

```
1 # Enable-AccountSecurityCompliance.ps1
2 # Written by: Jeff Brusoe
3 # Last Updated: January 4, 2021
4 #
5 # This account is designed to enable an Office365 mailbox
6 # after a user has passed their security compliance (HIPAA or IT Security)
7 # training.
8 [CmdletBinding()]
9 param (
10     [ValidateNotNullOrEmpty()]
11     [string]$DBName = "SecurityAwareness",
12
13     [ValidateNotNullOrEmpty()]
14     [string]$DBTableName = "EnableExchange",
15
16     [ValidateNotNullOrEmpty()]
17     [string]$SaveLogDirectory = "$PSScriptRoot\LogToSave\"
18 )
19
20 try {
21     Write-Output "Configuring HSC Environment"
22     $SessionTranscriptFile = Set-HSCEnvironment -ErrorAction Stop
23
24     Write-Output "Connecting to Exchange Online"
25     Connect-HSCExchangeOnlineV1 -ErrorAction Stop
26
27     $NewDisable = $false
28 }
29 catch {
30     Write-Warning "Unable to configure environment. Program is exiting."
31     Invoke-HSCExitCommand -ErrorCount $Error.Count
32 }
33
34 try {
35     Write-Output "Decrypting SQL Password & generating connection string"
36
37     $SQLPassword = Get-HSCSQLPassword -SOLEDB -Verbose -ErrorAction Stop
38
39     $GetHSCSQLConnectionStringParams = @{
40         Password = $SQLPassword
41         Database = $DBName
42         ErrorAction = "Stop"
43     }
44
45     $SQLConnectionString = Get-HSCSQLConnectionString
46     @GetHSCSQLConnectionStringParams
47 }
48 catch {
49     Write-Warning "Unable to decrypt SQL Password"
50     Invoke-HSCExitCommand -ErrorCount $Error.Count
51 }
52
53 try {
54     #Query table for any users who have passed compliance training
55     $Query = "select * from $DBTableName"
56
57     $InvokeSQLCmdParams = @{
58         Query = $Query
59         ConnectionString = $SQLConnectionString
```

```

59     ErrorAction = "Stop"
60 }
61
62 $AccountsToEnable = Invoke-SqlCmd @InvokeSQLCmdParams
63 }
64 catch {
65     Write-Warning "Unable to query SQL DB"
66     Invoke-HSCEExitCommand -ErrorCount $Error.Count
67 }
68
69 $TotalCount = 1
70
71 if (($AccountsToEnable | Measure-Object).Count -gt 0)
72 {
73     foreach ($AccountToEnable in $AccountsToEnable)
74     {
75         Write-Output "Total Count: $TotalCount"
76         $TotalCount++
77         $NewDisable = $false
78
79         Write-Output "New Disable Count: $NewDisableCount"
80
81         Write-Output $("Current email: " + $AccountToEnable.Email)
82         Write-Output $("Backup Email: " + $AccountToEnable.BackupEmail)
83
84         if (($AccountToEnable.Email -like "*mix.wvu.edu*") -AND
85             ([string]::IsNullOrEmpty($AccountToEnable.BackupEmail)))
86         {
87             Write-Output "Primary email is @mix.wvu.edu"
88
89             $DeleteQuery = "Delete From $DBTableName Where Email='" +
90                 $AccountToEnable.Email + "'"
91             Write-Output "Delete Query: $DeleteQuery"
92
93             $InvokeSQLCmdParams.Query = $DeleteQuery
94             try {
95                 Invoke-SqlCmd @InvokeSQLCmdParams
96             }
97             catch {
98                 Write-Warning "Unable top delete from DB"
99             }
100         }
101     }
102     else
103     {
104         try
105         {
106             $CasMailbox = Get-CASMailbox $AccountToEnable.Email -ErrorAction Stop
107
108             if ($CasMailbox.OWAEnabled -AND
109                 $CasMailbox.MAPIEnabled -AND
110                 $CasMailbox.ActiveSyncEnabled)
111             {
112                 Write-Output $("MapiEnabled: " + $CasMailbox.MapiEnabled)
113                 Write-Output $("ActiveSyncEnabled: " + $CasMailbox.ActiveSyncEnabled)
114                 Write-Output $("OWAEnabled: " + $CasMailbox.OWAEnabled)
115
116                 Write-Output "Mailbox is enabled."
117
118                 $DeleteQuery = "Delete From $DBTableName Where Email='" +

```

```

119         $AccountToEnable.Email + ""
120     Write-Output "Delete Query: $DeleteQuery"
121
122     $InvokeSQLCmdParams.Query = $DeleteQuery
123
124     Invoke-SqlCmd @InvokeSQLCmdParams
125 }
126 else
127 {
128     #Newly enabled mailbox
129
130     $SetCASMailBoxParams = @{
131         Identity = $AccountToEnable.Email
132         OWAEnabled = $true
133         MAPIEnabled = $true
134         ActiveSyncEnabled = $true
135         ErrorAction = "Stop"
136     }
137
138     Set-CASMailbox @SetCASMailBoxParams
139
140     $NewDisable = $true
141 }
142 }
143 catch
144 {
145     if ([string]::IsNullOrEmpty($AccountToEnable.BackupEmail)) {
146         Write-Output "Backup email is empty"
147     }
148     else
149     {
150         try {
151             $SetCASMailBoxParams.Identity = $AccountToEnable.BackupEmail
152
153             Set-CASMailbox @SetCASMailBoxParams
154             Get-CASMailbox $AccountToEnable.BackupEmail -ErrorAction Stop
155         }
156         catch {
157             Write-Warning "Unable to enable account"
158         }
159     }
160
161     try {
162         $DeleteQuery = "Delete From $DBTableName Where Email='" +
163             $AccountToEnable.Email + "'"
164         Write-Output "Delete Query: $DeleteQuery"
165
166         Write-Output "Running Delete Query"
167
168         $InvokeSQLCmdParams.Query = $DeleteQuery
169         Invoke-SqlCmd @InvokeSQLCmdParams
170     }
171     catch {
172         Write-Warning "Unable to execute delete query"
173     }
174 }
175 }
176
177 Write-Output "*****"
178 }

```

```
179 }
180 else {
181     Write-Output "No updates available at $(Get-Date)"
182 }
183
184 if ($NewDisable)
185 {
186     #An account is newly disabled. Being moved to Logs to keep directory.
187     try {
188         Copy-Item -Path $SessionTranscriptFile -Destination $SaveLogDirectory
189     }
190     catch {
191         Write-Warning "Error moving session transcript to logs to save directory"
192     }
193 }
194
195 Invoke-HSCExitCommand -ErrorCount $Error.Count
```