

```

1 #####
2 ### ECI EMI Server Management Module
3 ### ECI.EMI.VM.Mgmt.psml
4 #####
5
6
7 function Write-ServerMgmtRequesttoSQL
8 {
9     Param(
10         [Parameter(Mandatory = $True)][string]$VMName,
11         [Parameter(Mandatory = $True)][string]$vCenter,
12         [Parameter(Mandatory = $True)][string]$VMUUID,
13         [Parameter(Mandatory = $True)][string]$VMID,
14         [Parameter(Mandatory = $True)][string]$ServerMgmtOperation,
15         [Parameter(Mandatory = $True)][string]$ServerMgmtValue
16     )
17
18     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
19     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
20
21     Write-Host "Writing ServerMgmtRequest for VMName: $VMName" -ForegroundColor Cyan
22     $Query = "INSERT INTO
23     ServerMgmtRequest (VMName,vCenter,VMUUID,VMID,ServerMgmtOperation,ServerMgmtValue)
24     VALUES ('$VMName','$vCenter','$VMUUID','$VMID','$ServerMgmtOperation','$ServerMgmtValue') "
25
26     ### Open Database Connection
27     $Connection = New-Object System.Data.SqlClient.SqlConnection
28     $ConnectionString = $DevOps_DBConnectionString
29     $Connection.ConnectionString = $ConnectionString
30     $Connection.Open()
31     ### Insert Row
32     $cmd = New-Object System.Data.SqlClient.SqlCommand
33     $cmd.Connection = $Connection
34
35     $cmd.CommandText = $Query
36     $cmd.ExecuteNonQuery() #| Out-Null
37     $connection.Close()
38
39     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
40 }
41
42 function Get-ServerManagementRequestID
43 {
44     Param(
45         [Parameter(Mandatory = $True)][string]$VMName,
46         [Parameter(Mandatory = $True)][string]$vCenter,
47         [Parameter(Mandatory = $True)][string]$VMUUID,
48         [Parameter(Mandatory = $True)][string]$VMID
49     )
50
51     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
52     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
53
54     Write-Host "Getting ServerMgmtRequestID for HostName: $VMName" -ForegroundColor Cyan
55
56     $Query = "Select MAX(ServerMgmtRequestID) AS ServerMgmtRequestID FROM
57     ServerMgmtRequest WHERE VMName = '$VMName' AND vCenter = '$vCenter' AND VMUUID =
58     '$VMUUID' AND VMID = '$VMID' "
59
60     ### Execute DB Query
61     $connection = New-Object System.Data.SqlClient.SqlConnection
62     $ConnectionString = $DevOps_DBConnectionString
63     $connection.ConnectionString = $ConnectionString
64     $connection.Open()
65     $command = $connection.CreateCommand()
66     $command.CommandText = $Query
67     $result = $command.ExecuteReader()
68     $Datatable = new-object "System.Data.DataTable"

```

```

63 $Datatable.Load($result)
64 $Datatable | FT
65 $connection.Close()
66
67 if($Datatable.Rows.Count -eq 1)
68 {
69     $Column = $Datatable | Get-Member -MemberType Property,NoteProperty |
70     ForEach-Object {$_.Name} | Sort-Object -Property Name
71     $global:ServerMgmtRequestID = $Datatable.$Column
72 }
73 elseif($Datatable.Rows.Count -gt 1)
74 {
75     Write-Error -Message "ECI.ERROR: Too many Records Returned!" -ErrorAction
76     Continue
77     Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
78 }
79 elseif($Datatable.Rows.Count -eq 0)
80 {
81     Write-Error -Message "ECI.ERROR: No Records Found Matching Query!" -ErrorAction
82     Continue
83     Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
84 }
85
86 $ServerMgmtRequestID = $Datatable.$Column
87
88 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
89
90 # $global:ServerMgmtRequestID = $ServerMgmtRequestID
91 Return $ServerMgmtRequestID
92 }
93
94 function Identify-ECI.EMI.Automation.VM
95 {
96     Param(
97         [Parameter(Mandatory = $True)][string]$VMName,
98         [Parameter(Mandatory = $True)][string]$vCenter,
99         [Parameter(Mandatory = $True)][string]$VMUUID,
100         [Parameter(Mandatory = $True)][string]$VMID
101     )
102
103     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
104     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
105
106     Write-Host "Identifying VM: " $VMName -ForegroundColor Cyan
107
108     ### Does VM Exist?
