```powershell
###################################
### Configure Guest OS Module
### ECI.EMI.Automation.OS.Prod.psm1
###################################


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

    foreach($Module in (Get-Module -ListAvailable ECI.*)){Import-Module -Name
    $Module.Path -DisableNameChecking}


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

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

    ##########################################
    ### DESIRED STATE PARAMETERS
    ##########################################
    $PropertyName      = "NetworkInterfaceName"
    $DesiredState      = $NetworkInterfaceName
    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False   ### $True - $False

    ##########################################
    ### GET CURRENT CONFIGURATION STATE:
    ##########################################
    [scriptblock]$script:GetCurrentState =
    {
        $global:CurrentState = (Get-NetAdapter -Physical | Where-Object Status -eq
        'Up').Name
    }

    ##########################################
    ### SET DESIRED-STATE:
    ##########################################
    [scriptblock]$script:SetDesiredState =
    {
        Rename-NetAdapter (Get-NetAdapter -Name $CurrentState).Name -NewName
        $DesiredState
    }

    ##########################################
    ### CALL CONFIGURE DESIRED STATE:
    ##########################################
    ###----------------------------

    $Params = @{
        ServerID            = $ServerID
        HostName            = $HostName
        FunctionName        = $FunctionName
        PropertyName        = $PropertyName
        DesiredState        = $DesiredState
        GetCurrentState     = $GetCurrentState
        SetDesiredState     = $SetDesiredState
        ConfigurationMode   = $ConfigurationMode
        AbortTrigger        = $AbortTrigger
    }
    Configure-DesiredState @Params

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

```


function Configure-ECI.EMI.Configure.OS.WSMan
{


}

function Configure-ECI.EMI.Configure.OS.SMBv1
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray

    ##########################################
    ### DESIRED STATE PARAMETERS
    ##########################################
    $PropertyName      = "SMBv1"
    $DesiredState      = $SMBv1
    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False   ### $True - $False

    ##########################################
    ### GET CURRENT CONFIGURATION STATE:
    ##########################################
    [scriptblock]$script:GetCurrentState =
    {
        $KeyPath = "HKLM:\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters"
        $KeyName = "SMB1"

        try
        {
            $KeyValue = Get-ItemProperty -Path $KeyPath -Name $KeyName
        }
        catch
        {
            $global:CurrentState = $null
            Write-ECI.ErrorStack
        }

        if($KeyValue)
        {
            $global:CurrentState = $KeyValue
        }
        else
        {
            $global:CurrentState = $null
        }
    }

    ##########################################
    ### SET DESIRED-STATE:
    ##########################################
    [scriptblock]$script:SetDesiredState =
    {
        $KeyValue = $DesiredState
        $KeyPath  = "SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters"
        $KeyName  = "SMB1"
        Set-ItemProperty -Path $KeyPath -Name $Keyname -Value $KeyValue
    }

    ##########################################
    ### CALL CONFIGURE DESIRED STATE:
    ##########################################
    ###----------------------------
    $Params = @{
        ServerID            = $ServerID
        HostName            = $HostName
        FunctionName        = $FunctionName
        PropertyName        = $PropertyName
        DesiredState        = $DesiredState
        GetCurrentState     = $GetCurrentState
        SetDesiredState     = $SetDesiredState
        ConfigurationMode   = $ConfigurationMode
        AbortTrigger        = $AbortTrigger
    }
    Configure-DesiredState @Params
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

function Configure-ECI.EMI.Configure.OS.IPv6
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray

    ##########################################
    ### DESIRED STATE PARAMETERS
    ##########################################
    $PropertyName      = "IPv6Preference"
    $DesiredState      = $IPv6Preference
    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False  ### $True - $False

    ##########################################
    ### GET CURRENT CONFIGURATION STATE:
    ##########################################
    [scriptblock]$script:GetCurrentState =
    {
        ### Get Current Interface
        $CurrentInterface = (Get-NetAdapter -Physical | Where-Object {$_.Status -eq
        'Up'}).Name
        $IPv6State   = Get-NetAdapterBinding -InterfaceAlias $CurrentInterface
        -DisplayName "Internet Protocol Version 6 (TCP/IPv6)"
        $global:CurrentState = $IPv6State.Enabled ### Return True/False
    }

    ##########################################
    ### SET DESIRED-STATE:
    ##########################################

    [scriptblock]$script:SetDesiredState =
    {
        if ($DesiredState -eq $True)
        {
            ### Enable IPv6
            Enable-NetAdapterBinding -InterfaceAlias $CurrentInterfaceName -ComponentID
            MS_TCPIP6
        }
        elseif ($DesiredState -eq $False)
        {
            ### Disable IPv6
            Disable-NetAdapterBinding -InterfaceAlias $CurrentInterfaceName
            -ComponentID MS_TCPIP6
        }
    }

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

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

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

    ### Modify Parameter Values
    ### --------------------------------------
    ### Drive Letter Must End with Colon ":"
    $CDROMLetter = $CDROMLetter.Trim()
    $LastChar = $CDROMLetter.substring($CDROMLetter.length-1)
    if ($LastChar -ne ":"){$CDROMLetter = $CDROMLetter + ":"}

