```
1  ##########################################################################################
   ##########
2  #
   Disclaimer
          #
3  #
   ==========
          #
4  #
            #
5  # Please read this disclaimer carefully before using this script. This script is
   open-source and  #
6  # subject to the terms of the MIT
   license.                                                    #
7  #
            #
8  # This script is provided as a tool for IT Administrators to programatically  remove
   RingCentral  #
9  # legacy applications (RingCentral Phone, RingCentral Meetings and the legacy
   RingCentral App)    #
10 # from End Users' devices.  We also highly recommend that you first test this script to
   ensure    #
11 # that it achieves the desired
   results.                                                      #
12 #
            #
13 # RingCentral makes no representation as to the script containing any errors or bugs.
   Any bugs   #
14 # or errors in the script may produce an undesirable outcome.  Additionally, any
   modification or  #
15 # unintentional change made by you may have undesirable effects. If you discover any
   issue with   #
16 # the script, you should immediately cease use and manage the removal of RingCentral
   applications #
17 #
   manually.
          #
18 #
            #
19 # Your use of this software is undertaken at your own risk. To the full extent permitted
   under    #
20 # law, RingCentral will not be liable for any loss or damage of whatever nature (direct,
   indirect,#
21 # consequential or other) caused by the use of this
   script.                                       #
22 #
            #
23 ##########################################################################################
   ##########
24
25
26 ##########################################################################################
   ##########
27 # Change
   Log
   #
28 #
   ==========
          #
29 # Version  Date
   Reason                                                          #
30 # -------  -----------
   ------------------------------------------------------------------------ #
31 # 1.0      19-Mar-2021  Initial script created by Andy
   Connolly                                  #
32 # 2.0      07-May-2021  New version that runs in the administrator context
   only                     #
33 # 2.1      13-May-2021  Added additional locations based on customer
   feedback                        #
```

```
# 2.2      01-Jun-2021  Attempt to uninstall any remaining HKLM installations                     #
#                                                                                                 #
###################################################################################################

$ErrorActionPreference = "Stop"
$logfile = "C:\temp\$(gc env:computername)-remove-RCApps.log"
$dtFormat = 'dd-MMM-yyyy HH:mm:ss'
add-content $logfile -value "-----------------------------------------------------------------------------------------------"
add-content $logfile -value "$(Get-Date -Format $dtFormat) Attempting to remove RC apps"

$isAdmin = (New-Object Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent())).IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)
add-content $logfile -value "$(Get-Date -Format $dtFormat) Running in Administrator context: $isAdmin"

if (!$isAdmin){
    add-content $logfile -value "$(Get-Date -Format $dtFormat) Script must be executed as an administrator: powershell.exe -noprofile -executionpolicy Bypass -file `"admin.ps1`""
    exit(-5)
}

#Stop any of the applications that may be running
get-process | where-object {$_.Company -like "*RingCentral*" -or $_.Path -like "*RingCentral*"} | stop-process -ErrorAction ignore -Force

#Uninstall any RingCentral installed applications that the administrator can remove
foreach ($app in (Get-WmiObject -Class Win32_Product | Where-Object{$_.Vendor -like "*RingCentral*"})) {
    add-content $logfile -value "$(Get-Date -Format $dtFormat) Attempting to uninstall $($app)"
    try {
        $app.Uninstall() | Out-Null
    } catch {
        add-content $logfile -value $_
    }
}

#Remove any system uninstall keys
$paths = @("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall",
           "HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall")
foreach($path in $paths) {
    if (test-path($path)) {
