

```

1 #####
2 ### Configure Guest OS Module
3 ### ECI.EMI.Automation.OS.Prod.psm1
4 #####
5
6
7 function Import-ECI.EMI.Modules
8 {
9     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
10
11     foreach($Module in (Get-Module -ListAvailable ECI.*)){Import-Module -Name
    $Module.Path -DisableNameChecking}
12
13
14     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
15 }
16
17 function Configure-ECI.EMI.Configure.OS.NetworkInterface
18 {
19     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
    "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
20
21     #####
22     ### DESIRED STATE PARAMETERS
23     #####
24     $PropertyName = "NetworkInterfaceName"
25     $DesiredState = $NetworkInterfaceName
26     $ConfigurationMode = "Configure" ### Report - Configure
27     $AbortTrigger = $False ### $True - $False
28
29     #####
30     ### GET CURRENT CONFIGURATION STATE:
31     #####
32     [scriptblock]$script:GetCurrentState =
33     {
34         $global:CurrentState = (Get-NetAdapter -Physical | Where-Object Status -eq
    'Up').Name
35     }
36
37     #####
38     ### SET DESIRED-STATE:
39     #####
40     [scriptblock]$script:SetDesiredState =
41     {
42         Rename-NetAdapter (Get-NetAdapter -Name $CurrentState).Name -NewName
    $DesiredState
43     }
44
45     #####
46     ### CALL CONFIGURE DESIRED STATE:
47     #####
48     ###-----
49
50     $Params = @{}
51     ServerID = $ServerID
52     HostName = $HostName
53     FunctionName = $FunctionName
54     PropertyName = $PropertyName
55     DesiredState = $DesiredState
56     GetCurrentState = $GetCurrentState
57     SetDesiredState = $SetDesiredState
58     ConfigurationMode = $ConfigurationMode
59     AbortTrigger = $AbortTrigger
60 }
61 Configure-DesiredState @Params
62
63 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
64 }

```

```

65
66
67 function Configure-ECI.EMI.Configure.OS.WSMan
68 {
69
70
71 }
72
73 function Configure-ECI.EMI.Configure.OS.SMBv1
74 {
75     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host "`r`n('-' * 50)`r`n
"EXECUTING FUNCTION: " $FunctionName "`r`n('-' * 50) -ForegroundColor Gray
76
77     #####
78     ### DESIRED STATE PARAMETERS
79     #####
80     $PropertyName      = "SMBv1"
81     $DesiredState      = $SMBv1
82     $ConfigurationMode = "Configure" ### Report - Configure
83     $AbortTrigger      = $False    ### $True - $False
84
85     #####
86     ### GET CURRENT CONFIGURATION STATE:
87     #####
88     [scriptblock]$script:GetCurrentState =
89     {
90         $KeyPath = "HKLM:\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters"
91         $KeyName = "SMB1"
92
93         try
94         {
95             $KeyValue = Get-ItemProperty -Path $KeyPath -Name $KeyName
96         }
97         catch
98         {
99             $global:CurrentState = $null
100             Write-ECI.ErrorStack
101         }
102
103         if($KeyValue)
104         {
105             $global:CurrentState = $KeyValue
106         }
107         else
108         {
109             $global:CurrentState = $null
110         }
111     }
112
113     #####
114     ### SET DESIRED-STATE:
115     #####
116     [scriptblock]$script:SetDesiredState =
117     {
118         $KeyValue = $DesiredState
119         $KeyPath  = "SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters"
120         $KeyName  = "SMB1"
121         Set-ItemProperty -Path $KeyPath -Name $Keyname -Value $KeyValue
122     }
123
124     #####
125     ### CALL CONFIGURE DESIRED STATE:
126     #####
127     ###-----
128     $Params = @{
129         ServerID      = $ServerID
130         HostName      = $HostName
131         FunctionName  = $FunctionName
132         PropertyName  = $PropertyName

```

```

133         DesiredState           = $DesiredState
134         GetCurrentState         = $GetCurrentState
135         SetDesiredState         = $SetDesiredState
136         ConfigurationMode       = $ConfigurationMode
137         AbortTrigger            = $AbortTrigger
138     }
139     Configure-DesiredState @Params
140     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
141 }
142
143 function Configure-ECI.EMI.Configure.OS.IPv6
144 {
145     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
146     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
147
148     #####
149     ### DESIRED STATE PARAMETERS
150     #####
151     $PropertyName           = "IPv6Preference"
152     $DesiredState           = $IPv6Preference
153     $ConfigurationMode      = "Configure" ### Report - Configure
154     $AbortTrigger           = $False ### $True - $False
155
156     #####
157     ### GET CURRENT CONFIGURATION STATE:
158     #####
159     [scriptblock]$script:GetCurrentState =
160     {
161         ### Get Current Interface
162         $CurrentInterface = (Get-NetAdapter -Physical | Where-Object {$_.Status -eq
163         'Up'}).Name
164         $IPv6State = Get-NetAdapterBinding -InterfaceAlias $CurrentInterface
165         -DisplayName "Internet Protocol Version 6 (TCP/IPv6)"
166         $global:CurrentState = $IPv6State.Enabled ### Return True/False
167     }
168
169     #####
170     ### SET DESIRED-STATE:
171     #####
172
173     [scriptblock]$script:SetDesiredState =
174     {
175         if ($DesiredState -eq $True)
176         {
177             ### Enable IPv6
178             Enable-NetAdapterBinding -InterfaceAlias $CurrentInterfaceName -ComponentID
179             MS_TCPIP6
180         }
181         elseif ($DesiredState -eq $False)
182         {
183             ### Disable IPv6
184             Disable-NetAdapterBinding -InterfaceAlias $CurrentInterfaceName
185             -ComponentID MS_TCPIP6
186         }
187     }
188
189     #####
190     ### CALL CONFIGURE DESIRED STATE:
191     #####
192     $Params = @{
193         ServerID           = $ServerID
194         HostName            = $HostName
195         FunctionName        = $FunctionName
196         PropertyName        = $PropertyName
197         DesiredState        = $DesiredState
198         GetCurrentState      = $GetCurrentState
199         SetDesiredState      = $SetDesiredState
200         ConfigurationMode    = $ConfigurationMode
201         AbortTrigger        = $AbortTrigger

