AppleScript Support

On This Page
Controlling Tunnelblick with AppleScript
Controlling Tunnelblick from the Command Line
AppleScript Dictionary
Examples

Controlling Tunnelblick with AppleScript

This document is about making Tunnelblick do things via scripting. Tunnelblick itself also uses scripts; see Using Scripts.

AppleScripts may connect, disconnect, or list Tunnelblick configurations.

Please note that the verbs only *initiate* actions. The "connect" verb, for example, is the equivalent of clicking the "Connect" button. It returns immediately after starting an attempt to connect using a particular configuration and does not indicate the connection has been accomplished successfully.

For an example of using Tunnelblick's AppleScript support to solve a particular problem, see this discussion.

Controlling Tunnelblick from the Command Line

You can invoke AppleScript from the command line to control Tunnelblick. You can connect, disconnect, and list configurations.

One way is to store the AppleScript in a file and then invoke it with

```
osascript name-of-file
```

Another way is to include the AppleScript commands in the command line with, for example,

```
osascript –e "tell application \"/Applications/Tunnelblick.app\"" –e "connect \"configuration-name\"" –e "end tell"
```

replacing *configuration-name* with the name of your configuration.

AppleScript Dictionary

application *n* :

contains *configurations*.

configuration *n* : A VPN configuration.

contained by **application**.

properties

name (text, r/o) : Name of the configuration.

state (text, r/o) : State of the configuration.
'EXITING' means disconnecting or disconnected.
'CONNECTED' means connected.
Other values show progress towards a connection.

autoconnect (text, r/o):
'LAUNCH' means the configuration will be connected automatically when Tunnelblick launches.
'START' means the configuration will be connected automatically when the computer starts.
'NO' means the configuration will not be connected automatically.

bytesIn (text, r/o) :
The number of bytes that have come in through the connection since Tunnelblick was launched. (For client configurations only. This will return "0" for configurations that are functioning as a server.)

bytesOut (text, r/o) :
The number of bytes that have gone out through the connection since Tunnelblick was launched. (For client configurations only. This will return "0" for configurations that are functioning as a server.)

connect *v* : Connect a VPN configuration.

connect text : Name of configuration to connect.
-> boolean : Returns true if a connection is being attempted, false if the configuration is already connected.

disconnect *v* : Disconnect a VPN configuration.
disconnect text : Name of configuration to disconnect.
-> boolean : Returns true if a disconnection is being attempted, false if the configuration is already disconnected.

connect all *v* : Connect all unconnected VPN configurations.
-> integer : Returns the number of configurations for which a connection is being attempted.

disconnect all *v* : Disconnect all connected VPN configurations.
-> integer : Returns the number of configurations for which a disconnection is being attempted.

disconnect all except when computer starts *v* : Disconnect all connected VPN configurations except 'when computer starts' configurations.
-> integer : Returns the number of configurations for which a disconnection is being attempted.

added or removed configurations *v* : Tells Tunnelblick that configurations have been added or removed, so it can update the menu and the list of configurations in the "VPN Details" window.
-> boolean : Returns false.

changed openvpn configuration file for *v* : Tells Tunnelblick that an OpenVPN configuration file has been modified, so it can update settings in the "VPN Details" window.

changed openvpn configuration file for text : Name of configuration for which the OpenVPN configuration file has been modified.
-> boolean : Returns false.

Examples

tell application "Tunnelblick"

get configurations
-> {configuration "Work", configuration "Work-Development", configuration "Home", configuration "Test"}

disconnect all
-> 0

connect "Work-Development"
-> true

disconnect (get name of second configuration)
get name of configuration 2
-> "Work-Development"
disconnect "Work-Development"
-> true

connect (get name of first configuration where state = "EXITING")
get name of configuration 1 whose state = "EXITING"
-> "Work"
connect "Work"
-> true

disconnect all except when computer starts
-> 1

get state of second configuration
-> "EXITING"

get state of first configuration where name = "Work"
-> "EXITING"

get name of configurations
-> {"Work", "Work-Development", "Home", "Test"}

get state of configurations
-> {"EXITING", "EXITING", "EXITING", "EXITING"}

get autoconnect of configurations
-> {"NO", "NO", "NO", "NO"}

end tell