    ##########################################
    ### DESIRED STATE PARAMETERS
    ##########################################
    $PropertyName      = "CDROMLetter"
    $DesiredState      = $CDROMLetter
    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False   ### $True - $False

    ##########################################
    ### GET CURRENT CONFIGURATION STATE:
    ##########################################
    [scriptblock]$script:GetCurrentState =
    {
        $script:ComputerName = (Get-WmiObject Win32_ComputerSystem).Name
        $global:CurrentState = (Get-WMIObject -Class Win32_CDROMDrive -ComputerName
        $ComputerName).Drive
    }

    ##########################################
    ### SET DESIRED-STATE:
    ##########################################
    [scriptblock]$script:SetDesiredState =
    {
        $CDVolume = Get-WmiObject -Class Win32_Volume -ComputerName $ComputerName
        -Filter "DriveLetter='$CurrentState'"
        Set-WmiInstance -InputObject $CDVolume -Arguments @{DriveLetter =
        $DesiredState} | Out-Null
    }

    ##########################################
    ### CALL CONFIGURE DESIRED STATE:
    ##########################################
    $Params = @{
        ServerID             = $ServerID
        HostName             = $HostName
        FunctionName         = $FunctionName
        PropertyName         = $PropertyName
        DesiredState         = $DesiredState
        GetCurrentState      = $GetCurrentState
        SetDesiredState      = $SetDesiredState
        ConfigurationMode    = $ConfigurationMode
        AbortTrigger         = $AbortTrigger
    }
    Configure-DesiredState @Params
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

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

    ##########################################
    ### DESIRED STATE PARAMETERS
    ##########################################
    $PropertyName      = "ECIFolders"

    ### Create ECI Folders
    ###-------------------------------
    $ECIFolders = @()
    $ECIFolders += "C:\Scripts"
    $ECIFolders += "D:\Kits"

    $DesiredState      = $ECIFolders
    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False  ### $True - $False

    foreach($State in $DesiredState)
    {
        $DesiredState = $State

        ##################################################
        ### GET CURRENT CONFIGURATION STATE:
        ##################################################
        [scriptblock]$script:GetCurrentState =
        {
            if(Test-Path -Path $DesiredState){$global:CurrentState = $DesiredState}
            else{$global:CurrentState = $False}
        }

        ##################################################
        ### SET DESIRED-STATE:
        ##################################################
        [scriptblock]$script:SetDesiredState =
        {
            New-Item -ItemType Directory -Path $State -Force | Out-Null
        }

        ##########################################
        ### CALL CONFIGURE DESIRED STATE:
        ##########################################
        ###-------------------------
        $Params = @{
            ServerID             = $ServerID
            HostName             = $HostName
            FunctionName         = $FunctionName
            PropertyName         = $PropertyName
            DesiredState         = $DesiredState
            GetCurrentState      = $GetCurrentState
            SetDesiredState      = $SetDesiredState
            ConfigurationMode    = $ConfigurationMode
            AbortTrigger         = $AbortTrigger
        }
        Configure-DesiredState @Params
    }
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

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

    ### Modify Parameter Values
    ### --------------------------------------
    if($RemoteDesktopPreference -eq "False"){$RemoteDesktopPreferenceValue = "1"}
    elseif($RemoteDesktopPreference -eq "True"){$RemoteDesktopPreferenceValue = "0"}

    ##########################################
    ### DESIRED STATE PARAMETERS
    ##########################################
    $PropertyName      = "RemoteDesktopPreference"
    $DesiredState      = $RemoteDesktopPreferenceValue
    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False  ### $True - $False

    ##################################################
    ### GET CURRENT CONFIGURATION STATE:
    ##################################################
    [scriptblock]$script:GetCurrentState =
    {
        $global:CurrentState = (Get-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server' -name "fDenyTSConnections").fDenyTSConnections
    }

    ##################################################
    ### SET DESIRED-STATE:
    ##################################################
    [scriptblock]$script:SetDesiredState =
    {
        Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server' -name "fDenyTSConnections" -Value $RemoteDesktopPreferenceValue

        if($RemoteDesktopPreferenceValue -eq "0")
        {
            Enable-NetFirewallRule -DisplayGroup "Remote Desktop"
            Netsh advfirewall firewall set rule group=”remote desktop” new enable=yes
        }
        elseif($RemoteDesktopPreferenceValue -eq "1")
        {
            Disable-NetFirewallRule -DisplayGroup "Remote Desktop"
        }
    }

    ##########################################
    ### CALL CONFIGURE DESIRED STATE:
    ##########################################
    $Params = @{
        ServerID             = $ServerID
        HostName             = $HostName
        FunctionName         = $FunctionName
        PropertyName         = $PropertyName
        DesiredState         = $DesiredState
        GetCurrentState      = $GetCurrentState
        SetDesiredState      = $SetDesiredState
        ConfigurationMode    = $ConfigurationMode
        AbortTrigger         = $AbortTrigger
    }
    Configure-DesiredState @Params
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

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

    ##########################################
    ### DESIRED STATE PARAMETERS
    ##########################################
    $PropertyName      = "WindowsFirewallPreference"
    $DesiredState      = $WindowsFirewallPreference
    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False  ### $True - $False