```

```

197     }
198     Configure-DesiredState @Params
199     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
200 }
201
202 function Configure-ECI.EMI.Configure.OS.CDROM
203 {
204     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
205     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
206
207     ### Modify Parameter Values
208     ### -----
209     ### Drive Letter Must End with Colon ":"
210     $CDROMLetter = $CDROMLetter.Trim()
211     $LastChar = $CDROMLetter.substring($CDROMLetter.length-1)
212     if ($LastChar -ne ":"){$CDROMLetter = $CDROMLetter + ":"}
213
214     #####
215     ### DESIRED STATE PARAMETERS
216     #####
217     $PropertyName = "CDROMLetter"
218     $DesiredState = $CDROMLetter
219     $ConfigurationMode = "Configure" ### Report - Configure
220     $AbortTrigger = $False ### $True - $False
221
222     #####
223     ### GET CURRENT CONFIGURATION STATE:
224     #####
225     [scriptblock]$script:GetCurrentState =
226     {
227         $script:ComputerName = (Get-WmiObject Win32_ComputerSystem).Name
228         $global:CurrentState = (Get-WMIObject -Class Win32_CDROMDrive -ComputerName
229         $ComputerName).Drive
230     }
231
232     #####
233     ### SET DESIRED-STATE:
234     #####
235     [scriptblock]$script:SetDesiredState =
236     {
237         $CDVolume = Get-WmiObject -Class Win32_Volume -ComputerName $ComputerName
238         -Filter "DriveLetter='$CurrentState'"
239         Set-WmiInstance -InputObject $CDVolume -Arguments @{DriveLetter =
240         $DesiredState} | Out-Null
241     }
242
243     #####
244     ### CALL CONFIGURE DESIRED STATE:
245     #####
246     $Params = @{
247         ServerID = $ServerID
248         HostName = $HostName
249         FunctionName = $FunctionName
250         PropertyName = $PropertyName
251         DesiredState = $DesiredState
252         GetCurrentState = $GetCurrentState
253         SetDesiredState = $SetDesiredState
254         ConfigurationMode = $ConfigurationMode
255         AbortTrigger = $AbortTrigger
256     }
257     Configure-DesiredState @Params
258     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
259 }
260
261 function Configure-ECI.EMI.Configure.OS.Folders
262 {
263     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
264     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray

```

```

261 #####
262 ### DESIRED STATE PARAMETERS
263 #####
264 $PropertyName      = "ECIFolders"
265
266 ### Create ECI Folders
267 ###-----
268 $ECIFolders = @()
269 $ECIFolders += "C:\Scripts"
270 $ECIFolders += "D:\Kits"
271
272 $DesiredState      = $ECIFolders
273 $ConfigurationMode = "Configure" ### Report - Configure
274 $AbortTrigger      = $False  ### $True - $False
275
276 foreach($State in $DesiredState)
277 {
278     $DesiredState = $State
279
280     #####
281     ### GET CURRENT CONFIGURATION STATE:
282     #####
283     [scriptblock]$script:GetCurrentState =
284     {
285         if(Test-Path -Path $DesiredState){$global:CurrentState = $DesiredState}
286         else{$global:CurrentState = $False}
287     }
288
289     #####
290     ### SET DESIRED-STATE:
291     #####
292     [scriptblock]$script:SetDesiredState =
293     {
294         New-Item -ItemType Directory -Path $State -Force | Out-Null
295     }
296
297     #####
298     ### CALL CONFIGURE DESIRED STATE:
299     #####
300     ###-----
301     $Params = @{
302         ServerID      = $ServerID
303         HostName      = $HostName
304         FunctionName   = $FunctionName
305         PropertyName   = $PropertyName
306         DesiredState   = $DesiredState
307         GetCurrentState = $GetCurrentState
308         SetDesiredState = $SetDesiredState
309         ConfigurationMode = $ConfigurationMode
310         AbortTrigger   = $AbortTrigger
311     }
312     Configure-DesiredState @Params
313 }
314 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
315 }
316
317 function Configure-ECI.EMI.Configure.OS.RemoteDesktop
318 {
319     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
320     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
321
322     ### Modify Parameter Values
323     ### -----
324     if($RemoteDesktopPreference -eq "False"){ $RemoteDesktopPreferenceValue = "1"}
325     elseif($RemoteDesktopPreference -eq "True"){ $RemoteDesktopPreferenceValue = "0"}
326
327     #####
328     ### DESIRED STATE PARAMETERS
329     #####

```

```

329 $PropertyName = "RemoteDesktopPreference"
330 $DesiredState = $RemoteDesktopPreferenceValue
331 $ConfigurationMode = "Configure" ### Report - Configure
332 $AbortTrigger = $False ### $True - $False
333
334 #####
335 ### GET CURRENT CONFIGURATION STATE:
336 #####
337 [scriptblock]$script:GetCurrentState =
338 {
339     $global:CurrentState = (Get-ItemProperty -Path
340         'HKLM:\System\CurrentControlSet\Control\Terminal Server' -name
341         "fDenyTSConnections").fDenyTSConnections
342     }
343     #####
344     ### SET DESIRED-STATE:
345     #####
346     [scriptblock]$script:SetDesiredState =
347     {
348         Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server'
349         -name "fDenyTSConnections" -Value $RemoteDesktopPreferenceValue
350
351         if($RemoteDesktopPreferenceValue -eq "0")
352         {
353             Enable-NetFirewallRule -DisplayGroup "Remote Desktop"
354             Netsh advfirewall firewall set rule group="remote desktop" new enable=yes
355         }
356         elseif($RemoteDesktopPreferenceValue -eq "1")
357         {
358             Disable-NetFirewallRule -DisplayGroup "Remote Desktop"
359         }
360     }
361     #####
362     ### CALL CONFIGURE DESIRED STATE:
363     #####
364     $Params = @{
365         ServerID = $ServerID
366         HostName = $HostName
367         FunctionName = $FunctionName
368         PropertyName = $PropertyName
369         DesiredState = $DesiredState
370         GetCurrentState = $GetCurrentState
371         SetDesiredState = $SetDesiredState
372         ConfigurationMode = $ConfigurationMode
373         AbortTrigger = $AbortTrigger
374     }
375     Configure-DesiredState @Params
376     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
377 }
378 function Configure-ECI.EMI.Configure.OS.WindowsFirewallProfile
379 {
380     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
381     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
382     #####
383     ### DESIRED STATE PARAMETERS
384     #####
385     $PropertyName = "WindowsFirewallPreference"
386     $DesiredState = $WindowsFirewallPreference
387     $ConfigurationMode = "Configure" ### Report - Configure
388     $AbortTrigger = $False ### $True - $False
389
390     #####
391     ### GET CURRENT CONFIGURATION STATE:
392     #####
393     [scriptblock]$script:GetCurrentState =

