

Scripting the Unscriptable

GUI Scripts in AppleScript

Thursday, July 20 - 1:30pm

<https://sched.co/1MmYG>

2023
MACADMIN'S
CONFERENCE

Ross Matsuda Apple Systems Engineer at Ntiva

MacAdmins Slack: @Xirias

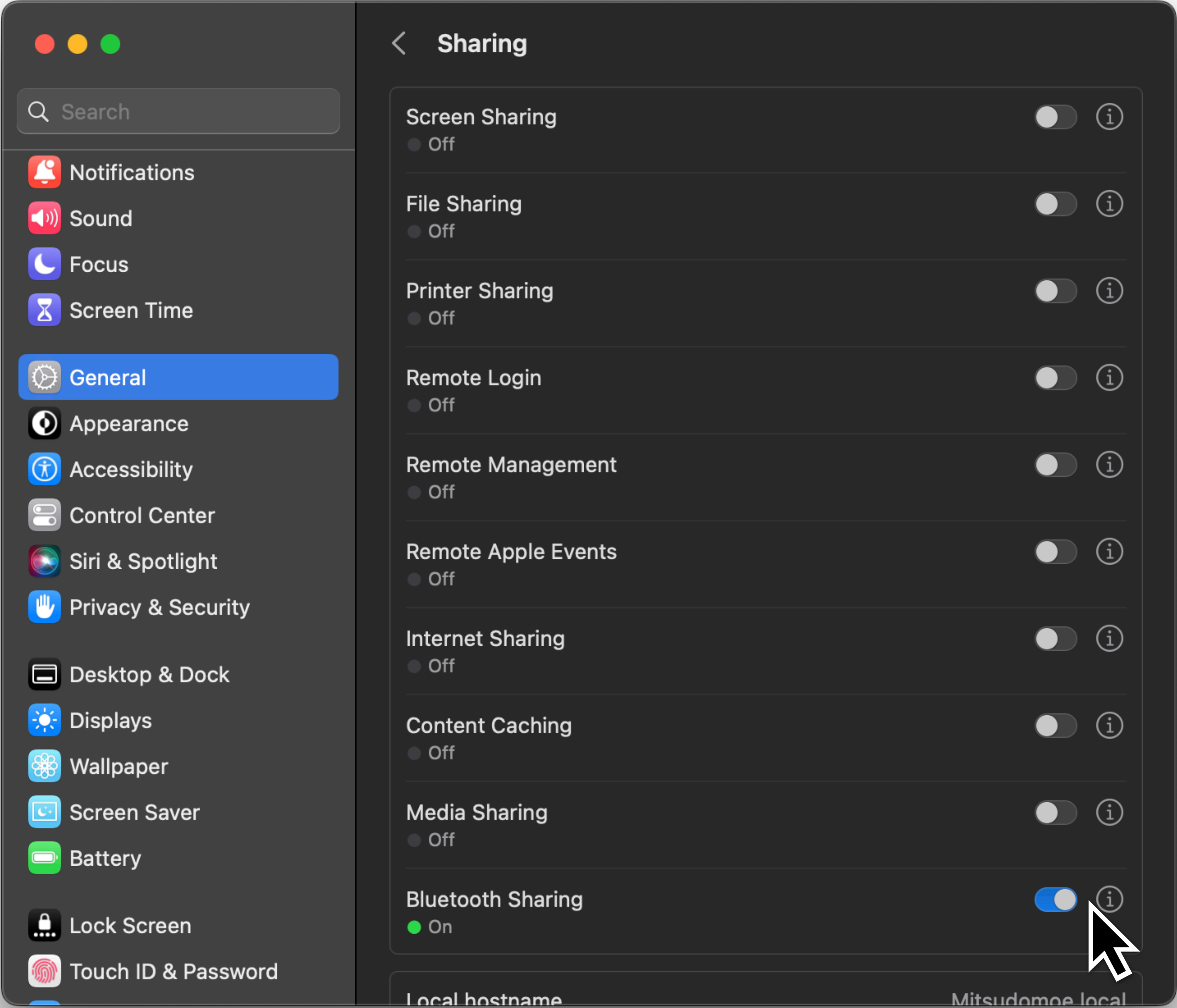
www.sudoade.com



What is GUI scripting?

