.EMI.Automation.ServerRecord
1679    {
1680        Param(
1681            [Parameter(Mandatory = $True)][string]$GPID,
1682            [Parameter(Mandatory = $True)][string]$CWID,
1683            [Parameter(Mandatory = $True)][string]$HostName
1684        )
1685
1686        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1687
1688        $ConnectionString = $DevOps_DBConnectionString
1689        $Connection = New-Object System.Data.SQLClient.SQLConnection
1690        $Connection.ConnectionString = $ConnectionString
1691        $Connection.Open()
1692        $Query = "SELECT * FROM Servers WHERE GPID = '$GPID' AND CWID = '$CWID' AND
            HostName = '$HostName'"
1693        $Command = New-Object System.Data.SQLClient.SQLCommand
1694        $Command.Connection = $Connection
1695        $Command.CommandText = $Query
1696        $Reader = $Command.ExecuteReader()
1697        $Datatable = New-Object System.Data.DataTable
1698        $Datatable.Load($Reader)
1699
1700        ### Check if Server Record Exists
1701        ###-------------------------------
1702        if(($Datatable.Rows.Count) -eq 0)
1703        {
1704            $global:ServerExists = $False
1705            #$global:ConfigurationMode =
                "Configure"
                                        ### <--- ConfigurationMode
1706            Write-Host "There is no existing ServerID record for this server."
                -ForegroundColor DarkCyan                                                ###
                <--- ConfigurationMode
1707            Write-Host `r`n('=' * 75)`r`n "This is a New Server Request: Running in
                Configure Mode!" `r`n('=' * 75)`r`n -ForegroundColor Yellow  ### <---
                ConfigurationMode
```

```
1708        }
1709        elseif(($Datatable.Rows.Count) -gt 0)
1710        {
1711            $global:ServerExists = $True
1712            #$global:ConfigurationMode =
                "Report"
                        ### <--- ConfigurationMode
1713            $global:ServerID = $ServerID
1714            Write-Host `r`n('=' * 75)`r`n "A Matching Server Record already Exists: Running
                in Report Mode!" `r`n('=' * 75)`r`n -ForegroundColor Yellow
1715
1716            Return $global:ServerID
1717        }
1718
1719        #Write-Host "ServerExists              : " $ServerExists -ForegroundColor
                Yellow                                          ### <--- ConfigurationMode
1720        #Write-Host "Setting ConfigurationMode : " $ConfigurationMode -ForegroundColor
                DarkYellow                                        ### <--- ConfigurationMode
1721
1722        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1723        #Return
            $global:ConfigurationMode
                                        ### <--- ConfigurationMode
1724    }
1725
1726    ### Write Server Status - SQL
1727    ### ------------------------------------------------
1728    function Create-ECI.EMI.Automation.ServerStatus
1729    {
1730        Param(
1731        [Parameter(Mandatory = $True)][int]$RequestID,
1732        [Parameter(Mandatory = $True)][string]$HostName,
1733        [Parameter(Mandatory = $True)][string]$ServerStatus
1734        )
1735
1736        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1737
1738        Write-Host "SQL - WRITING SERVER STATUS:"  -ForegroundColor Cyan
1739        Write-Host "RequestID    :" $RequestID    -ForegroundColor DarkCyan
1740        Write-Host "HostName     :" $HostName     -ForegroundColor DarkCyan
1741        Write-Host "ServerStatus :" $ServerStatus -ForegroundColor DarkCyan
1742
1743        $ConnectionString = $Portal_DBConnectionString
1744        $Query = "INSERT INTO
            ServerStatus(RequestID,ServerID,HostName,ServerStatus,ElapsedTime,VerifyErrorCount,Ab
            ort)
            VALUES('$RequestID','$ServerID','$HostName','$ServerStatus','$ElapsedTime','$VerifyEr
            rorCount','$Abort')"
1745        $Connection = New-Object System.Data.SQLClient.SQLConnection
1746        $Connection.ConnectionString = $ConnectionString
1747        $Connection.Open()
1748        $cmd = New-Object System.Data.SqlClient.SqlCommand
1749        $cmd.Connection = $connection
1750        $cmd.CommandText = $Query
1751        $cmd.ExecuteNonQuery() | Out-Null
1752        $connection.Close()
1753
1754        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1755    }
1756
1757    ### Update Server Status - SQL
1758    ### ------------------------------------------------
1759    function Update-ECI.EMI.Automation.ServerStatus
1760    {
1761        Param(
1762        [Parameter(Mandatory = $True)][int]$RequestID,
1763        [Parameter(Mandatory = $True)][int]$ServerID,
1764        [Parameter(Mandatory = $True)][string]$HostName,
```

```
1765        [Parameter(Mandatory = $True)][string]$VerifyErrorCount,
1766        [Parameter(Mandatory = $True)][string]$Abort,
1767        [Parameter(Mandatory = $False)][string]$ElapsedTime,
1768        [Parameter(Mandatory = $True)][string]$ServerStatus
1769        )
1770
1771        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1772
1773        Write-Host "SQL - UPDATE SERVER STATUS : "                -ForegroundColor DarkCyan
1774        Write-Host "ServerStatus               : " $ServerStatus  -ForegroundColor DarkCyan
1775        Write-Host "RequestID                  : " $RequestID     -ForegroundColor DarkCyan
1776        Write-Host "ServerID                   : " $ServerID      -ForegroundColor DarkCyan
1777        Write-Host "HostName                   : " $HostName      -ForegroundColor DarkCyan
1778        Write-Host "Verify                     : " $Verify        -ForegroundColor DarkCyan
1779        Write-Host "Abort                      : " $Abort         -ForegroundColor DarkCyan
1780        Write-Host "ElsapsedTime               : " $ElapsedTime   -ForegroundColor DarkCyan
1781
1782        $ConnectionString = $Portal_DBConnectionString
1783        $Query = "INSERT INTO
            ServerStatus(RequestID,ServerID,HostName,VerifyErrorCount,Abort,ElapsedTime,ServerSta
            tus)
            VALUES('$RequestID','$ServerID','$HostName','$VerifyErrorCount','$Abort','$ElapsedTim
            e','$ServerStatus')" #-f
            $RequestID,$ServerID,$HostName,$Verify,$Abort,$ElapsedTime,$ServerStatus
1784        $Connection = New-Object System.Data.SQLClient.SQLConnection
1785        $Connection.ConnectionString = $ConnectionString
1786        $Connection.Open()
1787        $cmd = New-Object System.Data.SqlClient.SqlCommand
1788        $cmd.Connection = $connection
1789        $cmd.CommandText = $Query
1790        $cmd.ExecuteNonQuery() | Out-Null
1791        $connection.Close()
1792
1793        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1794    }
1795
1796    ### Create Server Record - SQL
1797    ### -------------------------------------------------
1798    function Create-ECI.EMI.Automation.ServerRecord
1799    {
1800        Param(
1801        [Parameter(Mandatory = $True)][int]$RequestID,
1802        [Parameter(Mandatory = $True)][string]$GPID,
1803        [Parameter(Mandatory = $True)][string]$CWID,
1804        [Parameter(Mandatory = $True)][string]$HostName,
1805        [Parameter(Mandatory = $True)][string]$ServerRole,
1806        [Parameter(Mandatory = $True)][string]$BuildVersion
1807        )
1808
1809        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1810
1811        ###------------------------------
1812        ### Create Server Record
1813        ###------------------------------
1814        Write-Host "SQL - CREATING SERVER RECORD: " -ForegroundColor Cyan
1815        Write-Host "RequestID    : " $RequestID     -ForegroundColor DarkCyan
1816        Write-Host "GPID         : " $GPID          -ForegroundColor DarkCyan
1817        Write-Host "CWID         : " $CWID          -ForegroundColor DarkCyan
1818        Write-Host "HostName     : " $HostName      -ForegroundColor DarkCyan
1819        Write-Host "ServerRole   : " $ServerRole    -ForegroundColor DarkCyan
1820        Write-Host "BuildVersion : " $BuildVersion  -ForegroundColor DarkCyan
1821
1822        $ConnectionString = $DevOps_DBConnectionString
1823        $Connection = New-Object System.Data.SQLClient.SQLConnection
1824        $Connection.ConnectionString = $ConnectionString
1825        $Connection.Open()
1826        $cmd = New-Object System.Data.SqlClient.SqlCommand
```

```
1827        $cmd.Connection = $connection
1828        $Query = "INSERT INTO
            Servers(RequestID,GPID,CWID,HostName,ServerRole,InstanceLocation,IPv4Address,SubnetMa
            sk,DefaultGateway,PrimaryDNS,SecondaryDNS,ClientDomain,DomainUserName,BackupRecovery,
            DisasterRecovery,RequestDateTime,BuildVersion)
            VALUES('$RequestID','$GPID','$CWID','$HostName','$RequestServerRole','$InstanceLocati
            on','$IPv4Address','$SubnetMask','$DefaultGateway','$PrimaryDNS','$SecondaryDNS','$Cl
            ientDomain','$DomainUserName','$BackupRecovery','$DisasterRecovery','$RequestDateTime
            ','$BuildVersion')"
1829        $cmd.CommandText = $Query
1830        $cmd.ExecuteNonQuery() | Out-Null
1831        $connection.Close()
1832
1833        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1834    }
1835
1836    ### ------------------------------------------------
1837    function Update-ECI.EMI.Automation.ServerRecord
1838    {
1839        Param(
1840        [Parameter(Mandatory = $True)][string]$ServerID,
1841        [Parameter(Mandatory = $True)][string]$VMName,
1842        [Parameter(Mandatory = $True)][string]$ServerUUID,
1843        [Parameter(Mandatory = $True)][string]$vCenterUUID,
1844        [Parameter(Mandatory = $True)][string]$VMID
1845        )
1846
1847        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1848
1849        Write-Host "UPDATING SERVER RECORD - ServerID: $ServerID  VMName: $VMName
            ServerUUID: $ServerUUID vCenterUUID: $vCenterUUID VMID: $VMID   "
            -ForegroundColor Gray
1850        $ConnectionString = $DevOps_DBConnectionString
1851        $Query = "UPDATE Servers SET VMName = '$VMName',ServerUUID =
            '$ServerUUID',vCenterUUID = '$vCenterUUID',VMID = '$VMID'  WHERE ServerID =
            $ServerID"
1852        $Connection = New-Object System.Data.SQLClient.SQLConnection
1853        $Connection.ConnectionString = $ConnectionString
1854        $Connection.Open()
1855        $cmd = New-Object System.Data.SqlClient.SqlCommand
1856        $cmd.Connection = $connection
1857        $cmd.CommandText = $Query
1858        $cmd.ExecuteNonQuery() | Out-Null
1859        $connection.Close()
1860
1861        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1862    }
1863
1864
1865    ### Get Server ID - SQL                                    #<-------  (DATATABLE
        ROWS/COLUMNS!!!!!!)
1866    ### ------------------------------------------------
1867    function Get-ECI.EMI.Automation.ServerID
1868    {
1869        Param([Parameter(Mandatory = $True)][int]$RequestID)
1870
1871        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1872
1873        Write-Host "Getting Server ID for RequestID : " $RequestID
1874
1875        ### Parameters
1876        ###--------------------------
1877        $DataSetName = "ServerID"
1878        $ConnectionString = $DevOps_DBConnectionString
1879        $Query = "SELECT ServerID FROM Servers WHERE RequestID = '$RequestID'"
1880
1881        ### Execute Query
```

