

```
1 function Add-HSCProxyAddress
2 {
3     <#
4         .SYNOPSIS
5             This function adds a single (or multiple) proxy addresses to an AD Account.
6
7         .DESCRIPTION
8             The purpose of this function is to add a new proxy address to a specified
9             user which
10            can be passed into the function as either an ADUser object or a string
11            representing the SAMAccountName. The
12            function returns either true or false based on if this operation is
13            successful.
14
15        .PARAMETER ADUser
16            Specifies an ADUser object to add proxy address(es) to.
17
18        .PARAMETER SamAccountName
19            This parameter is a string representing an AD user's SAM account name
20            that will be searched for to add the proxy address.
21
22        .PARAMETER NewProxyAddress
23            The new proxy addresses to be added
24
25        .PARAMETER Primary
26            A switch parameter to indicate that this is a primary SMTP address
27
28        .EXAMPLE
29            PS C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository> Get-ADUser
30            jefftest -Properties * | select proxyAddresses -ExpandProperty proxyAddresses
31
32            PS C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository> Add-
33            HSCProxyAddress jefftest -NewProxyAddress "jefftest@hsc.wvu.edu" -Primary
34            Attempting to find user: jefftest
35            User found
36            Adding: SMTP:jefftest@hsc.wvu.edu
37            Successfully added new proxy address
38            True
39
40            PS C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository> Get-ADUser
41            jefftest -Properties * | select proxyAddresses -ExpandProperty proxyAddresses
42            SMTP:jefftest@hsc.wvu.edu
43
44        .EXAMPLE
45            PS C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository> Get-ADUser
46            jefftest -Properties * | select proxyAddresses -ExpandProperty proxyAddresses
47            SMTP:jefftest@hsc.wvu.edu
48
49            PS C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository> Add-
50            HSCProxyAddress jefftest -NewProxyAddress "jefftest2@hsc.wvu.edu"
51            Attempting to find user: jefftest
52            User found
53            Current Proxy Address from ADUser: SMTP:jefftest@hsc.wvu.edu
54            Adding: smtp:jefftest2@hsc.wvu.edu
55            Successfully added new proxy address
56            True
57
58            PS C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository> Get-ADUser
59            jefftest -Properties * | select proxyAddresses -ExpandProperty proxyAddresses
60            smtp:jefftest2@hsc.wvu.edu
```

```
52 SMTP:jefftest@hsc.wvu.edu
53
54 .NOTES
55     Written by: Jeff Brusoe
56     Last Updated: October 14, 2020
57 #>
58
59 [CmdletBinding(SupportsShouldProcess=$true)]
60 [OutputType([bool])]
61
62 param (
63     [Parameter(ValueFromPipeline=$false,
64         ParameterSetName="ADUser",
65         Mandatory=$true,
66         Position=0)]
67     [ValidateNotNullOrEmpty()]
68     [Microsoft.ActiveDirectory.Management.ADAccount]$ADUser,
69
70     [Parameter(ValueFromPipeline=$false,
71         ParameterSetName="SamAccountName",
72         Mandatory=$true,
73         Position=0)]
74     [ValidateNotNullOrEmpty()]
75     [string]$SAMAccountName,
76
77     [Parameter(ValueFromPipeline=$true,
78         Mandatory=$true,
79         Position=1)]
80     [Alias("NewProxyAddresses")]
81     [ValidateNotNullOrEmpty()]
82     [string[]]$NewProxyAddress,
83
84     [switch]$Primary
85 )
86
87 begin
88 {
89     Write-Verbose $("Parameter Set Name: " + $PSCmdlet.ParameterSetName)
90
91     if ($PSCmdlet.ParameterSetName -eq "ADUser") {
92         Write-Verbose $("ADUser SamAccountName: " + $ADUser.SamAccountName)
93     }
94     else {
95         Write-Verbose "SamAccountName: $SamAccountName"
96     }
97
98     $ProxyAdded = $false
99 }
100
101 process
102 {