        $list = Get-ItemProperty "$path\*" | Where-Object {$_.DisplayName -like "*RingCentral*"} | Select-Object -Property PSPath, UninstallString
        foreach($regkey in $list) {
            add-content $logfile -value "$(Get-Date -Format $dtFormat) Examining Registry Key $($regkey.PSpath)"
            try {
                $cmd = $regkey.UninstallString
                if ($cmd -like "msiexec.exe*") {
                    add-content $logfile -value "$(Get-Date -Format $dtFormat) Uninstall string is using msiexec.exe"
                    if ($cmd -notlike "*/X*") {
                        add-content $logfile -value "$(Get-Date -Format $dtFormat) no /X flag - this isn't for uninstalling"
                        $cmd = ""
                    } #don't do anything if it's not an uninstall
                    elseif ($cmd -notlike "*/qn*") {
                        add-content $logfile -value "$(Get-Date -Format $dtFormat) adding /qn flag to try and uninstall quietly"
                        $cmd = "$cmd /qn"
                    } #don't display UI
```

```
85                  }
86                  if ($cmd) {
87                      add-content $logfile -value "$(Get-Date -Format $dtFormat)
                        executing $($cmd)"
88                      cmd.exe /c "$($cmd)"
89                      add-content $logfile -value "$(Get-Date -Format $dtFormat)     done"
90                  }
91              } catch {
92                  add-content $logfile -value $_
93              }
94          }
95          $list = Get-ItemProperty "$path\*" | Where-Object {$_.DisplayName -like
            "*RingCentral*"} | Select-Object -Property PSPath
96          foreach($regkey in $list) {
97              add-content $logfile -value "$(Get-Date -Format $dtFormat) Removing Registry
                Key $($regkey.PSpath)"
98              try {
99                  remove-item $regkey.PSPath -recurse -force
100             } catch {
101                 add-content $logfile -value $_
102             }
103         }
104     } ##else { add-content $logfile -value "$(Get-Date -Format $dtFormat) Path $($item)
        not found" }
105 }
106
107 #Add shortcut to HKEY_USERS
108 New-PSDrive -PSProvider registry -Root HKEY_USERS        -Name HKU  | Out-Null
109
110 if (test-path(${Env:ProgramFiles(x86)})) { $pf86 = ${Env:ProgramFiles(x86)} }  else {
    $pf86 = "C:\Program Files (x86)" }
111 add-content $logfile -value "$(Get-Date -Format $dtFormat) Program Files (x86) location:
    $($pf86)"
112
113 if (test-path(${Env:ProgramFiles}))      { $pf = ${Env:ProgramFiles} }         else {
    $pf = "C:\Program Files" }
114 add-content $logfile -value "$(Get-Date -Format $dtFormat) Program Files location:
    $($pf)"
115
116 if (test-path(${Env:ProgramData}))       { $pd = ${Env:ProgramData} }          else {
    $pd = "C:\ProgramData" }
117 add-content $logfile -value "$(Get-Date -Format $dtFormat) ProgramData location: $($pd)"
118
119 if (test-path(${Env:PUBLIC}))            { $pub = ${Env:PUBLIC} }              else {
    $pub = "C:\Users\Public" }
120 add-content $logfile -value "$(Get-Date -Format $dtFormat) Public profile location:
    $($pub)"
121
122 if (test-path(${Env:SystemRoot}))        { $win = ${Env:SystemRoot} }          else {
    $win = "C:\Windows" }
123 add-content $logfile -value "$(Get-Date -Format $dtFormat) Windows root location:
    $($win)"
124
125 #Populate the lists of items to remove
126 $Brand = "RingCentral"  #RingCentral/TELUS/ATT/Avaya/BT/Rainbow/Unify
127 add-content $logfile -value "$(Get-Date -Format $dtFormat) Brand set to: $($Brand)"
128
129 $HKLM = [System.Collections.ArrayList]@()
130 $HKLM.add("HKLM:\SOFTWARE\$Brand") | Out-Null
131 $HKLM.add("HKLM:\SOFTWARE\Classes\.rcrecord") | Out-Null
132 $HKLM.add("HKLM:\SOFTWARE\Classes\.zoomrc") | Out-Null
133 $HKLM.add("HKLM:\SOFTWARE\Classes\MIME\Database\Content
    Type\application/x-rcmtg-launcher") | Out-Null
134 $HKLM.add("HKLM:\SOFTWARE\Classes\RCLauncher") | Out-Null
135 $HKLM.add("HKLM:\SOFTWARE\Classes\RingCentralMeetingsRecording") | Out-Null
136 $HKLM.add("HKLM:\SOFTWARE\Classes\rcapp") | Out-Null
137 $HKLM.add("HKLM:\SOFTWARE\Classes\rcmobile") | Out-Null
138 $HKLM.add("HKLM:\SOFTWARE\Classes\rcsp") | Out-Null
139 $HKLM.add("HKLM:\SOFTWARE\Classes\rcuk") | Out-Null
```

```
