```powershell
<#
.SYNOPSIS
Get the State of Computers. Online, Offline, All.

.DESCRIPTION
Queries AD for computer objects, then performs test for Ping, DNS, RDP, UpTime, etc, to
detrmine OnLine/OffLine state.

.PARAMETER OUFilter
[string[]] Filter for Get-ADUser command of AD OU's not to query.

.PARAMETER SearchBases
[array] Searchbases for Get-ADUser command.

.PARAMETER ADProperties
[array] Return AD Properties of computer objects.

.PARAMETER Online
[switch] Conditional Logoic Switch to get all ONLINE Computers.

.PARAMETER Offline
[switch] Conditional Logoic Switch to get all OFFLINE Computers.

.EXAMPLE
Get-ServerState -State OnLine
Get-ServerState -State OffLine -SearchBase $SearchBase

.NOTES
General notes
#>
function Get-ServerState {
    [CmdletBinding()]
    param (
        [Parameter(Mandatory=$false)]
        [string[]]
        $OUFilter = "OU=Deprovisioned,DC=memhosp,DC=com"
        ,
        [Parameter(Mandatory=$false)]
        [array]
        $SearchBases = @(
            "DC=memhosp,DC=com"
            #"OU=Servers,DC=memhosp,DC=com",
            #"OU=Workstations,DC=memhosp,DC=com",
            #"OU=Workstations W7,DC=memhosp,DC=com"
        )
        ,
        [Parameter(Mandatory=$false)]
        [array]
        $ADProperties = @(
            "SamAccountName",
            "Name",
            "Description",
            "DistinguishedName",
            "dNSHostName",
            "ObjectClass",
            "OperatingSystem",
            "LastLogon",
            "lastLogonTimestamp",
            "whenCreated"
        )
        ,
        [Parameter(Mandatory=$false)]
        [ValidateSet("All","Online","OffLine",$null)]
        [string]
        $State = $null
    )
    Begin{
        Clear-Host
        $TranscriptPath = "$PSScriptRoot\Transcript-Get-ServerState-$(Get-Date -F
```

```
        "MM-dd-yyyy").txt"
69      Start-Transcript -Path $TranscriptPath
70
71      ### Hide Progress Bar
72      $global:ProgressPreference = "SilentlyContinue"
73
74      ### Set Conditional Logic Statements
75      ###-----------------------------------
76      switch ($State)
77      {
78          $OnLine {
79              ### Host is OnLine
80              ###---------------
81              $LogicSwitch = '( $Computer.PingIPAddress -ne $null )
82                                  -AND
83                              ( $Computer.ResolveDNS -ne $null )
84                                  -AND
85                              ( $Computer.TcpTestSucceeded -ne $true )
86                                  -AND
87                              ( $Computer.LastLogonTimestamp -lt
                                $(Get-Date).AddDays(-30) )'
88          }
89          $OffLine {
90              ### Host is OffLine
91              ###---------------
92              $LogicSwitch = '( $Computer.PingIPAddress -eq $null )
93                                  -AND
94                              ( $Computer.ResolveDNS -eq $null )
95                                  -AND
96                              ( $Computer.TcpTestSucceeded -eq $true )
97                                  -AND
98                              ( $Computer.LastLogonTimestamp -lt
                                $(Get-Date).AddDays(-30) )'
99          }
100         { @("All","Default",$null) } {
101             $LogicSwitch = $true
102         }
103     }
104 }
105 Process {
106     $ADComputers = @()
107     foreach($SearchBase in $SearchBases){
108         $ADComputers += Get-ADComputer -Filter * -SearchBase $SearchBase
            -Properties $ADProperties |
109             Select-Object -Property $ADProperties  | Where-Object {
                $_.DistinguishedName -notlike "*,$OUFilter" }
110         Write-Host "$SearchBase Count: " $ADComputers.Count -ForegroundColor Magenta
111     }
112     $ALLComputers = @()
113     foreach($Computer in $ADComputers){
114         Write-Host "Computer Name       :" $Computer.Name -ForegroundColor Cyan
115         $LastBootUpTime     = $null
116         $UpTimeDays         = $null
117
118         ### Ping IP Address (ICMP Ping Test)
119         ###------------------------------------------------
120         try{
121             $PingTest = Test-Connection -ComputerName $Computer.Name -Count 1
                -ErrorAction Stop
122             [System.Net.IPAddress]$PingIPAddress =
                $PingTest.IPV4Address.IPAddressToString
123         }catch{
124             $PingIPAddress = $null
125         }
126         Write-Host "PingResult          :"  $PingIPAddress
127
128         ### Resolve DNS Name (NSLookup)
129         ###------------------------------------------------
130         try{
```

```
131                 $ResolveDNS = Resolve-DnsName -Name $Computer.Name -ErrorAction Stop
132                 [System.Net.IPAddress]$ResolveDNSIP = $ResolveDNS.IP4Address
133             }catch{
134                 $ResolveDNSIP = $null
135             }
136             Write-Host "ResolveDNSIP        :" $ResolveDNSIP
137
138             ### Get DNS Host Entry (Query DNS Server)
139             ###-------------------------------------------------
140             try{
141                 $GetDNSHostEntry =
                    [System.Net.Dns]::GetHostEntry($Computer.dNSHostName).HostName
142             }catch{
143                 $GetDNSHostEntry = $null
144             }
145             Write-Host "GetDNSHostEntry     :" $GetDNSHostEntry