109     ###-----
110     $VM = Get-VM -Name $VMName -ErrorAction SilentlyContinue ###<--- Use -ErrorAction
111     SilentlyContinue because VM may not exist.
112     if($VM)
113     {
114         $VMExists = $True
115         Write-Host `r`n('-' * 50)`r`n"VMExist: " $VMExists `r`n('-' * 50)`r`n
116         -ForegroundColor Green
117
118         ### Get Unique VM Values
119         Write-Host "Verifying VM Unique Identifiers:" -ForegroundColor DarkGray
120
121         $verifyVMName = $VM.Name
122         $verifyVMUUID = $VM | %{(Get-View $_.Id).config.uuid}
123         $verifyVMID = $VM.ID
124         $verifyvCenter = (($VM.UID).split("@")[1]).split(":")[0]
125
126         Write-Host "verifyVMName : " $verifyVMName -ForegroundColor DarkCyan
127         Write-Host "verifyVMUUID : " $verifyVMUUID -ForegroundColor DarkCyan
128         Write-Host "verifyVMID : " $verifyVMID -ForegroundColor DarkCyan
129         Write-Host "verifyvCenter : " $verifyvCenter -ForegroundColor DarkCyan

```

```

126
127     ### -----
128     ### Identify Unique VM
129     ### -----
130     if( ($VMName -eq $verifyVMName) -AND ($vCenter -eq $verifyvCenter) -AND
131         ($VMUUID -eq $verifyVMUUID) -AND ($VMID -eq $verifyVMID) )
132     {
133         $VMisUnique = $True
134         $UniqueColor = "Green"
135     }
136     else
137     {
138         $VMisUnique = $False
139         $UniqueColor = "Red"
140         Write-Error -Message "ECI.ERROR: VMName is not Unique." -ErrorAction
141         Continue -ErrorVariable +ECIError
142         Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
143     }
144     Write-Host `r`n('-' * 50)`r`n"VMisUnique: " $VMisUnique `r`n('-' * 50)`r`n
145     -ForegroundColor $UniqueColor
146 }
147 elseif(!$VM)
148 {
149     $VMEExists = $False
150     $VMisUnique = $False
151     Write-Error -Message "ECI.ERROR: VMName does not exist." -ErrorAction Continue
152     -ErrorVariable +ECIError
153     Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
154 }
155 $global:VMisUnique = $VMisUnique
156
157 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
158
159 Return $VMisUnique
160 }
161
162 function Get-ECI.EMI.Automation.VM.GuestState
163 {
164     Param(
165         [Parameter(Mandatory = $True)][string]$VMName,
166         [Parameter(Mandatory = $True)][string]$vCenter,
167         [Parameter(Mandatory = $True)][string]$VMUUID,
168         [Parameter(Mandatory = $True)][string]$VMID
169     )
170
171     $FunctionName = $(Get-PSCallStack)[0].Command; Write-Host `r`n('-' * 50)`r`n
172     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
173
174     Write-Host "Getting VM Current State: " $VMName -ForegroundColor Cyan
175
176     ###-----
177     ### Guest Ready State
178     ###-----
179     $VM = (Get-VM -Name $VMName -ErrorAction SilentlyContinue)
180     $GuestState = @{
181         "PowerState" =
182             $VM.PowerState                                     ### <---- RETRUN:
183             PoweredOn/PoweredOff
184         "GuestState" =
185             $VM.ExtensionData.guest.guestState               ### <---- RETRUN:
186             running/notRunning
187         "GuestOperationsReady" =
188             $VM.ExtensionData.guest.guestOperationsReady     ### <---- RETRUN:
189             True/False
190         "GuestStateChangeSupported" =
191             $VM.ExtensionData.guest.guestStateChangeSupported ### <---- RETRUN:
192             True/False
193     }
194 }

```

```

181 $GuestState = New-Object -TypeName PSObject -Property $GuestState
182
183 if(($GuestState.PowerState -eq "PoweredOn") -AND ($GuestState.GuestState -eq
"running") -AND ($GuestState.GuestOperationsReady -eq $True) -AND
($GuestState.GuestStateChangeSupported -eq $True))
184 {
185     $GuestReady = $True
186     $Color = "Green"
187 }
188 else
189 {
190     $GuestReady = $False
191     $Color = "Red"
192     Write-Error -Message "ECI.ERROR: VM Guest State not ready." -ErrorAction
Continue -ErrorVariable +ECIError
193     Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
194 }
195
196 $GuestState | Add-Member @{ECIGuestReady = $GuestReady}
197 $global:GuestState = $GuestState
198 $global:GuestReady = $GuestReady
199
200 Write-Host `r`n"GuestState: " ($GuestState | Out-String) -ForegroundColor DarkCyan
201 Write-Host "ECIGuestReady: " $GuestReady -ForegroundColor $Color
202
203 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
204
205 Return $GuestReady
206 }
207
208
209 function PowerOn-ECI.EMI.VM.Mgmt.PowerState
210 {
211     Param(
212         [Parameter(Mandatory = $True)][string]$ServerMgmtRequestID,
213         [Parameter(Mandatory = $True)][string]$ServerMgmtOperation,
214         [Parameter(Mandatory = $True)][string]$ServerMgmtValue,
215         [Parameter(Mandatory = $True)][string]$VMName,
216         [Parameter(Mandatory = $True)][string]$vCenter,
217         [Parameter(Mandatory = $True)][string]$VMUUID,
218         [Parameter(Mandatory = $True)][string]$VMID
219     )
220
221     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
"EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
222
223     Write-Host "Powering On VM Guest." -ForegroundColor Cyan
224
225     $ECIServerMgt = @{
226         VMName = $VMName
227         vCenter = $vCenter
228         VMUUID = $VMUUID
229         VMID = $VMID
230         ServerMgmtRequestID = $ServerMgmtRequestID
231     }
232
233     ### VM Current Power State
234     $VMPowerState = (Get-VM -Name $VMName).PowerState
235     Write-Host "VM Current PowerState: " $VMPowerState -ForegroundColor DarkCyan
236
237     if($VMPowerState -eq "PoweredOn")
238     {
239         Write-Host "The VM is already Powered On." -ForegroundColor Yellow
240     }
241     elseif($VMPowerState -ne "PoweredOff")
242     {
243         Write-Host "The VM is Not in a Powered Off State." -ForegroundColor Red
244     }
245     elseif($VMPowerState -eq "PoweredOff")

```

```

246 {
247     ###-----
248     ### Retry Loop
249     ###-----
250     $Retries          = 3
251     $RetryCounter     = 0
252     $RetryTime        = 60
253     $RetryTimeIncrement = ($RetryTime * 2)
254     $Success          = $False
255
256     while($Success -ne $True)
257     {
258         try
259         {
260             #####
261             ### PowerOn VM
262             #####
263             Write-Host "Powering ON VM: " $VMName -ForegroundColor Yellow
264
265             if($VMName.count -eq 1 -and $VMName -isnot [system.array])
