```
#-----------------------------------------------
#Get-HSCMailbox.ps1
#
#Written by: Jeff Brusoe
#
#Last Modified: May 21, 2020
#
#Version: 2.0
#
#Exports the HSC address book to import to the
#WVU Foundation's Office 365 tenant.
#-----------------------------------------------

<#
.SYNOPSIS
The purpose of this file is to export the HSC address book. The export file that is
generated will
then be imported into the WVU Foundation's address book.

.PARAMETER NoSessionTranscript
This parameter disables writing the session transcript (via Start-Transcript) log
file.

.PARAMETER LogFilePath
Specifies the path of the log file. The default is c:\AD-
Development\WVUFoundationAddressBook\Logs.

.PARAMETER StopOnError
Used for debugging purposes. Stops program execution if an error is found.

.PARAMETER DaysToKeepLogFiles
Default is 7. Specifies the number of days to keep log files before they are
deleted.

.PARAMETER HSCMailboxFile
The output file generated by the program. It should not be changed from the default
without also
changing the value in the import file.

.PARAMETER PasswordFile
The path to the secure password file used to send email. To decrypt it, you need to
be logged into the machine
where it was created with the user account that it was created with.

.NOTES
Written by: Jeff Brusoe
Last Updated: May 21, 2020
#>

[CmdletBinding()]
param (
  #Common HSC PowerShell Parameters
  [switch]$NoSessionTranscript,
    [string]$LogFilePath = "$PSScriptRoot\Logs\",
  [switch]$StopOnError, #$true is used for testing purposes
  [int]$DaysToKeepLogFiles = 7, #this value used to clean old log files

  #File specific parameters
  [string]$HSCMailboxFile = "$PSScriptRoot\Logs\" + (Get-Date -Format yyyy-MM-dd) +
"-HSCMailboxExport.csv"
```

```
  #[string]$PasswordFile = ".\EncryptedPassword.txt"
)

################################
#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

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

#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#
##########################################

#Initialization of output files
New-Item -path $HSCMailboxFile -type "file" -force #-force will cause a new file to
be created. If one exists, it will be overwritten.
```

```

#Get HSC mailboxes
Write-Output "Generating list of HSC mailboxes"
Write-Output "HSC Mailbox File: $HSCMailboxFile"

try
{
  $Mailboxes = Get-Recipient -identity "*" -Resultsize Unlimited -RecipientType UserMailbox | where {$_.CustomAttribute7 -eq "Yes365"}
  Write-Output "Done generating list of HSC mailboxes"
}
catch
{
  Write-Warning "Unable to generate HSC recipient array. Program is exiting."
  Exit-Command
}

Write-Output "Getting list of excluded mailboxes"
$ExcludedAccounts = Import-Csv HSCExcludedAccounts.csv

Write-Output "Excluded Mailboxes:"
Write-Output $ExcludedAccounts

Write-Output "`nBeginning to process HSC mailboxes"
Write-Output $( "Total Number of Mailboxes: " + ($Mailboxes | Measure).count)

$MailBoxCount = 1

foreach ($Mailbox in $Mailboxes)
{
  if ($StopOnError -AND $Error.Count -gt 0)
  {
    Write-Warning "Error has occurred. Error message:"
    $Error | fl

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

  Write-Output $("Current Mailbox: " + $Mailbox.PrimarySMTPAddress)
  Write-Output "Mailbox Number: $MailboxCount"
  $MailboxCount++

  #This block of code verifies that the following before allowing an account to be written to the output file.
  #1. First and last name must both have values.