```

```

394 {
395     $global:CurrentState = (Get-NetFirewallProfile -Name Domain).Enabled
396 }
397
398 #####
399 ### SET DESIRED-STATE:
400 #####
401 [scriptblock]$script:SetDesiredState =
402 {
403     Set-NetFirewallProfile -Profile Domain -Enabled $WindowsFirewallPreference
404 }
405
406 #####
407 ### CALL CONFIGURE DESIRED STATE:
408 #####
409 $Params = @{}
410     ServerID           = $ServerID
411     HostName           = $HostName
412     FunctionName       = $FunctionName
413     PropertyName       = $PropertyName
414     DesiredState       = $DesiredState
415     GetCurrentState    = $GetCurrentState
416     SetDesiredState    = $SetDesiredState
417     ConfigurationMode  = $ConfigurationMode
418     AbortTrigger       = $AbortTrigger
419 }
420 Configure-DesiredState @Params
421 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
422 }
423
424 function Configure-ECI.EMI.Configure.OS.InternetExplorerESC
425 {
426     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
427     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
428
429     ### Modify Parameter Values
430     ### -----
431     if($InternetExplorerESCPreference -eq "False") {$InternetExplorerESCValue = "0"}
432     elseif($InternetExplorerESCPreference -eq "True"){$InternetExplorerESCValue = "1"}
433
434     #####
435     ### DESIRED STATE PARAMETERS
436     #####
437     $PropertyName = "InternetExplorerESCPreference"
438     $script:DesiredState = $InternetExplorerESCPreference
439     $ConfigurationMode = "Configure" ### Report - Configure
440     $AbortTrigger = $False ### $True - $False
441
442     $Keys = @()
443     $Keys += "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed
444     Components\{A509B1A7-37EF-4b3f-8CFC-4F3A74704073}"
445     $Keys += "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed
446     Components\{A509B1A8-37EF-4b3f-8CFC-4F3A74704073}"
447
448     foreach($Key in $Keys)
449     {
450         #####
451         ### GET CURRENT CONFIGURATION STATE:
452         #####
453         [scriptblock]$script:GetCurrentState =
454         {
455             $global:CurrentState = (Get-ItemProperty -Path $Key -Name
456             "IsInstalled").IsInstalled
457         }
458
459         #####
460         ### SET DESIRED-STATE:
461         #####
462         [scriptblock]$script:SetDesiredState =

```

```

459     {
460         Set-ItemProperty -Path $Key -Name "IsInstalled" -Value $DesiredState -Force
461     }
462
463     #####
464     ### CALL CONFIGURE DESIRED STATE:
465     #####
466     $Params = @{
467         ServerID           = $ServerID
468         HostName           = $HostName
469         FunctionName       = $FunctionName
470         PropertyName       = $PropertyName
471         DesiredState       = $DesiredState
472         GetCurrentState    = $GetCurrentState
473         SetDesiredState    = $SetDesiredState
474         ConfigurationMode  = $ConfigurationMode
475         AbortTrigger       = $AbortTrigger
476     }
477     Configure-DesiredState @Params
478 }
479 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
480 }
481
482 function Initialize-ECI.EMI.Configure.OS.HardDisks
483 {
484     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
485     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
486
487     ### Initalzie Disk
488     ### -----
489     Write-Host "Initialize Disk: "
490     Get-Disk | Where-Object partitionstyle -eq 'raw' | Initialize-Disk -PartitionStyle
491     MBR -PassThru | New-Partition -AssignDriveLetter -UseMaximumSize | Format-Volume
492     -FileSystem NTFS -NewFileSystemLabel "SwapFile" -Confirm:$false
493     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
494 }
495
496 function Configure-ECI.EMI.Configure.OS.WindowsFeatures
497 {
498     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
499     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
500
501     #####
502     ### DESIRED STATE PARAMETERS
503     #####
504     $PropertyName       = "WindowsFeatures"
505     $ConfigurationMode  = "Configure" ### Report - Configure
506     $AbortTrigger       = $False ### $True - $False
507
508     switch ( $ServerRole )
509     {
510         "2016Server"
511         { $WindowsFeatures =
512             @(
513                 "NET-Framework-Features",
514                 "NET-Framework-Core",
515                 "GPMC",
516                 "Telnet-Client"
517             )
518         }
519         "2016FS"
520         { $WindowsFeatures =
521             @(
522                 "NET-Framework-Features",
523                 "NET-Framework-Core",
524                 "GPMC",
525                 "Telnet-Client",
526                 "RSAT"
527             )
528         }
529     }

