```
<#
.SYNOPSIS
  This file is used to create HS domain and Office 365 accounts.

.DESCRIPTION
  This file is used to create HS domain and Office 365 accounts. It loops
  through the microsoft account looking for new account emails. Once it finds one
  it checks that the account has already been created by SailPoint in the NewUsers
org unit.

  The following steps are taken during the account creation process:
  1. Parse up new account email
  2. Search for account with matching WVUID
  3. If match is found, process CSC request only if a difference between
    whenCreated and PasswordLastSet exists.
  4. Move user to correct OU
  5. Create home directory (and add permissions to it)
  6. Add primary SMTP address (ext15@hsc.wvu.edu)
  7. Add remaining proxy addresses
    a. samaccountname@hsc.wvu.edu
    b. samaccountname@WVUHSC.onmicrosoft.com
    c. samaccountname@wvumedicine.org (Only for HVI users)
  8. Add user to groups
    a. O365 License Group
    b. DUO MFA Group
    c. Groups from GroupMapping file
  9. Set ext7 to Yes365
  10. Set name and display name
  11. Notify CSC

  .PARAMETER Testing

  .NOTES
    Author: Jeff Brusoe
    Last Updated: July 22, 2020
#>

[CmdletBinding()]
[Diagnostics.CodeAnalysis.SuppressMessageAttribute("PSAvoidTrailingWhiteSpace","",Justification = "Not relevant")]
[Diagnostics.CodeAnalysis.SuppressMessageAttribute("PSAvoidUsingCmdletAliases","",Justification = "Only MS Approved Aliases are Used")]
param (
  [switch]$Testing
)

# Initialize environment block
$Error.Clear()
Set-HSCEnvironment

#Define Constants
$NameSpace = "MAPI"
$NewAccountSubject = "Add New Account"
$NewUsersOrgUnit = "OU=NewUsers,DC=HS,DC=WVU-AD,DC=WVU,DC=EDU"
# End of environment initialization block

############################################################
# Step 1: Look through Outlook emails for new account ones #
############################################################
Add-Type -AssemblyName microsoft.office.interop.outlook
```

```
$olFolders = "Microsoft.Office.Interop.Outlook.OlDefaultFolders" -as [type]

$Outlook = New-Object -ComObject outlook.application
$namespace = $Outlook.GetNameSpace("mapi")
$inbox = $namespace.getdefaultfolder($olFolders::olFolderInbox)

Write-Verbose "Defining Source and Target Folders"
$SourceFolder =
$inbox.Parent.Folders.Item("Cabinet").Folders.Item("Accounts").Folders.Item("Add
New Account Requests").Folders.Item("InProgress")
$TargetFolder =
$inbox.Parent.Folders.Item("Cabinet").Folders.Item("Accounts").Folders.Item("Add
New Account Requests").Folders.Item("Completed")

if ($SourceFolder.items.count -eq 0)
{
  Write-Output "No emails waiting to be processed"
}
else
{
  $TotalEmail = ($SourceFolder.items | Measure).Count
  Write-Output "Number of emails waiting to be processed: $TotalEmail"
}

$Count = 0
$SourceFolder.items | foreach {

  #Begin by resetting variables
  $UserHomeDirectory = ""
  $HomeDirectoryPath = ""
  $CorrectedPathToOU = ""
  $Results = ""

  if ($_.Subject -eq $NewAccountSubject)
  {
    ###############################
    # Process Email Block of Code #
    ###############################
    $Count++
    $mBody = $_.body
    #Splits the line before any previous replies are loaded
    $mBodySplit = $mBody -split "From:"
    #Assigns only the first message in the chain
    $mBodyLeft = $mbodySplit[0]

    $q = "From: " + $_.SenderName + ("`n") + " Message: " + $mBodyLeft

    $MessageBodyArray = $mBodyLeft.trim().split("`n")

    Write-Output "`n`nReceived Email:"
    $MessageBodyArray

    Write-Output "Parsed email fields:"

    $CSCEmail = ($MessageBodyArray[1] -Split " ")[1].Trim()
    Write-Output "CSCEmail: $CSCEmail"

    #Department
    #Email Format: new_dept:  CHSC Human Resources [HR.ADMIN.CHAR_DIV.HSC]
    #The following code needs to do these tasks:
```

```
    #1. Remove "new_dept:" and a (not fixed) number of spaces before the actual
department name
    #2. Parse up the department name (CHSC Human Resources)
    #3. Get the actual location in AD which the department needs to go
(HR.ADMIN.CHAR_DIV.HSC).