    ##########################################
    ### GET CURRENT CONFIGURATION STATE:
    ##########################################
    [scriptblock]$script:GetCurrentState =
    {
        $global:CurrentState = (Get-NetFirewallProfile -Name Domain).Enabled
    }

    ##########################################
    ### SET DESIRED-STATE:
    ##########################################
    [scriptblock]$script:SetDesiredState =
    {
        Set-NetFirewallProfile -Profile Domain -Enabled $WindowsFirewallPreference
    }

    ##########################################
    ### CALL CONFIGURE DESIRED STATE:
    ##########################################
    $Params = @{
        ServerID             = $ServerID
        HostName             = $HostName
        FunctionName         = $FunctionName
        PropertyName         = $PropertyName
        DesiredState         = $DesiredState
        GetCurrentState      = $GetCurrentState
        SetDesiredState      = $SetDesiredState
        ConfigurationMode    = $ConfigurationMode
        AbortTrigger         = $AbortTrigger
    }
    Configure-DesiredState @Params
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

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

    ### Modify Parameter Values
    ### --------------------------------------
    if($InternetExplorerESCPreference -eq "False")   {$InternetExplorerESCValue = "0"}
    elseif($InternetExplorerESCPreference -eq "True"){$InternetExplorerESCValue = "1"}

    ##########################################
    ### DESIRED STATE PARAMETERS
    ##########################################
    $PropertyName      = "InternetExplorerESCPreference"
    $script:DesiredState      = $InternetExplorerESCPreference
    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False  ### $True - $False

    $Keys = @()
    $Keys += "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed
    Components\{A509B1A7-37EF-4b3f-8CFC-4F3A74704073}"
    $Keys += "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed
    Components\{A509B1A8-37EF-4b3f-8CFC-4F3A74704073}"

    foreach($Key in $Keys)
    {
        ##########################################
        ### GET CURRENT CONFIGURATION STATE:
        ##########################################
        [scriptblock]$script:GetCurrentState =
        {
            $global:CurrentState = (Get-ItemProperty -Path $Key -Name
            "IsInstalled").IsInstalled
        }

        ##########################################
        ### SET DESIRED-STATE:
        ##########################################
       [scriptblock]$script:SetDesiredState =
        {
            Set-ItemProperty -Path $Key  -Name "IsInstalled" -Value $DesiredState -Force
        }

        ##########################################
        ### CALL CONFIGURE DESIRED STATE:
        ##########################################
        $Params = @{
            ServerID            = $ServerID
            HostName            = $HostName
            FunctionName        = $FunctionName
            PropertyName        = $PropertyName
            DesiredState        = $DesiredState
            GetCurrentState     = $GetCurrentState
            SetDesiredState     = $SetDesiredState
            ConfigurationMode   = $ConfigurationMode
            AbortTrigger        = $AbortTrigger
        }
        Configure-DesiredState @Params
    }
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

function Initialize-ECI.EMI.Configure.OS.HardDisks
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray

    ### Initalzie Disk
    ### --------------------------
    Write-Host "Initialize Disk: "
    Get-Disk | Where-Object partitionstyle -eq 'raw' | Initialize-Disk -PartitionStyle
    MBR -PassThru | New-Partition -AssignDriveLetter -UseMaximumSize | Format-Volume
    -FileSystem NTFS -NewFileSystemLabel "SwapFile" -Confirm:$false
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

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

    ##########################################
    ### DESIRED STATE PARAMETERS
    ##########################################
    $PropertyName      = "WindowsFeatures"
    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False  ### $True - $False

    switch ( $ServerRole )
    {
        "2016Server"
        { $WindowsFeatures =
            @(
                "NET-Framework-Features",
                "NET-Framework-Core",
                "GPMC",
                "Telnet-Client"
            )
        }
        "2016FS"
        { $WindowsFeatures =
            @(
                "NET-Framework-Features",
                "NET-Framework-Core",
                "GPMC",
                "Telnet-Client",
                "RSAT"
            )
        }
        "2016DC"
        { $WindowsFeatures =
        @(
            "NET-Framework-Features",
            "NET-Framework-Core",
            "GPMC",
            "Telnet-Client",
            "RSAT"
            )
        }
        "2016DCFS"
        { $WindowsFeatures =
        @(
            "NET-Framework-Features",
            "NET-Framework-Core",
            "GPMC",
            "Telnet-Client",
            "RSAT"
            )
        }
         "2016VDA"
         { $WindowsFeatures =
         @(
            "NET-Framework-Features",
            "NET-Framework-Core",
            "GPMC",
            "Telnet-Client",
            "RSAT"
            "AS-Net-Framework",
            "RDS-RD-Server"
            )
        }
        "2016SQL"
        { $WindowsFeatures =
        @(
            "NET-Framework-Features",
            "NET-Framework-Core",
            "GPMC",
            "Telnet-Client",
            "RSAT"
            )
        }
        "2016SQLOMS"
        { $WindowsFeatures =
        @(
            "NET-Framework-Features",
            "NET-Framework-Core",
            "GPMC",
            "Telnet-Client",
            "RSAT"
            )
        }
    }
    foreach ($Feature in $WindowsFeatures)
    {
        $script:DesiredState = $Feature
        write-host "DesiredState: " $DesiredState