  #2. Remove
  # a. admin accounts
  # b. test accounts.
  # c. retiree accounts
  # d. invalid addresses (@wvuhsc.onmicrosoft.com)
  # e. conference rooms
  # f. proxy accounts
  #   g. Hospital accounts

  if (([string]::IsNullOrEmpty($Mailbox.LastName)) -OR ([string]::IsNullOrEmpty($Mailbox.FirstName)))
  {
    if ([string]::IsNullOrEmpty($Mailbox.LastName))
    {
```

```powershell
      Write-Warning "Skipping mailbox because of missing last name"
    }
    else
    {
      Write-Warning "Skipping mailbox because of missing last name"
    }
  }
  elseif ($Mailbox.PrimarySMTPAddress.toLower().IndexOf("admin") -ge 0)
  {
    #Filter out admin accounts
    Write-Output "Admin Account - Skipping"
  }
  elseif (($Mailbox.DisplayName.toLower().IndexOf("test") -ge 0) -OR ($Mailbox.Title.toLower().IndexOf("test") -ge 0) -OR ($Mailbox.LastName.toLower().IndexOf("test") -ge 0))
  {
    #Filter out test accounts
    Write-Output "Test account - Skipping"
  }
  elseif (($Mailbox.Title.toLower().IndexOf("retiree") -ge 0) -OR ($Mailbox.PrimarySMTPAddress.toLower().IndexOf("-retiree") -ge 0) -OR ($Mailbox.PrimarySMTPAddress.toLower().IndexOf("-retired") -ge 0))
  {
    #Filter out retiree accounts
    Write-Output "Retiree Account - Skipping"
  }
  elseif ($Mailbox.PrimarySMTPAddress.toLower().IndexOf("onmicrosoft.com") -ge 0)
  {
    #Skip onmicrosoft.com accounts
    Write-Output "Invalid domain name - Skipping"
  }
  elseif (($Mailbox.DisplayName.toLower().IndexOf("conference") -ge 0) -OR ($Mailbox.PrimarySMTPAddress.toLower().IndexOf("conference") -ge 0))
  {
    #Filter out conference room accounts
    Write-Output "Skipping conference room account"
  }
  elseif ($Mailbox.PrimarySMTPAddress.toLower().IndexOf("proxy") -ge 0)
  {
    #Filter out proxy accounts
    Write-Output "Skipping proxy account"
  }
  elseif ($Mailbox.PrimarySMTPAddress.toLower().IndexOf("rni.") -ge 0 -OR $Mailbox.PrimarySMTPAddress.toLower().indexOf("wvurni") -ge 0)
  {
    Write-Output "RNI Mailbox - Skipping"
  }
  elseif ($ExcludedAccounts.MailboxAlias -contains $Mailbox.PrimarySMTPAddress)
  {
    #Filter out accounts from the excluded mailbox list
    Write-Output "Excluded Mailbox (from file) - Skipping"
  }
  else
  {
    Write-Host "Exporting user information" -ForegroundColor Green
    $Mailbox | select DisplayName,FirstName,LastName,Title,Department,Phone,PrimarySMTPAddress,Alias,WhenChanged,hiddenfromaddresslistsenabled | Export-Csv $HSCMailboxFile -Append
  }

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

Write-Output "HSC mailbox processing complete"
if ((Import-Csv $HSCMailboxFile | Measure).Count -gt 4000)
{
  Write-Output "Preparing to email address book file."
}
else
{
  Write-Warning "There was an error generating the file. Program is exiting."
  Send-MailMessage -Body "Error generating HSC mailbox for WVUF" -To
"jbrusoe@hsc.wvu.edu","microsoft@hsc.wvu.edu" -From "microsoft@hsc.wvu.edu" -
Subject "Error Generating HSC Mailbox File" -SmtpServer "hssmtp.hsc.wvu.edu" -
Verbose
  Exit-Command

  Write-Warning "Exit-Command didn't work."
  Stop-Transcript
  return
}

#######################
# Send File Via Email #
#######################

$Recipients = "microsoft@hsc.wvu.edu","jbrusoe@hsc.wvu.edu","RHilling@wvuf.org"
$Attachments = $HSCMailboxFile

$Subject = (Get-Date -format yyyy-MM-dd) + " HSC Address Book Export"
$MsgBody = "This is the HSC address book export."

try
{
  Send-MailMessage -Body $MsgBody -To $Recipients -From "microsoft@hsc.wvu.edu" -
Subject $Subject -Attachments $Attachments -SmtpServer "hssmtp.hsc.wvu.edu" -
Verbose -ErrorAction Stop
  Write-Output "Successfully sent email"
}
catch
{
  Write-Warning "There was an error attempting to send the email"
  Write-Output $Error | FL
}


if (!$NoSessionTranscript)
{
  Stop-Transcript
}

#Exit
```