266             {
267                 Start-VM -VM $VMName | Out-Null
268             }
269             else
270             {
271                 Write-Error -Message "ECI.Error: VMName is not Unique."
272                 -ErrorAction Continue -ErrorVariable +ECIError
273             }
274
275             Start-ECI.EMI.Automation.Sleep -t $WaitTime_GuestOSRestart -Message
276             "Powering On VM"
277             if((Get-VM -Name $VMName -ErrorAction SilentlyContinue).PowerState -eq
278             "PoweredOn")
279             {
280                 $Success = $True
281                 Write-Host "$FunctionName - Succeeded: " $Success -ForegroundColor
282                 Green
283             }
284         }
285         catch
286         {
287             if($RetryCounter -ge $Retries)
288             {
289                 Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID
290                 $ServerMgmtRequestID
291                 Throw "ECI.THROW.TERMINATING.ERROR: Power On Operation Failed! "
292             }
293             else
294             {
295                 ### Retry x Times
296                 ###-----
297                 $RetryCounter++
298
299                 ### Write ECI Error Log
300                 ###-----
301                 Write-Error -Message ("ECI.RETRY: PowerOn") -ErrorAction Continue
302                 -ErrorVariable +ECIError
303                 if(-NOT(Test-Path -Path $ECIErrorLogFile)) {(New-Item -ItemType
304                 file -Path $ECIErrorLogFile -Force | Out-Null)}
305                 $ECIError | Out-File -FilePath $ECIErrorLogFile -Append -Force
306
307                 ### Error Handling Action
308                 ###-----
309                 Start-ECI.EMI.Automation.Sleep -Message "Retry Power On Operation."
310                 -t $RetryTime
311
312                 $RetryTime = $RetryTime + $RetryTimeIncrement
313             }
314         }
315     }
316 }

```

```

307     }
308 }
309 else
310 {
311     Write-Error -Message ("ECI.ERROR: The VM Powered State was not determined.")
312     -ErrorAction Continue -ErrorVariable +ECIError
313     Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
314 }
315
316 ### Verify Power State
317 ###-----
318 $VerifyVMPowerState = (Get-VM -Name $VMName).PowerState
319 Write-Host "Verified VM Power State: " $VerifyVMPowerState -ForegroundColor Magenta
320 if($VerifyVMPowerState -eq "PoweredOn")
321 {
322     $OperationVerified = $True
323     $OperationVerifiedColor = "Green"
324 }
325 else
326 {
327     $OperationVerified = $False
328     $OperationVerifiedColor = "Red"
329     Write-Error -Message ("ECI.ERROR: The VM Powered State was not determined.")
330     -ErrorAction Continue -ErrorVariable +ECIError
331     Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
332 }
333 Write-Host "Verified VM Power State: " $OperationVerified -ForegroundColor
334 $OperationVerifiedColor
335
336 ### Report Power State
337 ###-----
338 $ServerMgmtUpdate = @{
339     ServerMgmtRequestID = $ServerMgmtRequestID
340     VMName               = $VMName
341     vCenter              = $vCenter
342     VMUUID               = $VMUUID
343     VMID                 = $VMID
344     ServerMgmtOperation  = $ServerMgmtOperation
345     ServerMgmtValue      = $ServerMgmtValue
346     OperationVerified    = $OperationVerified
347 }
348 Update-ECI.EMI.VM.Mgmt.ServerMgmtOperations-SQL @ServerMgmtUpdate
349
350 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
351 }
352
353
354 function PowerOff-ECI.EMI.VM.Mgmt.PowerState
355 {
356     Param(
357         [Parameter(Mandatory = $True)][string]$ServerMgmtRequestID,
358         [Parameter(Mandatory = $True)][string]$ServerMgmtOperation,
359         [Parameter(Mandatory = $True)][string]$ServerMgmtValue,
360         [Parameter(Mandatory = $True)][string]$VMName,
361         [Parameter(Mandatory = $True)][string]$vCenter,
362         [Parameter(Mandatory = $True)][string]$VMUUID,
363         [Parameter(Mandatory = $True)][string]$VMID
364     )
365
366     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
367     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
368
369     Write-Host "Powering Off VM Guest." -ForegroundColor Cyan
370
371     ### VM Current Power State

```

```

372 $VMPowerState = (Get-VM -Name $VMName).PowerState
373 Write-Host "VM Current PowerState: " $VMPowerState -ForegroundColor DarkCyan
374
375 if($VMPowerState -eq "Poweredff")
376 {
377     Write-Host "The VM is ia already Powerd off." -ForegroundColor Yellow
378 }
379 elseif($VMPowerState -ne "PoweredOn")
380 {
381     Write-Host "The VM is Not in a Powered On State." -ForegroundColor Red
382 }
383 elseif($VMPowerState -eq "PoweredOn")
384 {
385     ### Is GuestState Ready?