        ##########################################
        ### GET CURRENT CONFIGURATION STATE:
        ##########################################
        [scriptblock]$script:GetCurrentState =
        {
            $global:CurrentState = ((Get-WindowsFeature -Name $Feature) | Where-Object
            {$_.Installed -eq $True}).Name
        }

        ##########################################
        ### SET DESIRED-STATE:
        ##########################################
        [scriptblock]$script:SetDesiredState =
        {
            Write-Host "Installing Feature: " $Feature

            #$WindowsMediaSource = "R:\sources\sxs\microsoft-windows-netfx3-ondemand-package.cab"
            $WindowsMediaSource = "R:\sources\sxs"
            Install-WindowsFeature -Name $Feature -Source $WindowsMediaSource
        }

        ##########################################
        ### CALL CONFIGURE DESIRED STATE:
        ##########################################
        $Params = @{
            ServerID            = $ServerID
            HostName            = $HostName
            FunctionName        = $FunctionName
            PropertyName        = $PropertyName
            DesiredState        = $DesiredState
            GetCurrentState     = $GetCurrentState
            SetDesiredState     = $SetDesiredState
            ConfigurationMode   = $ConfigurationMode
            AbortTrigger        = $AbortTrigger
        }
        Configure-DesiredState @Params
    }
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}

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

    ##########################################
    ### DESIRED STATE PARAMETERS
    ##########################################
    $PropertyName      = "WindowsFirewallRules"
    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False  ### $True - $False

    ### Set Firewall Rules
    ###-------------------------
    $FireWallRules = @()
    $FireWallRules += "File and Printer Sharing (SMB-In)"
    $FireWallRules += "Windows Management Instrumentation (ASync-In)"
    $FireWallRules += "Windows Management Instrumentation (DCOM-In)"
    $FireWallRules += "Windows Management Instrumentation (WMI-In)"

    foreach($Rule in $FireWallRules)
    {
        ### Set Desired State Value
        ###-------------------------
        $script:DesiredState = $Rule

        ##########################################
        ### GET CURRENT CONFIGURATION STATE:
        ##########################################
        [scriptblock]$script:GetCurrentState =
        {
            $global:CurrentState = (Get-NetFirewallProfile -Name Domain | Get-NetFirewallRule | Where {$_.DisplayName -eq $Rule}).DisplayName
        }

        ##########################################
        ### SET DESIRED-STATE:
        ##########################################
```

```
            [scriptblock]$script:SetDesiredState =
            {
                Write-Host "Installing Feature: " $Feature
                Enable-NetFirewallRule -DisplayName $Rule
            }

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

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

    function Configure-ECI.EMI.Configure.OS.WindowsFirewallRules-orig-deleteme
    {
        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray

        $FireWallRules = @()
        $FireWallRules += "File and Printer Sharing (SMB-In)"
        $FireWallRules += "Windows Management Instrumentation (ASync-In)"
        $FireWallRules += "Windows Management Instrumentation (DCOM-In)"
        $FireWallRules += "Windows Management Instrumentation (WMI-In)"

        foreach($Rule in $FireWallRules)
        {
            #Write-Host "Getting FireWall Rule: " $Rule
            Get-NetFirewallRule |  Where {$_.DisplayName -eq $Rule}
            Enable-NetFirewallRule -DisplayName $Rule
        }

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

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

        ##########################################
        ### DESIRED STATE PARAMETERS
        ##########################################
        $PropertyName      = "PageFileLocation"
        $DesiredState      = $PageFileLocation
        $ConfigurationMode = "Configure" ### Report - Configure
        $AbortTrigger      = $False   ### $True - $False

        ### Modify Parameter Values
        ### --------------------------------------
        ### Drive Letter Must NOT End with Colon ":"
        $PageFileLocation = $PageFileLocation.Trim()
        $LastChar = $PageFileLocation.substring($PageFileLocation.length-1)
        if ($LastChar -eq ":"){$PageFileLocation = $PageFileLocation.Split(":")[0]}

        ##########################################
        ### GET CURRENT CONFIGURATION STATE:
```

```
        ##########################################
        [scriptblock]$script:GetCurrentState =
        {
            try
            {
                if(((Get-CimInstance -ClassName Win32_ComputerSystem).AutomaticManagedPagefile) -eq $True)
                {
                    $global:CurrentState = $Null
                    Write-Host "PageFile is set to AutomaticManagedPagefile"
                }
                else
                {
                    $global:CurrentState = ((Get-CimInstance -ClassName Win32_PageFileSetting).Name).Split(":")[0]
                }
            }
            catch
            {
                Write-ECI.ErrorStack
            }
        }

        ##########################################
        ### SET DESIRED-STATE:
        ##########################################
        [scriptblock]$script:SetDesiredState =
        {
            $script:DesiredState = $PageFileLocation

            # Disable Automatically Managed PageFile Setting
            ### ---------------------------------
            $ComputerSystem = Get-CimInstance -ClassName Win32_ComputerSystem
            if ($ComputerSystem.AutomaticManagedPagefile -eq "True")
            {
                Set-CimInstance -InputObject $ComputerSystem -Property @{AutomaticManagedPageFile = $False }
            }
            Write-Host "AutomaticManagedPagefile: " $(Get-CimInstance -ClassName Win32_ComputerSystem).AutomaticManagedPagefile

