```
<#
  .SYNOPSIS
    This module contains functions that are commonly used to do administative work wit
the HSC Office 365 tenant.

  .DESCRIPTION
    Functions contained in this file:
    1. Get-HSCConnectionAccount
    2. Connect-HSCOffice365 (AzureAD)
    3. Connect-HSCOffice365MSOL
    4. Connect-HSCExchangeOnline
    5. Connect-HSCExchangeOnlineV1
    6. Get-O365MailboxStatus
    7. Get-HSCGlobalAdminMember
    8. Get-HSCRoleMember
    9. Get-UserLicense *
    10. Set-UserLicense *
    11. Set-CommonUserParameter *
    12. Get-TenantName
    13. Get-TenantNameMSOL
    14. Set-BlockCredential (MSOL) *
    15. Enable-HSCCloudUser (AzureAD) *
    16. Disable-HSCPOP *
    17. Disable-HSCIMAP *

  .NOTES
    HSC-Office365Module.psm1
    Written by: Jeff Brusoe
    Last Modified by: Jeff Brusoe
    Last Modified: June 22, 2020
#>

#June 15, 2020 - This file is currently undergoing testing and updates. DO NOT USE IN
ANY FILE UNTIL THIS IS DONE!

[CmdletBinding()]
[Diagnostics.CodeAnalysis.SuppressMessageAttribute("PSAvoidTrailingWhiteSpace","",Just
fication = "Not relevant")]
[Diagnostics.CodeAnalysis.SuppressMessageAttribute("PSAvoidUsingCmdletAliases","",Just
fication = "Only MS Provided Aliases are Used")]
param()

function Get-HSCConnectionAccount
{
  <#
    .SYNOPSIS
      Randomly determines which of the microsoft accounts will be used to connected tc
the HSC Office 365 tenant

    .NOTES
      Written by: Jeff Brusoe
      Last Updated: June 16, 2020
  #>
```

```powershell
  [CmdletBinding()]
  [alias("Get-ConnectionAccount")]
  [OutputType([String])]
  param ()

  $AccountNumber = Get-Random -Minimum 1 -Maximum 5
  $ConnectionAccount = "microsoft$AccountNumber@hsc.wvu.edu"

  Write-Output "Connection Account: $ConnectionAccount" | Out-Host

  Return $ConnectionAccount
}

function Connect-HSCOffice365
{
  <#
    .SYNOPSIS
      This function establishes a connection to the HSC Office 365 tenant with the AzureAD cmdlets.

    .OUTPUTS
      True/False based on if connection was successful

    .PARAMETER EncryptedFilePath
      The path to the encrypted password file

    .NOTES
      Written by: Jeff Brusoe
      Last Updated: June 16, 2020
  #>

  [CmdletBinding()]
  [Alias("Connect-Office365")]
  [OutputType([bool])]
  param (
    [string]$EncryptedFilePath = $(Get-HSCEncryptedFilePath)
  )

  begin
  {
    $Error.Clear()

    Import-Module AzureAD

    Write-Output "Connecting to Office 365..." | Out-Host
  }

  process
  {
    $Account = Get-HSCConnectionAccount

    $Password = cat $EncryptedFilePath  | ConvertTo-SecureString
    $Credential = New-Object -Typename System.Management.Automation.PSCredential -ArgumentList $Account, $Password

```

```
103     try
104     {
105       Connect-AzureAD -Credential $Credential -ErrorAction Stop
106       Write-Output "Authenticated to Office 365`n" | Out-Host
107
108       return $true
109     }
110     catch
111     {
112       Write-Warning "Unable to authenticate to the Office 365 tenant with AzureAD
    cmdlets" | Out-Host
113       return $false
114     }
115   }
116 }
117
118 function Connect-HSCOffice365MSOL
119 {
120   <#
121     .SYNOPSIS
122       This function establishes a connection to the HSC Office 365 tenant with the MSC
    cmdlets.