    #4. Remove the square brackets around the location from 3

    $FullDepartment = ($MessageBodyArray[2] -Split " ")
    [string]$Department = $null

    for ($i=1;$i -lt $FullDepartment.length;$i++)
    {
      $Department += $FullDepartment[$i] + " "
    }

    $Department = $Department.Trim()
    Write-Output $("`nDepartment: " + $Department)

    #Note: [ is a special character and needs to be escaped to search for it.
    $DepartmentName = ($Department -Split "(\[)")[0].Trim()
    $DepartmentOrgUnit = ($Department -Split "(\[)")[2].Trim()
    $DepartmentOrgUnit = $DepartmentOrgUnit -replace "]", ""

    Write-Output "Department Name:  $DepartmentName"
    Write-Output "Org Unit: $DepartmentOrgUnit"

    $DepartmentOrgUnit = $DepartmentOrgUnit.Replace(".",",OU=")
    $DepartmentOrgUnit = "OU=" + $DepartmentOrgUnit

    Write-Output "Org Unit DN: $DepartmentOrgUnit"

    #First Name
    #Email Format: new_firstname: (Unknown amount of spaces) Robert
    $FirstName = ($MessageBodyArray[3] -Split " ")[1].Trim()
    Write-Output "`nFirst Name: $FirstName"

    #Last Name
    $LastName = ($MessageBodyArray[5] -Split " ")[1].Trim()
    Write-Output "`nLast Name: $LastName"

    #Title
    $Title = ($MessageBodyArray[6] -Split " ")[1].Trim()
    Write-Output "`nTitle: $Title"

    #Location
    $Location = ($MessageBodyArray[14] -Split ":")[1].Trim()
    Write-Output "`nLocation: $Location"

    #Phone
    $Phone = ($MessageBodyArray[15] -Split ":")[1].Trim()
    Write-Output "`nPhone: $Phone"

    #Fax
    $Fax = ($MessageBodyArray[16] -Split ":")[1].Trim()
    Write-Output "`nFax: $Fax"

    #Remote User
    $RemoteUser = ($MessageBodyArray[18] -Split ":")[1].Trim()
    Write-Output "`nRemoteUser: $RemoteUser"

```

```powershell
    #700 Number
    $WVUID = ($MessageBodyArray[20] -Split ":")[1].Trim()
    Write-Output "`n700 Number: $WVUID"

    #########################
    # Done Processing Email #
    #########################

    #Now search NewUsers to see if a 700 number exists
    $user = Get-ADUser -SearchBase $NewUsersOrgUnit -Properties
extensionAttribute11,whenCreated,PasswordLastSet -Filter * | where
{$_.extensionAttribute11 -eq $WVUID}

    [bool]$ProcessCSCRequest = $false

    if ($null -eq $user)
    {
      Write-Warning "The user was not found in NewUsers org unit."
      $Results += "Not Found: $WVUID`n"
    }
    else
    {
      Write-Output "User was found in NewUsers org unit"
      Write-Output $("Password Last Set: " + $user.PasswordLastSet)
      Write-Output $("When Created: " + $user.whenCreated)

      #Test time diffrence between password last set and when created
      [datetime]$PasswordLastSet = $user.PasswordLastSet
      [datetime]$WhenCreated = $user.whenCreated

      if ($PasswordLastSet.AddMinutes(-2) -le $WhenCreated)
      {
        #Unclaimed account
        Write-Output "Account is unclaimed. CSC request will not be processed"
      }
      else {
        #Claimed Account
        Write-Output "Account has been claimed. CSC request will be processed"
        $ProcessCSCRequest = $true
      }
    }

    if ($ProcessCSCRequest)
    {
      #Set user department
      Write-Output "Current Department Org Unit: $DepartmentOrgUnit"
      $TempCorrectedPathToOU = $DepartmentOrgUnit.Split(".")