            ### Delete Existing PageFile
            ### ---------------------------------
            $PageFile = Get-CimInstance -ClassName Win32_PageFileSetting
            $PageFile | Remove-CimInstance

            ### Calculate Page File Size
            ### ---------------------------------
            $Memory = (Get-CimInstance -ClassName Win32_PhysicalMemory | Measure-Object -Property Capacity -Sum | % {[Math]::Round(($_.sum / 1MB),2)})
            $NewPageFileSize = [Math]::Round(($Memory * $PageFileMultiplier)) # Memory Size Plus 20% - Round Up

            ### PageFile Minumin Size = 4GB
            ###----------------------------------
            if ($NewPageFileSize -lt "4096") {$NewPageFileSize = "4096"}

            ### Min/Max Multiplier - Currently 1/1
            ###----------------------------------
            [int]$script:InitialSize = ($NewPageFileSize * 1)
            [int]$script:MaximumSize = ($NewPageFileSize * 1)

            Write-Host "PageFile Location    : " $PageFileLocation
            Write-Host "PageFile InitialSize : " $InitialSize
            Write-Host "PageFile MaximumSize : " $MaximumSize

            ### -------------------------------------------------------------------------------
            ### Create New Page File
            ### ------------------------------------------------------------------------------
            [scriptblock]$CreatePageFile =
            {
                if(-NOT(Get-CimInstance -ClassName Win32_PageFileSetting))
                {
                    try
                    {
                        $PageFileName = $PageFileLocation + ":\pagefile.sys"
                        New-CimInstance -ClassName Win32_PageFileSetting -Property  @{ Name= $PageFileName } | Out-Null
                        Get-CimInstance -ClassName Win32_PageFileSetting | Set-CimInstance -Property @{InitialSize = $InitialSize; MaximumSize = $MaximumSize;} | Out-Null
                    }
                    catch
                    {
                        Write-ECI.ErrorStack
                    }
                }
                else
                {
                    Write-Host "A Page File Already Exists!"
                    $Abort = $True
                }
            }

            ### Check Avilable Disk Space
            ### --------------------------------
            [int]$FreeSpace = [Math]::Round(((Get-PSDrive $PageFileLocation).Free/1MB),2)

            if($FreeSpace -gt $NewPageFileSize)
            {
                Write-Host "Free Space is Available. Drive: $PageFileLocation FreeSpace: $FreeSpace NewPageFileSize: $NewPageFileSize"
                Invoke-Command -ScriptBlock $CreatePageFile
            }
            elseif($FreeSpace -le $NewPageFileSize)
            {
                Write-Host "Not Enough Avialable Space. Drive: $Drive FreeSpace: $FreeSpace NewPageFileSize: $NewPageFileSize `r`nNot Configuring!"
                $Abort = $True
            }
        }

        ##########################################
        ### CALL CONFIGURE DESIRED STATE:
        ##########################################
        ###------------------------------
        $Params = @{
            ServerID             = $ServerID
            HostName             = $HostName
            FunctionName         = $FunctionName
            PropertyName         = $PropertyName
            DesiredState         = $DesiredState
            GetCurrentState      = $GetCurrentState
            SetDesiredState      = $SetDesiredState
            ConfigurationMode    = $ConfigurationMode
            AbortTrigger         = $AbortTrigger
        }
        Configure-DesiredState @Params
        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
    }

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

```

```
##########################################
### DESIRED STATE PARAMETERS
##########################################
$PropertyName      = "PageFileSize"
$ConfigurationMode = "Configure" ### Report - Configure
$AbortTrigger      = $True  ### $True - $False

### Modify Parameter Values
### ---------------------------------------
### Drive Letter Must NOT End with Colon ":"
$PageFileLocation = $PageFileLocation.Trim()
$LastChar = $PageFileLocation.substring($PageFileLocation.length-1)
if ($LastChar -eq ":"){$PageFileLocation = $PageFileLocation.Split(":")[0]}

$Memory = (Get-CimInstance -ClassName Win32_PhysicalMemory | Measure-Object -Property Capacity -Sum | % {[Math]::Round(($_.sum / 1MB),2)})
$DesiredState = [Math]::Round(($Memory * $PageFileMultiplier)) # Memory Size Plus 20% - Round Up


##########################################
### GET CURRENT CONFIGURATION STATE:
##########################################
[scriptblock]$script:GetCurrentState =
{
    try
    {
        if(((Get-CimInstance -ClassName Win32_ComputerSystem).AutomaticManagedPagefile) -eq $True)
        {
            $global:CurrentState = $Null
            Write-Host "PageFile is set to AutomaticManagedPagefile"
        }
        else
        {
            $global:CurrentState = (Get-CimInstance -ClassName Win32_PageFileUsage).AllocatedBaseSize
        }
    }
    catch
    {
        Write-ECI.ErrorStack
    }
}

##########################################
### SET DESIRED-STATE:
##########################################
[scriptblock]$script:SetDesiredState =
{
    $script:DesiredState = $PageFileLocation

