```powershell
#########################################################
Server Management Module
VM.Mgmt.psm1
#########################################################


function Write-ServerMgmtRequesttoSQL
{
    Param(
        [Parameter(Mandatory = $True)][string]$VMName,
        [Parameter(Mandatory = $True)][string]$vCenter,
        [Parameter(Mandatory = $True)][string]$VMUUID,
        [Parameter(Mandatory = $True)][string]$VMID,
        [Parameter(Mandatory = $True)][string]$ServerMgmtOperation,
        [Parameter(Mandatory = $True)][string]$ServerMgmtValue
    )

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

    Write-Host "Writing ServerMgmtRequest for VMName: $VMName" -ForegroundColor Cyan
    $Query = "INSERT INTO
    ServerMgmtRequest(VMName,vCenter,VMUUID,VMID,ServerMgmtOperation,ServerMgmtValue)
    VALUES('$VMName','$vCenter','$VMUUID','$VMID','$ServerMgmtOperation','$ServerMgmtValu
    e')"

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

    $cmd.CommandText = $Query
    $cmd.ExecuteNonQuery() #| Out-Null
    $connection.Close()

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

function Get-ServerManagementRequestID
{
    Param(
        [Parameter(Mandatory = $True)][string]$VMName,
        [Parameter(Mandatory = $True)][string]$vCenter,
        [Parameter(Mandatory = $True)][string]$VMUUID,
        [Parameter(Mandatory = $True)][string]$VMID
    )

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

    Write-Host "Getting ServerMgmtRequestID for HostName: $VMName" -ForegroundColor Cyan

    $Query = "Select MAX(ServerMgmtRequestID) AS ServerMgmtRequestID FROM
    ServerMgmtRequest WHERE VMName = '$VMName' AND vCenter = '$vCenter' AND VMUUID =
    '$VMUUID' AND VMID = '$VMID' "

    ### Execute DB Query
    $connection = New-Object System.Data.SqlClient.SqlConnection
    $ConnectionString = $DevOps_DBConnectionString
    $connection.ConnectionString = $ConnectionString
    $connection.Open()
    $command = $connection.CreateCommand()
    $command.CommandText = $query
    $result = $command.ExecuteReader()
    $Datatable = new-object "System.Data.DataTable"
```

```powershell
    $Datatable.Load($result)
    $Datatable | FT
    $connection.Close()

    if($Datatable.Rows.Count -eq 1)
    {
        $Column = $Datatable | Get-Member -MemberType Property,NoteProperty | ForEach-Object {$_.Name} | Sort-Object -Property Name
        $global:ServerMgmtRequestID = $Datatable.$Column

    }
    elseif($Datatable.Rows.Count -gt 1)
    {
        Write-Error -Message "ERROR: Too many Records Returned!" -ErrorAction Continue
        Send-ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID

    }
    elseif($Datatable.Rows.Count -eq 0)
    {
        Write-Error -Message "ERROR: No Records Found Matching Query!" -ErrorAction Continue
        Send-ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
    }

    $ServerMgmtRequestID = $Datatable.$Column

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

    #$global:ServerMgmtRequestID = $ServerMgmtRequestID
    Return $ServerMgmtRequestID
}

function Identify-Automation.VM
{
    Param(
        [Parameter(Mandatory = $True)][string]$VMName,
        [Parameter(Mandatory = $True)][string]$vCenter,
        [Parameter(Mandatory = $True)][string]$VMUUID,
        [Parameter(Mandatory = $True)][string]$VMID
    )

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

    Write-Host "Idenifying VM: " $VMName -ForegroundColor Cyan

    ### Does VM Exist?
    ###------------------
    $VM = Get-VM -Name $VMName -ErrorAction SilentlyContinue   ###<--- Use -ErrorAction SilentlyContinue because VM may not exist.
    if($VM)
    {
        $VMExists = $True
        Write-Host `r`n('-' * 50)`r`n"VMExist: " $VMExists `r`n('-' * 50)`r`n -ForegroundColor Green

        ### Get Unique VM Values
        Write-Host "Verifying VM Unique Identifiers:" -ForegroundColor DarkGray

        $verifyVMName       = $VM.Name
        $verifyVMUUID       = $VM | %{(Get-View $_.Id).config.uuid}
        $verifyVMID         = $VM.ID
        $verifyvCenter      = (($VM.UID).split("@")[1]).split(":")[0]

        Write-Host "verifyVMName       : " $verifyVMName       -ForegroundColor DarkCyan
        Write-Host "verifyVMUUID       : " $verifyVMUUID       -ForegroundColor DarkCyan
        Write-Host "verifyVMID         : " $verifyVMID         -ForegroundColor DarkCyan
        Write-Host "verifyvCenter      : " $verifyvCenter      -ForegroundColor DarkCyan