- Simulating user action and input
- Handled in AppleScript
- A terrible idea






```
/usr/bin/sudo -u "$CURRENT_USER" \  
  /usr/bin/defaults -currentHost write \  
  com.apple.Bluetooth PrefKeyServicesEnabled \  
  -bool false
```

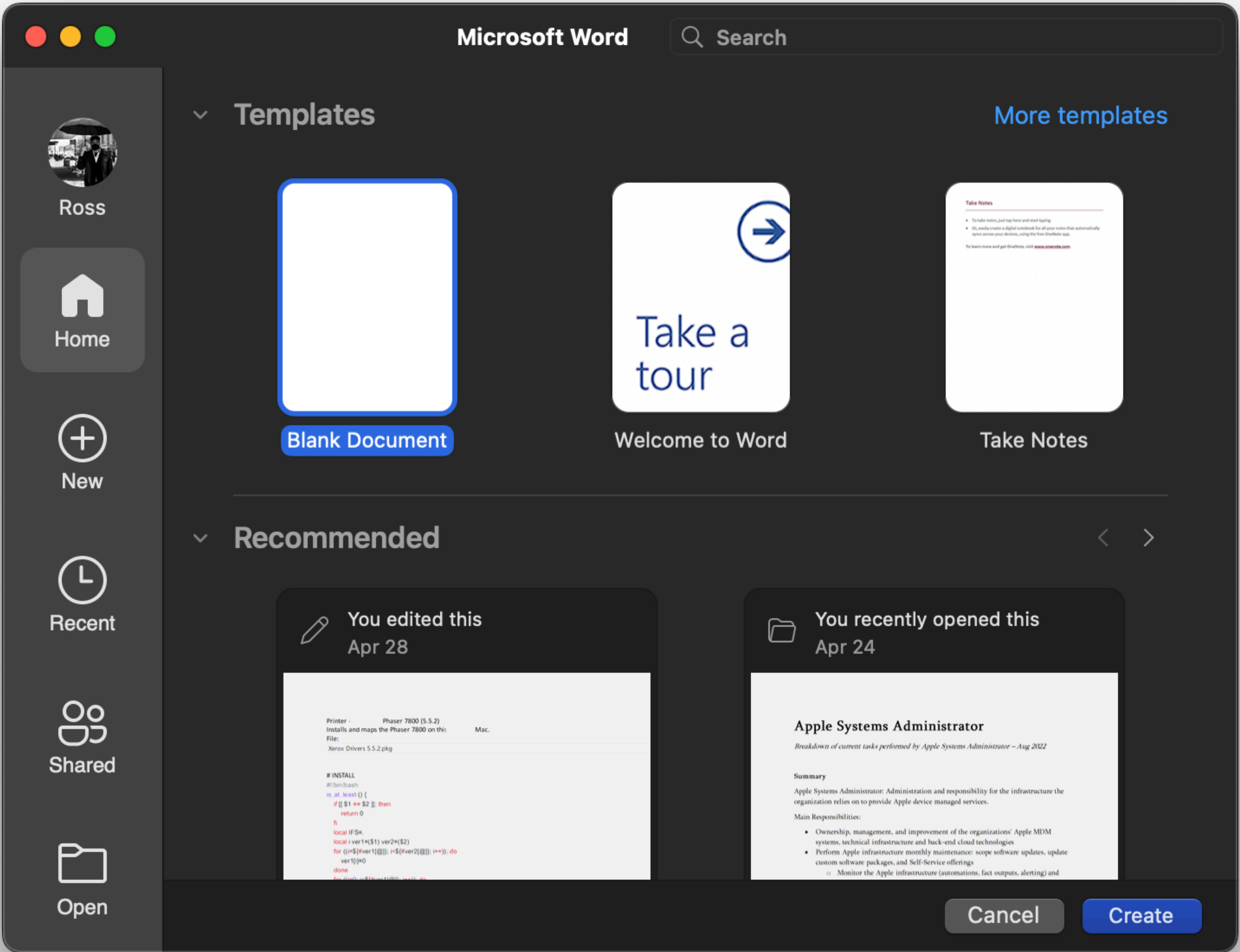

What is GUI scripting?

