```powershell
<#
.SYNOPSIS
  Compares the value of extensionAttribute15 with the primary SMTP address.

.DESCRIPTION
  This file gets the value from extensionAttribute15 (which should be
  the primary SMTP address email prefix) and generates a report by comparing
  it to the AD mail attribute and the primary SMTP address obtained
  from the proxyAddresses field. SailPoint populates extensionAttribute15, and
  this is not a field that we should manually change.

.PARAMETER SkipBlankExt15
  Tells the file to skip blank values for extensionAttribute 15. These are probably
  not maintained by SailPoint, and it doesn't make sense to run the comparison.

.NOTES
  Compared-Ext15WithPrimarySMTPAddress.ps1
  Written by: Jeff Brusoe
  Last Updated: August 10, 2020
#>

[CmdletBinding()]
[Diagnostics.CodeAnalysis.SuppressMessageAttribute("PSAvoidTrailingWhiteSpace","",Justification = "Not relevant")]
[Diagnostics.CodeAnalysis.SuppressMessageAttribute("PSAvoidUsingCmdletAliases","",Justification = "Only MS Provided Aliases are Used")]
param (
  [switch]$SkipBlankExt15 #A blank field normally implies that this account isn't maintained by SailPoint
)

#########################
# Configure Environment #
#########################
$Error.Clear()

Set-HSCEnvironment

$SummaryFile = "$PSScriptRoot\Logs\" + (Get-Date -format yyyy-MM-dd-HH-mm) + "-EmailFieldsSummary.csv"
New-Item -type File -Path $SummaryFile  -Force | Out-Null

$NoMatchFile = "$PSScriptRoot\Logs\" + (Get-Date -format yyyy-MM-dd-HH-mm) + "-NoMatchFile.csv"
New-Item -type File -Path $NoMatchFile -Force | Out-Null

##########################################
# End of environment configuration block #
##########################################

Write-Output "Generating list of AD users"
try
{
  $ADUsers = Get-ADUser -Filter * -Properties proxyAddresses,mail,extensionAttribute15 -ErrorAction Stop -SearchBase "OU=HSC,DC=hs,DC=wvu-ad,DC=wvu,DC=edu"
  Write-Output "Successfully generated AD user list"
}
catch
{
  Write-Warning "Unable to get list of AD users. Program is exiting."
  Invoke-HSCExitCommand
}

foreach ($ADUser in $ADUsers)
{
  Write-Output $("Current User: " + $ADUser.SamAccountName)

  #Get mail attribute
  try {
    [string]$ADMail = $ADUser.mail
    if ([string]::IsNullOrEmpty($ADMail))
    {
      Write-Verbose "AD mail field is empty"
      $ADMail = "Mail Attribute Not Present"
    }
  }
  catch {
    Write-Warning "Unable to get mail attribute"
    $ADMail = "Mail Attribute Not Present"
  }

  #Get ext15 attribute
  try {
    [string]$ext15 = $ADUser.extensionAttribute15
    if ([string]::IsNullOrEmpty($ext15))
    {
      Write-Verbose "extensionAttribute15 is empty"
      $ext15 = "Ext15 Not Present"
    }
  }
  catch {
    Write-Warning "Unable to retrieve extensionAttribute15"
    $ext15 = "Ext15 Not Present"
  }

  # Determine Primary SMTP Address from proxyAddresses field
  try {
    [string[]]$ProxyAddressArray = $ADUser  | select -ExpandProperty proxyAddresses
    Write-Output "Proxy Address Array:"
    $ProxyAddressArray
  }
  catch {
    Write-Warning "Unable to read proxy address field. Program is exiting"
    return
  }

  if ($null -ne $ProxyAddressArray)
  {
    $PrimarySMTPCount = ($ProxyAddressArray | where {$_ -clike "SMTP:*" } |
Measure).Count
    Write-Output "`nPrimary SMTP Address Cound: $PrimarySMTPCount"
  }
  else {
    $PrimarySMTPCount = 0
  }

  if ($PrimarySMTPCount -eq 0)
  {
    Write-Verbose "PrimarySMTPAddress wasn't found"
```

```powershell
    $PrimarySMTPAddress = "PrimarySMTP Not Present"
  }
  elseif ($PrimarySMTPCount -eq 1)
  {
    Write-Verbose "Single Primary SMTP address was found."
    $PrimarySMTPAddress = $ProxyAddressArray | where {$_ -clike "SMTP:*" }
    $PrimarySMTPAddress = $PrimarySMTPAddress -replace "SMTP:",""
  }
  else
  {
    #This shouldn't be reached, but I have seen a few cases of this.
    Write-Warning "Multiple Primary SMTP addresses were found."

    foreach ($ProxyAddress in ($ProxyAddressArray |  where {$_ -clike "SMTP:*"}))
    {
      Write-Output "Primary SMTP Address: $ProxyAddress"
    }

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

  Write-Output "Attribute Summary:"
  Write-Output "Mail Attribute: $ADMail"
  Write-Output "Primary SMTP Address: $PrimarySMTPAddress"
  Write-Output "extensionAttribute15: $ext15"

  #Begin to do comparison
  $UserInfo = New-Object -type PSObject

  $UserInfo | Add-Member -MemberTYpe NoteProperty -Name SamAccountName -Value $ADUser.SamAccountName
  $UserInfo | Add-Member -MemberType NoteProperty -Name ADMailAttribute -Value $ADMail
  $UserInfo | Add-Member -MemberType NoteProperty -Name PrimarySMTPAddress -Value $PrimarySMTPAddress
  $UserInfo | Add-Member -MemberTYpe NoteProperty -Name extensionAttribute15 -Value $ext15
  $UserInfo | Add-Member -MemberType NoteProperty -Name DistinguishedName -Value $ADUser.DistinguishedName

  if ($PrimarySMTPAddress.indexOf("@") -gt 0)
  {
    #This is a valid email address
    try {
      $EmailPrefix = $PrimarySMTPAddress.substring(0,$PrimarySMTPAddress.indexOf("@"))
      Write-Output "Email Prefix: $EmailPrefix"
    }
    catch {
      Write-Warning "Unable to generate email prefix"
      $EmailPrefix = "No email prefix"
    }
  }
  else
  {
    Write-Output "Unable to generate email prefix"
    $EmailPrefix = "No email prefix"
  }

  $UserInfo | Add-Member -MemberType NoteProperty -Name EmailPrefix -Value $EmailPrefix

  if ((!$SkipBlankExt15) -OR (($ext15 -ne "Ext15 Not Present") -AND ($SkipBlankExt15)))
  {
    $UserInfo | Export-Csv $SummaryFile -Append

    if ($EmailPrefix -ne $ext15)
    {
      $UserInfo | Export-Csv $NoMatchFile -Append
    }
  }

  $UserInfo = $null

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

Invoke-HSCExitCommand
```