123
124     .NOTES
125       Written by: Jeff Brusoe
126       Last Updated: June 23, 2020
127   #>
128
129   [CmdletBinding()]
130   [Alias("Connect-Office365MSOL")]
131   [OutputType([bool])]
132
133   param (
134     [string]$EncryptedFilePath = $(Get-EncryptedFilePath)
135   )
136
137   begin
138   {
139     $Error.Clear()
140     Write-Output "Connecting to Office 365 with MSOL cmdlets.." | Out-Host
141
142     Import-Module MSOnline
143
144     $Account = Get-HSCConnectionAccount
145   }
146
147   process
148   {
149     $Password = cat $EncryptedFilePath  | ConvertTo-SecureString
150     $Credential = New-Object -Typename System.Management.Automation.PSCredential
    -ArgumentList $Account, $Password
151
152     try
153     {
154       Connect-MSOLService -Credential $Credential -ErrorAction Stop
```

```
155       Write-Output "Authenticated to Office 365 with MSOL cmdlets`n" | Out-Host
156
157       return $true
158     }
159     catch
160     {
161       Write-Warning "Unable to authenticate to the Office 365 tenant with MSOL cmdlets
    | Out-Host
162
163       return $false
164     }
165   }
166 }
167
168 function Connect-HSCExchangeOnline
169 {
170   <#
171     .SYNOPSIS
172       This function establishes a connection to ExchangeOnline with V2 of the Exchange
    Online cmdlets
173
174     .NOTES
175       Written by: Jeff Brusoe
176       Last Updated: June 16, 2020
177   #>
178
179   [CmdletBinding()]
180   [OutputType([bool])]
181   param (
182     [string]$EncryptedFilePath = $(Get-HSCEncryptedFilePath)
183   )
184
185   Write-Output "Connecting to Exchange Online with V2 cmdlets.." | Out-Host
186
187   $Account = Get-HSCConnectionAccount
188
189   $Password = cat $EncryptedFilePath  | ConvertTo-SecureString
190   $Credential = New-Object -Typename System.Management.Automation.PSCredential
    -ArgumentList $Account, $Password
191
192   try
193   {
194     Connect-ExchangeOnline -Credential $Credential -ShowProgress $true -ErrorAction
    Stop
195
196     Write-Output "`nSuccessfully authenticated to Exchange Online with V2 cmdlets`n"
197
198     return $true
199   }
200   catch
201   {
202     Write-Warning "There was an error connecting to Exchange online with V2 cmdlets.`r
203
204     return $false
205   }
```

```
206
207 }
208
209 function Connect-HSCExchangeOnlineV1
210 {
211   <#
212     .SYNOPSIS
213       This function establishes a connection to ExchangeOnline using the older Exchang
    cmdlets
214
215     .NOTES
216       Written by: Jeff Brusoe
217       Last Updated: June 16, 2020
218   #>
219
220   [CmdletBinding()]
221   [Alias("Connect-ExchangeOnlineV1")]
222   [OutputType([bool])]
223
224   param (
225     [string]$EncryptedFilePath = $(Get-EncryptedFilePath)
226   )
227
228   $Error.Clear()
229
230   Import-Module MSOnline
231
232   Write-Output "Connecting to Exchange Online with V1 cmdlets.." | Out-Host
233
234   $Account = Get-HSCConnectionAccount
235
236   $Password = cat $EncryptedFilePath  | ConvertTo-SecureString
237   $Credential = New-Object -Typename System.Management.Automation.PSCredential
    -ArgumentList $Account, $Password
238
239   try
240   {
241     $Session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri
    https://ps.outlook.com/powershell/ -Credential $Credential -Authentication Basic
    -AllowRedirection -ErrorAction Stop
242     Import-Module (Import-PSSession -Session $Session -AllowClobber
    -DisableNameChecking -Verbose:$false) -Global
243     #Import-PSSession $Session
244
245     Export-ModuleMember -Variable $Session
246
247     Write-Output "`nSuccessfully authenticated to Exchange Online and downloadeded
    PowerShell cmdlets`n"
248     return $true
249   }
250   catch
251   {
252     Write-Warning "There was an error connecting to Exchange online.`n"
253     return $false
254   }
```

```
255 }
256
257 Function Get-HSCO365MailboxStatus
258 {
259   <#
260     .SYNOPSIS
261       This function gets the OWAEnabled, MAPIEnabled, and ActiveSyncEnabled values frc
    O365 of all mailboxes.