386     $VMIdentity = @{
387         #ServerMgmtOperation = $ServerMgmtOperation
388         VMName           = $VMName
389         vCenter           = $vCenter
390         VMUUID            = $VMUUID
391         VMID              = $VMID
392     }
393     Get-ECI.EMI.Automation.VM.GuestState @VMIdentity
394
395     if($GuestReady -eq $True)
396     {
397         ###-----
398         ### Retry Loop
399         ###-----
400         $Retries           = 3
401         $RetryCounter      = 0
402         $RetryTime         = 60
403         $RetryTimeIncrement = ($RetryTime * 1.5)
404         $Success           = $False
405
406         while($Success -ne $True)
407         {
408             try
409             {
410                 ### Initiate Command
411                 ###-----
412                 Write-Host "Powering OFF VM: " $VMName -ForegroundColor Yellow
413
414                 #####
415                 ### Soft Shutdown VM
416                 #####
417
418                 ### Soft Shutdown
419                 Shutdown-VMGuest -VM $VMName -confirm:$false
420                 Start-ECI.EMI.Automation.Sleep -t $RetryTime -Message "Issuing
Shutdown Command: $RetryCounter"
421
422                 if((Get-VM -Name $VMName -ErrorAction SilentlyContinue).PowerState
-eq "PoweredOff")
423                 {
424                     $Success = $True
425                     Write-Host "$FunctionName - Succeeded: " $Success
-ForegroundColor Green
426                 }
427             }
428             catch
429             {
430                 if($RetryCounter -ge $Retries)
431                 {
432                     #####
433                     ### Hard Shutdown
434                     #####
435                     Stop-VM -VM $VMName -Confirm:$false
436
437                     ### Multiply Wait time by 4x

```

```

438 Start-ECI.EMI.Automation.Sleep -t ($RetryTime * 4.5) -Message
439 "Issuing Final Shutdown Command: $RetryCounter"
440
441 if((Get-VM -Name $VMName -ErrorAction
442 SilentlyContinue).PowerState -eq "PoweredOff")
443 {
444     $Success = $True
445     Write-Host "$FunctionName - Succeeded: " $Success
446     -ForegroundColor Green
447 }
448 elseif((Get-VM -Name $VMName -ErrorAction
449 SilentlyContinue).PowerState -ne "PoweredOff")
450 {
451     ### Wait 60 Minutes
452     Start-ECI.EMI.Automation.Sleep -t 3600 -Message "Issuing
453     Stop"
454     if((Get-VM -Name $VMName -ErrorAction
455     SilentlyContinue).PowerState -eq "PoweredOff")
456     {
457         $Success = $True
458         Write-Host "$FunctionName - Succeeded: " $Success
459         -ForegroundColor Green
460     }
461     elseif((Get-VM -Name $VMName -ErrorAction
462     SilentlyContinue).PowerState -ne "PoweredOff")
463     {
464         #####
465         ### Kill
466         #####
467         Stop-VM -VM $VMName -Kill -Confirm:$false
468         Start-ECI.EMI.Automation.Sleep -t 7200 -Message "Issued
469         Kill Command"
470         if((Get-VM -Name $VMName -ErrorAction
471         SilentlyContinue).PowerState -eq "PoweredOff")
472         {
473             $Success = $True
474             Write-Host "$FunctionName - Succeeded: " $Success
475             -ForegroundColor Green
476         }
477         elseif((Get-VM -Name $VMName -ErrorAction
478         SilentlyContinue).PowerState -ne "PoweredOff")
479         {
480             Write-Error -Message "ECI.ERROR: Power Off