    # Disable Automatically Managed PageFile Setting
    ### ---------------------------------
    $ComputerSystem = Get-CimInstance -ClassName Win32_ComputerSystem
    if ($ComputerSystem.AutomaticManagedPagefile -eq "True")
    {
        Set-CimInstance -InputObject $ComputerSystem -Property @{AutomaticManagedPageFile = $False }
    }
    Write-Host "AutomaticManagedPagefile: " $(Get-CimInstance -ClassName Win32_ComputerSystem).AutomaticManagedPagefile

    ### Delete Existing PageFile
    ### ---------------------------------
    $PageFile = Get-CimInstance -ClassName Win32_PageFileSetting
    $PageFile | Remove-CimInstance

    ### Calculate Page File Size
```

```
        ### ---------------------------------
        $Memory = (Get-CimInstance -ClassName Win32_PhysicalMemory | Measure-Object -Property Capacity -Sum | % {[Math]::Round(($_.sum / 1MB),2)})
        $NewPageFileSize = [Math]::Round(($Memory * $PageFileMultiplier)) # Memory Size Plus 20% - Round Up

        ### PageFile Minumin Size = 4GB
        ###----------------------------------
        if ($NewPageFileSize -lt "4096") {$NewPageFileSize = "4096"}

        ### Min/Max Multiplier - Currently 1/1
        ###----------------------------------
        [int]$script:InitialSize = ($NewPageFileSize * 1)
        [int]$script:MaximumSize = ($NewPageFileSize * 1)

        Write-Host "PageFile Location    : " $PageFileLocation
        Write-Host "PageFile InitialSize : " $InitialSize
        Write-Host "PageFile MaximumSize : " $MaximumSize

        ### --------------------------------------------------------------------------------
        ### Create New Page File
        ### --------------------------------------------------------------------------------
        [scriptblock]$CreatePageFile =
        {
            if(-NOT(Get-CimInstance -ClassName Win32_PageFileSetting))
            {
                try
                {
                    $PageFileName = $PageFileLocation + ":\pagefile.sys"
                    New-CimInstance -ClassName Win32_PageFileSetting -Property  @{ Name= $PageFileName } | Out-Null
                    Get-CimInstance -ClassName Win32_PageFileSetting | Set-CimInstance -Property @{InitialSize = $InitialSize; MaximumSize = $MaximumSize;} | Out-Null
                }
                catch
                {
                    Write-ECI.ErrorStack
                }
            }
            else
            {
                Write-Host "A Page File Already Exists!"
                $Abort = $True
            }
        }

        ### Check Avilable Disk Space
        ### ---------------------------------
        [int]$FreeSpace = [Math]::Round(((Get-PSDrive $PageFileLocation).Free/1MB),2)

        if($FreeSpace -gt $NewPageFileSize)
        {
            Write-Host "Free Space is Available. Drive: $PageFileLocation FreeSpace: $FreeSpace NewPageFileSize: $NewPageFileSize"
            Invoke-Command -ScriptBlock $CreatePageFile
        }
        elseif($FreeSpace -le $NewPageFileSize)
        {
            Write-Host "Not Enough Avialable Space. Drive: $Drive FreeSpace: $FreeSpace NewPageFileSize: $NewPageFileSize `r`nNot Configuring!"
            $Abort = $True
        }
    }

    ##########################################
    ### CALL CONFIGURE DESIRED STATE:
```

```
    ########################################
    ###----------------------------
    $Params = @{
        ServerID             = $ServerID
        HostName             = $HostName
        FunctionName         = $FunctionName
        PropertyName         = $PropertyName
        DesiredState         = $DesiredState
        GetCurrentState      = $GetCurrentState
        SetDesiredState      = $SetDesiredState
        ConfigurationMode    = $ConfigurationMode
        AbortTrigger         = $AbortTrigger
    }
    Configure-DesiredState @Params
    Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
}


function Configure-PageFile-old #<---- need to complete function!!!!!!!!!!!!!!!!
{
    $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray

    ### Modify Parameter Values
    ### --------------------------------------
    ### Drive Letter Must NOT End with Colon ":"
    $LastChar = $PageFileLocation.substring($PageFileLocation.length-1)
    if ($LastChar -eq ":"){$PageFileLocation = $PageFileLocation.Split(":")[0]}

    ########################################
    ### DESIRED STATE PARAMETERS
    ########################################
    $PropertyName      = "PageFileSize"

    $ConfigurationMode = "Configure" ### Report - Configure
    $AbortTrigger      = $False  ### $True - $False

    $DesiredState = @()
    $DesiredState += $PageFileSize
    $DesiredState += $PageFileLocation

    ########################################
    ### GET CURRENT CONFIGURATION STATE:
    ########################################
    [scriptblock]$script:GetCurrentState =
    {
        ### Calculate Desired Page File Size
        $Memory = (Get-WMIObject -class Win32_PhysicalMemory | Measure-Object -Property
        Capacity -Sum | % {[Math]::Round(($_.sum / 1GB),2)})
        $DesiredPageFileSize = [Math]::Round(($Memory * $PageFileMultiplier)) # Memory
        Size Plus 20% - Round Up
        if($Memory -lt "4") {$NewPageFileSize = "4"} ### Set a Minimun 4GB PageFile Size


        $DesiredState = ($DesiredPageFileSize * 1000)
        $PageFile = Get-WmiObject -Class Win32_PageFileUsage -Computer "LocalHost"
        $CurrentState = $PageFile.MaximumSize
        $Size = ($CurrentState -eq $DesiredState)