```
1882        ###--------------------------
1883        $Connection = New-Object System.Data.SQLClient.SQLConnection
1884        $Connection.ConnectionString = $ConnectionString
1885        $Connection.Open()
1886        $Command = New-Object System.Data.SQLClient.SQLCommand
1887        $Command.Connection = $Connection
1888        $Command.CommandText = $Query
1889        $Reader = $Command.ExecuteReader()
1890        $DataTable = New-Object System.Data.DataTable
1891        $DataTable.Load($Reader)
1892        $Connection.Close()
1893
1894        ### Return Values
1895        ###--------------------------
1896        foreach ($Datarow in $DataTable.Rows)
1897        {
1898           #Write-Host "Value: " $DataTable.Columns
1899           #Write-Host "Value: " $DataRow[0]
1900           Set-Variable -Name $DataTable.Columns -Value $DataRow[0] -Scope Global
1901        }
1902
1903        $global:ServerID = $ServerID
1904        Write-Host "ServerID                          : " $ServerID -ForegroundColor Cyan
1905
1906        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1907        Return $global:ServerID
1908    }
1909
1910    ### ------------------------------------------------  2!!!!
1911    function Create-ECI.EMI.Automation.CurrentStateRecord
1912    {
1913        Param([Parameter(Mandatory = $True)][int]$ServerID)
1914
1915        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1916
1917        ###-------------------------------
1918        ### Create Current State Record
1919        ###-------------------------------
1920        Write-Host "CREATING CURRENT STATE RECORD : "            -ForegroundColor Cyan
1921        Write-Host "ServerID                      : " $ServerID  -ForegroundColor Cyan
1922        Write-Host "HostName                      : " $HostName  -ForegroundColor Cyan
1923
1924        # Open Database Connection
1925        $ConnectionString = $DevOps_DBConnectionString
1926        $Connection = New-Object System.Data.SQLClient.SQLConnection
1927        $Connection.ConnectionString = $ConnectionString
1928        $Connection.Open()
1929        $cmd = New-Object System.Data.SqlClient.SqlCommand
1930        $cmd.Connection = $connection
1931        $Query = "INSERT INTO ServerCurrentState(ServerID,HostName)
            VALUES('$ServerID','$HostName')"
1932        $cmd.CommandText = $Query
1933        $cmd.ExecuteNonQuery() | Out-Null
1934        $connection.Close()
1935
1936        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1937    }
1938
1939
1940    ### ------------------------------------------------
1941    function Get-ECI.EMI.Automation.ServerRequest-SQL
1942    {
1943        Param([Parameter(Mandatory = $True)][int]$RequestID)
1944
1945        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1946
1947        Write-Host "`nGetting Server Request - RequestID: $RequestID " `r`n('-' * 50)`r`n
```

```
        -ForegroundColor Cyan
1948
1949        $ConnectionString = $Portal_DBConnectionString
1950        $Query = "SELECT * FROM ServerRequest WHERE RequestID = '$RequestID'"
1951        $connection = New-Object System.Data.SqlClient.SqlConnection
1952        $connection.ConnectionString = $ConnectionString
1953        $connection.Open()
1954        $command = $connection.CreateCommand()
1955        $command.CommandText = $query
1956        $result = $command.ExecuteReader()
1957        $table = new-object "System.Data.DataTable"
1958        $table.Load($result)
1959        $table | FL
1960        $connection.Close()
1961
1962        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1963    }
1964
1965    ### ------------------------------------------------
1966    function Get-ECI.EMI.Automation.ServerRecord-SQL
1967    {
1968        Param([Parameter(Mandatory = $True)][int]$ServerID)
1969
1970        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1971
1972        Write-Host "`nGetting Server Record - ServerID: $ServerID " `r`n('-' * 50)`r`n
        -ForegroundColor Cyan
1973
1974        $ConnectionString = $DevOps_DBConnectionString
1975        $Query = "SELECT * FROM Servers WHERE ServerID = '$ServerID'"
1976        $connection = New-Object System.Data.SqlClient.SqlConnection
1977        $connection.ConnectionString = $ConnectionString
1978        $connection.Open()
1979        $command = $connection.CreateCommand()
1980        $command.CommandText = $query
1981        $result = $command.ExecuteReader()
1982        $table = new-object "System.Data.DataTable"
1983        $table.Load($result)
1984        $table | FL
1985        $connection.Close()
1986
1987        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1988    }
1989
1990    ### ------------------------------------------------
1991    function Get-ECI.EMI.Automation.ServerCurrentState-SQL
1992    {
1993        Param([Parameter(Mandatory = $True)][int]$ServerID)
1994
1995        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
1996
1997        Write-Host "`r`nGetting Server Current State - ServerID: $ServerID " `r`n('-' *
        50)`r`n -ForegroundColor Cyan
1998
1999        $ConnectionString = $DevOps_DBConnectionString
2000        $Query = "SELECT * FROM ServerCurrentState WHERE ServerID = '$ServerID'"
2001        $connection = New-Object System.Data.SqlClient.SqlConnection
2002        $connection.ConnectionString = $ConnectionString
2003        $connection.Open()
2004        $command = $connection.CreateCommand()
2005        $command.CommandText = $query
2006        $result = $command.ExecuteReader()
2007        $table = new-object "System.Data.DataTable"
2008        $table.Load($result)
2009        $table | FL
2010        $connection.Close()
2011
```

```
2012        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
2013    }
2014
2015    ### ------------------------------------------------
2016    function Get-ECI.EMI.Automation.ServerDesiredState-SQL
2017    {
2018        Param([Parameter(Mandatory = $True)][int]$ServerID)
2019
2020        Write-Host "`r`nGetting Server Desired State - ServerID: $ServerID " `r`n('-' * 50)
            -ForegroundColor Cyan
2021
2022        $ConnectionString = $DevOps_DBConnectionString
2023        $Query = "SELECT * FROM ServerDesiredState WHERE ServerID = '$ServerID'"
2024        $connection = New-Object System.Data.SqlClient.SqlConnection
2025        $connection.ConnectionString = $ConnectionString
2026        $connection.Open()
2027        $command = $connection.CreateCommand()
2028        $command.CommandText = $query
2029        $result = $command.ExecuteReader()
2030        $table = new-object "System.Data.DataTable"
2031        $table.Load($result)
2032        $table  | FT -AutoSize -Property
            HostName,PropertyName,CurrentState,DesiredState,Verify,Abort
2033        $connection.Close()
2034
2035        $ServerDesiredState = $table
2036
2037        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
2038    }
2039
2040    ### ------------------------------------------------
2041    function Get-ECI.EMI.Automation.ServerConfigLog-SQL
2042    {
2043        Write-Host "`r`nGetting Server ConfigLog - ServerID: $ServerID " `r`n('-' * 50)`r`n
            -ForegroundColor Cyan
2044
2045        $ConnectionString = $DevOps_DBConnectionString
2046        $Query = "SELECT * FROM ServerConfigLog WHERE ServerID = '$ServerID'"
2047        $connection = New-Object System.Data.SqlClient.SqlConnection
2048        $connection.ConnectionString = $ConnectionString
2049        $connection.Open()
2050        $command = $connection.CreateCommand()
2051        $command.CommandText = $query
2052        $result = $command.ExecuteReader()
2053        $table = new-object "System.Data.DataTable"
2054        $table.Load($result)
2055        $table | FT -AutoSize -Property HostName,FunctionName,PropertyName,Verify,Abort
2056        $connection.Close()
2057
2058        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
2059    }
2060
2061    ### ------------------------------------------------
2062    function Get-DecommissionData
2063    {
2064        Param([Parameter(Mandatory = $True)][int]$ServerID)
2065
2066        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
2067
2068        ### Parameters
2069        ###--------------------------
2070        $ConnectionString = $DevOps_DBConnectionString
2071        $Query = "SELECT * FROM Servers WHERE ServerID = '$ServerID'"
2072
2073        ### Execute Query
2074        ###--------------------------
2075        $Connection = New-Object System.Data.SQLClient.SQLConnection
2076        $Connection.ConnectionString = $ConnectionString
```