$HKLM.add("HKLM:\SOFTWARE\Classes\rcvdt") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\Classes\RingCentral.callto") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\Classes\RingCentral.fax") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\Classes\RingCentral.rcmobile") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\Classes\RingCentral.rcsp") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\Classes\RingCentral.rcuk") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\Classes\RingCentral.tel") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\Classes\zoomrc") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\IM Providers\RCIM") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\InternetPrinters") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\Microsoft\Office\Outlook\Addins\RingCentralForOutlook") |
Out-Null
$HKLM.add("HKLM:\SOFTWARE\RingCentralForOutlook") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\RingCentralInternetFax") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\RingCentralMeetings") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\$Brand") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\.rcrecord") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\.zoomrc") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\MIME\Database\Content
Type\application/x-rcmtg-launcher") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\RCLauncher") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\RingCentralMeetingsRecording") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\rcapp") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\rcmobile") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\rcsp") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\rcuk") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\rcvdt") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\RingCentral.callto") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\RingCentral.fax") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\RingCentral.rcmobile") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\RingCentral.rcsp") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\RingCentral.rcuk") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\RingCentral.tel") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Classes\zoomrc") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\IM Providers\RCIM") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\InternetPrinters") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\Microsoft\Office\Outlook\Addins\RingCentralForOutlo
ok") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\RingCentralForOutlook") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\RingCentralInternetFax") | Out-Null
$HKLM.add("HKLM:\SOFTWARE\WOW6432Node\RingCentralMeetings") | Out-Null

foreach ($regkey in $HKLM) {
    try {
        if (test-path($regkey)) {
            add-content $logfile -value "$(Get-Date -Format $dtFormat) Removing Registry
            Key $($regkey)"
            remove-item $regkey -recurse -force
        } ##else { add-content $logfile -value "$(Get-Date -Format $dtFormat) Registry
        Key $($regkey) not found" }
    } catch {
        add-content $logfile -value $_
    }
}

$MachineFolders = [System.Collections.ArrayList]@()
$MachineFolders.add("$pd\Glip") | Out-Null
$MachineFolders.add("$pd\Microsoft\Windows\Start Menu\Programs\*RingCentral*") | Out-Null
$MachineFolders.add("$pf86\$Brand\SoftPhoneApp") | Out-Null
$MachineFolders.add("$pf86\Common Files\RingCentral") | Out-Null
$MachineFolders.add("$pf86\Glip") | Out-Null
$MachineFolders.add("$pf86\RingCentral Classic Installer") | Out-Null
$MachineFolders.add("$pf86\RingCentral Installer") | Out-Null
$MachineFolders.add("$pf86\RingCentralForOutlook") | Out-Null
$MachineFolders.add("$pf86\RingCentralMeetings") | Out-Null
$MachineFolders.add("$pf\$Brand\SoftPhoneApp") | Out-Null
$MachineFolders.add("$pf\Common Files\RingCentral") | Out-Null
$MachineFolders.add("$pf\Glip") | Out-Null
$MachineFolders.add("$pf\RingCentral Classic Installer") | Out-Null
```

```
$MachineFolders.add("$pf\RingCentral Installer") | Out-Null
$MachineFolders.add("$pf\RingCentralForOutlook") | Out-Null
$MachineFolders.add("$pf\RingCentralMeetings") | Out-Null
$MachineFolders.add("$pub\Desktop\RingCentral*.lnk") | Out-Null
$MachineFolders.add("$win\Prefetch\*GLIP*.pf") | Out-Null
$MachineFolders.add("$win\Prefetch\*RINGCENTRAL*.pf") | Out-Null
$MachineFolders.add("$win\Prefetch\*SOFTPHONE*.pf") | Out-Null

