```powershell
#Update-WVUMAddressList.ps1
#Written by: Jeff Brusoe
#Last Updated: April 27, 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
    [string]$ImportFile = "$PSScriptRoot\Recipients.csv", #Only needed with $Testing flag
    [string]$APPTtitleFile = "$PSScriptRoot\AllAPPTitles.csv",
    [string]$PhysicianTitleFile = "$PSScriptRoot\AllPhysicianTitles.csv",
    [switch]$Testing #$true ==> Read recipients from csv (see comments for this below)
    instead of with Get-Recipient
)

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

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

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

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

#Both from HSC common code module
Set-Environment
Set-WindowTitle

if (!$Testing)
{
    #See this page to understand what is going on here.

    #https://www.thecloudjournal.net/2016/07/create-your-own-powershell-module-for-exchan
```

```
    ge-online/
    ConnectTo-Office365 #from Office 365 module
    Import-Module ExchangeOnline -Force #comes from HSC-OFfice365Module.psm1
}

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

############################
# Begin main part of code #
############################

#Create output files
Write-Output "Creating output files"

$PhysicianOutputFile = "$PSScriptRoot\OutputFiles\" + (Get-Date -format
yyyy-MM-dd-HH-mm) + "-PhysicianOutputFile.csv"
New-Item -type file -Path $PhysicianOutputFile -Force

$ResidentOutputFile = "$PSScriptRoot\OutputFiles\" + (Get-Date -format
yyyy-MM-dd-HH-mm) + "-ResidentOutputFile.csv"
New-Item -type file -Path $ResidentOutputFile -Force

$APPOutputFile = "$PSScriptRoot\OutputFiles\" + (Get-Date -format yyyy-MM-dd-HH-mm) +
"-APPOutputFile.csv"
New-Item -type file -Path $APPOutputFile -Force

#Read title files and generate arrays of job titles
Write-Output "Reading title files"
$APPTitles = Import-Csv $APPTTitleFile

$APPJobTitles = @()
foreach ($APPTitle in $APPTitles."Job Title")
{
    Write-Output $APPTitle
    Write-Output $APPTitle.substring(0,$AppTitle.indexOf(":")).Trim()
    $APPJobTitles += $APPTitle.substring(0,$AppTitle.indexOf(":")).Trim()
    Write-Output "***************"
}
$APPJobTitles = $APPJobTitles | Select -Unique
$APPJobTitles

#Physican Job Titles
$PhysicianTitles = Import-Csv $PhysicianTitleFile

$PhysicianJobTitles = @()
foreach ($PhysicianTitle in $PhysicianTitles."Job Title")
{
    Write-Output $PhysicianTitle
    Write-Output $PhysicianTitle.substring(0,$PhysicianTitle.indexOf(":")).Trim()
    $PhysicianJobTitles +=
    $PhysicianTitle.substring(0,$PhysicianTitle.indexOf(":")).Trim()
    Write-Output "*****************"
}
$PhysicianJobTitles = $PhysicianJobTitles | Select -Unique

Write-Output "`n`nPhysician Job Titles"
Write-Output $("Count: " + $PhysicianJobTitles.Count)
Write-Output $PhysicianJobTitles

Write-Output "`n`nAPP Job Titles:"
Write-Output $("Count: " + $APPJobTitles.Count)
Write-Output $APPJobTitles

#Generate lists of O365 objects
if ($Testing)
```

```
{
    #Just read CSV to speed up testing
    #CSV was generated from Get-Recipient -ResultSize Unlimited | Export-Csv
    Recipients.csv
    try
    {
        Write-Output "Path to Import File: $ImportFile"
        $Recipients = Import-Csv $ImportFile -ErrorAction Stop
    }
    catch
    {
        Write-Warning "Unable to find import file. Program is exiting."
        Exit-Command
    }
}
else
{
    #Get-Recipient takes about half an hour to generate list due to the number of
    recipients (~45000).
    #For testing purposes, choose the -Testing switch to read from the previously
    generated CSV file.
    try
    {
        Write-Output "`n`nGenerating recipient list"
        $Recipients = Get-Recipient -ResultSize Unlimited -ErrorAction Stop
        Write-Output "Successfully generated recipient list"
    }
    catch
    {
        Write-Warning "Error generating mailbox list. Program is exiting"
        Exit-Command
    }
}

#################################
# Looping through O365 objects #
#################################

$RecipientCount = 0
Write-Output "Looping through O365 Recipients"

foreach ($Recipient in $Recipients)
{
    Write-Output $("Current Recipient: " + $Recipient.PrimarySMTPAddress)
    Write-Output $("Recipient Title String: " + $Recipient.Title)

    $RecipientCount++
    Write-Output "Recipient Count: $RecipientCount"

    #An array of titles is created here to account for the possibility of multiple
    #titles appearing in the title field. A comma is used to delimit fields like this.
    $TitleArray = $Recipient.Title.split(",")
    Write-Output "`n`nTitle Array:"
    Write-Output $TitleArray

    #Determine extensionAttribute7
    if ([string]::IsNullOrEmpty($Recipient.CustomAttribute7))
    {
        $ext7 = "Blank"
    }
    else
    {
        $ext7 = $Recipient.CustomAttribute7
    }
    Write-Output "`nextensionAttribute7: $ext7"

    foreach ($Title in $TitleArray)
```

```
    {
        Write-Output "Current Title: $Title"

        try
        {
            $Title = $Title.substring(0,$Recipient.Title.indexOf(":")).Trim()
            Write-Output "Cleaned Title: $Title"
        }
        catch
        {
            #This is most likely going to happen when the user is not in the list which
            the hospital sent over.
            Write-Warning "Unable to clean title"
        }

        if (($PhysicianJobTitles -contains $Title) -AND ($Title -like "*resident*"))
        {
            #This is a resident
            Write-Output "Writing to resident output file"
            $Recipient | Select -Property
            Name,Title,PrimarySMTPAddress,@{Name="extensionAttribute7";Expression =
            {$ext7}} | Export-Csv $ResidentOutputFile -Append -NoTypeInformation
            break
        }
        elseif ($PhysicianJobTitles -contains $Title)
        {
            #Physician
            Write-Output "Writing to physician output file"
            $Recipient | Select
            Name,Title,PrimarySMTPAddress,@{Name="extensionAttribute7";Expression =
            {$ext7}} | Export-Csv $PhysicianOutputFile -Append -NoTypeInformation
            break
        }
        elseif ($APPJobTitles -contains $Title)
        {
            #APP
            Write-Output "Writing to APP output file"
            $Recipient | Select
            Name,Title,PrimarySMTPAddress,@{Name="extensionAttribute7";Expression =
            {$ext7}} | Export-Csv $APPOutputFile -Append -NoTypeInformation
            break
        }
        else
        {
            #No match
            Write-Output "No match"
        }
    }

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

    if ($StopOnError)
    {
        $Error | FL
        Exit-Command
    }
}

Stop-Transcript
```