```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 SearchBase
    The distinguished name of where to being searching for users

  .PARAMETER Properties
    These are the properties in addition to the default AD user
    properties that are pulled.

  .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 11, 2021
#>

[CmdletBinding()]
param (
  [ValidateNotNullOrEmpty()]
  [string]$SearchBase = "OU=HSC,DC=hs,DC=wvu-ad,DC=wvu,DC=edu",

  [ValidateNotNullOrEmpty()]
  [string[]]$Properties = @("proxyAddresses",
              "mail",
              "extensionAttribute15"
            ),

  #Blank normally implies that account isn't maintained by SailPoint
  [switch]$SkipBlankExt15
)

try {
  Write-Output "Configuring Environment"
  Set-HSCEnvironment -ErrorAction Stop

  Write-Output "Search Base: $SearchBase"
}
catch {
  Write-Warning "Unable to configure environment"
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

try {
  Write-Output "Configuring summary files"

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

  $NoMatchFile = "$PSScriptRoot\Logs\" +
        (Get-Date -format yyyy-MM-dd-HH-mm) +
        "-NoMatchFile.csv"
  New-Item -type File -Path $NoMatchFile -Force -ErrorAction Stop
}
catch {
  Write-Warning "Error creating log files"
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

try {
  Write-Output "Generating list of AD users"

  $GetADUserParams = @{
    Filter = "*"
    SearchBase = $SearchBase
    Properties = $Properties
    ErrorAction = "Stop"
  }

  $ADUsers = Get-ADUser @GetADUserParams
  Write-Output "Successfully generated AD user list"
}
catch {
  Write-Warning "Unable to get list of AD users. Program is exiting."
  Invoke-HSCExitCommand -ErrorCount $Error.Count
}

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

  #Get mail attribute
  try {
    [string]$ADMail = $ADUser.mail

    if ([string]::IsNullOrEmpty($ADMail)) {
      Write-Output "AD mail field is empty"
      $ADMail = "Mail Attribute Not Present"
    }
    else {
      Write-Output "AD mail field: $ADMail"
    }
  }
  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-Output "extensionAttribute15 is empty"
      $ext15 = "Ext15 Not Present"
    }
    else {
      Write-Output "extensionAttribute15: $ext15"
    }
  }
  catch {
    Write-Warning "Unable to retrieve extensionAttribute15"
    $ext15 = "Ext15 Not Present"
  }

  # Determine Primary SMTP Address
  try {
    $GetHSCParams = @{
      UserNames = $ADUser.SamAccountName
      ErrorAction = "Stop"
    }
    $PrimarySMTPAddress = (Get-HSCPrimarySMTPAddress @GetHSCParams).PrimarySMTPAddress
  }
  catch {
    Write-Warning "Unable to determine PrimarySMTPAddress field"
  }

  Write-Output "`n`nAttribute Summary:"
  Write-Output "Mail Attribute: $ADMail"

  if ($null -eq $PrimarySMTPAddress) {
    Write-Output "Primary SMTP Address is null"
  }
  else {
    Write-Output "Primary SMTP Address: $PrimarySMTPAddress"
  }

  Write-Output "extensionAttribute15: $ext15"

  #Begin to do comparison
  $UserInfo = [PSCustomObject]@{
    SamAccountName = $ADUser.SamAccountName
    ADMailAttribute = $ADMail
    PrimarySMTPAddress = $PrimarySMTPAddress
    extensionAttribute15 = $ext15
    DistinguishedName = $ADUser.DistinguishedName
  }

  if ($PrimarySMTPAddress -AND $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"
  }

  try {
    $AddMemberParams = @{
      MemberType = "NoteProperty"
      Name = "EmailPrefix"
      Value = $EmailPrefix
      ErrorAction = "Stop"
    }
    $UserInfo | Add-Member @AddMemberParams
  }
  catch {
    Write-Warning "Unable to add email prefix to user object"
  }

  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 -ErrorCount $Error.Count
```