```
2077        $Connection.Open()
2078        $Command = New-Object System.Data.SQLClient.SQLCommand
2079        $Command.Connection = $Connection
2080        $Command.CommandText = $Query
2081        $Reader = $Command.ExecuteReader()
2082        $DataTable = New-Object System.Data.DataTable
2083        $DataTable.Load($Reader)
2084
2085        ### Return Values
2086        ###---------------------------
2087
2088        $DecomData = @()
2089        foreach ($Datarow in $DataTable.Rows)
2090        {
2091           #Write-Host "Name: " $DataTable.Columns
2092           #Write-Host "Value: " $DataRow[0]
2093           Set-Variable -Name $DataTable.Columns -Value $DataRow[0] -Scope Global
2094           $DecomData += ($DataTable.Columns,$DataRow[0])
2095        }
2096        Write-Host "DecomData: " $DecomData
2097    }
2098
2099    ### ------------------------------------------------
2100    function GetDBSize
2101    {
2102        use "devops"
2103        exec sp_spaceused
2104    }
2105
2106
2107
2108    function Start-ECI.EMI.Automation.Sleep
2109    {
2110        Param(
2111        [Parameter(Mandatory = $False)][int16]$t,
2112        [Parameter(Mandatory = $False)][string]$Message,
2113        [Parameter(Mandatory = $False)][switch]$ShowRemaining
2114        )
2115
2116        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor DarkGray
2117
2118        If(!$t){$t = $WaitTime_StartSleep}
2119
2120        if($Message)
2121        {
2122            Write-Host `r`n('- ' * 25)`r`n "START-ECI.SLEEP: $t seconds. $Message" `r`n('-
                ' * 25)`r`n -ForegroundColor Cyan
2123        }
2124        else
2125        {
2126            Write-Host `r`n('- ' * 25)`r`n "START-ECI.SLEEP: $t seconds" `r`n('- ' *
                25)`r`n -ForegroundColor Cyan
2127        }
2128
2129        for ($i=$t; $i -gt 1; $i--)
2130        {
2131            if($ShowRemaining)
2132            {
2133                $a = [math]::Round((($i/$t)/1)*100)    ### Amount Remaining
2134            }
2135            else
2136            {
2137                $a = [math]::Round( (($t-$i)/$t)*100) ### Amount Completed
2138            }
2139
2140            if($Message){$Status = "Waiting for... $Message"}else{$Status = "Waiting for...
                "}
2141
```

```
2142            Write-Progress -Activity "ECI START-SLEEP - for $t seconds: " -SecondsRemaining
                $i -CurrentOperation "Completed: $a%" -Status $Status
2143            Start-Sleep 1
2144        }
2145        Write-Host "Done Sleeping." -ForegroundColor DarkGray
2146        Write-Progress -Activity 'Sleeping...' -Completed
2147    }
2148
2149
2150    function Generate-ECI.RandomAlphaNumeric
2151    {
2152        Param([Parameter(Mandatory = $False)][int]$Length)
2153
2154        if(!$Length){[int]$Length = 15}
2155
2156        ##ASCII
2157        #48 -> 57 :: 0 -> 9
2158        #65 -> 90 :: A -> Z
2159        #97 -> 122 :: a -> z
2160
2161        for ($i = 1; $i -lt $Length; $i++) {
2162
2163            $a = Get-Random -Minimum 1 -Maximum 4
2164
2165            switch ($a)
2166            {
2167                1 {$b = Get-Random -Minimum 48 -Maximum 58}
2168                2 {$b = Get-Random -Minimum 65 -Maximum 91}
2169                3 {$b = Get-Random -Minimum 97 -Maximum 123}
2170            }
2171
2172            [string]$c += [char]$b
2173        }
2174
2175        Return $c
2176    }
2177
2178    ########################################
2179    ### Function: Set-TranscriptPath
2180    ########################################
2181    function Start-ECI.EMI.Automation.Transcript
2182    {
2183        Param(
2184        [Parameter(Mandatory = $False)][string]$TranscriptPath,
2185        [Parameter(Mandatory = $False)][string]$TranscriptName
2186        )
2187
2188        function Generate-RandomAlphaNumeric
2189        {
2190            Param([Parameter(Mandatory = $False)][int]$Length)
2191
2192            if(!$Length){[int]$Length = 15}
2193
2194            ##ASCII
2195            #48 -> 57 :: 0 -> 9
2196            #65 -> 90 :: A -> Z
2197            #97 -> 122 :: a -> z
2198
2199            for ($i = 1; $i -lt $Length; $i++)
2200            {
2201                $a = Get-Random -Minimum 1 -Maximum 4
2202                switch ($a)
2203                {
2204                    1 {$b = Get-Random -Minimum 48 -Maximum 58}
2205                    2 {$b = Get-Random -Minimum 65 -Maximum 91}
2206                    3 {$b = Get-Random -Minimum 97 -Maximum 123}
2207                }
2208                [string]$c += [char]$b
2209            }
```

```
2210
2211            Return $c
2212        }
2213
2214        ### Stop Transcript if its already running
2215        try {Stop-transcript -ErrorAction SilentlyContinue} catch {}
2216
2217        $TimeStamp  = Get-Date -format "yyyyMMddhhmss"
2218        $Rnd = (Generate-RandomAlphaNumeric)
2219
2220        ### Set Default Path
2221        if(!$TranscriptPath){$TranscriptPath = "C:\Scripts\Transcripts"}
2222
2223        ### Make sure path ends in "\"
2224        $LastChar = $TranscriptPath.substring($TranscriptPath.length-1)
2225        if ($LastChar -ne "\"){$TranscriptPath = $TranscriptPath + "\"}
2226
2227        ### Create File Name
2228        if($TranscriptName)
2229        {
2230            $TranscriptFile = $TranscriptPath + "PowerShell_transcript" + "." +
                $TranscriptName + "." + $Rnd + "." + $TimeStamp + ".txt"
2231        }
2232        else
2233        {
2234            $TranscriptFile = $TranscriptPath + "PowerShell_transcript" + "." + $Rnd + "."
                + $TimeStamp + ".txt"
2235        }
2236        ### Start Transcript Log
2237        Start-Transcript -Path $TranscriptFile -NoClobber
2238    }
2239
2240
2241
2242    ###################################################################
2243    ### ERROR HANDELING
2244    ###################################################################
2245
2246    ### Write Error Stack - Try/Catch
2247    ### -------------------------------------------------
2248    function Write-ECI.ErrorStack
2249    {
2250        Param(
2251        [Parameter(Mandatory = $False)][switch]$Details,
2252        [Parameter(Mandatory = $False)][switch]$NoExit
2253        )
2254
2255        ###
            https://kevinmarquette.github.io/2017-04-10-Powershell-exceptions-everything-you-ever
            -wanted-to-know/
2256
2257    <#
2258
2259
2260            ### Guest State
2261            ###----------------------------
2262            $VM = (Get-VM -Name $VMName -ErrorAction SilentlyContinue)
2263            $VMguestState                       =
                $VM.ExtensionData.guest.guestState                              ### <---- RETRUN:
                running/notRunning
2264            $VMguestOperationsReady             =
                $VM.ExtensionData.guest.guestOperationsReady                    ### <---- RETRUN:
                True/False
2265            $VMinteractiveGuestOperationsReady =
                $VM.ExtensionData.guest.interactiveGuestOperationsReady         ### <---- RETRUN:
                True/False
2266            $VMguestStateChangeSupported        =
                $VM.ExtensionData.guest.guestStateChangeSupported
2267            Write-Host "VMguestState                        :"
```

```
        $VMguestState                                   -ForegroundColor DarkGray
        Write-Host "VMguestOperationsReady            :"
        $VMguestOperationsReady                         -ForegroundColor DarkGray
        Write-Host "VMinteractiveGuestOperationsReady :"
        $VMinteractiveGuestOperationsReady              -ForegroundColor DarkGray
        Write-Host "VMguestStateChangeSupported       :"
        $VMguestStateChangeSupported                    -ForegroundColor DarkGray
#>

    $Abort    = $True
    #$ECIError = $True
    $Alert    = $True

    if($global:Error.Count -eq 0)
    {
        Write-Host "No Errors in the PS-ErrorStack" -ForegroundColor Green
    }
    elseif($global:error.Count -gt 0)
    {
        ### $Error Variable
        ###-----------------------------------------
        Write-Host `r`n`r`n('=' * 100)`r`n "START - ECI ERROR STACK: `r`n" ('=' *
        100)`r`n               -ForegroundColor Red
        Write-Host "PS-Error.Count:"
        $global:error.Count                                 -ForegroundColor Red
        Write-Host "PS-ERRROR[0]:"                          `r`n('-' *
        20)`r`n                                 -ForegroundColor Red
        Write-Host "PS-Error.InvocationInfo    [0]    : "
        ($global:Error[0].InvocationInfo.Line).Trim()  -ForegroundColor Red
        Write-Host "PS-Error.Exception         [0]    : "
        $global:Error[0].TargetObject                   -ForegroundColor Red
        Write-Host "PS-Error.ExceptionType     [0]    : "
        $global:Error[0].Exception.GetType().fullname  -ForegroundColor Red
        Write-Host "PS-Error.Exception.Message [0]    : "
        $global:Error[0].Exception.Message              -ForegroundColor Red
        #Write-Host "Error.ScriptStackTrace[0]     : "
        $global:Error[0].ScriptStackTrace                   -ForegroundColor DarkGray

        if($Details -eq $True)
        {
            ### ScriptStackTrace
            ###-----------------------------------------
            $StackTrace = $global:Error.ScriptStackTrace
            foreach($Trace in $StackTrace)
            {
                Write-Host  ('-' * 50)                      -ForegroundColor Red
                Write-Host  "PS-ScriptStackTrace:"          -ForegroundColor Red

                $Call       = $Trace.Split()[1]
                $LineNumber = $Trace.Split()[4]
                $ScriptPath = (Split-Path(($Trace.Split()[2]).split(":")[0]) -Parent) +
                "\"
                $ScriptName = Split-Path ($Trace.Split()[2]) -Leaf

                #Write-Host "Trace       : " $Trace       -ForegroundColor yellow
                foreach($Item in $Trace)
                {
                    $Call       = $Item.Split()[1]
                    $LineNumber = $Item.Split()[4]
                    $ScriptPath = (Split-Path(($Item.Split()[2]).split(":")[0])
                    -Parent) + "\"
                    $ScriptName = Split-Path ($Item.Split()[2]) -Leaf

                    Write-Host  ('-' * 50)                      -ForegroundColor Red
                    Write-Host "PS-Call       : " $Call       -ForegroundColor Red
                    Write-Host "PS-ScriptPath : " $ScriptPath -ForegroundColor Red
                    Write-Host "PS-ScriptName : " $ScriptName -ForegroundColor Red
                    Write-Host "PS-LineNumber : " $LineNumber -ForegroundColor Red
                }
```

