```powershell
#Update-AddressList.ps1
#Written by: Jeff Brusoe
#Last Updated: April 28, 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
    )

$Error.Clear()
Clear-Host
Set-StrictMode -Version Latest

##$SessionTranscriptFile = "$PSScriptRoot\Logs\" + (Get-Date -Format yyyy-MM-dd-HH-mm)
+ "-SessionTranscript.txt"
##Start-Transcript $SessionTranscriptFile

########################################
# 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 Office 365 Module
$Office365Module = $PathToHSCPowerShellModules + "\HSC-Office365Module.psm1"
Write-Output "Path to HSC Office 365 module: $Office365Module"
Import-Module $Office365Module -Force

if ($Error.Count -gt 0)
{
    #This would imply an error loading the HSC PS modules.
    Write-Warning "Error loading HSC PowerShell modules. Program is exiting."
    Stop-Transcript
    return
}

################################################
# End of block to load HSC PowerShell Modules #
################################################

##############################
#Configure environment block#
##############################
Write-Output "Getting Parameter Information"
Get-Parameter -ParameterList $PSBoundParameters #Here for logging purposes

Set-Environment
Set-WindowTitle

#See this page to understand what is going on here.
#https://www.thecloudjournal.net/2016/07/create-your-own-powershell-module-for-exchange-online/
ConnectTo-Office365 #from Office 365 module
```

```
Import-Module ExchangeOnline -Force #comes from HSC-OFfice365Module.psm1

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

#Attempting to set contact information
Function Set-HSCMailContact
{
    Write-Output "`n`nConfiguring Mail Contacts"

    try
    {
        Write-Output "Generating list of mail contacts"
        $Contacts = Get-Contact -ResultSize Unlimited -ErrorAction Stop #| Select
        -First 100
    }
    catch
    {
        Write-Warning "Unable to generate list of contacts"
        $Contacts = $null
    }

    $ContactCount = 0
    foreach ($Contact in $Contacts)
    {
        Write-Output $("Current Contact: " + $Contact.WindowsEmailAddress)
        Write-Output $("Current Simple Display Name: " + $Contact.SimpleDisplayName)
        Write-Output "Contact Count: $ContactCount"
        $ContactCount++

        try
        {
            $Contact | Set-Contact -SimpleDisplayName $Contact.SimpleDisplayName
            -ErrorAction Stop
            Write-Output "Successfully updated contact"
        }
        catch
        {
            Write-Warning "Error attempting to set mail contact. Trying again."

            try
            {
                Set-Contact -Identity $Contact.WindowsEmailAddress -SimpleDisplayName
                $Contact.SimpleDisplayName -ErrorAction Stop
                Write-Output "Successfully updated contact"
            }
            catch
            {
                Write-Error "Unable to set mail contact"

                if ($StopOnError)
                {
                    return
                }
            }
        }

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

    Write-Output "Done setting mail contacts"
} #End mail contact function


Function Set-HSCMailbox
{
```

```
128         Write-Output "`n`nConfiguring Mailboxes"
129
130         try
131         {
132             Write-Output "Generating list of mailboxes"
133             $Mailboxes = Get-Mailbox -ResultSize Unlimited -ErrorAction Stop #| select
                -first 100
134         }
135         catch
136         {
137             Write-Warning "Unable to generate list of mailboxes"
138             $Mailboxes = $null
139         }
140
141         $MailboxCount = 0
142         foreach ($Mailbox in $Mailboxes)
143         {
144             Write-Output $( "Current Mailbox: " +$Mailbox.PrimarySMTPAddress)
145             Write-Output "Mailbox Count: $MailboxCount"
146             $MailboxCount++
147
148             try
149             {
150                 $Mailbox | Set-Mailbox -ApplyMandatoryProperties -ErrorAction Stop
151                 Write-Output "Successfully updated mailbox"
152             }
153             catch
154             {
155                 Write-Warning "Error attempting to set mailbox. Trying again."
156
157                 try
158                 {
159                     Set-Mailbox -Identity $Mailbox.PrimarySMTPAddress
                        -ApplyMandatoryProperties -ErrorAction Stop
160                     Write-Output "Successfully updated mailbox"
161                 }
162                 catch
163                 {
164                     Write-Error "Unable to set mailbox"
165
166                     if ($StopOnError)
167                     {
168                         return
169                     }
170                 }
171             }
172
173             Write-Output "********************************"
174         }
175     } #End set mailbox function
176
177     Function Set-HSCMailUser
178     {
179         Write-Output "`n`nSetting properties for mail users"
180         try
181         {
182             Write-Output "Generatng list of mail users"
183             $MailUsers = Get-MailUser -Resultsize Unlimited -ErrorAction Stop #| select
                -first 100
184         }
185         catch
186         {
187             Write-Warning "Unable to generate list of mail users"
188             $MailUsers = $null
189         }
190
191         $MailUserCount = 0
```

```
192         foreach ($MailUser in $MailUsers)
193         {
194             Write-Output $("Current MailUser: " + $MailUser.PrimarySMTPAddress)
195             Write-Output $("Current Simple Display Name: " + $MailUser.SimpleDisplayName)
196             Write-Output "Mail User Count: $MailUserCount"
197             $MailUserCount++
198 
199             try
200             {
201                 $MailUser | Set-MailUser -SimpleDisplayName $MailUser.SimpleDisplayName
                    -ErrorAction Stop
202                 Write-Output "Successfully updated mailuser"
203             }
204             catch
205             {
206                 Write-Warning "Error attempting to set mailuser. Trying again."
207 
208                 try
209                 {
210                     Set-MailUser -Identity $MailUser.PrimarySMTPAddress -SimpleDisplayName
                        $MailUser.SimpleDisplayName -ErrorAction Stop
211                     Write-Output "Successfully updated mailuser"
212                 }
213                 catch
214                 {
215                     Write-Error "Unable to set mailuser"
216 
217                     if ($StopOnError)
218                     {
219                         return
220                     }
221                 }
222             }
223 
224             Write-Output "*********************************"
225         }
226     }
227 
228     #This switch stateent is here because of some sporadic issues with timesouts that were
        occurring.
229     switch ((Get-Date).Day%3)
230     {
231         0 { Set-HSCMailContact ; Set-HSCMailbox ; Set-HSCMailUser }
232         1 { Set-HSCMailbox ; Set-HSCMailUser ; Set-HSCMailContact }
233         2 { Set-HSCMailUser ; Set-HSCMailContact ; Set-HSCMailbox }
234     }
235 
236     if (!$NoSessionTranscript)
237     {
238         Write-Verbose "Stopping Transcript"
239         Stop-Transcript
240     }
241 
242     Exit
```