foreach ($item in $MachineFolders) {
    try {
        if (test-path($item)) {
            add-content $logfile -value "$(Get-Date -Format $dtFormat) Removing $($item)"
            remove-item $item -recurse -force
        } ##else { add-content $logfile -value "$(Get-Date -Format $dtFormat) Path
        $($item) not found" }
    } catch {
        add-content $logfile -value $_
    }
}

#Loop through HKLM key to remove RC entries
$paths = @("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\Folders",
           "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\UFH\ARP",

           "HKLM:\SYSTEM\ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\Fi
           rewallRules",

           "HKLM:\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolic
           y\FirewallRules")
foreach($path in $paths) {
    if (test-path($path)) {
        add-content $logfile -value "$(Get-Date -Format $dtFormat) Checking registry
        path: $($path)"
        Get-Item -Path $path | Select-Object -ExpandProperty Property | % {
            $propValue = (Get-ItemProperty -Path "$path" -Name "$_")."$_"
            if (($_ -like "*RingCentral*") -or ($propValue -like "*RingCentral*")) {
                try {
                    add-content $logfile -value "$(Get-Date -Format $dtFormat)
                    Removing property: $($_) containing value: $($propValue)"
                    Remove-ItemProperty -path "$path" -Name $_
                } catch {
                    add-content $logfile -value $_
                }
            }
        }
    } ##else { add-content $logfile -value "$(Get-Date -Format $dtFormat) Path $($item)
    not found" }
}

#Build list of items that need to be removed for each user profile
$HKU = [System.Collections.ArrayList]@()
$HKU.add("HKU:\%SID%\SOFTWARE\$Brand") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\584acf4c-ebc3-56fa-9cfd-586227f098ba") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\Clients\Internet Call\RingCentral for Windows") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\IM Providers\RCIM") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\Microsoft\Internet Explorer\ProtocolExecute\zoomrc") |
Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\Microsoft\Internet Explorer\Zoom") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\Microsoft\Windows\CurrentVersion\ApplicationAssociationToas
ts\rcapp_rcapp") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\Microsoft\Windows\CurrentVersion\ApplicationAssociationToas
ts\zoomrc_zoomrc") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\MozillaPlugins\@ringcentral.com/RingCentralMeetingsPlugin")
 | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\RingCentral Softphone") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\$Brand") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\Classes\MIME\Database\Content
Type\application/x-rcmtg-launcher") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\Classes\rcapp") | Out-Null
```

```
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\Classes\rcmobile") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\Classes\rcvdt") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\Classes\zoomrc") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\IM Providers\RCIM") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\Clients\Internet Call\RingCentral for Windows") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\Microsoft\Internet Explorer\ProtocolExecute\zoomrc") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\Microsoft\Internet Explorer\Zoom") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\ApplicationAssociationToasts\rcapp_rcapp") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\ApplicationAssociationToasts\zoomrc_zoomrc") | Out-Null
$HKU.add("HKU:\%SID%\SOFTWARE\WOW6432Node\RingCentral Softphone") | Out-Null
$HKU.add("HKU:\%SID%_classes\.rcrecord") | Out-Null
$HKU.add("HKU:\%SID%_classes\.zoomrc") | Out-Null
$HKU.add("HKU:\%SID%_classes\MIME\Database\Content Type\application/x-rcmtg-launcher") | Out-Null
$HKU.add("HKU:\%SID%_classes\RCLauncher") | Out-Null
$HKU.add("HKU:\%SID%_classes\RingCentralMeetingsRecording") | Out-Null
$HKU.add("HKU:\%SID%_classes\RingCentralMeetingsRecording") | Out-Null
$HKU.add("HKU:\%SID%_classes\rcapp") | Out-Null
$HKU.add("HKU:\%SID%_classes\rcmobile") | Out-Null
$HKU.add("HKU:\%SID%_classes\rcsp") | Out-Null