262       It also exports to a CSV and returns an array of these values.
263
264     .NOTES
265       Needed for Export-ToSole.ps1
266       Originally Written by: Matt Logue(?)
267       Last Updated by: Jeff Brusoe
268       Last Updated: June 17, 2020
269   #>
270
271   [CmdletBinding()]
272   [Alias("Get-O365MailboxStatus")]
273   [OutputType([PSObject])]
274   param (
275     [string]$ExportFile = $($MyInvocation.PSScriptRoot + "\O365MailboxStatus.csv")
276   )
277
278     if (Test-Path $ExportFile)
279     {
280         Write-Verbose "Cleaning Up Old Export File" | Out-Host
281         Remove-Item -Path $ExportFile -Force
282     }
283
284     Write-Verbose "Getting Mailboxes in Office 365" | Out-Host
285     $users = get-casmailbox -resultsize unlimited | Where { $_.PrimarySMTPAddress
    -notlike "*rni.*" -OR $_.PrimarySMTPAddress -notlike "*wvurni" }
286
287     [PSObject[]]$MailboxStatus = @()
288     [PSObject[]]$O365Enabled = @()
289
290     Write-Verbose "Getting information for users with SIP addresses...." | Out-Host
291     foreach ($user in $users)
292     {
293         $userArray = New-Object -TypeName psobject
294
295     if ($user.emailaddresses -clike "SMTP:*")
296         {
297             $usersmtp = $user.EmailAddresses -clike "SMTP:*" | Select-String 'SMTP:'
298             $usersmtp = $usersmtp -replace "SMTP:"
299             $usersmtp = $usersmtp.ToLower()
300
301
302             $userArray | Add-Member -Name "O365EmailAddress" -Value $usersmtp
    -MemberType NoteProperty
303             $userArray | Add-Member -Name "OWAEnabled" -Value $user.OWAEnabled
    -MemberType NoteProperty
304             $userArray | Add-Member -Name "MAPIEnabled" -Value $user.MAPIEnabled
    -MemberType NoteProperty
```

```
305             $userArray | Add-Member -Name "ActiveSyncEnabled" -Value
    $user.ActiveSyncEnabled -MemberType NoteProperty
306             $MailboxStatus += $userArray
307
308             if (($userArray.OWAEnabled -eq $true) -and ($userArray.MAPIEnabled -eq
    $true) -and ($userArray.ActiveSyncEnabled -eq $true))
309             {
310                 $O365Enabled += $userArray
311             }
312
313         }
314         elseif ($user.emailaddresses -clike "SIP:*")
315         {
316             $usersip = $user.EmailAddresses | Select-String 'sip:'
317             $usersip = $usersip -replace "sip:"
318             $usersip = $usersip.ToLower()
319
320
321             $userArray | Add-Member -Name "O365EmailAddress" -Value $usersip
    -MemberType NoteProperty
322             $userArray | Add-Member -Name "OWAEnabled" -Value $user.OWAEnabled
    -MemberType NoteProperty
323             $userArray | Add-Member -Name "MAPIEnabled" -Value $user.MAPIEnabled
    -MemberType NoteProperty
324             $userArray | Add-Member -Name "ActiveSyncEnabled" -Value
    $user.ActiveSyncEnabled -MemberType NoteProperty
325
326             $MailboxStatus += $userArray
327
328             if (($userArray.OWAEnabled -eq $true) -and ($userArray.MAPIEnabled -eq
    $true) -and ($userArray.ActiveSyncEnabled -eq $true))
329             {
330                 $O365Enabled += $userArray
331             }
332         }
333     }
334
335     Write-Verbose '$MailboxStatus Array created, Exporting to CSV' | Out-Host
336     $MailboxStatus | select o365emailaddress,owaenabled,mapienabled,activesyncenabled
    Export-Csv -Path $ExportFile -NoTypeInformation
337     Write-Verbose "CSV Exported - Mailbox Count: $(($MailboxStatus.O365EmailAddress |
    Measure-Object).Count)" | Out-Host
338
339     return $O365Enabled
340 }
341
342 function Get-HSCGlobalAdminMember
343 {
344   <#
345     .SYNOPSIS
346       This function returns an array of AzureADUsers that are Global Admins in the HSC
    Office 365 Tenant.