146
147             ### Get IP DNS Host Entry (Reverse Lookup by IP)
148             ###-------------------------------------------------
149             try{
150                 $ReversePDNSLookup =
                    [System.Net.Dns]::GetHostEntry($PingIPAddress).HostName
151             }catch{
152                 $ReversePDNSLookup = $null
153             }
154             Write-Host "ReversePDNSLookup   :" $ReversePDNSLookup
155
156             ### Test RDP Port 3389 (RDP Port 3389 Is Open)
157             ###-------------------------------------------------
158             try{
159                 $TestConnection = Test-NetConnection -ComputerName $Computer.Name -Port
                    3389 -ErrorAction Stop
160                 [boolean]$RDPPort3389 = $TestConnection.TcpTestSucceeded
161             }catch{
162                 $RDPPort3389 = $null
163             }
164             Write-Host "RDPPort3389         :" $RDPPort3389
165
166             ### Test Host Name UNC Path (UNC Administrative Share by HostName)
167             ###-------------------------------------------------
168             try{
169                 $TestHostNameUNCPath = Test-Path -Path "\\$($Computer.dNSHostName)\C$"
                    -ErrorAction Stop
170             }catch{
171                 $TestHostNameUNCPath = $null
172             }
173             Write-Host "TestHostNameUNCPath :" $TestHostNameUNCPath
174
175             ### Last Login Date
176             ###-------------------------------------------------
177             [DateTime]$LastLogon =
                [datetime]::FromFileTime($Computer.LastLogon).ToString('MM/dd/yyyy')
178             [DateTime]$LastLogonTimestamp =
                [DateTime]::FromFileTime($Computer.LastLogonTimestamp).ToString('MM/dd/yyyy')
179             $LastLogonDays = $(Get-Date) - $LastLogonTimestamp
180             $LastLogonDays = $LastLogonDays.Days
181             Write-Host "LastLogon           :" $LastLogon
182             Write-Host "LastLogonTimestamp  :" $LastLogonTimestamp
183
184             ### Get UpTime
185             ###-------------------------------------------------
186             if($GetDNSHostEntry){
187                 try{
188                     $WSMan = Test-WSMan -ComputerName $GetDNSHostEntry -ErrorAction Stop
189                 }catch{
190                     $WSMan = $null
191                 }
192                 if($WSMan){
193                     Write-Host "WSMan       : " $WSMan -ForeGroundColor Green
```

```
                    try{
                        $OS = Get-WmiObject Win32_OperatingSystem -ComputerName $GetDNSHostEntry
                        $LastBootUpTime = $OS.ConvertToDateTime($OS.LastBootUpTime)
                        $UpTimeDays = ((Get-Date) -  $LastBootUpTime).Days
                    }catch{
                        $LastBootUpTime = $null
                        $UpTimeDays     = $null
                    }
                    Write-Host "LastBootUpTime      : " $LastBootUpTime -ForeGroundColor Yellow
                    Write-Host "UpTimeDays          : " $UpTimeDays -ForeGroundColor Yellow
                }else{
                    Write-Host "WSMan       : False" -ForeGroundColor Red
                }
            }
            $Computer = [PSCustomObject]@{
                Name                = $Computer.Name
                dNSHostName         = $Computer.dNSHostName
                Description         = $Computer.Description
                DistinguishedName   = $Computer.DistinguishedName
                ObjectClass         = $Computer.ObjectClass
                OperatingSystem     = $Computer.OperatingSystem
                PingIPAddress       = $PingIPAddress
                ResolveDNSIP        = $ResolveDNSIP
                GetDNSHostEntry     = $GetDNSHostEntry
                ReversePDNSLookup   = $ReversePDNSLookup
                RDPPort3389         = $RDPPort3389
                TestHostNameUNCPath = $TestHostNameUNCPath
                TestIPUNCPath       = $TestIPUNCPath
                LastLogon           = $LastLogon
                LastLogonTimestamp  = $LastLogonTimestamp
                LastLogonDays       = $LastLogonDays
                WSMan               = $WSMan
                LastBootUpTime      = $LastBootUpTime
                UpTimeDays          = $UpTimeDays
                whenCreated         = $Computer.whenCreated
            }
            $AllComputers += $Computer
        }

        ### Execute Conditional Logic Statements
        ###-------------------------------------
        $Computers = @()
        foreach($Computer in $AllComputers){
            if($LogicSwitch){
                $Computers += $Computer
            }
        }
    }
    End{
        ### Show Computer Counts
        ###----------------------
        Write-Host "All Computers Count  ;" $ADComputers.Count -ForegroundColor Magenta
        Write-Host "Live Computers Count ;" $Computers.Count -ForegroundColor Magenta

        ### Export Data File
        ###----------------------
        $ExportPath = "$PSSCriptRoot\MemHosp-Depro-Computers-$(Get-Date -f MM-dd-yyyy).csv"
        Write-Host "ExportPath: " $ExportPath
        $Computers | Export-CSV -Path $ExportPath -NoTypeInformation -Delimiter ';'
        Start-Process $ExportPaths

        Stop-Transcript
    }
}
Get-ServerState -State Online
```