      for ($i=$TempCorrectedPathToOU.Length-1;$i -ge 0; $i--)
      {
        $OrgUnitName = $TempCorrectedPathToOU[$i]
        $CorrectedPathToOU += "/" + $TempCorrectedPathToOU[$i]
      }

      #Now move user to new org unit
      Write-Output "`nMoving user from NewUsers to correct org unit"
      Write-Output "Corrected Path To Org Unit: $CorrectedPathToOU"

      $CanonicalName = "hs.wvu-ad.wvu.edu" + $CorrectedPathToOU
      Write-Output "Canonical Name: $CanonicalName"

      $OUDistinguishedName = Get-ADOrganizationalUnit -Filter * | Where {$_.DistinguishedName -like "*$DepartmentOrgUnit*"}

      if ($OUDistinguishedName -eq $null)
      {
        Write-Warning "Org unit not found"
      }
      else
      {
        Write-Output "Org unit found"
        Write-Output $OUDistinguishedName.DistinguishedName
      }

      try
      {
        Write-Output "Attempting to move user from NewUsers to:"
        Write-Output $OUDistinguishedName.DistinguishedName

        Write-Output "Successfully moved user"

        if (!$Testing)
        {
          $user | Move-ADObject -TargetPath $OUDistinguishedName.DistinguishedName -ErrorAction Stop
          Write-Output "Moved User"
        }
      }
      catch
      {
        Write-Warning "Error moving user"
      }

      Start-HSCCountdown -Message "Delay to allow time for user move to sync" -Seconds 10

      Write-Output "`nGenerating new user object after move"
      $usr = Get-ADUser -Filter * -Properties extensionAttribute15 | where {$_.UserPrincipalName -eq $user.UserPrincipalName}

      ##################################
      # Determine home directory path #
      ##################################

      [string]$HomeDirectoryPath = $null
      [string]$UserDN = $usr.DistinguishedName

      Write-Output $UserDN

      $HomeDirectoryInformation = Get-HSCDirectoryMapping -UserDN $UserDN -DetermineFullPath

      if ($HomeDirectoryInformation.FullPath)
      {
        $HomeDirectoryPath = $HomeDirectoryInformation.DirectoryPath
      }
      else {
        $HomeDirectoryPath = $null
      }

      #####################################################
      #The following block of code adds the proxy addresses.
      #####################################################

      #Note: This is being replaced with the Add-HSCProxyAddress function
      Write-Output "Adding ProxyAddresses"

      #Primary SMTP Address
      if ($usr.DistinguishedName.indexOf("HVI") -lt 0)
      {
        $PrimarySMTPAddress = $usr.extensionAttribute15 + "@hsc.wvu.edu"
      }
      else
      {
        $PrimarySMTPAddress = $usr.extensionAttribute15 + "@wvumedicine.org"
      }

      Write-Output "Setting Primary SMTP Address: $PrimarySMTPAddress"

      $PrimarySMTPAddress = "SMTP:" + $PrimarySMTPAddress
      $usr | Set-ADUser -Add @{ProxyAddresses=$PrimarySMTPAddress}
      $PrimarySMTPAddress = $PrimarySMTPAddress -replace "SMTP:",""

      Start-HSCCountdown -Message "Delay after adding primary SMTP address" -
Seconds 5
      #Primary SMTP has been added at this point.