```



```

524     }
525     "2016DC"
526     { $WindowsFeatures =
527     @(
528         "NET-Framework-Features",
529         "NET-Framework-Core",
530         "GPMC",
531         "Telnet-Client",
532         "RSAT"
533     )
534     }
535     "2016DCFS"
536     { $WindowsFeatures =
537     @(
538         "NET-Framework-Features",
539         "NET-Framework-Core",
540         "GPMC",
541         "Telnet-Client",
542         "RSAT"
543     )
544     }
545     "2016VDA"
546     { $WindowsFeatures =
547     @(
548         "NET-Framework-Features",
549         "NET-Framework-Core",
550         "GPMC",
551         "Telnet-Client",
552         "RSAT"
553         "AS-Net-Framework",
554         "RDS-RD-Server"
555     )
556     }
557     "2016SQL"
558     { $WindowsFeatures =
559     @(
560         "NET-Framework-Features",
561         "NET-Framework-Core",
562         "GPMC",
563         "Telnet-Client",
564         "RSAT"
565     )
566     }
567     "2016SQLOMS"
568     { $WindowsFeatures =
569     @(
570         "NET-Framework-Features",
571         "NET-Framework-Core",
572         "GPMC",
573         "Telnet-Client",
574         "RSAT"
575     )
576     }
577 }
578 foreach ($Feature in $WindowsFeatures)
579 {
580     $script:DesiredState = $Feature
581     write-host "DesiredState: " $DesiredState
582
583     #####
584     ## GET CURRENT CONFIGURATION STATE:
585     #####
586     [scriptblock]$script:GetCurrentState =
587     {
588         $global:CurrentState = ((Get-WindowsFeature -Name $Feature) | Where-Object
589             {$_.Installed -eq $True}).Name
590     }
591     #####

```

```

592     ### SET DESIRED-STATE:
593     #####
594     [scriptblock]$script:SetDesiredState =
595     {
596         Write-Host "Installing Feature: " $Feature
597
598         # $WindowsMediaSource =
599         "R:\sources\sxs\microsoft-windows-netfx3-ondemand-package.cab"
600         $WindowsMediaSource = "R:\sources\sxs"
601         Install-WindowsFeature -Name $Feature -Source $WindowsMediaSource
602     }
603     #####
604     ### CALL CONFIGURE DESIRED STATE:
605     #####
606     $Params = @{
607         ServerID           = $ServerID
608         HostName           = $HostName
609         FunctionName       = $FunctionName
610         PropertyName       = $PropertyName
611         DesiredState       = $DesiredState
612         GetCurrentState    = $GetCurrentState
613         SetDesiredState    = $SetDesiredState
614         ConfigurationMode  = $ConfigurationMode
615         AbortTrigger       = $AbortTrigger
616     }
617     Configure-DesiredState @Params
618 }
619 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
620 }
621
622 function Configure-ECI.EMI.Configure.OS.WindowsFirewallRules
623 {
624     $FunctionName = $((Get-PSCallStack)[0].Command); Write-Host `r`n('-' * 50)`r`n
625     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
626
627     #####
628     ### DESIRED STATE PARAMETERS
629     #####
630     $PropertyName       = "WindowsFirewallRules"
631     $ConfigurationMode  = "Configure" ### Report - Configure
632     $AbortTrigger       = $False ### $True - $False
633
634     ### Set Firewall Rules
635     ###-----
636     $FireWallRules = @()
637     $FireWallRules += "File and Printer Sharing (SMB-In)"
638     $FireWallRules += "Windows Management Instrumentation (ASync-In)"
639     $FireWallRules += "Windows Management Instrumentation (DCOM-In)"
640     $FireWallRules += "Windows Management Instrumentation (WMI-In)"
641
642     foreach($Rule in $FireWallRules)
643     {
644         ### Set Desired State Value
645         ###-----
646         $script:DesiredState = $Rule
647
648         #####
649         ### GET CURRENT CONFIGURATION STATE:
650         #####
651         [scriptblock]$script:GetCurrentState =
652         {
653             $global:CurrentState = (Get-NetFirewallProfile -Name Domain |
654             Get-NetFirewallRule | Where {$_.DisplayName -eq $Rule}).DisplayName
655         }
656
657         #####
658         ### SET DESIRED-STATE:
659         #####

```

```

658 [scriptblock]$script:SetDesiredState =
659 {
660     Write-Host "Installing Feature: " $Feature
661     Enable-NetFirewallRule -DisplayName $Rule
662 }
663
664 #####
665 ### CALL CONFIGURE DESIRED STATE:
666 #####
667 $Params = @{
668     ServerID           = $ServerID
669     HostName           = $HostName
670     FunctionName       = $FunctionName
671     PropertyName       = $PropertyName
672     DesiredState       = $DesiredState
673     GetCurrentState    = $GetCurrentState
674     SetDesiredState    = $SetDesiredState
675     ConfigurationMode  = $ConfigurationMode
676     AbortTrigger       = $AbortTrigger
677 }
678 Configure-DesiredState @Params
679 }
680
681 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
682 }
683
684 function Configure-ECI.EMI.Configure.OS.WindowsFirewallRules-orig-deleteme
685 {
686     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
687     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
688
689     $FireWallRules = @()
690     $FireWallRules += "File and Printer Sharing (SMB-In)"
691     $FireWallRules += "Windows Management Instrumentation (ASync-In)"
692     $FireWallRules += "Windows Management Instrumentation (DCOM-In)"
693     $FireWallRules += "Windows Management Instrumentation (WMI-In)"
694
695     foreach($Rule in $FireWallRules)
696     {
697         #Write-Host "Getting FireWall Rule: " $Rule
698         Get-NetFirewallRule | Where {$_.DisplayName -eq $Rule}
699         Enable-NetFirewallRule -DisplayName $Rule
700     }
701
702     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
703 }
704
705 function Configure-ECI.EMI.Configure.OS.PageFileLocation
706 {
707     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
708     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
709
710     #####
711     ### DESIRED STATE PARAMETERS
712     #####
713     $PropertyName       = "PageFileLocation"
714     $DesiredState       = $PageFileLocation
715     $ConfigurationMode  = "Configure" ### Report - Configure
716     $AbortTrigger       = $False ### $True - $False
717
718     ### Modify Parameter Values
719     ### -----
720     ### Drive Letter Must NOT End with Colon ":"
721     $PageFileLocation = $PageFileLocation.Trim()
722     $LastChar = $PageFileLocation.substring($PageFileLocation.length-1)
723     if ($LastChar -eq ":"){$PageFileLocation = $PageFileLocation.Split(":")[0]}
724
725     #####
726     ### GET CURRENT CONFIGURATION STATE:

```

```

725 #####
726 [scriptblock]$script:GetCurrentState =
727 {
728     try
729     {
730         if(((Get-CimInstance -ClassName
731             Win32_ComputerSystem).AutomaticManagedPagefile) -eq $True)
732         {
733             $global:CurrentState = $Null
734             Write-Host "PageFile is set to AutomaticManagedPagefile"
735         }
736         else
737         {
738             $global:CurrentState = ((Get-CimInstance -ClassName
739                 Win32_PageFileSetting).Name).Split(":")[0]
740         }
741     }
742     catch
743     {
744         Write-ECI.ErrorStack
745     }
746 }
747 #####
748 ### SET DESIRED-STATE:
749 #####
750 [scriptblock]$script:SetDesiredState =
751 {
752     $script:DesiredState = $PageFileLocation
753
754     # Disable Automatically Managed PageFile Setting
755     ### -----
756     $ComputerSystem = Get-CimInstance -ClassName Win32_ComputerSystem
757     if ($ComputerSystem.AutomaticManagedPagefile -eq "True")
758     {
759         Set-CimInstance -InputObject $ComputerSystem -Property
760             @{AutomaticManagedPageFile = $False }
761     }
762     Write-Host "AutomaticManagedPagefile: " $(Get-CimInstance -ClassName
763         Win32_ComputerSystem).AutomaticManagedPagefile
764
765     ### Delete Existing PageFile
766     ### -----
767     $PageFile = Get-CimInstance -ClassName Win32_PageFileSetting
768     $PageFile | Remove-CimInstance
769
770     ### Calculate Page File Size
771     ### -----
772     $Memory = (Get-CimInstance -ClassName Win32_PhysicalMemory | Measure-Object
773         -Property Capacity -Sum | % {[Math]::Round(($_.sum / 1MB),2)})
774     $NewPageFileSize = [Math]::Round(($Memory * $PageFileMultiplier)) # Memory Size
775     Plus 20% - Round Up
776
777     ### PageFile Minumin Size = 4GB
778     ###-----
779     if ($NewPageFileSize -lt "4096") {$NewPageFileSize = "4096"}
780
781     ### Min/Max Multiplier - Currently 1/1
782     ###-----
783     [int]$script:InitialSize = ($NewPageFileSize * 1)
784     [int]$script:MaximumSize = ($NewPageFileSize * 1)
785
786     Write-Host "PageFile Location      : " $PageFileLocation
787     Write-Host "PageFile InitialSize : " $InitialSize
788     Write-Host "PageFile MaximumSize : " $MaximumSize
789
790     ###
791     -----
792     ### Create New Page File

```

```

787     ###
788     [scriptblock]$CreatePageFile =
789     {
790         if(-NOT(Get-CimInstance -ClassName Win32_PageFileSetting))
791         {
792             try
793             {
794                 $PageFileName = $PageFileLocation + ":\pagefile.sys"
795                 New-CimInstance -ClassName Win32_PageFileSetting -Property @{
796                     Name= $PageFileName } | Out-Null
797                 Get-CimInstance -ClassName Win32_PageFileSetting | Set-CimInstance
798                     -Property @{InitialSize = $InitialSize; MaximumSize =
799                     $MaximumSize;} | Out-Null
800             }
801             catch
802             {
803                 Write-ECI.ErrorStack
804             }
805         }
806         else
807         {
808             Write-Host "A Page File Already Exists!"
809             $Abort = $True
810         }
811     }
812     ### Check Available Disk Space
813     ### -----
814     [int]$FreeSpace = [Math]::Round(((Get-PSDrive $PageFileLocation).Free/1MB),2)
815
816     if($FreeSpace -gt $NewPageFileSize)
817     {
818         Write-Host "Free Space is Available. Drive: $PageFileLocation FreeSpace:
819         $FreeSpace NewPageFileSize: $NewPageFileSize"
820         Invoke-Command -ScriptBlock $CreatePageFile
821     }
822     elseif($FreeSpace -le $NewPageFileSize)
823     {
824         Write-Host "Not Enough Available Space. Drive: $Drive FreeSpace: $FreeSpace
825         NewPageFileSize: $NewPageFileSize `r`nNot Configuring!"
826         $Abort = $True
827     }
828 }
829
830 #####
831 ### CALL CONFIGURE DESIRED STATE:
832 #####
833 ###-----
834 $Params = @{
835     ServerID           = $ServerID
836     HostName           = $HostName
837     FunctionName       = $FunctionName
838     PropertyName       = $PropertyName
839     DesiredState       = $DesiredState
840     GetCurrentState    = $GetCurrentState
841     SetDesiredState    = $SetDesiredState
842     ConfigurationMode  = $ConfigurationMode
843     AbortTrigger       = $AbortTrigger
844 }
845 Configure-DesiredState @Params
846 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
847 }
848
849 function Configure-ECI.EMI.Configure.OS.PageFileSize
850 {
851     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
852     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray

```

```

849 #####
850 ### DESIRED STATE PARAMETERS
851 #####
852 $PropertyName = "PageFileSize"
853 $ConfigurationMode = "Configure" ### Report - Configure
854 $AbortTrigger = $True ### $True - $False
855
856 ### Modify Parameter Values
857 #####
858 ### Drive Letter Must NOT End with Colon ":"
859 $PageFileLocation = $PageFileLocation.Trim()
860 $LastChar = $PageFileLocation.substring($PageFileLocation.length-1)
861 if ($LastChar -eq ":"){$PageFileLocation = $PageFileLocation.Split(":")[0]}
862
863 $Memory = (Get-CimInstance -ClassName Win32_PhysicalMemory | Measure-Object
864 -Property Capacity -Sum | % {[Math]::Round(($_sum / 1MB),2)})
865 $DesiredState = [Math]::Round(($Memory * $PageFileMultiplier)) # Memory Size Plus
866 20% - Round Up
867
868 #####
869 ### GET CURRENT CONFIGURATION STATE:
870 #####
871 [scriptblock]$script:GetCurrentState =
872 {
873     try
874     {
875         if(((Get-CimInstance -ClassName
876 Win32_ComputerSystem).AutomaticManagedPagefile) -eq $True)
877         {
878             $global:CurrentState = $Null
879             Write-Host "PageFile is set to AutomaticManagedPagefile"
880         }
881         else
882         {
883             $global:CurrentState = (Get-CimInstance -ClassName
884 Win32_PageFileUsage).AllocatedBaseSize
885         }
886     }
887     catch
888     {
889         Write-ECI.ErrorStack
890     }
891 }
892
893 #####
894 ### SET DESIRED-STATE:
895 #####
896 [scriptblock]$script:SetDesiredState =
897 {
898     $script:DesiredState = $PageFileLocation
899
900     # Disable Automatically Managed PageFile Setting
901     ### -----
902     $ComputerSystem = Get-CimInstance -ClassName Win32_ComputerSystem
903     if ($ComputerSystem.AutomaticManagedPagefile -eq "True")
904     {
905         Set-CimInstance -InputObject $ComputerSystem -Property
906         @{AutomaticManagedPageFile = $False }
907     }
908     Write-Host "AutomaticManagedPagefile: " $(Get-CimInstance -ClassName
909 Win32_ComputerSystem).AutomaticManagedPagefile
910
911     ### Delete Existing PageFile
912     ### -----
913     $PageFile = Get-CimInstance -ClassName Win32_PageFileSetting
914     $PageFile | Remove-CimInstance
915
916     ### Calculate Page File Size

```

```

912     ### -----
913     $Memory = (Get-CimInstance -ClassName Win32_PhysicalMemory | Measure-Object
914     -Property Capacity -Sum | % {[Math]::Round(($_sum / 1MB),2)})
915     $NewPageFileSize = [Math]::Round(($Memory * $PageFileMultiplier)) # Memory Size
916     Plus 20% - Round Up
917
918     ### PageFile Minumin Size = 4GB
919     ###-----
920     if ($NewPageFileSize -lt "4096") {$NewPageFileSize = "4096"}
921
922     ### Min/Max Multiplier - Currently 1/1
923     ###-----
924     [int]$script:InitialSize = ($NewPageFileSize * 1)
925     [int]$script:MaximumSize = ($NewPageFileSize * 1)
926
927     Write-Host "PageFile Location      : " $PageFileLocation
928     Write-Host "PageFile InitialSize : " $InitialSize
929     Write-Host "PageFile MaximumSize : " $MaximumSize
930
931     ###
932     -----
933     ### Create New Page File
934     ###
935     -----
936     [scriptblock]$CreatePageFile =
937     {
938         if(-NOT(Get-CimInstance -ClassName Win32_PageFileSetting))
939         {
940             try
941             {
942                 $PageFileName = $PageFileLocation + ":\pagefile.sys"
943                 New-CimInstance -ClassName Win32_PageFileSetting -Property @{
944                     Name= $PageFileName } | Out-Null
945                 Get-CimInstance -ClassName Win32_PageFileSetting | Set-CimInstance
946                     -Property @{InitialSize = $InitialSize; MaximumSize =
947                         $MaximumSize;} | Out-Null
948             }
949             catch
950             {
951                 Write-ECI.ErrorStack
952             }
953         }
954         else
955         {
956             Write-Host "A Page File Already Exists!"
957             $Abort = $True
958         }
959     }
960
961     ### Check Avilable Disk Space
962     ### -----
963     [int]$FreeSpace = [Math]::Round(((Get-PSDrive $PageFileLocation).Free/1MB),2)
964
965     if($FreeSpace -gt $NewPageFileSize)
966     {
967         Write-Host "Free Space is Available. Drive: $PageFileLocation FreeSpace:
968         $FreeSpace NewPageFileSize: $NewPageFileSize"
969         Invoke-Command -ScriptBlock $CreatePageFile
970     }
971     elseif($FreeSpace -le $NewPageFileSize)
972     {
973         Write-Host "Not Enough Avialable Space. Drive: $Drive FreeSpace: $FreeSpace
974         NewPageFileSize: $NewPageFileSize `r`nNot Configuring!"
975         $Abort = $True
976     }
977 }
978
979 #####
980 ### CALL CONFIGURE DESIRED STATE:

```

```

972 #####
973 ###-----
974 $Params = @{
975     ServerID           = $ServerID
976     HostName           = $HostName
977     FunctionName       = $FunctionName
978     PropertyName       = $PropertyName
979     DesiredState       = $DesiredState
980     GetCurrentState    = $GetCurrentState
981     SetDesiredState    = $SetDesiredState
982     ConfigurationMode  = $ConfigurationMode
983     AbortTrigger       = $AbortTrigger
984 }
985 Configure-DesiredState @Params
986 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
987 }
988
989
990 function Configure-PageFile-old #<---- need to complete function!!!!!!!!!!!!!!!!!!!!!!
991 {
992     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
993     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
994
995     ### Modify Parameter Values
996     ### -----
997     ### Drive Letter Must NOT End with Colon ":"
998     $LastChar = $PageFileLocation.substring($PageFileLocation.length-1)
999     if ($LastChar -eq ":"){$PageFileLocation = $PageFileLocation.Split(":")[0]}
1000
1001     #####
1002     ### DESIRED STATE PARAMETERS
1003     #####
1004     $PropertyName = "PageFileSize"
1005
1006     $ConfigurationMode = "Configure" ### Report - Configure
1007     $AbortTrigger      = $False ### $True - $False
1008
1009     $DesiredState = @()
1010     $DesiredState += $PageFileSize
1011     $DesiredState += $PageFileLocation
1012
1013     #####
1014     ### GET CURRENT CONFIGURATION STATE:
1015     #####
1016     [scriptblock]$script:GetCurrentState =
1017     {
1018         ### Calculate Desired Page File Size
1019         $Memory = (Get-WMIObject -class Win32_PhysicalMemory | Measure-Object -Property
1020         Capacity -Sum | % {[Math]::Round(($_.sum / 1GB),2)})
1021         $DesiredPageFileSize = [Math]::Round(($Memory * $PageFileMultiplier)) # Memory
1022         Size Plus 20% - Round Up
1023         if($Memory -lt "4") {$NewPageFileSize = "4"} ### Set a Minimun 4GB PageFile Size
1024
1025         $DesiredState = ($DesiredPageFileSize * 1000)
1026         $PageFile = Get-WmiObject -Class Win32_PageFileUsage -Computer "LocalHost"
1027         $CurrentState = $PageFile.MaximumSize
1028         $Size = ($CurrentState -eq $DesiredState)
1029
1030         ### PageFile Location
1031         $script:DesiredState = $PageFileLocation
1032         $PageFile = Get-CimInstance -ClassName Win32_PageFileSetting
1033         $CurrentState = ($PageFile.Name).Split(":")[0]
1034         $Location = ($CurrentState -eq $DesiredState)
1035
1036         ### Set $CurrentState
1037         $script:CurrentState = $True
1038         $script:DesiredState = $True
1039         if (($Size -eq $False) -OR ($Location -eq $False))

