VpnInit AppleScript: Override and Restore Default VPN-Routes on OS X

Years ago I wrote a small script to restore local default route after connecting the RAS VPN on Windows.

Now, I made up a similar script to do the same on OS X.

The basic idea, now and then, was that I would not want to send all traffic through the VPN.
Thus the script will assist in restoring your local default route after the VPN connection is established.
Furthermore, it'll add some specific routes directed to the VPN.

This way, all your usual traffic (internet, surfing, skype, whatever) is sent through your default gateway, while more specific routes (your business stuff) is sent through the VPN.

Below's the code for the initial release. It may lack some details yet, like auto-detecting the tunnel device name, but it does the job already.

Just copy the code into Apple Script Editor and save it to a convenient location. Make sure you save it as "Application" and not as "Script" (which is the default). You you don't, double-clicking the script will open the Script Editor instead of executing the script. Surely, not what you want.

Pay attention to the variables on the top, which need to be edited before you save the Script: _vpn_name, _default_gw, _networks, and _sudo_password (optional).
I hope directions are clear enough from the comments sections.

```
# VpnInit
# ---
# an AppleScript utility to connect your vpn,
# restore local default route and add selected
# routes directed to the VPN only
# thus you'll end up sending only selected
# traffic through the VPN, while the rest
# goes through your local default gateway
# as usual
# ---
# released "as is" under the terms of GPL v2.
# Copyright © 2011 Gianpaolo Del Matto
#
# r0.1 initial release 2011/12/29
#
# ToDo:
# - hardcoding the "sudo" password is a bad idea, maybe
# need a better way to get away with it
# - vpn tunnel (utun0) is still hardcoded,
# should be auto-detected
#


# the name of your vpn connection
#
```

```
set _vpn_name to "My VPN"

# your local default gateway
#
set _default_gw to "192.168.1.1"

# your remote networks to pass via VPN, separate multiple with comma
# like so: {"1.2.3.4/30", "5.6.7.8/30"}
#
set _networks to {"192.168.2.1/24"}


# your super-user (root) password
# actually needed to bypass the prompts
# leave empty to get prompted
#
set _sudo_password to ""


# ######################################################################
# DO NOT CHANGE ANYTHING BELOW
# ######################################################################


# kindly borrowed from
# http://www.macosxautomation.com/applescript/uiscripting/index.html
# make sure that support for assistive devices is enabled
#
tell application "System Events"
 if UI elements enabled is false then
  tell application "System Preferences"
   activate
   set current pane to pane id "com.apple.preference.universalaccess"
   display dialog "This script requires access for assistive devices be enabled." & return & return & "To continue, click the OK button and enter an administrative password in the forthcoming security dialog." with icon 1
  end tell
  set UI elements enabled to true
  if UI elements enabled is false then
   display dialog "This script cannot run while access for assistive devices is disabled." & return & "Exiting now." buttons {"OK"} with icon 2
   return "user cancelled"
  end if
 end if
end tell

# now dive into the VPN setup part
#
tell application "System Events"
 set _if_tunnel to "utun0" # do not change, will be auto-detected, just giving a reasonable default
 tell current location of network preferences
```

```
 if exists service _vpn_name then
  # try to connect the VPN service if it's disconnected
  #
  if current configuration of service _vpn_name is not connected then
   connect service _vpn_name
  end if

  # give it some time to settle
  #
  set _retval to false
  repeat until (_retval) is true
   set counter to 0
   repeat while counter is less than 16
    # exit if we get connected
    #
    if current configuration of service _vpn_name is connected then
     set _retval to true
     exit repeat
    end if

    # opt for exit if still not connected after 15 seconds
    #
    if counter is equal to 15 then
     display dialog "VPN '" & _vpn_name & "' is still not connected after 15 seconds. Do you want to keep waiting?" with title "VPN still not connected" buttons {"Yes", "No"}
     if button returned of result is "No" then
      # bail out if user decided not to wait any longer
      #
      set _retval to true
      return
     else
      # otherwise reset the counter so we can trigger again
      #
      set counter to 0
     end if
    end if

    set counter to counter + 1
    delay 1
   end repeat
  end repeat

  # now go to post processing and to the following:
  # - delete default route via vpn
  # - restore original default route
  # - add specific routes to vpn
  #
  if current configuration of service _vpn_name is connected then
   # restore local default route
   #
```

```
    do shell script "route delete default" password _sudo_password with administrator privileges
    do shell script "route add default " & _default_gw password _sudo_password with administrator privileges

    # inject custom routes via VPN
    #
    repeat with _network in _networks
     do shell script "route add -interface " & _network & " utun0" password _sudo_password with administrator privileges
    end repeat
   end if
  else
   # bail out if the VPN service does not exist
   #
   display dialog "Given VPN '" & _vpn_name & "' does not exist. Please check the name"
  end if
 end tell
end tell
```