$HKU.add("HKU:\%SID%_classes\rcuk") | Out-Null
$HKU.add("HKU:\%SID%_classes\rcvdt") | Out-Null
$HKU.add("HKU:\%SID%_classes\RingCentral.callto") | Out-Null
$HKU.add("HKU:\%SID%_classes\RingCentral.fax") | Out-Null
$HKU.add("HKU:\%SID%_classes\RingCentral.rcmobile") | Out-Null
$HKU.add("HKU:\%SID%_classes\RingCentral.rcsp") | Out-Null
$HKU.add("HKU:\%SID%_classes\RingCentral.rcuk") | Out-Null
$HKU.add("HKU:\%SID%_classes\RingCentral.tel") | Out-Null
$HKU.add("HKU:\%SID%_classes\zoomrc") | Out-Null

$UserFolders = [System.Collections.ArrayList]@()
$UserFolders.add("%desktop%\RingCentral*.lnk") | Out-Null
$UserFolders.add("%local%\$Brand") | Out-Null
$UserFolders.add("%local%\$Brand\SoftPhoneApp") | Out-Null
$UserFolders.add("%local%\Glip") | Out-Null
$UserFolders.add("%local%\Programs\RingCentral") | Out-Null
$UserFolders.add("%local%\RingCentral") | Out-Null
$UserFolders.add("%local%\RingCentral\RingCentral Classic") | Out-Null
$UserFolders.add("%local%\RingCentral\RingCentral") | Out-Null
$UserFolders.add("%local%\SquirrelTemp") | Out-Null
$UserFolders.add("%local%\Temp\Glip Crashes") | Out-Null
$UserFolders.add("%local%\ringcentral-updater") | Out-Null
$UserFolders.add("%roaming%\.rc-persist") | Out-Null
$UserFolders.add("%roaming%\Glip") | Out-Null
$UserFolders.add("%roaming%\JabraSDK") | Out-Null
$UserFolders.add("%roaming%\Microsoft\Windows\Start Menu\Programs\RingCentral Meetings") | Out-Null
$UserFolders.add("%roaming%\Microsoft\Windows\Start Menu\Programs\RingCentral") | Out-Null
$UserFolders.add("%roaming%\Microsoft\Windows\Start Menu\Programs\RingCentral*.lnk") | Out-Null
$UserFolders.add("%roaming%\Microsoft\Windows\Start Menu\Programs\RingCentral\RingCentral Classic.lnk") | Out-Null
$UserFolders.add("%roaming%\Microsoft\Windows\Start Menu\Programs\Startup\RingCentral*.lnk") | Out-Null
$UserFolders.add("%roaming%\RingCentral") | Out-Null
$UserFolders.add("%roaming%\RingCentralMeetings") | Out-Null
$UserFolders.add("%roaming%\RingCentral\logs") | Out-Null
$UserFolders.add("%roaming%\ZoomSDK") | Out-Null
$UserFolders.add("%roaming%\com.ringcentral.rcoutlook") | Out-Null

#Look at every user profile on the computer and remove the registry keys and associated folders for each RC application
add-content $logfile -value "$(Get-Date -Format $dtFormat) Removing applications for all user profiles"
```

```
317 #$PatternSID = 'S-1-5-21-\d+-\d+\-\d+\-\d+$'
318 #$ProfileList = Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows
    NT\CurrentVersion\ProfileList\*' | Where-Object {$_.PSChildName -match $PatternSID} |
    Select-Object @{name="SID";expression={$_.PSChildName}},
    @{name="UserProfile";expression={"$($_.ProfileImagePath)"}},
    @{name="Username";expression={$_.ProfileImagePath -replace '^(.*[\\\/])', ''}}
319 $ProfileList = Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows
    NT\CurrentVersion\ProfileList\*' | Select-Object
    @{name="SID";expression={$_.PSChildName}},
    @{name="UserProfile";expression={"$($_.ProfileImagePath)"}},
    @{name="Username";expression={$_.ProfileImagePath -replace '^(.*[\\\/])', ''}}
320 $DefaultProfile = "" | Select-Object SID, UserProfile, Username
321 $DefaultProfile.SID = ".DEFAULT"
322 $DefaultProfile.UserProfile = "$pub\..\Default"
323 $DefaultProfile.UserName = "Default"
324 $ProfileList += $DefaultProfile
325 $LoadedHives = Get-ChildItem HKU:\ | Select-Object
    @{name="SID";expression={$_.PSChildName}}
326 $UnloadedHives = Compare-Object $ProfileList.SID $LoadedHives.SID | Select-Object
    @{name="SID";expression={$_.InputObject}}, UserHive, Username
327 foreach ($item in $ProfileList) {
328     try {
329         if ($item.SID -in $UnloadedHives.SID) {
330             add-content $logfile -value "$(Get-Date -Format $dtFormat) Loading profile
                $($item.username) - located at $($item.UserProfile)\ntuser.dat"
331             reg load HKU\$($item.SID) "$($item.UserProfile)\ntuser.dat" | Out-Null
332         } else {
333             add-content $logfile -value "$(Get-Date -Format $dtFormat) Checking profile
                $($item.username)"
334         }
335
336         $folders =
            "HKU:\$($item.sid)\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell
            Folders"
337         $desktop = (Get-Item -Path $folders).GetValue("Desktop",
            "$($item.UserProfile)\Desktop", "DoNotExpandEnvironmentNames") -replace