347
348     .NOTES
349       Written by: Jeff Brusoe
```

```
350       Last Updated: June 18, 2020
351   #>
352
353   [CmdletBinding()]
354   [Alias("Get-GlobalAdminMember")]
355   [OutputType([PSObject[]])]
356   param()
357
358   try
359   {
360     $GlobalAdminRole = Get-AzureADDirectoryRole | where {$_.DisplayName -like "Company
    Administrator"} -ErrorAction Stop
361
362     $GlobalAdmins = Get-AzureADDirectoryRoleMember -ObjectId $GlobalAdminRole.ObjectID
    -ErrorAction Stop
363     return $GlobalAdmins
364   }
365   catch
366   {
367     Write-Warning "Unable to get list of global admins" | Out-Host
368     return $null
369   }
370 }
371
372 function Get-HSCRoleMember
373 {
374   <#
375     .SYNOPSIS
376       This function returns an array of AzureADUsers that are Global Admins in the HSC
    Office 365 Tenant.
377
378     .NOTES
379       Written by: Jeff Brusoe
380       Last Updated: June 18, 2020
381   #>
382
383   [CmdletBinding()]
384   [OutputType([PSObject[]])]
385   param(
386     [parameter(Mandatory=$true)]
387     [string]$DisplayName
388   )
389
390   Write-Verbose "Search for: $DisplayName" | Out-Host
391
392   try
393   {
394     $HSCRole = Get-AzureADDirectoryRole | where {$_.DisplayName -eq $DisplayName}
    -ErrorAction Stop
395
396     $HSCRoleMembers = Get-AzureADDirectoryRoleMember -ObjectId $HSCRole.ObjectID
    -ErrorAction Stop
397     return $HSCRoleMembers
398   }
399   catch
```

```
400   {
401     Write-Warning "Unable to get member list" | Out-Host
402     rteturn $null
403   }
404
405   return $null
406 }
407
408 function Get-HSCUserLicense
409 {
410   <#
411     .SYNOPSIS
412       Returns the licese information for a user with AzureAD
413
414     .NOTES
415       Written by: Jeff Brusoe
416       Last Updated: June 19, 2020
417   #>
418
419   [CmdletBinding()]
420   [OutputType([PSObject])]
421   param(
422     [parameter(Mandatory=$true)]
423     [string]$UserName
424   )
425
426   Write-Verbose "Getting license information for: $UserName" | Out-Host
427
428   try
429   {
430
431   }
432   catch
433   {
434     Write-Warning "Unable to find license information for user: $UserName" | Out-Host
435     return $null
436   }
437 }
438
439 function Set-HSCUserLicense
440 {
441   return $null
442 }
443
444 function Set-HSCCommonUserParameter
445 {
446   return $null
447 }
448
449 function Get-HSCTenantName
450 {
451   <#
452     .SYNOPSIS
453       Returns the name of the currently logged in tenant with AzureAD
454
```

```powershell
    .NOTES
      Written by: Jeff Brusoe
      Last Updated: June 22, 2020
  #>

  [CmdletBinding()]
  [Alias("Get-TenantName")]
  [OutputType([string])]
  param ()

  try
  {
    $TenantDetail = Get-AzureADTenantDetail -ErrorAction Stop
  }
  catch
  {
    Write-Warning "Unable to get AzureAD tenant information" | Out-Host