```



```

1038     {
1039         $script:CurrentState = $False
1040     }
1041 }
1042
1043 #####
1044 ### SET DESIRED-STATE:
1045 #####
1046 [scriptblock]$script:SetDesiredState =
1047 {
1048     $script:DesiredState = $PageFileSize
1049
1050     # Disable Automatically Managed PageFile Setting
1051     $ComputerSystem = Get-CimInstance -ClassName Win32_ComputerSystem
1052     if ($ComputerSystem.AutomaticManagedPagefile -eq "True")
1053     {
1054         Set-CimInstance -Property @{ AutomaticManagedPageFile = $False }
1055     }
1056     Write-Host "AutomaticManagedPagefile : " $(Get-CimInstance -ClassName
1057     Win32_ComputerSystem).AutomaticManagedPagefile
1058
1059     ### Delete Existing PageFile
1060     ### -----
1061     $PageFile = Get-CimInstance -ClassName Win32_PageFileSetting
1062     $PageFile | Remove-CimInstance
1063
1064     ### Calculate Page File Size
1065     ### -----
1066     $Memory = (Get-WMIObject -class Win32_PhysicalMemory | Measure-Object -Property
1067     Capacity -Sum | % {[Math]::Round(($_.sum / 1GB),2)})
1068     $NewPageFileSize = [Math]::Round(($Memory * $PageFileSize)) # Memory Size Plus
1069     20% - Round Up
1070     if ($Memory -lt "4") {$NewPageFileSize = "4"}
1071     [int]$NewPageFileSize = ($NewPageFileSize * 1000)
1072     [int]$InitialSize      = ($NewPageFileSize * 1)
1073     [int]$MaximumSize      = ($NewPageFileSize * 1)
1074
1075     ### Create New Page File
1076     ### -----
1077     [scriptblock]$CreatePageFile =
1078     {
1079         if(-NOT(Get-CimInstance -ClassName Win32_PageFileSetting))
1080         {
1081             $PageFileName = $PageFileLocation + ":\pagefile.sys"
1082             Write-Host "Creating New Page File: PageFileLocation: $PageFileName
1083             InitialSize: $InitialSize MaximumSize: $MaximumSize "
1084             New-CimInstance -ClassName Win32_PageFileSetting -Property @{ Name=
1085             $PageFileName } | Out-Null
1086             Get-CimInstance -ClassName Win32_PageFileSetting | Set-CimInstance
1087             -Property @{InitialSize = $InitialSize; MaximumSize = $MaximumSize;} |
1088             Out-Null
1089         }
1090     }
1091
1092     ### Check Available Disk Space
1093     ### -----
1094     $FreeSpace = (Get-PSDrive $PageFileLocation).Free
1095
1096     #[int]$FreeSpace = $FreeSpace
1097
1098     if($FreeSpace -gt $NewPageFileSize)
1099     {
1100         Write-Host "Free Space Available. Drive: $Drive FreeSpace: $FreeSpace
1101         NewPageFileSize: $NewPageFileSize"
1102         Invoke-Command -ScriptBlock $CreatePageFile
1103     }
1104     elseif($FreeSpace -le $NewPageFileSize)

```

```

1098     {
1099         Write-Host "Not Enough Avialable Space. Drive: $Drive FreeSpace: $FreeSpace
NewPageFileSize: $NewPageFileSize `r`nNot Configuring!"
1100     }
1101
1102
1103 }
1104
1105 Configure-DesiredState -GetCurrentState $GetCurrentState -SetDesiredState
$SetDesiredState -ConfigurationMode $ConfigurationMode -AbortTrigger $AbortTrigger
1106 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1107 }
1108
1109 function Rename-ECI.EMI.Configure.OS.GuestComputer
1110 {
1111
1112     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
"EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
1113
1114     #####
1115     ### DESIRED STATE PARAMETERS
1116     #####
1117     $PropertyName      = "HostName"
1118     $DesiredState      = $HostName
1119     $ConfigurationMode = "Configure" ### Report - Configure
1120     $AbortTrigger      = $True   ### $True - $False
1121
1122     #####
1123     ### GET CURRENT CONFIGURATION STATE:
1124     #####
1125     [scriptblock]$script:GetCurrentState =
1126     {
1127         $global:CurrentState = (Get-ItemProperty -Path
HKLM:\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName).ComputerName
1128
1129         # $ActiveComputerName = (Get-ItemProperty -Path
HKLM:\SYSTEM\CurrentControlSet\Control\ComputerName\ActiveComputerName).ComputerN
ame
1130         # $ComputerName = (Get-ItemProperty -Path
HKLM:\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName).ComputerName
1131
1132     }
1133
1134     #####
1135     ### SET DESIRED-STATE:
1136     #####
1137     [scriptblock]$script:SetDesiredState =
1138     {
1139         #Rename-Computer -ComputerName $(Get-CIMInstance CIM_ComputerSystem).Name
-NewName $HostName
1140         #Rename-Computer -ComputerName $(Get-WmiObject Win32_Computersystem).Name
-NewName $HostName
1141         Rename-Computer -ComputerName . -NewName $HostName
1142     }
1143
1144     #####
1145     ### CALL CONFIGURE DESIRED STATE:
1146     #####
1147     ###-----
1148     $Params = @{}
1149     ServerID      = $ServerID
1150     HostName      = $HostName
1151     FunctionName  = $FunctionName
1152     PropertyName  = $PropertyName
1153     DesiredState  = $DesiredState
1154     GetCurrentState = $GetCurrentState
1155     SetDesiredState = $SetDesiredState
1156     ConfigurationMode = $ConfigurationMode
1157     AbortTrigger  = $AbortTrigger