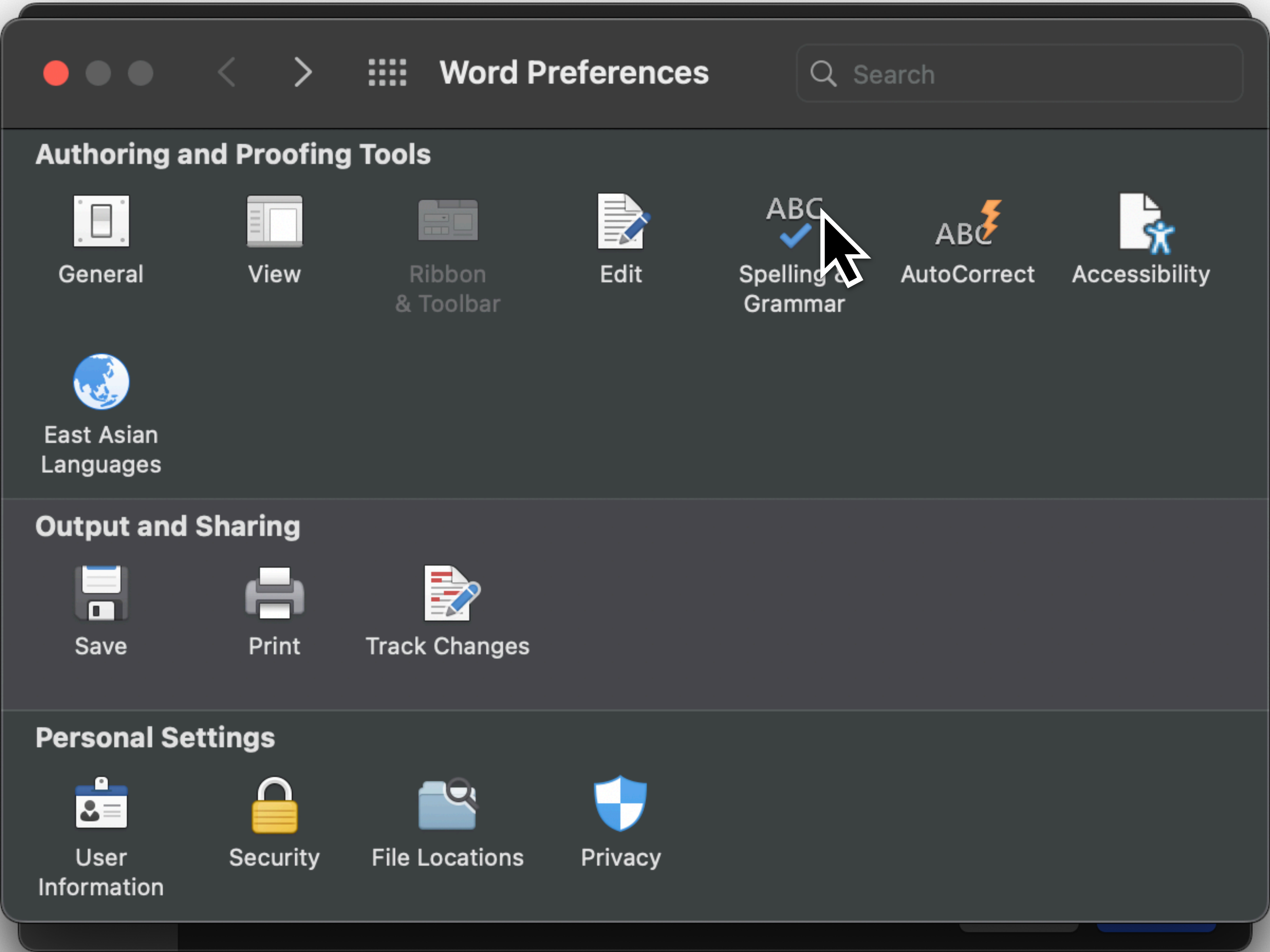
Example 1