481             Operation Failed!" -ErrorAction Continue
482             -ErrorVariable +ECIError
483             Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID
484             $ServerMgmtRequestID
485         }
486     }
487 }
488 else
489 {
490     ### Retry x Times
491     ###-----
492     $RetryCounter++
493
494     ### Write ECI Error Log
495     ###-----
496     Write-Error -Message ("ECI.RETRY: Retry PowerOff") -ErrorAction
497     Continue -ErrorVariable +ECIError
498     if(-NOT(Test-Path -Path $ECIErrorLogFile)) {(New-Item -ItemType
499     file -Path $ECIErrorLogFile -Force | Out-Null)}
500     $ECIError | Out-File -FilePath $ECIErrorLogFile -Append -Force
501
502     ### Error Handling Action
503     ###-----
504     Start-ECI.EMI.Automation.Sleep -Message "Retry Power Off
505     Operation." -t $RetryTime

```



```

489
490         $RetryTime = $RetryTime + $RetryTimeIncrement
491     }
492 }
493
494 }
495 else
496 {
497     Write-Error -Message ("ECI.ERROR: Guest State NOT Ready.") -ErrorAction
498     Continue -ErrorVariable +ECIError
499     Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
500 }
501 else
502 {
503     Write-Error -Message ("ECI.ERROR: The VM Powered State was not determined.")
504     -ErrorAction Continue -ErrorVariable +ECIError
505     Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
506 }
507
508 ### Verify Power State
509 ###-----
510 function Verify-ServerState
511 {
512     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
513     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
514
515     Write-Host "Verifying VM Power State." -ForegroundColor Cyan
516
517     $VerifyVMPowerState = (Get-VM -Name $VMName).PowerState
518
519     if($VerifyVMPowerState -eq "PoweredOff")
520     {
521         $OperationVerified = $True
522         $OperationVerifiedColor = "Green"
523     }
524     else
525     {
526         $OperationVerified = $False
527         $OperationVerifiedColor = "Red"
528         Write-Error -Message ("ECI.ERROR: The VM Powered State was not
529         determined.") -ErrorAction Continue -ErrorVariable +ECIError
530         Send-ECI.EMI.ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
531     }
532     Write-Host "VerifyVMPowerState : " $VerifyVMPowerState -ForegroundColor DarkCyan
533     Write-Host "OperationVerified : " $OperationVerified -ForegroundColor
534     $OperationVerifiedColor
535
536     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor
537     DarkGray
538
539     $global:OperationVerified = $OperationVerified
540     Return $OperationVerified
541 }
542
543 Verify-ServerState
544
545 ### Report Power State
546 ###-----
547 $ServerMgmtUpdate = @{
548     ServerMgmtRequestID = $ServerMgmtRequestID
549     VMName = $VMName
550     vCenter = $vCenter
551     VMUUID = $VMUUID
552     VMID = $VMID
553     ServerMgmtOperation = $ServerMgmtOperation
554     ServerMgmtValue = $ServerMgmtValue
555     OperationVerified = $OperationVerified

```

```

552     }
553     Update-ECI.EMI.VM.Mgmt.ServerMgmtOperations-SQL @ServerMgmtUpdate
554
555     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
556 }
557
558 function Update-ECI.EMI.VM.Mgmt.ServerMgmtOperations-SQL
559 {
560     Param(
561         [Parameter(Mandatory = $True)][string]$ServerMgmtRequestID,
562         [Parameter(Mandatory = $True)][string]$ServerMgmtOperation,