      #Add samaccountname@wvuhsc.onmicrosoft.com
      $OnMicrosoftProxy = $usr.samaccountname + "@WVUHSC.onmicrosoft.com"
      Write-Output "Adding: $OnMicrosoftProxy"
      $OnMicrosoftProxy = "smtp:" + $OnMicrosoftProxy
      $usr | Set-ADUser -Add @{ProxyAddresses=$OnMicrosoftProxy}
      $OnMicrosoftProxy = $OnMicrosoftProxy -replace "smtp:",""

      #Verify samaccountname@hsc.wvu.edu is a proxy address
      $SAMProxyAddress = $usr.SamAccountName + "@hsc.wvu.edu"

      if ($SAMProxyAddress -ne $PrimarySMTPAddress)
      {
        Write-Output "Adding Proxy Address: $SAMProxyAddress"
        $SAMProxyAddress = "smtp:" + $SAMProxyAddress
        $usr | Set-ADUser -Add @{ProxyAddresses = $SAMProxyAddress}
      }

      if ($PrimarySMTPAddress.indexOf("wvumedicine.org") -gt 0)
      {
        $HVISAMProxyAddress = $SAMProxyAddress -replace
"hsc.wvu.edu","wvumedicine.org"

        if ($HVISAMProxyAddress -ne $PrimarySMTPAddress) {
          Write-Output "Adding Proxy Address: $HVISAMProxyAddress"
          $HVISAMProxyAddress = "smtp:" + $HVISAMProxyAddress
          $usr | Set-ADUser -Add @{ProxyAddresses = $HVISAMProxyAddress}

        }
      }

      Start-HSCCountdown -Message "Delay to allow proxyaddresses to sync before
adding SIP address" -Seconds 5

```

```
      $SIPAddress = "SIP:" + $usr.SamAccountName + "@hsc.wvu.edu"
      Write-Output "Adding SIP address: $SIPAddress"

      $usr | Set-ADUser -Add @{ProxyAddresses = $SIPAddress}

      Start-HSCCountdown -Message "Delay after adding SIP address" -Seconds 5

      ######################################################
      #End of proxy address block
      ######################################################

      ######################
      # Add user to groups #
      ######################

      #Step 2: Add users to the DUO MFA and O365 license groups
      #Add to O365 License Group
      $UserDN = $usr.DistinguishedName

      Set-HSCGroupMembership -UserDN $UserDN

      ##############################################
      # End of block of code to add user to groups #
      ##############################################

      #########################
      # Create home directory #
      #########################

      Write-Output $HomeDirectoryPath

      if ($HomeDirectoryPath -eq "NoHomeDirectory")
      {

      }
      else
      {
        $UserHomeDirectory = $HomeDirectoryPath + "\" + $usr.samaccountname

        Write-Output "Creating Home Directory: $UserHomeDirectory"
        if (!$Testing)
        {
          New-Item -ItemType directory -Path $UserHomeDirectory
        }
        else
        {
          Write-Warning "Code testing is being done. Home directory will not be
created"
        }

        Start-HSCCountdown -Message "Home Directory Created. Delay before adding
permissions." -Seconds 10

        #Now add file system permissions
        if (!$Testing)
        {
          $UserName = "HS\" + $usr.samaccountname
          $Acl = (Get-Item $UserHomeDirectory).GetAccessControl('Access')
          $Ar = New-Object
System.Security.AccessControl.FileSystemAccessRule($Username,
'FullControl','ContainerInherit,ObjectInherit', 'None', 'Allow')
          $Acl.SetAccessRule($Ar)
          Set-Acl -path $UserHomeDirectory -AclObject $Acl
        }
      }

      #Add Yes365
      Write-Output "Setting extensionAttribute7 to Yes365"
      $usr | Set-ADUser -Add @{extensionAttribute7="Yes365"}

      #If users is in RedCapUsers OU, set extensionAttribute7 to No365
      #To do: Verify if this if statement is still needed
      if ($DepartmentOrgUnit -like "*RedCapUsers*")
      {
        $usr | Set-Aduser -Replace @{extensionAttribute7="No365"}
      }

      Start-HSCCountdown -Message "Synchronization Delay" -seconds 2

      #Set name and display name
      Write-Output "Setting name and display name"
      $DisplayName = $user.Surname + ", " + $user.GivenName
      $usr | Set-ADUser -DisplayName $DisplayName

      Start-HSCCountdown -Message "Delay to allow time for all user information to
sync" -Seconds 10

      #Add user to file that sends out email
      #$Output = $usr.mail + "," + $CSCEmail + "," + $user.samaccountname
      #Add-Content -path $LicenseFile -value $Output

      #$Results += "Created: $UserName`n"

      #Move email to correct destination folder in Outlook
      $_.Move($TargetFolder)

      #Now send CSC email
      Send-HSCNewAccountEmail -CSCEmail $CSCEmail -NewUserSamAccountName
$usr.SamAccountName -PrimarySMTPAddress

      Invoke-HSCExitCommand
      exit
    }
  } #End parsing of single email
} #End of foreach looping through the inbox

Stop-Transcript
```