```
2323                }
2324
2325                for($i = 0; $i -le $global:error.count -1; $i++)
2326                {
2327                    Write-Host  ('-' * 50) -ForegroundColor DarkRed
2328                    Write-Host "PS-Error.InvocationInfo    $i : "
                        ($global:Error[$i].InvocationInfo.Line).Trim() -ForegroundColor DarkRed
2329                    Write-Host "PS-Error.Exception.Message $i : "
                        $global:Error[$i].Exception.Message -ForegroundColor DarkRed
2330                    Write-Host "PS-Error.TargetObject      $i : "
                        $global:Error[$i].TargetObject -ForegroundColor DarkRed
2331                    Write-Host "PS-Error.ExceptionType     $i : "
                        $global:Error[$i].Exception.GetType().fullname  -ForegroundColor DarkRed
2332                    Write-Host "PS-Error.ScriptStackTrace  $i : "
                        $global:Error[$i].ScriptStackTrace -ForegroundColor DarkRed
2333                }
2334            }
2335
2336            ### CallStack
2337            ###-------------------------------------------
2338            $CallStack = Get-PSCallStack -Verbose -Debug
2339            Write-Host `r`n`r`n "PS-Callstack - Count             : " $CallStack.Count
                -ForegroundColor DarkGray
2340            for($i = 0; $i -le $CallStack.count -1; $i++)
2341            {
2342                Write-Host  ('-' * 50) -ForegroundColor DarkGray
2343                Write-Host "PS-Callstack.ScriptName[$i]       : " (Split-Path -Path
                    $CallStack[$i].ScriptName -Leaf) -ForegroundColor DarkGray
2344                Write-Host "PS-Callstack.Command[$i]          : " $CallStack[$i].Command
                    -ForegroundColor DarkGray
2345
2346                if(($CallStack[$i].Command) -ne ($CallStack[$i].FunctionName))
2347                {
2348                    Write-Host "PS-Callstack.FunctionName[$i]     : "
                        $CallStack[$i].FunctionName -ForegroundColor DarkGray
2349                }
2350            }
2351
2352            Write-Host `r`n`r`n('=' * 100)`r`n "END - ERROR STACK: " `r`n('=' * 100)
                -ForegroundColor Red
2353
2354
2355            ### Send Alert Message
2356            ###-------------------------------------------
2357            if($Alert)
2358            {
2359                #Send-ECI.ServerStatus -ServerID $ServerID -Abort $Abort -VerifyErrorCount
                    $VerifyErrorCount
2360                Send-ECI.Alert -ErrorMsg $Error[0]
2361            }
2362
2363            if($NoExit)
2364            {
2365                Continue
2366            }
2367            elseif(!$NoExit)
2368            {
2369                 ### Exit
2370                ###------------------
2371                Throw "ABORT ERROR THROWN"     ### <--- Throw Terminating Error
2372                #[Environment]::Exit(1)        ### <--- Exits PS Session
2373                #Exit
2374            }
2375        }
2376    }
2377
2378
2379    ### Throw Abort Error
2380    ### ------------------------------------------------
```

```powershell
function Throw-ECI.AbortError
{

    #https://kevinmarquette.github.io/2017-04-10-Powershell-exceptions-everything-you-ever-wanted-to-know/
    Param(
        [Parameter(Mandatory = $False)][string]$ServerID,
        [Parameter(Mandatory = $False)][string]$HostName,
        [Parameter(Mandatory = $False)][string]$VMName,
        [Parameter(Mandatory = $False)][string]$FunctionName,
        [Parameter(Mandatory = $False)][string]$Verify,
        [Parameter(Mandatory = $False)][string]$AbortTrigger,
        [Parameter(Mandatory = $False)][string]$Abort
        )

    Write-Host `r`n("-=" * 50)`r`n(" " * 37)"THROWING ABORT ERROR!!!"`r`n("-+" * 50)`r`n  -ForegroundColor Red
    Write-Host "The scripts encountered an Abort Level Error." `n`n  -ForegroundColor Red


    #Write-Host "THROW-ABORTERROR: " "ServerID: " $ServerID "HostName: " $HostName "VMName: " $VMName "FunctionName:" $FunctionName "Verify:" $Verify "AbortTrigger:" $AbortTrigger "Abort:" $Abort -ForegroundColor Red
    Write-Host "PS-ERRORVAR.Count : " $global:Error.Count -ForegroundColor Yellow
    Write-Host "PS-ERRORVAR[0]    : " $global:Error[0] -ForegroundColor Yellow
    Write-Host `r`n`r`n

    #Write-AbortErrorLog
    #Write-AbortErrortoSQL
    #Write-ECI.ErrorStack

    Send-ECI.ServerStatus

    ### Exit
    ###-------------------
    Throw "ABORT ERROR THROWN"      ### <--- Throw Terminating Error
    #[Environment]::Exit(1)         ### <--- Exits PS Session
    #Exit                           ### <--- Exits Current Context
}

function Send-ECI.ServerStatus
{
    Param(
    [Parameter(Mandatory = $False)][int]$ServerID,
    [Parameter(Mandatory = $True)][bool]$Abort,
    [Parameter(Mandatory = $True)][int]$VerifyErrorCount
    )

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

    ### Abort Status
    ###---------------------------
    if($Abort -eq $False)
    {
        $Status      = "Success"
        $StatusColor = "Green"
    }
    elseif($Abort -eq $True)
    {
        $Status      = "Failure"
        $StatusColor = "Red"
    }

    ### Abort Status
    ###---------------------------
    if($ECIError -eq $False)
    {
        $ECIStatus      = "Success"
```

```
2444            $ECIStatusColor = "Green"
2445        }
2446        elseif($ECIError -eq $True)
2447        {
2448            $ECIStatus      = "Failure"
2449            $ECIStatusColor = "Red"
2450        }
2451
2452        ### Verify Error Count
2453        ###--------------------------
2454        if($VerifyErrorCount -eq 0)
2455        {
2456            $VerifyStatus      = "No Verify Errors"
2457            $VerifyStatusColor = "Green"
2458        }
2459        elseif($VerifyErrorCount -gt 0)
2460        {
2461            $VerifyStatus      = "Verify Errors"
2462            $VerifyStatusColor = "Yellow"
2463        }
2464
2465        ### Message Header
2466        ###-------------------------------------------------------------------------
2467        $Message = $Null
2468
2469        $Header = "
2470            <style>
2471            #BODY{font-family: Lucida Console, Consolas, Courier New,
                monospace;font-size:9;font-color: #000000;text-align:left;}
2472            BODY{font-family: Verdana, Arial, Helvetica, sans-serif;font-size:9;font-color:
                #000000;text-align:left;}
2473            TABLE {border-width: 0px; border-style: hidden; border-color: white;
                border-collapse: collapse;}
2474            TH {border-width: 0px; padding: 3px; border-style: hidden; border-color: white;
                background-color: #6495ED;}
2475            TD {border-width: 0px; padding: 3px; border-style: hidden; border-color: white;}
2476
2477            </style>
2478        "
2479        $Message += $Header
2480        $Message += "<html><body>"
2481        $Message += "<table>"
2482        $Message += "<tr>"
2483        $Message += "<td align='right'>" + "Status : </td>"
2484        $Message += "<td align='left'><font size='4';color='$StatusColor'>" + $Status   +
            "</font></td>"
2485        $Message += "</tr>"
2486        $Message += "<tr>"
2487        $Message += "<td align='right'>" + "HostName : </td>"
2488        $Message += "<td align='left'><font size='4';color='$StatusColor'>" + $HostName +
            "</font></td>"
2489        $Message += "</tr>"
2490        $Message += "</table>"
2491        $Message += "<br>"
2492        $Message += "FOR ECI INTERNAL USE ONLY."                                    + "`r`n" +
            "<br><br>"
2493
2494
2495        ### SERVER PROVISIONING STATUS
2496        ###----------------------------------------------
2497        $Message += "-------------------------------------------------------------------------"
            + "`r`n" + "<br>"
2498        $Message += "<b>SERVER PROVISIONING STATUS: "
            + "`r`n" + "</b><br>"
2499        $Message += "-------------------------------------------------------------------------"
            + "`r`n" + "<br>"
2500
2501        $StatusParams = [ordered]@{
2502            Status           = $Status
```

```
2503            HostName          = $HostName
2504            VerifyStatus      = $VerifyStatus
2505            PSErrorVar        = $global:Error.Count
2506            ECIErrorVar       = $global:ECIError.Count
2507            VerifyErrorCount  = $VerifyErrorCount
2508            Abort             = $Abort
2509        }
2510
2511        $StatusMsg = "<table>"
2512        foreach($Param in $StatusParams.GetEnumerator())
2513        {
2514            $StatusMsg += "<tr>"
2515            $StatusMsg += "<td align='right'>" + $Param.Name + "  : </td>"
2516            $StatusMsg += "<td align='left'>  "  + $Param.Value + "</td>"
2517            $StatusMsg += "</tr>"
2518        }
2519        $StatusMsg += "</table>"
2520        $Message += $StatusMsg
2521
2522        ### SERVER SPECIFICATIONS
2523        ###-----------------------------------------------
2524        $Message += "<br><br>"
2525        $Message += "------------------------------------------------------------------------"
        + "`r`n" + "<br>"
2526        $Message += "<b>SERVER SPECIFICATIONS : "
        + "`r`n" + "</b><br>"
2527        $Message += "------------------------------------------------------------------------"
        + "`r`n" + "<br>"
2528
2529        $DetailParams = [ordered]@{
2530            RequestDateTimeUTC    = $RequestDateTime
2531            Hostame               = $HostName
2532            VMame                 = $VMName
2533            GPID                  = $GPID
2534            RequestID             = $RequestID
2535            ServerID              = $ServerID
2536            InstanceLocation      = $InstanceLocation
2537            ServerRole            = $ServerRole
2538            ServerRoleDescription = $ServerRoleDescription
2539            BuildVersion          = $BuildVersion
2540            vCPUCount             = $vCPUCount
2541            vMemoryGB             = $vMemoryGB
2542            OSVolumeCapacityGB    = $OSVolumeCapacityGB
2543            SwapVolumeCapacityGB  = $SwapVolumeCapacityGB
2544            IPv4Address           = $IPv4Address
2545            ClientDomain          = $ClientDomain
2546            VMProvisionTime       = $VMElapsedTime
2547            OSConfigurationTime   = $OSElapsedTime
2548            RoleConfigurationTime = "00:00:00" # $RoleElapsedTime
2549            DeploymentQATime      = $QAElapsedTime
2550            TotalAutomationTime   = ((Get-Date) - $AutomationStartTime)
2551        }
2552        $details = "<table>"
2553
2554
2555        foreach($Param in $DetailParams.GetEnumerator())
2556        {
2557            $details += "<tr>"
2558            $details += "<td align='right'>" + $Param.Name + "  : </td>"
2559            $details += "<td align='left'>  "  + $Param.Value + "</td>"
2560            $details += "</tr>"
2561        }
2562        $details += "</table>"
2563        $Message += $details
2564        $Message += "<br><br>"
2565
2566
2567        ### Display Desired State SQL Record
2568        ###-------------------------------------------------------------------------
```