563         [Parameter(Mandatory = $True)][string]$ServerMgmtValue,
564         [Parameter(Mandatory = $True)][string]$VMName,
565         [Parameter(Mandatory = $True)][string]$vCenter,
566         [Parameter(Mandatory = $True)][string]$VMUUID,
567         [Parameter(Mandatory = $True)][string]$VMID,
568         [Parameter(Mandatory = $True)][string]$OperationVerified
569     )
570
571     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
572     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
573
574     $ServerMgmtFunction = $FunctionName
575     $Query = "INSERT INTO
576     ServerMgmtOperations (ServerMgmtRequestID,VMName,vCenter,VMUUID,VMID,ServerMgmtOperati
577     on,ServerMgmtValue,OperationVerified)
578     VALUES ('$ServerMgmtRequestID','$VMName','$vCenter','$VMUUID','$VMID','$ServerMgmtOper
579     ation','$ServerMgmtValue','$OperationVerified')"
580     $ConnectionString = $DevOps_DBConnectionString
581     $Connection = New-Object System.Data.SqlClient.SqlConnection
582     $Connection.ConnectionString = $ConnectionString
583     $Connection.Open()
584     $cmd = New-Object System.Data.SqlClient.SqlCommand
585     $cmd.Connection = $connection
586     $cmd.CommandText = $Query
587     $cmd.ExecuteNonQuery() | Out-Null
588     $connection.Close()
589
590     Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
591 }
592
593 function Send-ECI.EMI.ServerMgmtAlert
594 {
595     Param(
596         [Parameter(Mandatory = $True)][int]$ServerMgmtRequestID,
597         [Parameter(Mandatory = $False)][string]$Status,
598         [Parameter(Mandatory = $False)][string]$HostName
599     )
600
601     $FunctionName = $((Get-PSCallStack)[0].Command);Write-Host `r`n('-' * 50)`r`n
602     "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
603
604     ### Message Header
605     ###-----
606     $Message = $Null
607
608     $Header = "
609     <style>
610     BODY{font-family: Verdana, Arial, Helvetica, sans-serif;font-size:9;font-color:
611     #000000;text-align:left;}
612     TABLE {border-width: 0px; border-style: hidden; border-color: white;
613     border-collapse: collapse;}
614     TH {border-width: 0px; padding: 3px; border-style: hidden; border-color: white;
615     background-color: #6495ED;}
616     TD {border-width: 0px; padding: 3px; border-style: hidden; border-color: white;}
617
618     </style>
619     "

```

```

612
613 $Message += $Header
614 $Message += "<html><body>"
615
616
617 $Message += "<font size='3';color='gray'><i>ECI EMI Server Automation</i></font><br>"
618 $Message += "<font size='5';color='NAVY'><b>ECI Error Alert</b></font><br>"
619 $Message += "<font size='2'>Request Date:" + (Get-Date) + "</font>"
620 $Message += "<br><br><br><br>"
621 $Message += "<font size='3';color='black'>WARNING: This Server <b> COULD NOT </b>
be provisioned.</font>"
622 $Message += "<br><br>"
623 $Message += "<font size='3';color='red'><br><b>ERROR MESSAGE: </b></font><br>"
624
625
626 $Message += "<font size='3';color='red'>" + $Error[0] + " </font>"
627 # $Message += "<font size='3';color='red'>" + $ErrorMsg + " </font>"
628
629 $Message += "<br><br><br>"
630
631 $Message += "<table>"
632 $Message += "<tr>"
633 $Message += "<td align='right'>" + "Status : </td>"
634 $Message += "<td align='left'><font size='4';color='$black'>" + $Status +
"</font></td>"
635 $Message += "</tr>"
636 $Message += "<tr>"