    return $null
  }

  try
  {
    $TenantName = $TenantDetail.VerifiedDomain
    Write-Verbose "Verified Domain: $TenantName" | Out-Host

    $TenantName = $TenantName -replace ".onmicrosoft.com"
    Write-Output "Tenant Name: $TenantName" | Out-Host

    return $TenantName
  }
  catch
  {
    Write-Warning "Error reading AzureAD tenant name" | Out-Host
    return $null
  }

  return $null
}

function Get-HSCTenantNameMSOL
{
  <#
    .SYNOPSIS
      Returns the name of the currently logged in tenant with MSOnline

    .NOTES
      Written by: Jeff Brusoe
      Last Updated: June 22, 2020
  #>

  [CmdletBinding()]
  [Alias("Get-TenantNameMSOL")]
  [OutputType([string])]
  param ()

```

```
510   try
511   {
512     $TenantName = Get-MsolDomain | where {($_.Name.Split(".").Length -eq 3) -AND
    ($_.Name -like "*onmicrosoft.com*")}
513     Write-Output "Tenant Name: $TenantName" | Out-Host
514
515     return $TenantName
516
517   }
518   catch
519   {
520     Write-Warning "Unable to get MSOL tenant name" | Out-Host
521     return $null
522   }
523 }
524
525 function Set-HSCBlockCredential
526 {
527   <#
528     .SYNOPSIS
529       Sets the block credential using MSOnline
530
531     .NOTES
532       Written by: Jeff Brusoe
533       Last Updated: June 22, 2020
534   #>
535
536   [CmdletBinding()]
537   [OutputType([bool])]
538   param ()
539
540   return $false
541 }
542
543 function Enable-HSCCloudUser
544 {
545   <#
546     .SYNOPSIS
547       Enables a user with AzureAD
548
549     .NOTES
550       Written by: Jeff Brusoe
551       Last Updated: June 22, 2020
552   #>
553
554   [CmdletBinding()]
555   [OutputType([bool])]
556   param ()
557
558   return $false
559 }
560
561 function Disable-HSCPOP
562 {
563   return $null
```

```
564 }
565
566 function Disable-HSCIMAP
567 {
568   return $null
569 }
570
571 ####################
572 # Export Functions #
573 ####################
574
575 #Connect Modules
576 Export-ModuleMember -Function "Connect-HSCOffice365" -Alias "Connect-Office365"
577 Export-ModuleMember -Function "Connect-HSCOffice365MSOL" -Alias "Connect-Office365MSOL
578 Export-ModuleMember -Function "Connect-HSCExchangeOnline"
579 Export-ModuleMember -Function "Connect-HSCExchangeOnlineV1" -Alias "Connectd-
    ExchangeOnlineV1"
580
581 #Get Modules
582 Export-ModuleMember -Function "Get-HSCConnectionAccount" -Alias "Get-ConnectionAccount
583 Export-ModuleMember -Function "Get-HSCO365MailboxStatus" -Alias "Get-
    HSCO365MailboxStatus"
584 Export-ModuleMember -Function "Get-HSCGlobalAdminMember" -Alias "Get-
    HSCGlobalAdminMember"
585 Export-ModuleMember -Function "Get-HSCRoleMember"
586 Export-ModuleMember -Function "Get-HSCTenantName" -Alias "Get-TenantName"
587 Export-ModuleMember -Function "Get-HSCTenantNameMSOL" -Alias "Get-TenantNameMSOL"
```