```powershell
#Set-ManagerField.ps1
#Written by:Jeff Brusoe
#Last Updated: May 1, 2020

[CmdletBinding()]
param (
    #Common HSC PowerShell Parameters
    [switch]$NoSessionTranscript,
    [string]$LogFilePath = "$PSScriptRoot\Logs",
    [switch]$StopOnError, #$true is used for testing purposes
    [int]$DaysToKeepLogFiles = 5, #this value used to clean old log files

    #File specific parameters
    [switch]$Testing,
    [int]$TestDelay = 2
    )

#Reset environment
Clear-Host
$Error.Clear()
Set-Location $PSScriptRoot

################################
#Import HSC PowerShell Modules#
################################

#Build path to HSC PowerShell Modules
$PathToHSCPowerShellModules = $PSScriptRoot
$PathToHSCPowerShellModules =
$PathToHSCPowerShellModules.substring(0,$PathToHSCPowerShellModules.lastIndexOf("\")+1)
$PathToHSCPowerShellModules += "1HSC-PowerShell-Modules"
Write-Output $PathToHSCPowerShellModules

#Attempt to load common code module
$CommonCodeModule = $PathToHSCPowerShellModules + "\HSC-CommonCodeModule.psm1"
Write-Output "Path to common code module: $CommonCodeModule"
Import-Module $CommonCodeModule -Force -ArgumentList
$NoSessionTranscript,$LogFilePath,$true,$DaysToKeepLogFiles

#Attempt to load HSC Active Directory Module
$ActiveDirectoryModule = $PathToHSCPowerShellModules + "\HSC-ActiveDirectoryModule.psm1"
Write-Output "Path to HSC Active Directory Module: $ActiveDirectoryModule"
Import-Module $ActiveDirectoryModule -Force

if ($Error.Count -gt 0)
{
    #Any errors at this point are from loading modules. Program must stop.
    Write-Warning "There was an error configuring the environment. Program is exiting."
    Exit-Commands
}

########################################
# End of Import HSC PowerShell Modules #
########################################

##############################
#Configure environment block#
##############################
Write-Output "Getting Parameter Information"
Get-Parameter -ParameterList $PSBoundParameters

Set-Environment
Set-WindowTitle

########################################
#End of environment configuration block#
########################################
```

```
66
67  #extensionAttribute4 -> Manager's EmployeeNumber
68  #EmployeeNumber _> Users's EmployeeNumber
69  Write-Output "Generating list of AD users"
70  try
71  {
72      $users = Get-ADUser -filter * -Properties extensionAttribute4,EmployeeNumber
        -ErrorAction Stop |
73          where {$_.extensionAttribute4 -ne $null}
74  }
75  catch
76  {
77      Write-Warning "Unable to generate AD user list. Program is exiting."
78      Exit-Command
79  }
80
81  $ManagerFoundCount = 0
82  $ManagerNotFoundCount = 0
83  $TotalCount = 0
84
85  foreach ($user in $users)
86  {
87      Write-Output $("SamAccountName: " + $user.SamAccountName)
88      Write-Output $("EmployeeNumber: " + $user.EmployeeNumber)
89      Write-Output $("extensionAttribute4: " + $user.extensionAttribute4)
90
91      $Manager = $users | where {$user.extensionAttribute4 -eq $_.EmployeeNumber}
92
93      if (($Manager | Measure).Count -gt 1)
94      {
95          #This is a safety measure to make sure one unique value is returned
96          Write-Warning "Unable to find one unique value for the manager field."
97
98          if ($StopOnError)
99          {
100             Write-Warning "Program is exiting."
101             Exit-Command
102         }
103     }
104     elseif ($Manager -eq $null)
105     {
106         Write-Output "Unable to find manager"
107         $ManagerNotFoundCount++
108     }
109     else
110     {
111         Write-Output $("Manager: " + $Manager.SamAccountName)
112         $ManagerFoundCount++
113
114         if ($Testing)
115         {
116             #Short delay for testing purposes
117             Start-Sleep -s $TestDelay
118         }
119
120         try
121         {
122             $user | Set-ADUser -Manager $Manager -ErrorAction Stop
123             Write-Output "Successfully set manager field"
124         }
125         catch
126         {
127             Write-Warning "Error setting manager field"
128             $Error | fl
129
130             if ($StopOnError)
131             {
```

```
                Write-Warning "Program is exiting."
                Exit-Command
            }
        }
    }

    $TotalCount++

    Write-Output "Manager Found Count: $ManagerFoundCount"
    Write-Output "Manager Not Found Count: $ManagerNotFoundCount"
    Write-Output "Total Count: $TotalCount"

    Write-Output "**************************"
}

if (!$StopOnError)
{
    Stop-Transcript
}
```