```
2569        $Message += "<font size='3';><b>SERVER DESIRED CONFIGURATION STATE:</b>"
            + "</font><br>"
2570        $Header = "
2571            <style>
2572            BODY{font-family: Verdana, Arial, Helvetica, sans-serif;font-size:9;font-color:
                #000000;text-align:left;}
2573            TABLE {border-width: 1px; border-style: solid; border-color: black;
                border-collapse: collapse;}
2574            </style>
2575        "
2576        $Message += $Header
2577
2578        $DataSetName = "DesiredState"
2579        $ConnectionString = "Server=automate1.database.windows.net;Initial
            Catalog=DevOps;User ID=devops;Password=JKFLKA8899*(*(32faiuynv;"
2580        $Query = "SELECT
            ServerID,HostName,PropertyName,DesiredState,CurrentState,Verify,Abort,RecordDateTimeU
            TC FROM ServerDesiredState WHERE ServerID = '$ServerID'"
2581        $Connection = New-Object System.Data.SQLClient.SQLConnection
2582        $Connection.ConnectionString = $ConnectionString
2583        $Connection.Open()
2584        $Command = New-Object System.Data.SQLClient.SQLCommand
2585        $Command.Connection = $Connection
2586        $Command.CommandText = $Query
2587        $Reader = $Command.ExecuteReader()
2588        $DataTable = New-Object System.Data.DataTable
2589        $DataTable.Load($Reader)
2590        $dt = $DataTable
2591        $dt | ft
2592
2593        $DesiredState += "<table>"
2594        $DesiredState +="<tr>"
2595        for($i = 0;$i -lt $dt.Columns.Count;$i++)
2596        {
2597            $DesiredState += "<b><u><td>"+$dt.Columns[$i].ColumnName+"</td></u></b>"
2598        }
2599        $DesiredState +="</tr>"
2600
2601        for($i=0;$i -lt $dt.Rows.Count; $i++)
2602        {
2603            $DesiredState +="<tr>"
2604            for($j=0; $j -lt $dt.Columns.Count; $j++)
2605            {
2606                $DesiredState += "<td>"+$dt.Rows[$i][$j].ToString()+"</td>"
2607            }
2608            $DesiredState +="</tr>"
2609        }
2610
2611        $DesiredState += "</table></body></html>"
2612        $Message += $DesiredState
2613        $Message += "<br><br>"
2614
2615
2616        ### Transcript Log
2617        ###---------------------
2618        $TranscriptURL = "cloud-portal01.eci.cloud/vmautomationlogs/" + $HostName + "/" +
            (Split-Path $TranscriptFile -Leaf)
2619        $Message += "Transcript Log: " + "<a href=http://" + $TranscriptURL + ">" +
            $TranscriptURL + "</a>"
2620        $Message += "<br><br>"
2621        $Message += "Server Build Date: " + (Get-Date)
2622        $Message += "<br>"
2623
2624
2625        ### ECI-Error Array
2626        ###---------------------
2627        if($global:ECIError)
2628        {
2629            $Message += "<br><b>ECI-ErrorVar:  " + $global:ECIError.count  + " </b><br>"
```

```
2630            for($i = 0; $i -le $global:ECIError.count -1; $i++)
2631            {
2632                $Message += "ECIError: $i<br>"
2633                $Message += ($global:ECIError[$i])
2634                $Message += "<br>"
2635            }
2636        }
2637
2638        ### ECI-Error Log
2639        ###--------------------
2640        # Get-Content C:\scripts\_vmautomationlog\hostname\ecierrorlogs.log
2641
2642
2643        ### PS-Error Array
2644        ###--------------------
2645        if($global:Error)
2646        {
2647            $Message += "<br><b>PS-ErrorVar:  " + $global:error.count + "</b><br>"
2648            for($i = 0; $i -le $global:error.count -1; $i++)
2649            {
2650                $Message += "ERROR: $i<br>"
2651                $Message += ($global:Error[$i])
2652                $Message += "<br>"
2653            }
2654        }
2655
2656        ### Close Message
2657        ###--------------------
2658        $Message += "</body></html>"
2659
2660        ### Email Constants
2661        ###--------------------
2662        $From    = "cbrennan@eci.com"
2663        if($Abort -eq $False)
2664        {
2665            $To      = "cbrennan@eci.com,sdesimone@eci.com,wercolano@eci.com,rgee@eci.com"
2666        }
2667        elseif($Abort -eq $True)
2668        {
2669            $To     = "cbrennan@eci.com"
2670            $Message += "PS-Error.Count: " + $Error.Count
2671            $Message += $Error
2672        }
2673
2674        $SMTP    = "alertmx.eci.com"
2675        #$SMTP   = $SMTPServer
2676        $Subject = "SERVER PROVISIONING STATUS: " + $Status
2677
2678        ### Email Message
2679        ###-------------------------------------------------------------------------------
2680        Write-Host `r`n`r`n`r`n("=" * 50)`n"SENDING NOTIFICATION MESSAGE:" $Status`r`n("="
          * 50)`r`n`r`n -ForegroundColor $StatusColor
2681
2682        #Write-Host `n "MESSAGE: " $Message `n -ForegroundColor $StatusColor
2683        Write-Host "TO: " $To
2684        Send-MailMessage -To ($To -split ",") -From $From -Body $Message -Subject $Subject
          -BodyAsHtml -SmtpServer $SMTP
2685
2686
2687        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
2688    }
2689
2690    function ECIHeader
2691    {
2692        $Header  = "<style>"
2693        $Header += "BODY{font-family: Verdana, Arial, Helvetica,
          sans-serif;font-size:9;font-color: #000000;text-align:left;}"
2694        $Header += "TABLE{border-width: 1px;border-style: solid;border-color:
          black;border-collapse: collapse;}"
```

```
2695        $Header += "TH{border-width: 1px;padding: 0px;border-style: solid;border-color:
            black;background-color: #D2B48C}"
2696        $Header += "TD{border-width: 1px;padding: 0px;border-style: solid;border-color:
            black;background-color: #FFEFD5}"
2697        $Header += "</style>"
2698
2699        ### HTML Header
2700        ###--------------------
2701        $Header = "
2702            <style>
2703            BODY{font-family: Lucida Console, Consolas, Courier New,
            monospace;font-size:9;font-color: #000000;text-align:left;}
2704            #TABLE {border-width: 1px; border-style: solid; border-color: black;
            border-collapse: collapse;}
2705            #TH {border-width: 1px; padding: 3px; border-style: solid; border-color: black;
            background-color: #6495ED;}
2706            #TD {border-width: 1px; padding: 3px; border-style: solid; border-color: black;}
2707            </style>
2708        "
2709    }
2710
2711    function Set-ECI.PS.BufferSize
2712    {
2713        #$BufferSize = $host.UI.RawUI.BufferSize
2714        $host.UI.RawUI.BufferSize = New-Object
            System.Management.Automation.Host.Size(160,5000)
2715    }
2716
2717    function Write-ECI.Function
2718    {
2719        Param(
2720        [Parameter(Mandatory = $True)][switch]$Head,
2721        [Parameter(Mandatory = $True)][switch]$Tail
2722        )
2723
2724        if($Head)
2725        {
2726            $FunctionName = $((Get-PSCallStack)[0].Command)
2727            Write-Host `r`n('-' * 75)`r`n "EXECUTING FUNCTION: " $FunctionName `r`n('-' *
            75) -ForegroundColor Gray
2728        }
2729        if($Tail)
2730        {
2731            $FunctionName = $((Get-PSCallStack)[0].Command)
2732            Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor
            DarkGray
2733        }
2734    }
2735
2736
2737    function Report-ECI.EMI.ReadOnlyVMReport
2738    {
2739        Param(
2740            [Parameter(Mandatory = $True)][string]$InstanceLocation,
2741            [Parameter(Mandatory = $True)][string]$GPID,
2742            [Parameter(Mandatory = $True)][string]$VMName,
2743            [Parameter(Mandatory = $True)][string]$ConfigurationMode,
2744            [Parameter(Mandatory = $True)][string]$ECIVMTemplate,
2745            [Parameter(Mandatory = $True)][string]$OSCustomizationSpecName,
2746            [Parameter(Mandatory = $True)][string]$ResourcePool,
2747            [Parameter(Mandatory = $True)][string]$PortGroup,
2748            [Parameter(Mandatory = $True)][string]$OSDataStore,
2749            [Parameter(Mandatory = $True)][string]$SwapDataStore,
2750            [Parameter(Mandatory = $True)][string]$vCPU,
2751            [Parameter(Mandatory = $True)][string]$vMemory,
2752            [Parameter(Mandatory = $True)][string]$IPv4Address,
2753            [Parameter(Mandatory = $True)][string]$SubnetMask,
2754            [Parameter(Mandatory = $True)][string]$DefaultGateway,
2755            [Parameter(Mandatory = $True)][string]$PrimaryDNS,
```