        ### PageFile Location
        $script:DesiredState = $PageFileLocation
        $PageFile = Get-CimInstance -ClassName Win32_PageFileSetting
        $CurrentState = ($PageFile.Name).Split(":")[0]
        $Location =  ($CurrentState -eq $DesiredState)

        ### Set $CurrentState
        $script:CurrentState = $True
        $script:DesiredState = $True
        if (($Size -eq $False) -OR ($Location -eq $False))
```

```powershell
            {
                $script:CurrentState = $False
            }
        }

        ##########################################
        ### SET DESIRED-STATE:
        ##########################################
        [scriptblock]$script:SetDesiredState =
        {
            $script:DesiredState = $PageFileSize

            # Disable Automatically Managed PageFile Setting
            $ComputerSystem = Get-CimInstance -ClassName Win32_ComputerSystem
            if ($ComputerSystem.AutomaticManagedPagefile -eq "True")
            {
                Set-CimInstance -Property @{ AutomaticManagedPageFile = $False }
            }
            Write-Host "AutomaticManagedPagefile : " $(Get-CimInstance -ClassName Win32_ComputerSystem).AutomaticManagedPagefile

            ### Delete Existing PageFile
            ### ---------------------------------
            $PageFile = Get-CimInstance -ClassName Win32_PageFileSetting
            $PageFile | Remove-CimInstance


            ### Calculate Page File Size
            ### ---------------------------------
            $Memory = (Get-WMIObject -class Win32_PhysicalMemory | Measure-Object -Property Capacity -Sum | % {[Math]::Round(($_.sum / 1GB),2)})
            $NewPageFileSize = [Math]::Round(($Memory * $PageFileSize)) # Memory Size Plus 20% - Round Up
            if ($Memory -lt "4") {$NewPageFileSize = "4"}
            [int]$NewPageFileSize = ($NewPageFileSize * 1000)
            [int]$InitialSize     = ($NewPageFileSize * 1)
            [int]$MaximumSize     = ($NewPageFileSize * 1)

            ### Create New Page File
            ### -------------------------------------------------------------------------------
            [scriptblock]$CreatePageFile =
            {
                if(-NOT(Get-CimInstance -ClassName Win32_PageFileSetting))
                {
                    $PageFileName = $PageFileLocation + ":\pagefile.sys"
                    Write-Host "Creating New Page File: PageFileLocation: $PageFileName InitialSize: $InitialSize MaximumSize: $MaximumSize "
                    New-CimInstance -ClassName Win32_PageFileSetting -Property  @{ Name= $PageFileName } | Out-Null
                    Get-CimInstance -ClassName Win32_PageFileSetting | Set-CimInstance -Property @{InitialSize = $InitialSize; MaximumSize = $MaximumSize;} | Out-Null
                }
            }

            ### Check Avilable Disk Space
            ### ---------------------------------
            $FreeSpace = (Get-PSDrive $PageFileLocation).Free

            #[int]$FreeSpace = $FreeSpace

            if($FreeSpace -gt $NewPageFileSize)
            {
                Write-Host "Free Space Available. Drive: $Drive FreeSpace: $FreeSpace NewPageFileSize: $NewPageFileSize"
                Invoke-Command -ScriptBlock $CreatePageFile
            }
            elseif($FreeSpace -le $NewPageFileSize)
            {
                Write-Host "Not Enough Avialable Space. Drive: $Drive FreeSpace: $FreeSpace
                NewPageFileSize: $NewPageFileSize `r`nNot Configuring!"
            }


        }

        Configure-DesiredState -GetCurrentState $GetCurrentState -SetDesiredState
        $SetDesiredState -ConfigurationMode $ConfigurationMode -AbortTrigger $AbortTrigger
        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
    }

    function Rename-ECI.EMI.Configure.OS.GuestComputer
    {

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

        ##########################################
        ### DESIRED STATE PARAMETERS
        ##########################################
        $PropertyName      = "HostName"
        $DesiredState      = $HostName
        $ConfigurationMode = "Configure" ### Report - Configure
        $AbortTrigger      = $True   ### $True - $False

        ##########################################
        ### GET CURRENT CONFIGURATION STATE:
        ##########################################
        [scriptblock]$script:GetCurrentState =
        {
            $global:CurrentState =  (Get-ItemProperty -Path
            HKLM:\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName).ComputerName

            #$ActiveComputerName = (Get-ItemProperty -Path
            HKLM:\SYSTEM\CurrentControlSet\Control\ComputerName\ActiveComputerName).ComputerN
            ame
            #$ComputerName = (Get-ItemProperty -Path
            HKLM:\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName).ComputerName

        }

        ##########################################
        ### SET DESIRED-STATE:
        ##########################################
        [scriptblock]$script:SetDesiredState =
        {
            #Rename-Computer -ComputerName $(Get-CIMInstance CIM_ComputerSystem).Name
            -NewName $HostName
            #Rename-Computer -ComputerName $(Get-WmiObject Win32_Computersystem).Name
            -NewName $HostName
            Rename-Computer -ComputerName . -NewName $HostName
        }