```

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

179             "GuestStateChangeSupported"  =
                $VM.ExtensionData.guest.guestStateChangeSupported   ### <---- RETRUN:
                True/False
180         }
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 "ERROR: VM Guest State not ready." -ErrorAction Continue
                -ErrorVariable +Error
193             Send-ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
194         }
195
196         $GuestState | Add-Member @{GuestReady = $GuestReady}
197         $global:GuestState    = $GuestState
198         $global:GuestReady = $GuestReady
199
200         Write-Host `r`n"GuestState: " ($GuestState | Out-String) -ForegroundColor DarkCyan
201         Write-Host "GuestReady: " $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-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         $ServerMgt = @{
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 "Error: VMName is not Unique." -ErrorAction
                    Continue -ErrorVariable +Error
272                 }
273
274                 Start-Automation.Sleep -t $WaitTime_GuestOSRestart -Message "Powering
                On VM"
275                 if((Get-VM -Name $VMName -ErrorAction SilentlyContinue).PowerState -eq
                "PoweredOn")
276                 {
277                     $Success = $True
278                     Write-Host "$FunctionName - Succeded: " $Success -ForegroundColor
                    Green
279                 }
280             }
281             catch
282             {
283                 if($RetryCounter -ge $Retries)
284                 {
285                     Send-ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
286                     Throw "THROW.TERMINATING.ERROR: Power On Operation Failed! "
287                 }
288                 else
289                 {
290                     ### Retry x Times
291                     ###--------------
292                     $RetryCounter++
293
294                     ### Write  Error Log
295                     ###--------------------------------
296                     Write-Error -Message ("RETRY: PowerOn") -ErrorAction Continue
                    -ErrorVariable +Error
297                     if(-NOT(Test-Path -Path $ErrorLogFile)) {(New-Item -ItemType file
                    -Path $ErrorLogFile -Force | Out-Null)}
298                     $Error | Out-File -FilePath $ErrorLogFile -Append -Force
299
300                     ### Error Handling Action
301                     ###--------------------------------
302                     Start-Automation.Sleep -Message "Retry Power On Operation." -t
                    $RetryTime
303
304                     $RetryTime = $RetryTime + $RetryTimeIncrement
305                 }
306             }
307         }
308     }
```

```
309         else
310         {
311             Write-Error -Message ("ERROR: The VM Powered State was not determined.")
                -ErrorAction Continue -ErrorVariable +Error
312             Send-ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
313         }
314
315         ### Verify Power State
316         ###-------------------------------
317         $VerifyVMPowerState = (Get-VM -Name $VMName).PowerState
318         Write-Host "Verified VM Power State: " $VerifyVMPowerState -ForegroundColor Magenta
319         if($VerifyVMPowerState -eq "PoweredOn")
320         {
321             $OperationVerified = $True
322             $OperationVerifiedColor = "Green"
323         }
324         else
325         {
326             $OperationVerified = $False
327             $OperationVerifiedColor = "Red"
328             Write-Error -Message ("ERROR: The VM Powered State was not determined.")
                -ErrorAction Continue -ErrorVariable +Error
329             Send-ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
330         }
331         Write-Host "Verified VM Power State: " $OperationVerified -ForegroundColor
            $OperationVerifiedColor
332
333
334         ### Report Power State
335         ###-------------------------------
336         $ServerMgmtUpdate = @{
337             ServerMgmtRequestID  = $ServerMgmtRequestID
338             VMName               = $VMName
339             vCenter              = $vCenter
340             VMUUID               = $VMUUID
341             VMID                 = $VMID
342             ServerMgmtOperation  = $ServerMgmtOperation
343             ServerMgmtValue      = $ServerMgmtValue
344             OperationVerified    = $OperationVerified
345
346         }
347         Update-VM.Mgmt.ServerMgmtOperations-SQL @ServerMgmtUpdate
348
349
350         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
351     }
352
353
354     function PowerOff-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
            "EXECUTING FUNCTION: " $FunctionName `r`n('-' * 50) -ForegroundColor Gray
367
368         Write-Host "Powering Off VM Guest." -ForegroundColor Cyan
369
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        = $VMID
392             }
393             Get-Automation.VM.GuestState @VMIdentity
394
395             if($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-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 - Succeded: " $Success
                                -ForegroundColor Green
426                         }
427                     }
428                     catch
429                     {
430                         if($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-Automation.Sleep -t ($RetryTime * 4.5) -Message "Issuing
                                Final Shutdown Command: $RetryCounter"
```

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