```
2756            [Parameter(Mandatory = $True)][string]$SecondaryDNS,
2757            [Parameter(Mandatory = $False)][string]$ReportMode_Timeout
2758        )
2759
2760        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
2761
2762        ############################
2763        ### HTML Header
2764        ############################
2765        $Header  = "<style>"
2766        $Header += "BODY{font-family: Verdana, Arial, Helvetica,
            sans-serif;font-size:9;font-color: #000000;text-align:left;}"
2767        $Header += "TABLE{border-width: 1px;border-style: solid;border-color:
            black;border-collapse: collapse;}"
2768        $Header += "TH{border-width: 1px;padding: 0px;border-style: solid;border-color:
            black;background-color: #D2B48C}"
2769        $Header += "TD{border-width: 1px;padding: 0px;border-style: solid;border-color:
            black;background-color: #FFEFD5}"
2770        $Header += "</style>"
2771
2772        ############################
2773        ### Format Report Data
2774        ############################
2775
2776        $Message = $Null
2777
2778        $Header = "
2779            <style>
2780            BODY{font-family: Verdana, Arial, Helvetica, sans-serif;font-size:9;font-color:
                #000000;text-align:left;}
2781            TABLE {border-width: 0px; border-style: hidden; border-color: white;
                border-collapse: collapse;}
2782            TH {border-width: 0px; padding: 3px; border-style: hidden; border-color: white;
                background-color: #6495ED;}
2783            TD {border-width: 0px; padding: 3px; border-style: hidden; border-color: white;}
2784
2785            </style>
2786        "
2787        $Message += $Header
2788
2789        $Message += "<font size='3';color='gray'><i>Server Provisioning - Report
            Mode:</i></font><br>"
2790        $Message += "<font size='5';color='NAVY'><b>Parameters for New Server</b></font><br>"
2791        $Message += "<font size='2'>Request Date:" + (Get-Date) + "</font><br>"
2792        $Message += "<font size='2'>Requested By:" +  "</font><br><br><br>"
2793        $Message += "<font size='3';color='red'>This Server <b> WAS NOT </b> provisioned.
            These are the parameters that would have been used to create the new server: </font>"
2794        $Message += "<br><br><br>"
2795
2796        ### Server Specs
2797        $Message += "<font size='2';color='NAVY'><b>SERVER SPECIFICATIONS:</b></font><br>"
2798        $Message += "<table>"
2799        $Message += "<tr><td align='right'>VMName : </td><td
            align='left'>"                    + $VMName                    + "</td></tr>"
2800        $Message += "<tr><td align='right'>ConfigurationMode : </td><td
            align='left'>"       + $ConfigurationMode        + "</td></tr>"
2801        $Message += "<tr><td align='right'>ECIVMTemplate : </td><td
            align='left'>"           + $ECIVMTemplate            + "</td></tr>"
2802        $Message += "<tr><td align='right'>OSCustomizationSpecName : </td><td
            align='left'>" + $OSCustomizationSpecName  + "</td></tr>"
2803        $Message += "</table>"
2804
2805        $Message += "<br>"
2806
2807        ### VCenter Resources
2808        $Message += "<font size='2';color='NAVY'><b>VCENTER RESOURCES:</b></font><br>"
2809        $Message += "<table border='1'>"
2810
```

```
$Message += "<tr><td align='right'>vCPU : </td><td align='left'>"                    + $vCPU                    + "</td></tr>"
$Message += "<tr><td align='right'>vMemory : </td><td align='left'>"                 + $vMemory                 + "</td></tr>"

$Message += "<font size='2.5';color='NAVY'>"

$Message += "<tr><td align='right'>ResourcePool : </td><td align='left'>"           + $ResourcePool            + "</td></tr>"
$Message += "<tr><td align='right'>PortGroup : </td><td align='left'>"              + $PortGroup               + "</td></tr>"
$Message += "<tr><td align='right'>OSDataStore : </td><td align='left'>"            + $OSDataStore             + "</td></tr>"
$Message += "<tr><td align='right'>SwapDataStore : </td><td align='left'>"         + $SwapDataStore           + "</td></tr>"
$Message += "</font>"
$Message += "</table>"

$Message += "<br>"

### OS Customization Spec
$Message += "<font size='2';color='NAVY'><b>OS CUSTOMIZATION SPEC:</b></font><br>"
$DataSetName = "VMOSCustomizationSpec"
$Query = "OPEN SYMMETRIC KEY SQLSymmetricKey DECRYPTION BY CERTIFICATE SelfSignedCertificate; SELECT BuildVersion,ServerRole,OSCustomizationSpecName,OSCustomizationSpecDescription,OSType,Type,NamingScheme,FullName,OrgName,ChangeSid,DeleteAccounts,TimeZone,ProductKey,LicenseMode,Workgroup,EncryptedPassword, CONVERT(varchar, DecryptByKey(EncryptedPassword)) AS 'DecryptedPassword' FROM definitionVMOSCustomizationSpec WHERE ServerRole = '$ServerRole' AND BuildVersion = '$BuildVersion'"
#$Query = "OPEN SYMMETRIC KEY SQLSymmetricKey  DECRYPTION BY CERTIFICATE SelfSignedCertificate; SELECT *, CONVERT(varchar, DecryptByKey(EncryptedPassword)) AS 'DecryptedPassword' FROM definitionVMOSCustomizationSpec WHERE ServerRole = '$ServerRole' AND BuildVersion = '$BuildVersion'"
$Connection = New-Object System.Data.SQLClient.SQLConnection
$Connection.ConnectionString = $DevOps_DBConnectionString
$Connection.Open()
$Command = New-Object System.Data.SQLClient.SQLCommand
$Command.Connection = $Connection
$Command.CommandText = $Query
$Reader = $Command.ExecuteReader()
$DataTable = New-Object System.Data.DataTable
$DataTable.Load($Reader)
$dt = $DataTable
$dt | ft

$VMOSCustomizationSpec += "<table>"
$VMOSCustomizationSpec +="<tr>"


for($i = 0;$i -lt $dt.Columns.Count;$i++)
{
    $VMOSCustomizationSpec += "<b><u><td>"+$dt.Columns[$i].ColumnName+"</td></u></b>"
}
$VMOSCustomizationSpec +="</tr>"

for($i=0;$i -lt $dt.Rows.Count; $i++)
{
    $VMOSCustomizationSpec +="<tr>"
    for($j=0; $j -lt $dt.Columns.Count; $j++)
    {
        $VMOSCustomizationSpec += "<td>"+$dt.Rows[$i][$j].ToString()+"</td>"
    }
    $VMOSCustomizationSpec +="</tr>"
}

$VMOSCustomizationSpec += "</table></body></html>"
```

```
2863        $Message += $VMOSCustomizationSpec
2864
2865        $Message += "<br>"
2866
2867        ### OS CUSTOMIZATION NIC MAPPING:
2868        $Message += "<font size='2';color='NAVY'><b>OS CUSTOMIZATION NIC
            MAPPING:</b></font><br>"
2869        $Message += "<table>"
2870        $Message += "<tr><td align='right'>IPv4Address : </td><td
            align='left'>"              + $IPv4Address              + "</td></tr>"
2871        $Message += "<tr><td align='right'>SubnetMask : </td><td
            align='left'>"              + $SubnetMask               + "</td></tr>"
2872        $Message += "<tr><td align='right'>DefaultGateway : </td><td
            align='left'>"          + $DefaultGateway           + "</td></tr>"
2873        $Message += "<tr><td align='right'>PrimaryDNS : </td><td
            align='left'>"              + $PrimaryDNS               + "</td></tr>"
2874        $Message += "<tr><td align='right'>SecondaryDNS : </td><td
            align='left'>"            + $SecondaryDNS             + "</td></tr>"
2875        $Message += "</table>"
2876
2877        $Message += "<br>"
2878
2879        ### OS Config Parameters
2880        $Message += "<font size='2';color='NAVY'><b>OS CONFIGURATION
            PARAMETERS:</b></font><br>"
2881        $DataSetName = "OSConfiguration"
2882        $Query = "SELECT * FROM definitionOSParameters WHERE ServerRole = '$ServerRole' AND
            BuildVersion = '$BuildVersion'"
2883
2884        $Connection = New-Object System.Data.SQLClient.SQLConnection
2885        $Connection.ConnectionString = $DevOps_DBConnectionString
2886        $Connection.Open()
2887        $Command = New-Object System.Data.SQLClient.SQLCommand
2888        $Command.Connection = $Connection
2889        $Command.CommandText = $Query
2890        $Reader = $Command.ExecuteReader()
2891        $DataTable = New-Object System.Data.DataTable
2892        $DataTable.Load($Reader)
2893        $dt = $DataTable
2894        $dt | ft
2895
2896        $Message += "<table>"
2897        $Message +="<tr>"
2898
2899
2900        for($i = 0;$i -lt $dt.Columns.Count;$i++)
2901        {
2902            $Message += "<b><u><td>"+$dt.Columns[$i].ColumnName+"</td></u></b>"
2903        }
2904        $Message +="</tr>"
2905
2906        for($i=0;$i -lt $dt.Rows.Count; $i++)
2907        {
2908            $Message +="<tr>"
2909            for($j=0; $j -lt $dt.Columns.Count; $j++)
2910            {
2911                $Message += "<td>"+$dt.Rows[$i][$j].ToString()+"</td>"
2912            }
2913            $Message +="</tr>"
2914        }
2915
2916        $Message += "</table></body></html>"
2917        $Message += "<br>"
2918
2919
2920        ### Write Report to SQL
2921        ###-----------------------
2922        Write-Host ("=" * 50)`n"Writing Report to SQL"`n("=" * 50)`n -ForegroundColor
            DarkCyan
```

