

```

1 #Update-WVUMAddressList.ps1
2 #Written by: Jeff Brusoe
3 #Last Updated: April 27, 2020
4
5 [CmdletBinding()]
6 param (
7     #Common HSC PowerShell Parameters
8     [switch]$NoSessionTranscript,
9     [string]$LogFilePath = "$PSScriptRoot\Logs",
10    [switch]$StopOnError, #true is used for testing purposes
11    [int]$DaysToKeepLogFiles = 5, #this value used to clean old log files
12
13    #File specific parameters
14    [string]$ImportFile = "$PSScriptRoot\Recipients.csv", #Only needed with $Testing flag
15    [string]$APPTtitleFile = "$PSScriptRoot\AllAPPTitles.csv",
16    [string]$PhysicianTitleFile = "$PSScriptRoot\AllPhysicianTitles.csv",
17    [switch]$Testing #true ==> Read recipients from csv (see comments for this below)
18    instead of with Get-Recipient
19 )
20
21 #Reset environment
22 Clear-Host
23 $Error.Clear()
24 Set-Location $PSScriptRoot
25 Set-StrictMode -Version Latest
26
27 #####
28 # Import HSC PowerShell Modules #
29 #####
30
31 #Build path to HSC PowerShell Modules
32 $PathToHSCPowerShellModules = $PSScriptRoot
33 $PathToHSCPowerShellModules =
34 $PathToHSCPowerShellModules.substring(0,$PathToHSCPowerShellModules.lastIndexOf("\")+1)
35 $PathToHSCPowerShellModules += "1HSC-PowerShell-Modules"
36 Write-Output $PathToHSCPowerShellModules
37
38 #Attempt to load common code module
39 $CommonCodeModule = $PathToHSCPowerShellModules + "\HSC-CommonCodeModule.psm1"
40 Write-Output "Path to common code module: $CommonCodeModule"
41 Import-Module $CommonCodeModule -Force -ArgumentList
42 $NoSessionTranscript,$LogFilePath,$true,$DaysToKeepLogFiles
43
44 #Attempt to load HSC Office 365 Module
45 $Office365Module = $PathToHSCPowerShellModules + "\HSC-Office365Module.psm1"
46 Write-Output "Path to HSC Office 365 module: $Office365Module"
47 Import-Module $Office365Module -Force
48
49 #####
50 # End of Import HSC PowerShell Modules #
51 #####
52
53 #####
54 # Configure environment block #
55 #####
56 Write-Output "Getting Parameter Information"
57 Get-Parameter -ParameterList $PSBoundParameters
58
59 #Both from HSC common code module
60 Set-Environment
61 Set-WindowTitle
62
63 if (!$Testing)
64 {
65     #See this page to understand what is going on here.
66
67     #https://www.thecloudjournal.net/2016/07/create-your-own-powershell-module-for-exchan

```

```

64     ge-online/
65     ConnectTo-Office365 #from Office 365 module
66     Import-Module ExchangeOnline -Force #comes from HSC-Office365Module.psm1
67 }
68 #####
69 # End of environment configuration block #
70 #####
71
72 #####
73 # Begin main part of code #
74 #####
75
76 #Create output files
77 Write-Output "Creating output files"
78
79 $PhysicianOutputFile = "$PSScriptRoot\OutputFiles\" + (Get-Date -format
80 yyyy-MM-dd-HH-mm) + "-PhysicianOutputFile.csv"
81 New-Item -type file -Path $PhysicianOutputFile -Force
82
83 $ResidentOutputFile = "$PSScriptRoot\OutputFiles\" + (Get-Date -format
84 yyyy-MM-dd-HH-mm) + "-ResidentOutputFile.csv"
85 New-Item -type file -Path $ResidentOutputFile -Force
86
87 $APPOutputFile = "$PSScriptRoot\OutputFiles\" + (Get-Date -format yyyy-MM-dd-HH-mm) +
88 "-APPOutputFile.csv"
89 New-Item -type file -Path $APPOutputFile -Force
90
91 #Read title files and generate arrays of job titles
92 Write-Output "Reading title files"
93 $APPTitles = Import-Csv $APPTTitleFile
94
95 $APPJobTitles = @()
96 foreach ($APPTTitle in $APPTitles."Job Title")
97 {
98     Write-Output $APPTTitle
99     Write-Output $APPTTitle.substring(0,$APPTTitle.indexOf(":")).Trim()
100     $APPJobTitles += $APPTTitle.substring(0,$APPTTitle.indexOf(":")).Trim()
101     Write-Output "*****"
102 }
103 $APPJobTitles = $APPJobTitles | Select -Unique
104 $APPJobTitles
105
106 #Physician Job Titles
107 $PhysicianTitles = Import-Csv $PhysicianTitleFile
108
109 $PhysicianJobTitles = @()
110 foreach ($PhysicianTitle in $PhysicianTitles."Job Title")
111 {
112     Write-Output $PhysicianTitle
113     Write-Output $PhysicianTitle.substring(0,$PhysicianTitle.indexOf(":")).Trim()
114     $PhysicianJobTitles +=
115     $PhysicianTitle.substring(0,$PhysicianTitle.indexOf(":")).Trim()
116     Write-Output "*****"
117 }
118 $PhysicianJobTitles = $PhysicianJobTitles | Select -Unique
119
120 Write-Output "`n`nPhysician Job Titles"
121 Write-Output $("Count: " + $PhysicianJobTitles.Count)
122 Write-Output $PhysicianJobTitles
123
124 Write-Output "`n`nAPP Job Titles:"
125 Write-Output $("Count: " + $APPJobTitles.Count)
126 Write-Output $APPJobTitles
127
128 #Generate lists of O365 objects
129 if ($Testing)