103     if (!(Test-HSCADModuleLoaded))
104     {
105         Write-Verbose "Attempting to load AD Module"
106
107         try {
108             Import-Module ActiveDirectory -ErrorAction Stop
109             $ProxyAdded = $true
110         }
111         catch {
```

```
112     Write-Warning "Unable to load Active Directory module"
113     $ProxyAdded = $false
114 }
115 }
116 else {
117     Write-Verbose "AD module is already loaded"
118     $ProxyAdded = $true
119 }
120
121 Write-Verbose $("In process block - Parameter Set Name: " +
$PSCmdlet.ParameterSetName)
122
123 if ($PSCmdlet.ParameterSetName -eq "SamAccountName" -AND $ProxyAdded)
124 {
125     try {
126         Write-Output "Attempting to find user: $SamAccountName" | Out-Host
127
128         $LDAPFilter = "&(objectCategory=person)(objectClass=user)
(sAMAccountName=$SamAccountName)"
129         Write-Verbose "LDAP Filter: $LDAPFilter"
130
131         $ADUser = Get-ADUser -LDAPFilter $LDAPFilter -properties proxyAddresses -
ErrorAction Stop
132
133         if ($null -eq $ADUser) {
134             Write-Warning "Unable to find AD User"
135             $ProxyAdded = $false
136         }
137         else {
138             Write-Output "User found" | Out-Host
139             Write-Verbose "Distinguished Name:"
140             Write-Verbose $ADUser.DistinguishedName
141
142             $ProxyAdded = $true
143         }
144     }
145     catch {
146         Write-Warning "Unable to find AD user based on SamAccountName"
147     }
148 }
149
150 if ($ProxyAdded)
151 {
152     #Verify one unique user was found
153     try {
154         $ADUserCount = ($ADUser | Measure-Object).Count
155         Write-Verbose "AD User Count: $ADUserCount"
156
157         if ($ADUserCount -eq 1)
158         {
159             Write-Verbose "One unique AD User was found"
160         }
161         else {
162             Write-Warning "AD user count doesn't equal 1"
163             $ProxyAdded = $false
164         }
165     }
166     catch {
167         Write-Warning "Unable to determine AD User Count"
168         $ProxyAdded = $false
169     }
170 }
```

```

169     }
170 }
171
172 if ($ProxyAdded)
173 {
174     #At this point, one unique AD user object should be found.
175     #Now get proxy addresses to verify that the proxy address isn't already
present
176     #or that it's not the primary SMTP address
177     $ProxyAddresses = $ADUser.proxyAddresses
178
179     foreach ($ProxyAddress in $ProxyAddresses)
180     {
181         Write-Output "Current Proxy Address from ADUser: $ProxyAddress" | Out-Host
182
183         if (($ProxyAddress.indexOf($NewProxyAddress) -ge 0) -AND ($ProxyAddress -
clike "*SMTP*"))
184         {
185             Write-Output "New Proxy address is the primary SMTP address."
186             $ProxyAdded = $false
187         }
188         elseif ($ProxyAddress.indexOf($NewProxyAddress) -ge 0)
189         {
190             Write-Output "New proxy address is already a proxy address" | Out-Host
191             $ProxyAdded = $false
192         }
193     }
194 }
195
196 if ($ProxyAdded)
197 {
198     #Now proxy address will be added to AD account
199     $NewProxyAddress = $NewProxyAddress.Trim()
200
201     if ($Primary)
202     {
203         $NewProxyAddress = "SMTP:" + $NewProxyAddress
204     }
205     else {
206         $NewProxyAddress = "smtp:" + $NewProxyAddress
207     }
208
209     Write-Output "Adding: $NewProxyAddress"
210
211     try {
212         if ($PSCmdlet.ShouldProcess("Adding new proxy address"))
213         {
214             Write-Verbose "About to set proxy address"
215             $ADUser | Set-ADUser -Add @{proxyAddresses=$NewProxyAddress} -ErrorAction
Stop
216         }
217
218         Write-Output "Successfully added new proxy address" | Out-Host
219     }
220     catch {
221         Write-Warning "Unable to add new proxy address"
222         $ProxyAdded = $false
223     }
224 }
225 }

```

```
226  
227     end  
228     {  
229         return $ProxyAdded  
230     }  
231 }
```