            "%USERPROFILE%", $item.UserProfile
338         add-content $logfile -value "$(Get-Date -Format $dtFormat)     User Desktop
            location: $($desktop)"
339         $local = (Get-Item -Path $folders).GetValue("Local AppData",
            "$($item.UserProfile)\AppData\Local", "DoNotExpandEnvironmentNames") -replace
            "%USERPROFILE%", $item.UserProfile
340         add-content $logfile -value "$(Get-Date -Format $dtFormat)     User Local
            AppData location: $($local)"
341         $roaming = (Get-Item -Path $folders).GetValue("AppData",
            "$($item.UserProfile)\AppData\Roaming", "DoNotExpandEnvironmentNames") -replace
            "%USERPROFILE%", $item.UserProfile
342         add-content $logfile -value "$(Get-Date -Format $dtFormat)     User AppData
            location: $($roaming)"
343
344         if ($item.SID -in $UnloadedHives.SID) {
345             add-content $logfile -value "$(Get-Date -Format $dtFormat) Loading user
                classes for profile $($item.username) - located at
                $($local)\Microsoft\Windows\UsrClass.dat"
346             reg load HKU\$($item.SID)_classes "$($local)\Microsoft\Windows\UsrClass.dat"
                | Out-Null
347         }
348
349         foreach ($regkey in $HKU) {
350             try {
351                 $key = $regkey -replace "%SID%", $item.SID
352                 if (test-path($key)) {
353                     add-content $logfile -value "$(Get-Date -Format $dtFormat)
                        Removing Registry Key $($key)"
354                     remove-item $key -recurse -force
355                 } ##else { add-content $logfile -value "$(Get-Date -Format
                    $dtFormat)     Registry Key $($key) not found" }
356             } catch {
357                 add-content $logfile -value $_
```

```
358             }
359         }
360
361         foreach ($path in $UserFolders) {
362             $temp = (($path -replace "%roaming%", $roaming) -replace "%local%", $local)
                -replace "%desktop%", $desktop
363             try {
364                 if (test-path($temp)) {
365                     add-content $logfile -value "$(Get-Date -Format $dtFormat)
                        Removing $($temp)"
366                     remove-item $temp -recurse -force
367                 } ##else { add-content $logfile -value "$(Get-Date -Format
                    $dtFormat)     Path $($temp) not found" }
368             } catch {
369                 add-content $logfile -value $_
370             }
371         }
372
373         #Remove any user uninstall keys
374         $paths =
            @("HKU:\$($item.sid)\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninst
            all", "HKU:\$($item.sid)\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall")
375         foreach($path in $paths) {
376             if (test-path($path)) {
377                 $list = Get-ItemProperty "$path\*" | Where-Object {$_.DisplayName -like
                    "*RingCentral*"} | Select-Object -Property PSPath
378                 foreach($regkey in $list) {
379                     add-content $logfile -value "$(Get-Date -Format $dtFormat)
                        Removing Uninstall Registry Key $($regkey.PSPath)"
380                     try {
381                         remove-item $regkey.PSPath -recurse -force
382                     } catch {
383                         add-content $logfile -value $_
384                     }
385                 }
386             } ##else { add-content $logfile -value "$(Get-Date -Format $dtFormat) Path
                $($item) not found" }
387         }
388
389         #Remove any user install keys - this is done both in the user hive and the user
            data part of the local machine
390         $paths =
            @("HKU:\$($item.sid)\SOFTWARE\WOW6432Node\Microsoft\Installer\Products",
            "HKU:\$($item.sid)\SOFTWARE\Microsoft\Installer\Products")
391         foreach($path in $paths) {
392             if (test-path($path)) {
393                 $list = Get-ItemProperty "$path\*" | Where-Object {$_.ProductName -like
                    "*RingCentral*"} | Select-Object -Property PSPath
394                 foreach($regkey in $list) {
395                     add-content $logfile -value "$(Get-Date -Format $dtFormat)
                        Removing Install Registry Key $($regkey.PSPath)"