```

```

126 {
127     #Just read CSV to speed up testing
128     #CSV was generated from Get-Recipient -ResultSize Unlimited | Export-Csv
    Recipients.csv
129     try
130     {
131         Write-Output "Path to Import File: $ImportFile"
132         $Recipients = Import-Csv $ImportFile -ErrorAction Stop
133     }
134     catch
135     {
136         Write-Warning "Unable to find import file. Program is exiting."
137         Exit-Command
138     }
139 }
140 else
141 {
142     #Get-Recipient takes about half an hour to generate list due to the number of
    recipients (~45000).
143     #For testing purposes, choose the -Testing switch to read from the previously
    generated CSV file.
144     try
145     {
146         Write-Output "`n`nGenerating recipient list"
147         $Recipients = Get-Recipient -ResultSize Unlimited -ErrorAction Stop
148         Write-Output "Successfully generated recipient list"
149     }
150     catch
151     {
152         Write-Warning "Error generating mailbox list. Program is exiting"
153         Exit-Command
154     }
155 }
156
157 #####
158 # Looping through O365 objects #
159 #####
160
161 $RecipientCount = 0
162 Write-Output "Looping through O365 Recipients"
163
164 foreach ($Recipient in $Recipients)
165 {
166     Write-Output $("Current Recipient: " + $Recipient.PrimarySMTPAddress)
167     Write-Output $("Recipient Title String: " + $Recipient.Title)
168
169     $RecipientCount++
170     Write-Output "Recipient Count: $RecipientCount"
171
172     #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.
173     $TitleArray = $Recipient.Title.split(",")
174     Write-Output "`n`nTitle Array:"
175     Write-Output $TitleArray
176
177     #Determine extensionAttribute7
178     if ([string]::IsNullOrEmpty($Recipient.CustomAttribute7))
179     {
180         $ext7 = "Blank"
181     }
182     else
183     {
184         $ext7 = $Recipient.CustomAttribute7
185     }
186     Write-Output "`nextensionAttribute7: $ext7"
187
188     foreach ($Title in $TitleArray)

```

```

190 {
191     Write-Output "Current Title: $Title"
192
193     try
194     {
195         $Title = $Title.substring(0,$Recipient.Title.indexOf(":")).Trim()
196         Write-Output "Cleaned Title: $Title"
197     }
198     catch
199     {
200         #This is most likely going to happen when the user is not in the list which
201         the hospital sent over.
202         Write-Warning "Unable to clean title"
203     }
204
205     if (($PhysicianJobTitles -contains $Title) -AND ($Title -like "*resident*"))
206     {
207         #This is a resident
208         Write-Output "Writing to resident output file"
209         $Recipient | Select -Property
210         Name,Title,PrimarySMTPAddress,@{Name="extensionAttribute7";Expression =
211         {$ext7}} | Export-Csv $ResidentOutputFile -Append -NoTypeInfoation
212         break
213     }
214     elseif ($PhysicianJobTitles -contains $Title)
215     {
216         #Physician
217         Write-Output "Writing to physician output file"
218         $Recipient | Select
219         Name,Title,PrimarySMTPAddress,@{Name="extensionAttribute7";Expression =
220         {$ext7}} | Export-Csv $PhysicianOutputFile -Append -NoTypeInfoation
221         break
222     }
223     elseif ($APPJobTitles -contains $Title)
224     {
225         #APP
226         Write-Output "Writing to APP output file"
227         $Recipient | Select
228         Name,Title,PrimarySMTPAddress,@{Name="extensionAttribute7";Expression =
229         {$ext7}} | Export-Csv $APPOutputFile -Append -NoTypeInfoation
230         break
231     }
232     else
233     {
234         #No match
235         Write-Output "No match"
236     }
237 }
238
239 Write-Output "*****"
240
241 if ($StopOnError)
242 {
243     $Error | FL
244     Exit-Command
245 }
246
247 Stop-Transcript

```