```
2923
2924        $SQLParams = {
2925            [Parameter(Mandatory = $True)][string]$InstanceLocation,
2926            [Parameter(Mandatory = $True)][string]$GPID,
2927            [Parameter(Mandatory = $True)][string]$VMName,
2928            [Parameter(Mandatory = $True)][string]$ConfigurationMode,
2929            [Parameter(Mandatory = $True)][string]$ECIVMTemplate,
2930            [Parameter(Mandatory = $True)][string]$OSCustomizationSpecName,
2931            [Parameter(Mandatory = $True)][string]$ResourcePool,
2932            [Parameter(Mandatory = $True)][string]$PortGroup,
2933            [Parameter(Mandatory = $True)][string]$OSDataStore,
2934            [Parameter(Mandatory = $True)][string]$SwapDataStore,
2935            [Parameter(Mandatory = $True)][string]$vCPU,
2936            [Parameter(Mandatory = $True)][string]$vMemory,
2937            [Parameter(Mandatory = $True)][string]$IPv4Address,
2938            [Parameter(Mandatory = $True)][string]$SubnetMask,
2939            [Parameter(Mandatory = $True)][string]$DefaultGateway,
2940            [Parameter(Mandatory = $True)][string]$PrimaryDNS,
2941            [Parameter(Mandatory = $True)][string]$SecondaryDNS
2942            }
2943
2944        $Query = "INSERT INTO
            [Servers-ReadOnly](GPID,VMName,ServerRole,BuildVersion,ECIVMTemplate,OSCustomizationS
            pecName,vCPU,vMemory,ResourcePool,PortGroup,OSDatastore,SwapDataStore,IPv4Address,Sub
            netMask,DefaultGateway,PrimaryDNS,SecondaryDNS)
            VALUES('$GPID','$VMName','$ServerRole','$BuildVersion','$ECIVMTemplate','$OSCustomiza
            tionSpecName','$vCPU','$vMemory','$ResourcePool','$PortGroup','$OSDatastore','$SwapDa
            taStore','$IPv4Address','$SubnetMask','$DefaultGateway','$PrimaryDNS','$SecondaryDNS'
            )"
2945
2946        $Connection = New-Object System.Data.SQLClient.SQLConnection
2947        $Connection.ConnectionString = $DevOps_DBConnectionString
2948        $Connection.Open()
2949        $cmd = New-Object System.Data.SqlClient.SqlCommand
2950        $cmd.Connection = $connection
2951        $cmd.CommandText = $Query
2952        $cmd.ExecuteNonQuery() | Out-Null
2953        $connection.Close()
2954
2955
2956
2957        ### Email HTML Report
2958        ###------------------------
2959        $From    = "cbrennan@eci.com"
2960        $SMTP = $SMTPServer
2961        $Subject = "Server Provisioning- Report Mode"
2962
2963        #$To      = "cbrennan@eci.com,sdesimone@eci.com,wercolano@eci.com,rgee@eci.com"
2964        $To     = "cbrennan@eci.com"
2965
2966        ### Email Message
2967        ###------------------------------------------------------------------------------
2968        Write-Host `r`n`r`n`r`n("=" * 50)`n "Sending New Server Report" `r`n("=" *
            50)`r`n`r`n -ForegroundColor Magenta
2969        Write-Host "TO: " $To
2970        #Write-Host "MESSAGE:" $Message -ForegroundColor Magenta
2971
2972
2973        Send-MailMessage -To ($To -split ",") -From $From -Body $Message -Subject $Subject
            -BodyAsHtml -SmtpServer $SMTP
2974
2975        #if(-NOT(Test-Path -Path $ReportFile)) {(New-Item -ItemType file -Path $ReportFile
            -Force | Out-Null)}
2976        #$NewVMParameters | Out-File $ReportFile -Force
2977        #start-process $ReportFile
2978
2979        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
2980
2981        if(!$ReportMode_Timeout)
```

```
2982        {
2983            $ReportMode_Timeout = 1
2984        }
2985        Start-ECI.EMI.Automation.Sleep -Message "Exiting after Sending Report." -t
            $ReportMode_Timeout
2986        [Environment]::Exit(0)
2987     }
2988
2989    function PreCheck-ECI.EMI.Automation
2990    {
2991        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
2992
2993        Write-Host "Performing System Pre-Check Verification:" -ForegroundColor Cyan
2994
2995        $PreCheckState = @{}
2996
2997        ### Get ECI Modules & Version
2998        foreach($Module in (Get-Module ECI*))
2999        {
3000            $PreCheckState += $Module.Name
3001            $PreCheckState += $Module.Version
3002        }
3003
3004        ### Get PowerCLI Modules & Version
3005        foreach($Module in (Get-Module $VMModulesPath vmware*))
3006        {
3007            $PreCheckState += $Module.Name
3008            $PreCheckState += $Module.Version
3009        }
3010
3011
3012
3013
3014
3015        ### Connect vCenter
3016        Connect-VIServer -Server $vCenter -User $vCenter_Account -Password $vCenter_Password
3017
3018
3019        Get-Folder -Server $ECIvCenter -Name $vCenterFolder
3020
3021
3022     #$PreCheckState = [ordered]@{
3023     #}
3024
3025     $PreCheckState = New-Object -TypeName PSObject -Property $PreCheckState
3026
3027    <#
3028        Portal-Int
3029        vCenterFolder
3030        Portal_DBConnectionString
3031        DevOps_DBConnectionString
3032
3033        ISODatastore
3034        Windows ISO
3035        VMTools ISO
3036        SMTPServer
3037    #>
3038
3039    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
3040
3041    }
3042
3043    function Get-VMLogs
3044    {
3045
3046        ### Get vCenter VM Logs On Failurer/Abort
3047        ###-------------------------------------
3048
```

```
3049
3050
3051        # export virtual machine logs from a Datastore to local computer.
3052
3053        $vm = get-vm "VM NAME HERE"
3054        $target = New-Item -ItemType Directory -Force -Path c:\VM_Logs\$vm
3055        $datastore = get-vm $vm | Get-Datastore
3056        New-PSDrive -Location $datastore -Name ds -PSProvider VimDatastore -Root "\"
3057        Set-Location ds:\
3058        cd $vm
3059        Copy-DatastoreItem -Item *.log -Destination $target
3060        set-Location C:
3061        Remove-PSDrive -Name ds -Confirm:$false
3062
3063
3064
3065    }
3066
3067    function Get-MachineSID
3068    {
3069      param(
3070      [string]$HostName,
3071      [switch]$DomainSID
3072      )
3073
3074      ### SOURCE: https://gist.github.com/IISResetMe/36ef331484a770e23a81
3075      $VMTemplateSID = "S-1-5-21-1341700647-1908522465-1290903906-501"
3076      Write-Host "VMTemplateSID: " $VMTemplateSID -ForegroundColor DarkCyan
3077
3078      # Retrieve the Win32_ComputerSystem class and determine if machine is a Domain
         Controller
3079      $WmiComputerSystem = Get-WmiObject -Class Win32_ComputerSystem
3080      $IsDomainController = $WmiComputerSystem.DomainRole -ge 4
3081
3082      if($IsDomainController -or $DomainSID)
3083      {
3084        # We grab the Domain SID from the DomainDNS object (root object in the default NC)
3085        $Domain    = $WmiComputerSystem.Domain
3086        $SIDBytes = ([ADSI]"LDAP://$Domain").objectSid |%{$_}
3087        $SID = New-Object System.Security.Principal.SecurityIdentifier -ArgumentList
           ([Byte[]]$SIDBytes),0
3088        Return $SID.Value
3089      }
3090      else
3091      {
3092        # Going for the local SID by finding a local account and removing its Relative ID
           (RID)
3093        $LocalAccountSID = Get-WmiObject -ComputerName $HostName -Query "SELECT SID FROM
           Win32_UserAccount WHERE LocalAccount = 'True'" | Select-Object -First 1
           -ExpandProperty SID
3094        $MachineSID      = ($p = $LocalAccountSID -split "-")[0..($p.Length-2)]-join"-"
3095        $SID = New-Object System.Security.Principal.SecurityIdentifier -ArgumentList
           $MachineSID
3096        Return $SID.Value
3097      }
3098    }
3099
3100    function Send-ECI.Alert
3101    {
3102        Param(
3103            [Parameter(Mandatory = $False)][string]$Alert,
3104            [Parameter(Mandatory = $False)][string]$ErrorMsg,
3105            [Parameter(Mandatory = $False)][switch]$Exit
3106        )
3107
3108        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 75)`r`n
           "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 75) -ForegroundColor Gray
3109
3110        ############################
```

```
3111        ### HTML Header
3112        ############################
3113        $Header  = "<style>"
3114        $Header += "BODY{font-family: Verdana, Arial, Helvetica,
            sans-serif;font-size:9;font-color: #000000;text-align:left;}"
3115        $Header += "TABLE{border-width: 1px;border-style: solid;border-color:
            black;border-collapse: collapse;}"
3116        $Header += "TH{border-width: 1px;padding: 0px;border-style: solid;border-color:
            black;background-color: #D2B48C}"
3117        $Header += "TD{border-width: 1px;padding: 0px;border-style: solid;border-color:
            black;background-color: #FFEFD5}"
3118        $Header += "</style>"
3119
3120        ############################
3121        ### Format Report Data
3122        ############################
3123
3124        $Message = $Null
3125
3126        $Header = "
3127            <style>
3128            BODY{font-family: Verdana, Arial, Helvetica, sans-serif;font-size:9;font-color:
            #000000;text-align:left;}
3129            TABLE {border-width: 0px; border-style: hidden; border-color: white;
            border-collapse: collapse;}
3130            TH {border-width: 0px; padding: 3px; border-style: hidden; border-color: white;
            background-color: #6495ED;}
3131            TD {border-width: 0px; padding: 3px; border-style: hidden; border-color: white;}
3132
3133            </style>
3134        "
3135        $Message += $Header
3136        $Message += "<font size='3';color='gray'><i>ECI EMI Server Automation</i></font><br>"
3137        $Message += "<font size='5';color='NAVY'><b>ECI Automation Error
            Alert</b></font><br>"
3138        $Message += "<font size='2'>Alert Date:  " + (Get-Date) + "</font>"
3139        $Message += "<br><br><br><br>"
3140        $Message += "<font size='3';color='black'>WARNING: This Server <b> COULD NOT </b>
            be provisioned.</font>"
3141        $Message += "<br><br>"
3142
3143
3144        if($Alert)
3145        {
3146            $Message += "<font size='3';color='red'><br><b>ALERT: </b></font><br>"
3147            $Message += "<font size='3';color='red'><b>" + $Alert +  "</b></font>"
3148        }
3149        if($ErrorMsg)
3150        {
3151            $Message += "<font size='3';color='red'><br><b>ERROR MESSAGE: </b></font><br>"
3152            $Message += "<font size='3';color='red'>" + $ErrorMsg +  " </font>"
3153        }
3154        if($ECIError)
3155        {
3156            $Message += "<font size='3';color='red'><br><b>ECI ERROR: </b></font><br>"
3157            $Message += "<font size='3';color='red'>" + $ECIError +  " </font>"
3158        }
3159
3160
3161        $Message += "<br><br><br>"
3162
3163        ### Server Specs
3164        $Message += "<font size='3';color='NAVY'><b>SERVER SPECIFICATIONS:</b></font><br>"
3165        $Message += "<table>"
3166        $Message += "<font size='2';color='NAVY'>"
3167        $Message += "<tr><td align='right'>RequestID : </td><td
            align='left'>"               + $RequestID               + "</td></tr>"
3168        $Message += "<tr><td align='right'>HostName : </td><td
            align='left'>"               + $HostName                + "</td></tr>"
```

