

```
1 <#
2 .SYNOPSIS
3     The purpose of this file is to export the HSC address book.
4     The export file that is generated will then be imported into
5     the WVU Foundation's address book.
6
7 .PARAMETER MinimumFileRecipients
8     This parameter is used as a safety value to ensure that
9     an empty address book isn't sent to WVUF.
10
11 .PARAMETER Testing
12     Default is 7. Specifies the number of days to keep log files
13     before they are deleted.
14
15 .PARAMETER HSCMailboxFile
16     The output file generated by the program. It should not be
17     changed from the default without also changing the value in
18     the import file.
19
20 .NOTES
21     Written by: Jeff Brusoe
22     Last Updated: November 24, 2020
23 #>
24
25 [CmdletBinding()]
26 param (
27     [ValidateRange(1,[int]::MaxValue)]
28     [int]$MinimumFileRecipients = 4000,
29
30     [ValidateNotNullOrEmpty()]
31     [string]$HSCMailboxFile = "$PSScriptRoot\Logs\" +
32         (Get-Date -Format yyyy-MM-dd) +
33         "-HSCMailboxExport.csv",
34
35     [ValidateNotNullOrEmpty()]
36     [string]$ExcludedAccountFile = "$PSScriptRoot\HSCExcludedAccounts.csv",
37
38     [ValidateNotNullOrEmpty()]
39     [string[]]$ErrorRecipients = @("jbrusoe@hsc.wvu.edu","microsoft@hsc.wvu.edu"),
40
41     [ValidateNotNullOrEmpty()]
42     [string[]]$FileRecipients = @(
43         "microsoft@hsc.wvu.edu",
44         "jbrusoe@hsc.wvu.edu",
45         "RHilling@wvuf.org"
46     ),
47
48     [ValidateNotNullOrEmpty()]
49     [string]$From = "microsoft@hsc.wvu.edu",
50
51     [ValidateNotNullOrEmpty()]
52     [string]$SMTPServer = "hssmtp.hsc.wvu.edu",
53
54     [switch]$Testing
55 )
56
57 # Configure PS environment
58 try {
59     Set-HSCEnvironment -ErrorAction Stop
60     Connect-HSCExchangeOnline -ErrorAction Stop
```

```
61 }
62 catch {
63     Write-Warning "Unable to configure environment. Program is exiting."
64     Invoke-HSCExitCommand -ErrorCount $Error.Count
65 }
66
67 # Initialization of output files
68 New-Item -path $HSCMailboxFile -Type "file" -Force
69
70 #Get HSC mailboxes
71 Write-Output "Generating list of HSC mailboxes"
72 Write-Output "HSC Mailbox File: $HSCMailboxFile"
73
74 try
75 {
76     $Properties = @(
77         "FirstName",
78         "LastName",
79         "Title",
80         "Department",
81         "Phone",
82         "WhenChanged",
83         "HiddenFromAddressListsEnabled",
84         "CustomAttribute7"
85     )
86
87     $EXOMailboxParams = @{
88         ResultSize = "Unlimited"
89         Properties = $Properties
90         RecipientType = "UserMailbox"
91         ErrorAction = "Stop"
92         Verbose = $true
93     }
94     $Mailboxes = Get-EXORecipient @EXOMailboxParams |
95         Where-Object {
96             ($_.CustomAttribute7 -eq "Yes365") -AND
97             ($_.PrimarySMTPAddress -notlike "*rni.*") -AND
98             ($_.PrimarySMTPAddress -notlike "*wvurni*")
99         }
100
101     Write-Output "Done generating list of HSC mailboxes"
102 }
103 catch
104 {
105     Write-Warning "Unable to generate HSC recipient array. Program is exiting."
106     Invoke-HSCExitCommand -ErrorCount $Error.Count
107 }
108
109 try {
110     Write-Output "Getting list of excluded mailboxes"
111     $ExcludedAccounts = Import-Csv $ExcludedAccountFile -ErrorAction Stop
112 }
113 catch {
114     Write-Warning "Unable to import exclude accounts file"
115     Invoke-HSCExitCommand -ErrorCount $Error.Count
116 }
117
118 Write-Output "Excluded Mailboxes:"
119 Write-Output $ExcludedAccounts
120
```

```
121 Write-Output "`nBeginning to process HSC mailboxes"
122 Write-Output $("Total Number of Mailboxes: " + ($Mailboxes | Measure-Object).count)
123
124 $MailBoxCount = 1
125
126 foreach ($Mailbox in $Mailboxes)
127 {
128     if ($Testing -AND $Error.Count -gt 0)
129     {
130         Write-Warning "Error has occurred. Error message:"
131         $Error | Format-List
132
133         Write-Warning "`nProgram is exiting."
134         Invoke-HSCExitCommand -ErrorCount $Error.Count
135     }
136
137     Write-Output $("Current Mailbox: " + $Mailbox.PrimarySMTPAddress)
138     Write-Output "Mailbox Number: $MailboxCount"
139     $MailboxCount++
140
141     #This block of code verifies that the following before allowing
142     #an account to be written to the output file.
143     #1. First and last name must both have values.
144     #2. Remove
145     # a. admin accounts
146     # b. test accounts.