        ##########################################
        ### CALL CONFIGURE DESIRED STATE:
        ##########################################
        ###----------------------------
        $Params = @{
            ServerID             = $ServerID
            HostName             = $HostName
            FunctionName         = $FunctionName
            PropertyName         = $PropertyName
            DesiredState         = $DesiredState
            GetCurrentState      = $GetCurrentState
            SetDesiredState      = $SetDesiredState
            ConfigurationMode    = $ConfigurationMode
            AbortTrigger         = $AbortTrigger
        }
        Configure-DesiredState @Params
        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
    }

    function Restart-ECI.EMI.Configure.OS.GuestComputer
    {
        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray

        $t = 10

        Write-Host "Restarting Guest OS in $t seconds . . . "
        Shutdown /r -t $t

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

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

        ##########################################
        ### DESIRED STATE PARAMETERS
        ##########################################
        $PropertyName      = "ClientDomain"
        $DesiredState      = $ClientDomain
        $ConfigurationMode = "Configure" ### Report - Configure
        $AbortTrigger      = $False   ### $True - $False

        ##########################################
        ### GET CURRENT CONFIGURATION STATE:
        ##########################################
        [scriptblock]$script:GetCurrentState =
        {
            $global:CurrentState = (Get-CimInstance -ClassName Win32_ComputerSystem).Domain
        }

        ##########################################
        ### SET DESIRED-STATE:
        ##########################################
        [scriptblock]$script:SetDesiredState =
        {
            $AdministrativePassword = ConvertTo-SecureString $AdministrativePassword
            -AsPlainText -Force
            $PSCredentials =  New-Object System.Management.Automation.PSCredential
            ($AdministrativeUserName, $AdministrativePassword)

            #$OUPath = "OU=Servers,OU=London,DC=ercolanomgmt,DC=corp"
            #Add-Computer -ComputerName $HostName -DomainName $ClientDomain -OUPath $OUPath
            -Credential $PSCredentials -Force -Verbose

            Add-Computer -ComputerName $HostName -DomainName $ClientDomain -Credential
            $PSCredentials -Force -Verbose
        }

        ##########################################
        ### CALL CONFIGURE DESIRED STATE:
        ##########################################
        ###----------------------------
        ### Test Variables passed from #$Variable# substitution
        $Params = @{
            ServerID            = $ServerID
            HostName            = $HostName
            FunctionName        = $FunctionName
            PropertyName        = $PropertyName
            DesiredState        = $DesiredState
            GetCurrentState     = $GetCurrentState
            SetDesiredState      = $SetDesiredState
            ConfigurationMode    = $ConfigurationMode
            AbortTrigger         = $AbortTrigger
        }
        Configure-DesiredState @Params
        Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
    }

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

        $t = 30
        Write-Host "Restarting Guest OS on Server - $Hostname in $t seconds:"
        -ForegroundColor Magenta
        Start-Sleep -Seconds $t
        Restart-Computer -ComputerName . -Force #-Wait -For PowerShell -Timeout 300 -Delay
        $t -Verbose
    }

    function
    Rename-LocalAdministrator-test
                                                                              #
    <------------------------------ deleteme??????
    {
        $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
        "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray

            ### Use .NET to Find the Current Local Administrator Account
            Add-Type -AssemblyName System.DirectoryServices.AccountManagement
            $ComputerName = [System.Net.Dns]::GetHostName()
            $PrincipalContext = New-Object
            System.DirectoryServices.AccountManagement.PrincipalContext([System.DirectoryServ
            ices.AccountManagement.ContextType]::Machine, $ComputerName)
            $UserPrincipal = New-Object
            System.DirectoryServices.AccountManagement.UserPrincipal($PrincipalContext)
            $Searcher = New-Object
            System.DirectoryServices.AccountManagement.PrincipalSearcher
            $Searcher.QueryFilter = $UserPrincipal

            ### The Administrator account is the only account that has a SID that ends with
            "-500"
            $Account = $Searcher.FindAll() | Where-Object {$_.Sid -Like "*-500"}
            $script:CurrentAdminName = $Account.Name

            Write-Host "1CurrentAdminName              : " $CurrentAdminName
            Write-Host "1LocalAdministrator            : " $LocalAdministrator
            Write-Host "1PreConfig_LocalAdminAccount   : " $PreConfig_LocalAdminAccount
            Write-Host "1PostConfig_LocalAdminAccount  : " $PostConfig_LocalAdminAccount


            ### Check if Local Admin is already renamed
            if($CurrentAdminName -eq $LocalAdministrator)
            {
                #Write-Host "Local Admin Names are the same -  CurrentAdminName:
                $CurrentAdminName NewAdminName: $LocalAdministrator"
                #$RebootRequired = $False
            }
            elseif($CurrentAdminName -ne $NewLocalAdminName)
            {

               # $RebootRequired = $True
                #Write-Host "Renaming Local Admin Account: Current Admin: $CurrentAdminName
                New Admin: $LocalAdministrator"
                Rename-LocalUser -Name $CurrentAdminName -NewName $LocalAdministrator
                -ErrorAction SilentlyContinue | Out-Null


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

    function Configure-ECI.EMI.Configure.OS.RegisterDNS
    {
        Ipconfig /registerdns
    }

```