```
3169        $Message += "<tr><td align='right'>VMName : </td><td
            align='left'>"                  + $VMName                   + "</td></tr>"
3170        $Message += "<tr><td align='right'>InstanceLocation : </td><td
            align='left'>"        + $InstanceLocation         + "</td></tr>"
3171        $Message += "<tr><td align='right'>GPID : </td><td
            align='left'>"                    + $GPID                     + "</td></tr>"
3172        $Message += "<tr><td align='right'>ServerRole : </td><td
            align='left'>"              + $ServerRole               + "</td></tr>"
3173        $Message += "</table>"
3174        $Message += "<br>"
3175
3176    <#
3177        ### Server Specs
3178        $Message += "<font size='3';color='NAVY'><b>SERVER SPECIFICATIONS:</b></font><br>"
3179        $Message += "<table>"
3180        $Message += "<font size='2';color='NAVY'>"
3181        $Message += "<tr><td align='right'>RequestID : </td><td
            align='left'>"               + $RequestID                + "</td></tr>"
3182        $Message += "<tr><td align='right'>VMName : </td><td
            align='left'>"                  + $VMName                   + "</td></tr>"
3183        $Message += "<tr><td align='right'>InstanceLocation : </td><td
            align='left'>"        + $InstanceLocation         + "</td></tr>"
3184        $Message += "<tr><td align='right'>GPID : </td><td
            align='left'>"                    + $GPID                     + "</td></tr>"
3185        $Message += "<tr><td align='right'>ServerRole : </td><td
            align='left'>"              + $ServerRole               + "</td></tr>"
3186        $Message += "<tr><td align='right'>Pod : </td><td
            align='left'>"                    + $Pod                      + "</td></tr>"
3187        $Message += "</table>"
3188        $Message += "<br>"
3189
3190        ### VCenter Resources
3191        $Message += "<font size='3';color='NAVY'><b>VCENTER RESOURCES:</b></font><br>"
3192        $Message += "<table border='1'>"
3193        $Message += "<font size='2';color='NAVY'>"
3194        $Message += "<tr><td align='right'>ResourcePool : </td><td
            align='left'>"           + $ResourcePool             + "</td></tr>"
3195        $Message += "<tr><td align='right'>PortGroup : </td><td
            align='left'>"               + $PortGroup                + "</td></tr>"
3196        $Message += "<tr><td align='right'>osDatastoreCluster : </td><td
            align='left'>"      + $osDatastoreCluster       + "</td></tr>"
3197        $Message += "<tr><td align='right'>swapDatastoreCluster : </td><td
            align='left'>"    + $swapDatastoreCluster     + "</td></tr>"
3198        $Message += "<tr><td align='right'>dataDatastoreCluster : </td><td
            align='left'>"    + $dataDatastoreCluster     + "</td></tr>"
3199        $Message += "<tr><td align='right'>logDatastoreCluster : </td><td
            align='left'>"     + $logDatastoreCluster      + "</td></tr>"
3200        $Message += "<tr><td align='right'>swapDatastoreCluster : </td><td
            align='left'>"    + $swapDatastoreCluster     + "</td></tr>"
3201        $Message += "<tr><td align='right'>sysDatastoreCluster : </td><td
            align='left'>"     + $sysDatastoreCluster      + "</td></tr>"
3202        $Message += "</font>"
3203        $Message += "</table>"
3204        $Message += "<br>"
3205    #>
3206
3207        ### Display Server Request Record
3208        ###-------------------------------------------------------------------------
3209        $Message += "<font size='3';color='NAVY'><b>SERVER REQUEST
            INFORMATION:</b>"           + "</font><br>"
3210        $Header = "
3211            <style>
3212            BODY{font-family: Verdana, Arial, Helvetica, sans-serif;font-size:9;font-color:
            #000000;text-align:left;}
3213            TABLE {border-width: 1px; border-style: solid; border-color: black;
            border-collapse: collapse;}
3214            </style>
3215        "
3216        $Message += $Header
```

```
3217
3218        $DataSetName = "ServerRequest"
3219        $ConnectionString = $Portal_DBConnectionString
3220        $Query = "SELECT * FROM ServerRequest WHERE RequestID = '$RequestID'"
3221        $Connection = New-Object System.Data.SQLClient.SQLConnection
3222        $Connection.ConnectionString = $ConnectionString
3223        $Connection.Open()
3224        $Command = New-Object System.Data.SQLClient.SQLCommand
3225        $Command.Connection = $Connection
3226        $Command.CommandText = $Query
3227        $Reader = $Command.ExecuteReader()
3228        $DataTable = New-Object System.Data.DataTable
3229        $DataTable.Load($Reader)
3230        $dt = $DataTable
3231        $dt | ft
3232
3233        $ServerRequest += "<table>"
3234        $ServerRequest +="<tr>"
3235        for($i = 0;$i -lt $dt.Columns.Count;$i++)
3236        {
3237            $ServerRequest += "<b><u><td>"+$dt.Columns[$i].ColumnName+"</td></u></b>"
3238        }
3239        $ServerRequest +="</tr>"
3240
3241        for($i=0;$i -lt $dt.Rows.Count; $i++)
3242        {
3243            $ServerRequest +="<tr>"
3244            for($j=0; $j -lt $dt.Columns.Count; $j++)
3245            {
3246                $ServerRequest += "<td>"+$dt.Rows[$i][$j].ToString()+"</td>"
3247            }
3248            $ServerRequest +="</tr>"
3249        }
3250
3251        $ServerRequest += "</table></body></html>"
3252        $Message += $ServerRequest
3253        $Message += "<br><br>"
3254
3255        ### Transcript Log
3256        ###---------------------
3257        $TranscriptURL = "cloud-portal01.eci.cloud/vmautomationlogs/" + $HostName + "/" +
            (Split-Path $TranscriptFile -Leaf)
3258        $Message += "Transcript Log: " + "<a href=http://" + $TranscriptURL + ">" +
            $TranscriptURL + "</a>"
3259        $Message += "<br><br>"
3260        $Message += "Server Build Date: " + (Get-Date)
3261        $Message += "<br>"
3262
3263
3264        ### ECI-Error Array
3265        ###---------------------
3266        if($global:ECIError)
3267        {
3268            $Message += "<br><b>ECI-ErrorVar:  " + $global:ECIError.count  + " </b><br>"
3269            for($i = 0; $i -le $global:ECIError.count -1; $i++)
3270            {
3271                $Message += "ECIError: $i<br>"
3272                $Message += ($global:ECIError[$i])
3273                $Message += "<br>"
3274            }
3275        }
3276
3277        ### ECI-Error Log
3278        ###---------------------
3279        # Get-Content C:\scripts\_vmautomationlog\hostname\ecierrorlogs.log
3280
3281
3282        ### PS-Error Array
3283        ###---------------------
```

```
3284        if($global:Error)
3285        {
3286            $Message += "<br><b>PS-ErrorVar:  " + $global:error.count + "</b><br>"
3287            for($i = 0; $i -le $global:error.count -1; $i++)
3288            {
3289                $Message += "ERROR: $i<br>"
3290                $Message += ($global:Error[$i])
3291                $Message += "<br>"
3292            }
3293        }
3294        ### Close Message
3295        ###---------------------
3296        $Message += "</body></html>"
3297
3298    <#
3299
3300        ### Write Report to SQL
3301        ###------------------------
3302        Write-Host ("=" * 50)`n"Writing Report to SQL"`n("=" * 50)`n -ForegroundColor
            DarkCyan
3303
3304        $Query = "INSERT INTO
            Errors(GPID,VMName,ServerRole,ResourcePool,PortGroup,osDatastoreCluster,swapDatastore
            Cluster)
            VALUES('$GPID','$VMName','$ServerRole','$ResourcePool','$PortGroup','$osDatastoreClus
            ter','$swapDatastoreCluster')"
3305
3306        $Connection = New-Object System.Data.SQLClient.SQLConnection
3307        $Connection.ConnectionString = $DevOps_DBConnectionString
3308        $Connection.Open()
3309        $cmd = New-Object System.Data.SqlClient.SqlCommand
3310        $cmd.Connection = $connection
3311        $cmd.CommandText = $Query
3312        $cmd.ExecuteNonQuery() | Out-Null
3313        $connection.Close()
3314    #>
3315
3316
3317        ### Email HTML Report
3318        ###------------------------
3319        #$From    = "cbrennan@eci.com"
3320        $From    = $SMTPFrom
3321        $SMTP    = $SMTPServer
3322        $Subject = "Server Provisioning Alert"
3323        $To      = $SMTPTo
3324        #$To      = "cbrennan@eci.com,sdesimone@eci.com,wercolano@eci.com,rgee@eci.com"
3325        #$To      = "cbrennan@eci.com,sdesimone@eci.com"
3326        $To      = "cbrennan@eci.com"
3327
3328        ### Email Message
3329        ###-----------------------------------------------------------------------------
3330        Write-Host `r`n`r`n`r`n("=" * 50)`n "Sending Alert - $ErrorMsg" `r`n("=" *
            50)`r`n`r`n -ForegroundColor Yellow
3331        Write-Host "TO: " $To
3332
3333
3334
3335        Send-MailMessage -To ($To -split ",") -From $From -Body $Message -Subject $Subject
            -BodyAsHtml -SmtpServer $SMTP
3336
3337
3338        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
3339
3340        if($Exit)
3341        {
3342            [Environment]::Exit(1)
3343        }
3344        else
3345        {
```

```
3346            Throw $ErrorMsg  ### DO NOT USE THROW if you want to exit PSSession
3347        }
3348
3349    }
```