```

```

1158     }
1159     Configure-DesiredState @Params
1160     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1161 }
1162
1163 function Restart-ECI.EMI.Configure.OS.GuestComputer
1164 {
1165     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
1166     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
1167
1168     $t = 10
1169
1170     Write-Host "Restarting Guest OS in $t seconds . . . "
1171     Shutdown /r -t $t
1172
1173     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1174 }
1175
1176 function Configure-ECI.EMI.Configure.OS.JoinDomain
1177 {
1178     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
1179     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
1180
1181     #####
1182     ### DESIRED STATE PARAMETERS
1183     #####
1184     $PropertyName      = "ClientDomain"
1185     $DesiredState       = $ClientDomain
1186     $ConfigurationMode = "Configure" ### Report - Configure
1187     $AbortTrigger       = $False    ### $True - $False
1188
1189     #####
1190     ### GET CURRENT CONFIGURATION STATE:
1191     #####
1192     [scriptblock]$script:GetCurrentState =
1193     {
1194         $global:CurrentState = (Get-CimInstance -ClassName Win32_ComputerSystem).Domain
1195     }
1196
1197     #####
1198     ### SET DESIRED-STATE:
1199     #####
1200     [scriptblock]$script:SetDesiredState =
1201     {
1202         $AdministrativePassword = ConvertTo-SecureString $AdministrativePassword
1203         -AsPlainText -Force
1204         $PSCredentials = New-Object System.Management.Automation.PSCredential
1205         ($AdministrativeUserName, $AdministrativePassword)
1206
1207         #OUPath = "OU=Servers,OU=London,DC=ercolanomgmt,DC=corp"
1208         #Add-Computer -ComputerName $HostName -DomainName $ClientDomain -OUPath $OUPath
1209         -Credential $PSCredentials -Force -Verbose
1210
1211         Add-Computer -ComputerName $HostName -DomainName $ClientDomain -Credential
1212         $PSCredentials -Force -Verbose
1213     }
1214
1215     #####
1216     ### CALL CONFIGURE DESIRED STATE:
1217     #####
1218     ###-----
1219     ### Test Variables passed from #Variable# substitution
1220     $Params = @{
1221         ServerID           = $ServerID
1222         HostName           = $HostName
1223         FunctionName       = $FunctionName
1224         PropertyName       = $PropertyName
1225         DesiredState       = $DesiredState
1226         GetCurrentState    = $GetCurrentState

```

```

1221         SetDesiredState      = $SetDesiredState
1222         ConfigurationMode     = $ConfigurationMode
1223         AbortTrigger          = $AbortTrigger
1224     }
1225     Configure-DesiredState @Params
1226     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
1227 }
1228
1229 function Restart-ECI.GuestOS
1230 {
1231     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
1232     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
1233
1234     $t = 30
1235     Write-Host "Restarting Guest OS on Server - $Hostname in $t seconds:"
1236     -ForegroundColor Magenta
1237     Start-Sleep -Seconds $t
1238     Restart-Computer -ComputerName . -Force #-Wait -For PowerShell -Timeout 300 -Delay
1239     $t -Verbose
1240 }
1241
1242 function
1243 Rename-LocalAdministrator-test
1244
1245 #
1246 <----- delete me?????
1247 {
1248     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
1249     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
1250
1251     ### Use .NET to Find the Current Local Administrator Account
1252     Add-Type -AssemblyName System.DirectoryServices.AccountManagement
1253     $ComputerName = [System.Net.Dns]::GetHostName()
1254     $PrincipalContext = New-Object
1255     System.DirectoryServices.AccountManagement.PrincipalContext([System.DirectoryServ
1256     ices.AccountManagement.ContextType]::Machine, $ComputerName)
1257     $UserPrincipal = New-Object
1258     System.DirectoryServices.AccountManagement.UserPrincipal($PrincipalContext)
1259     $Searcher = New-Object
1260     System.DirectoryServices.AccountManagement.PrincipalSearcher
1261     $Searcher.QueryFilter = $UserPrincipal
1262
1263     ### The Administrator account is the only account that has a SID that ends with
1264     "-500"
1265     $Account = $Searcher.FindAll() | Where-Object {$_.Sid -Like "*-500"}
1266     $script:CurrentAdminName = $Account.Name
1267
1268     Write-Host "1CurrentAdminName           : " $CurrentAdminName
1269     Write-Host "1LocalAdministrator           : " $LocalAdministrator
1270     Write-Host "1PreConfig_LocalAdminAccount   : " $PreConfig_LocalAdminAccount
1271     Write-Host "1PostConfig_LocalAdminAccount  : " $PostConfig_LocalAdminAccount
1272
1273     ### Check if Local Admin is already renamed
1274     if($CurrentAdminName -eq $LocalAdministrator)
1275     {
1276         #Write-Host "Local Admin Names are the same - CurrentAdminName:
1277         $CurrentAdminName NewAdminName: $LocalAdministrator"
1278         # $RebootRequired = $False
1279     }
1280     elseif($CurrentAdminName -ne $NewLocalAdminName)
1281     {
1282         # $RebootRequired = $True
1283         #Write-Host "Renaming Local Admin Account: Current Admin: $CurrentAdminName
1284         New Admin: $LocalAdministrator"
1285         Rename-LocalUser -Name $CurrentAdminName -NewName $LocalAdministrator
1286         -ErrorAction SilentlyContinue | Out-Null
1287     }
1288 }

```

```
1275     }
1276     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor
    DarkGray
1277 }
1278
1279 function Configure-ECI.EMI.Configure.OS.RegisterDNS
1280 {
1281     Ipconfig /registerdns
1282 }
1283
```