147     # c. retiree accounts
148     # d. invalid addresses (@wvuhsc.onmicrosoft.com)
149     # e. conference rooms
150     # f. proxy accounts
151     # g. Hospital accounts
152
153     if ([string]::IsNullOrEmpty($Mailbox.LastName) -OR
154         [string]::IsNullOrEmpty($Mailbox.FirstName))
155     {
156         if ([string]::IsNullOrEmpty($Mailbox.LastName))
157         {
158             Write-Warning "Skipping mailbox because of missing last name"
159         }
160         else
161         {
162             Write-Warning "Skipping mailbox because of missing last name"
163         }
164     }
165     elseif ($Mailbox.PrimarySMTPAddress.ToLower().IndexOf("admin") -ge 0)
166     {
167         #Filter out admin accounts
168         Write-Output "Admin Account - Skipping"
169     }
170     elseif (($Mailbox.DisplayName.ToLower().IndexOf("test") -ge 0) -OR
171             ($Mailbox.Title.ToLower().IndexOf("test") -ge 0) -OR
172             ($Mailbox.LastName.ToLower().IndexOf("test") -ge 0))
173     {
174         #Filter out test accounts
175         Write-Output "Test account - Skipping"
176     }
177     elseif (($Mailbox.Title.ToLower().IndexOf("retiree") -ge 0) -OR
178             ($Mailbox.PrimarySMTPAddress.ToLower().IndexOf("-retiree") -ge 0) -OR
179             ($Mailbox.PrimarySMTPAddress.ToLower().IndexOf("-retired") -ge 0))
180     {
```

```
181     #Filter out retiree accounts
182     Write-Output "Retiree Account - Skipping"
183 }
184 elseif ($Mailbox.PrimarySMTPAddress.ToLower().IndexOf("onmicrosoft.com") -ge 0)
185 {
186     #Skip onmicrosoft.com accounts
187     Write-Output "Invalid domain name - Skipping"
188 }
189 elseif (($Mailbox.DisplayName.ToLower().IndexOf("conference") -ge 0) -OR
190         ($Mailbox.PrimarySMTPAddress.ToLower().IndexOf("conference") -ge 0))
191 {
192     #Filter out conference room accounts
193     Write-Output "Skipping conference room account"
194 }
195 elseif ($Mailbox.PrimarySMTPAddress.ToLower().IndexOf("proxy") -ge 0)
196 {
197     #Filter out proxy accounts
198     Write-Output "Skipping proxy account"
199 }
200 elseif ($Mailbox.PrimarySMTPAddress.ToLower().IndexOf("rni.") -ge 0 -OR
201         $Mailbox.PrimarySMTPAddress.ToLower().IndexOf("wvurni") -ge 0)
202 {
203     Write-Output "RNI Mailbox - Skipping"
204 }
205 elseif ($ExcludedAccounts.MailboxAlias -contains $Mailbox.PrimarySMTPAddress)
206 {
207     #Filter out accounts from the excluded mailbox list
208     Write-Output "Excluded Mailbox (from file) - Skipping"
209 }
210 else
211 {
212     Write-Output "Exporting user information"
213
214     $SelectProperties = @(
215         "DisplayName",
216         "FirstName",
217         "LastName",
218         "Title",
219         "Department",
220         "Phone",
221         "PrimarySMTPAddress",
222         "Alias",
223         "WhenChanged",
224         "hiddenfromaddresslistsenabled"
225     )
226
227     $Mailbox | Select-Object -Property $SelectProperties |
228         Export-Csv $HSCMailboxFile -Append
229 }
230
231 Write-Output "*****"
232 }
233
234 Write-Output "HSC mailbox processing complete"
235 if ((Import-Csv $HSCMailboxFile | Measure-Object).Count -gt $MinimumFileRecipients)
236 {
237     Write-Output "Preparing to email address book file."
238 }
239 else
240 {
```

```
241 Write-Warning "There was an error generating the file. Program is exiting."
242
243 $ErrorMailParams = @{
244     Body = "Error generating HSC mailbox for WVUF"
245     To = $ErrorRecipients
246     From = $From
247     Subject = "Error Generating HSC Mailbox File"
248     SMTPServer = $SMTPServer
249     Verbose = $true
250 }
251 Send-MailMessage @ErrorMailParams
252
253 Invoke-HSCExitCommand -ErrorCount $Error.Count
254 }
255
256 #####
257 # Send File Via Email #
258 #####
259
260 $SuccessEmailParams = @{
261     Body = "This is the HSC address book export."
262     To = $FileRecipients
263     From = $From
264     Subject = "$((Get-Date -format yyyy-MM-dd) + " HSC Address Book Export")
265     Attachments = $HSCMailboxFile
266     SmtServer = $SMTPServer
267     Verbose = $true
268     ErrorAction = "Stop"
269 }
270
271 try
272 {
273     Send-MailMessage @SuccessEmailParams
274     Write-Output "Successfully sent email"
275 }
276 catch
277 {
278     Write-Warning "There was an error attempting to send the email"
279     Write-Output $Error | Format-List
280 }
281
282 Invoke-HSCExitCommand -ErrorCount $Error.Count
```