```powershell
<#
  .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 MinimumFileRecipients
    This parameter is used as a safety value to ensure that
    an empty address book isn't sent to WVUF.

  .PARAMETER Testing
    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.

  .NOTES
    Written by: Jeff Brusoe
    Last Updated: November 24, 2020
#>

[CmdletBinding()]
param (
  [ValidateRange(1,[int]::MaxValue)]
  [int]$MinimumFileRecipients = 4000,

  [ValidateNotNullOrEmpty()]
  [string]$HSCMailboxFile = "$PSScriptRoot\Logs\" +
                (Get-Date -Format yyyy-MM-dd) +
                "-HSCMailboxExport.csv",

  [ValidateNotNullOrEmpty()]
  [string]$ExcludedAccountFile = "$PSScriptRoot\HSCExcludedAccounts.csv",

  [ValidateNotNullOrEmpty()]
  [string[]]$ErrorRecipients = @("jbrusoe@hsc.wvu.edu","microsoft@hsc.wvu.edu"),

  [ValidateNotNullOrEmpty()]
  [string[]]$FileRecipients = @(
              "microsoft@hsc.wvu.edu",
              "jbrusoe@hsc.wvu.edu",
              "RHilling@wvuf.org"
              ),

  [ValidateNotNullOrEmpty()]
  [string]$From = "microsoft@hsc.wvu.edu",

  [ValidateNotNullOrEmpty()]
  [string]$SMTPServer = "hssmtp.hsc.wvu.edu",

  [switch]$Testing
)

# Configure PS environment
try {
  Set-HSCEnvironment -ErrorAction Stop
  Connect-HSCExchangeOnline -ErrorAction Stop
```

```powershell
}
catch {
  Write-Warning "Unable to configure environment. Program is exiting."
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

# Initialization of output files
New-Item -path $HSCMailboxFile -Type "file" -Force

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

try
{
  $Properties = @(
    "FirstName",
    "LastName",
    "Title",
    "Department",
    "Phone",
    "WhenChanged",
    "HiddenFromAddressListsEnabled",
    "CustomAttribute7"
  )

  $EXOMailboxParams = @{
    ResultSize = "Unlimited"
    Properties = $Properties
    RecipientType = "UserMailbox"
    ErrorAction = "Stop"
    Verbose = $true
  }
  $Mailboxes = Get-EXORecipient @EXOMailboxParams |
    Where-Object {
      ($_.CustomAttribute7 -eq "Yes365") -AND
      ($_.PrimarySMTPAddress -notlike "*rni.*") -AND
      ($_.PrimarySMTPAddress -notlike "*wvurni*")
    }

  Write-Output "Done generating list of HSC mailboxes"
}
catch
{
  Write-Warning "Unable to generate HSC recipient array. Program is exiting."
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

try {
  Write-Output "Getting list of excluded mailboxes"
  $ExcludedAccounts = Import-Csv $ExcludedAccountFile -ErrorAction Stop
}
catch {
  Write-Warning "Unable to import exclude accounts file"
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

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

```

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

$MailBoxCount = 1

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

    Write-Warning "`nProgram is exiting."
    Invoke-HSCExitCommand -ErrorCount $Error.Count
  }

  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($Mailbox.LastName))
    {
      Write-Warning "Skipping mailbox because of missing last name"
    }
    else
    {
      Write-Warning "Skipping mailbox because of missing last name"
    }
  }
  elseif ($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-Output "Exporting user information"

    $SelectProperties = @(
      "DisplayName",
      "FirstName",
      "LastName",
      "Title",
      "Department",
      "Phone",
      "PrimarySMTPAddress",
      "Alias",
      "WhenChanged",
      "hiddenfromaddresslistsenabled"
    )

    $Mailbox | Select-Object -Property $SelectProperties |
      Export-Csv $HSCMailboxFile -Append
  }

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

Write-Output "HSC mailbox processing complete"
if ((Import-Csv $HSCMailboxFile | Measure-Object).Count -gt $MinimumFileRecipients)
{
  Write-Output "Preparing to email address book file."
}
else
{
```

```powershell
  Write-Warning "There was an error generating the file. Program is exiting."

  $ErrorMailParams = @{
    Body = "Error generating HSC mailbox for WVUF"
    To = $ErrorRecipients
    From = $From
    Subject = "Error Generating HSC Mailbox File"
    SMTPServer = $SMTPServer
    Verbose = $true
  }
  Send-MailMessage @ErrorMailParams

  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

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

$SuccessEmailParams = @{
  Body = "This is the HSC address book export."
  To = $FileRecipients
  From = $From
  Subject = $((Get-Date -format yyyy-MM-dd) + " HSC Address Book Export")
  Attachments = $HSCMailboxFile
  SmtpServer = $SMTPServer
  Verbose = $true
  ErrorAction = "Stop"
}

try
{
  Send-MailMessage @SuccessEmailParams
  Write-Output "Successfully sent email"
}
catch
{
  Write-Warning "There was an error attempting to send the email"
  Write-Output $Error | Format-List
}

Invoke-HSCExitCommand -ErrorCount $Error.Count
```