```
            }
        }
        else
        {
            Write-Error -Message ("ERROR: Guest State NOT Ready.") -ErrorAction
            Continue -ErrorVariable +Error
            Send-ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
        }
    }
    else
    {
        Write-Error -Message ("ERROR: The VM Powered State was not determined.")
        -ErrorAction Continue -ErrorVariable +Error
        Send-ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
    }

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

        Write-Host "Verifying VM Power State." -ForegroundColor Cyan

        $VerifyVMPowerState = (Get-VM -Name $VMName).PowerState

        if($VerifyVMPowerState -eq "PoweredOff")
        {
            $OperationVerified = $True
            $OperationVerifiedColor = "Green"
        }
        else
        {
            $OperationVerified = $False
            $OperationVerifiedColor = "Red"
            Write-Error -Message ("ERROR: The VM Powered State was not determined.")
            -ErrorAction Continue -ErrorVariable +Error
            Send-ServerMgmtAlert -ServerMgmtRequestID $ServerMgmtRequestID
        }
        Write-Host "VerifyVMPowerState : " $VerifyVMPowerState -ForegroundColor DarkCyan
        Write-Host "OperationVerified  : " $OperationVerified -ForegroundColor
        $OperationVerifiedColor

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

        $global:OperationVerified = $OperationVerified
        Return $OperationVerified

    }
    Verify-ServerState

    ### Report Power State
    ###-----------------------------------
    $ServerMgmtUpdate = @{
        ServerMgmtRequestID  = $ServerMgmtRequestID
        VMName               = $VMName
        vCenter              = $vCenter
        VMUUID               = $VMUUID
        VMID                 = $VMID
        ServerMgmtOperation  = $ServerMgmtOperation
        ServerMgmtValue      = $ServerMgmtValue
        OperationVerified    = $OperationVerified