637 $Message += "<td align='right'>" + "HostName : </td>"
638 $Message += "<td align='left'><font size='4';color='black'>" + $VMName +
"</font></td>"
639 $Message += "</tr>"
640 $Message += "</table>"
641 $Message += "<br>"
642 $Message += "FOR ECI INTERNAL USE ONLY." + "`r`n" +
"<br><br>"
643
644
645 ### Message Status
646 ###-----
647 $Message += "-----"
+ "`r`n" + "<br>"
648 $Message += "<b>SERVER PROVISIONING STATUS: "
+ "`r`n" + "</b><br>"
649 $Message += "-----"
+ "`r`n" + "<br>"
650
651 ### Display Desired State SQL Record
652 ###-----
653 $Message += "<font size='3';><b>SERVER CONFIGURATION STATE:</b>" +
"</font><br>"
654 $Header = "
<style>
BODY{font-family: Verdana, Arial, Helvetica, sans-serif;font-size:9;font-color:
#000000;text-align:left;}
TABLE {border-width: 1px; border-style: solid; border-color: black;
border-collapse: collapse;}
</style>
"
655
656 $Message += $Header
657
658 $DataSetName = "DesiredState"
659 $ConnectionString = "Server=automatel.database.windows.net;Initial
Catalog=DevOps;User ID=devops;Password=JKFLKA8899*(*(32faiuynv;"
660 $Query = "SELECT * FROM ServermgmtRequest WHERE ServerMgmtRequestID =
'$ServerMgmtRequestID'"
661
662 $Connection = New-Object System.Data.SqlClient.SqlConnection
663 $Connection.ConnectionString = $ConnectionString
664 $Connection.Open()
665 $Command = New-Object System.Data.SqlClient.SqlCommand

```

```

669 $Command.Connection = $Connection
670 $Command.CommandText = $Query
671 $Reader = $Command.ExecuteReader()
672 $DataTable = New-Object System.Data.DataTable
673 $DataTable.Load($Reader)
674 $dt = $DataTable
675 $dt | ft
676
677 $Message += "<table>"
678 $Message += "<tr>"
679 for($i = 0;$i -lt $dt.Columns.Count;$i++)
680 {
681     $Message += "<b><u><td>"+$dt.Columns[$i].ColumnName+"</td></u></b>"
682 }
683 $Message += "</tr>"
684
685 for($i=0;$i -lt $dt.Rows.Count; $i++)
686 {
687     $Message += "<tr>"
688     for($j=0; $j -lt $dt.Columns.Count; $j++)
689     {
690         $Message += "<td>"+$dt.Rows[$i][$j].ToString()+"</td>"
691     }
692     $Message += "</tr>"
693 }
694
695 $Message += "</table></body></html>"
696 $Message += "<br><br>"
697
698 ### Transcript Log
699 ###-----
700 $TranscriptURL = "cloud-portal01.eci.cloud/vmautomationlogs/Transcripts/" +
701 (Split-Path $TranscriptFile -Leaf)
702 $Message += "Transcript Log: " + "<a href=http://" + $TranscriptURL + ">" +
703 $TranscriptURL + "</a>"
704 $Message += "<br><br>"
705 $Message += "Server Build Date: " + (Get-Date)
706 $Message += "<br>"
707
708 ### Close Message
709 ###-----
710 $Message += "</body></html>"
711
712 ### Email Constants
713 ###-----
714 $From = "cbrennan@eci.com"
715 $To = "cbrennan@eci.com,sdesimone@eci.com,wercolano@eci.com,rgee@eci.com"
716 $To = "cbrennan@eci.com"
717 $SMTP = "alertmx.eci.com"
718 $SMTP = $SMTPServer
719 $Subject = "SERVER PROVISIONING STATUS: " + $Status
720
721 ### Email Message
722 ###-----
723 Write-Host `r`n`r`n`r`n("=" * 50)`n"SENDING ALERT:" $Status`r`n("=" * 50)`r`n`r`n
724 -ForegroundColor Yellow
725
726 #Write-Host `n "MESSAGE: " $Message `n -ForegroundColor $StatusColor
727 Write-Host "TO: " $To
728 Send-MailMessage -To ($To -split ",") -From $From -Body $Message -Subject $Subject
729 -BodyAsHtml -SmtpServer $SMTP
730
731 Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
732
733 }

```

734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752