396                     try {
397                         remove-item $regkey.PSPath -recurse -force
398                     } catch {
399                         add-content $logfile -value $_
400                     }
401                 }
402             } ##else { add-content $logfile -value "$(Get-Date -Format $dtFormat) Path
                $($item) not found" }
403         }
404         $paths =
            @("HKLM:\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Installer\UserData
            \$($item.sid)\Products",
405
                       "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData\$(
                       $item.sid)\Products")
406         foreach($path in $paths) {
407             if (test-path($path)) {
408                 $list = Get-ItemProperty "$path\*\*" | Where-Object {$_.Publisher -like
```

```
                    "*RingCentral*"} | Select-Object -Property PSParentPath
409                 foreach($regkey in $list) {
410                     add-content $logfile -value "$(Get-Date -Format $dtFormat)
                        Removing Install Registry Key $($regkey.PSParentPath)"
411                     try {
412                         remove-item $regkey.PSParentPath -recurse -force
413                     } catch {
414                         add-content $logfile -value $_
415                     }
416                 }
417             } ##else { add-content $logfile -value "$(Get-Date -Format $dtFormat) Path
                $($item) not found" }
418         }
419
420         #Loop through the keys and remove any RC entries
421         $paths = @("HKU:\$($item.sid)_classes\Local
            Settings\Software\Microsoft\Windows\Shell\MuiCache",
422
                    "HKU:\$($item.sid)\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\
                    FeatureUsage\AppBadgeUpdated",
423                 "HKU:\$($item.sid)\SOFTWARE\Microsoft\Windows\CurrentVersion\UFH\SHC")
424         foreach($path in $paths) {
425             if (test-path($path)) {
426                 add-content $logfile -value "$(Get-Date -Format $dtFormat)     Checking
                    registry path: $($path)"
427                 Get-Item -Path $path | Select-Object -ExpandProperty Property | % {
428                     $propValue = (Get-ItemProperty -Path "$path" -Name "$_")."$_"
429                     if (($_ -like "*RingCentral*") -or ($propValue -like
                        "*RingCentral*")) {
430                         try {
431                             add-content $logfile -value "$(Get-Date -Format
                                $dtFormat)         Removing property: $($_) containing
                                value: $($propValue)"
432                             Remove-ItemProperty -path "$path" -Name $_
433                         } catch {
434                             add-content $logfile -value $_
435                         }
436                     }
437                 }
438             } ##else { add-content $logfile -value "$(Get-Date -Format $dtFormat) Path
                $($item) not found" }
439         }
440
441         if ($item.SID -in $UnloadedHives.SID) {
442             [gc]::Collect()
443             add-content $logfile -value "$(Get-Date -Format $dtFormat) Unloading profile"
444             reg unload HKU\$($item.SID) | Out-Null
445             reg unload HKU\$($item.SID)_classes | Out-Null
446         }
447     } catch {
448         add-content $logfile -value $_
449     }
450 }
451 add-content $logfile -value "$(Get-Date -Format $dtFormat) End of removal script"
452
453 #add-content $logfile -value "$(Get-Date -Format $dtFormat) Installing required
    applications"
454
455 #add-content $logfile -value "$(Get-Date -Format $dtFormat)     Installing RingCentral
    MSI app quietly"
456 #cmd.exe /c 'MSIEXEC.EXE /i "RingCentral-x64.msi" /qn'
457
458 #add-content $logfile -value "$(Get-Date -Format $dtFormat)     Installing RingCentral
    Meetings MSI app quietly"
459 #cmd.exe /c 'MSIEXEC.EXE /i "RCMeetingsClientSetup.msi" /qn'
460
461 #add-content $logfile -value "$(Get-Date -Format $dtFormat)     Installing RingCentral
    Phone MSI app quietly"
462 #cmd.exe /c 'MSIEXEC.exe /i "RingCentral-Phone-21.1.0.msi" ALLUSERS=1 /qn'
```

```
463
464   #add-content $logfile -value "$(Get-Date -Format $dtFormat) End of install script"
```