    }
    Update-VM.Mgmt.ServerMgmtOperations-SQL @ServerMgmtUpdate

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

```powershell
}

function Update-VM.Mgmt.ServerMgmtOperations-SQL
{
    Param(
        [Parameter(Mandatory = $True)][string]$ServerMgmtRequestID,
        [Parameter(Mandatory = $True)][string]$ServerMgmtOperation,
        [Parameter(Mandatory = $True)][string]$ServerMgmtValue,
        [Parameter(Mandatory = $True)][string]$VMName,
        [Parameter(Mandatory = $True)][string]$vCenter,
        [Parameter(Mandatory = $True)][string]$VMUUID,
        [Parameter(Mandatory = $True)][string]$VMID,
        [Parameter(Mandatory = $True)][string]$OperationVerified
    )

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

    $ServerMgmtFunction = $FunctionName
    $Query = "INSERT INTO
    ServerMgmtOperations(ServerMgmtRequestID,VMName,vCenter,VMUUID,VMID,ServerMgmtOperati
    on,ServerMgmtValue,OperationVerified)
    VALUES('$ServerMgmtRequestID','$VMName','$vCenter','$VMUUID','$VMID','$ServerMgmtOper
    ation','$ServerMgmtValue','$OperationVerified')"
    $ConnectionString  = $DevOps_DBConnectionString
    $Connection = New-Object System.Data.SQLClient.SQLConnection
    $Connection.ConnectionString = $ConnectionString
    $Connection.Open()
    $cmd = New-Object System.Data.SqlClient.SqlCommand
    $cmd.Connection = $connection
    $cmd.CommandText = $Query
    $cmd.ExecuteNonQuery() | Out-Null
    $connection.Close()

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


function Send-ServerMgmtAlert
{
    Param(
    [Parameter(Mandatory = $True)][int]$ServerMgmtRequestID,
    [Parameter(Mandatory = $False)][string]$Status,
    [Parameter(Mandatory = $False)][string]$HostName
    )

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

    ### Message Header
    ###-------------------------------------------------------------------------
    $Message = $Null

    $Header = "
        <style>
        BODY{font-family: Verdana, Arial, Helvetica, sans-serif;font-size:9;font-color:
        #000000;text-align:left;}
        TABLE {border-width: 0px; border-style: hidden; border-color: white;
        border-collapse: collapse;}
        TH {border-width: 0px; padding: 3px; border-style: hidden; border-color: white;
        background-color: #6495ED;}
        TD {border-width: 0px; padding: 3px; border-style: hidden; border-color: white;}

        </style>
    "

    $Message += $Header
    $Message += "<html><body>"

```

```
616
617         $Message += "<font size='3';color='gray'><i>  Server Automation</i></font><br>"
618         $Message += "<font size='5';color='NAVY'><b> 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  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 = "
655             <style>
656             BODY{font-family: Verdana, Arial, Helvetica, sans-serif;font-size:9;font-color:
                #000000;text-align:left;}
657             TABLE {border-width: 1px; border-style: solid; border-color: black;
                border-collapse: collapse;}
658             </style>
659         "
660         $Message += $Header
661
662         $DataSetName = "DesiredState"
663         $ConnectionString = "Server=automate1.database.windows.net;Initial
            Catalog=DevOps;User ID=devops;Password=JKFLKA8899*(*(32faiuynv;"
664         $Query = "SELECT * FROM ServermgmtRequest WHERE ServerMgmtRequestID =
            '$ServerMgmtRequestID'"
665         $Connection = New-Object System.Data.SQLClient.SQLConnection
666         $Connection.ConnectionString = $ConnectionString
667         $Connection.Open()
668         $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.cloud/vmautomationlogs/Transcripts/" + (Split-Path
            $TranscriptFile -Leaf)
701         $Message += "Transcript Log: " + "<a href=http://" + $TranscriptURL + ">" +
            $TranscriptURL + "</a>"
702         $Message += "<br><br>"
703         $Message += "Server Build Date: " + (Get-Date)
704         $Message += "<br>"
705 
706         ### Close Message
707         ###---------------------
708         $Message += "</body></html>"
709 
710         ### Email Constants
711         ###---------------------
712         $From    = "cbrennan@com"
713         #$To      = "cbrennan@com,sdesimone@com,wercolano@com,rgee@com"
714         $To      = "cbrennan@com"
715         $SMTP    = "alertmx.com"
716         #$SMTP   = $SMTPServer
717         $Subject = "SERVER PROVISIONING STATUS: " + $Status
718 
719         ### Email Message
720         ###-------------------------------------------------------------------------------
721         Write-Host `r`n`r`n`r`n("=" * 50)`n"SENDING ALERT:" $Status`r`n("=" * 50)`r`n`r`n
            -ForegroundColor Yellow
722 
723         #Write-Host `n "MESSAGE: " $Message `n -ForegroundColor $StatusColor
724         Write-Host "TO: " $To
725         Send-MailMessage -To ($To -split ",") -From $From -Body $Message -Subject $Subject
            -BodyAsHtml -SmtpServer $SMTP
726 
727 
728         Write-Host `r`n('-' * 50)`r`n "END FUNCTION:" $FunctionName -ForegroundColor DarkGray
729 
730     }
731 
732 
733 
734 
735 
736 
737 
```