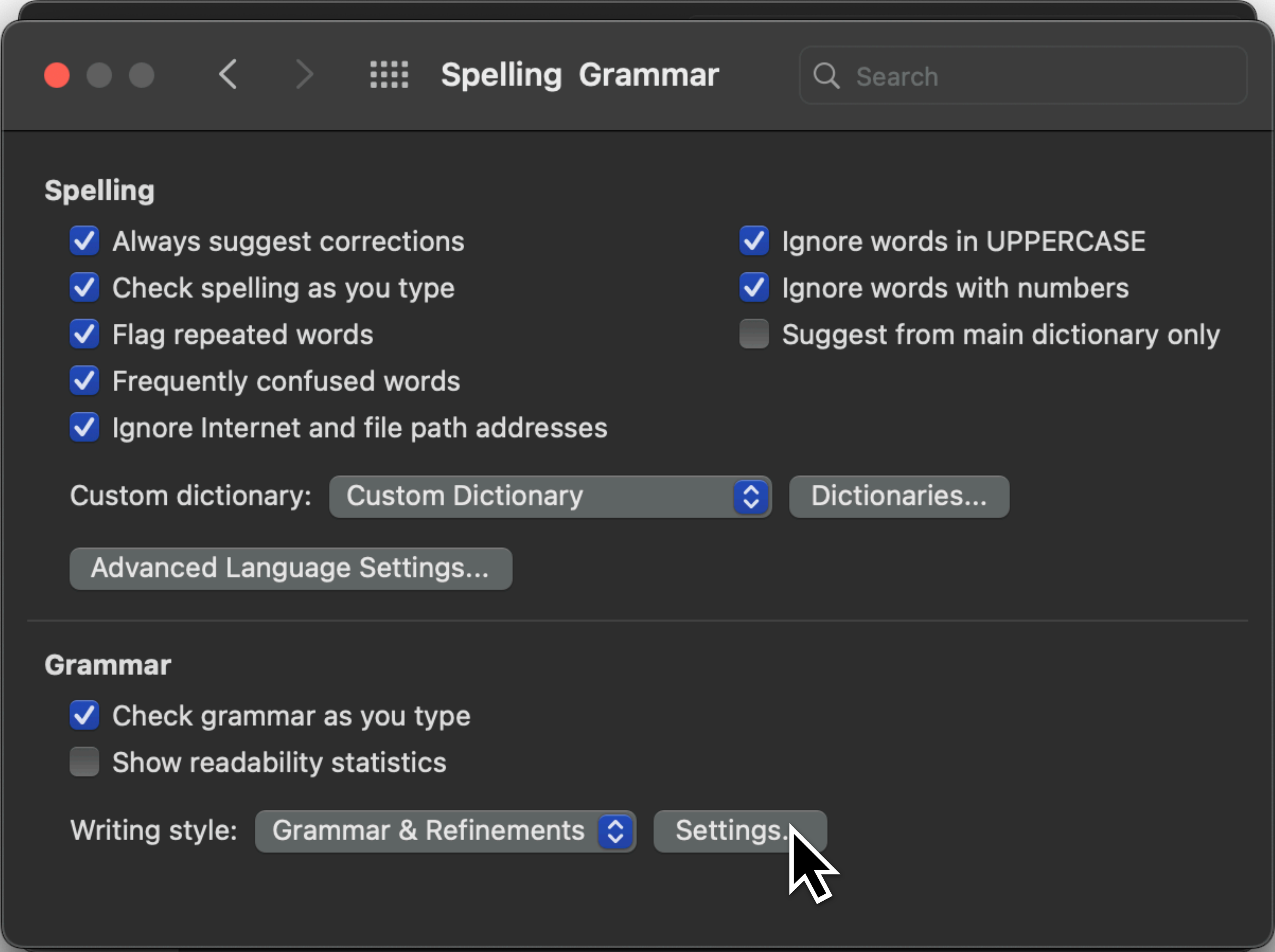
Microsoft Word:

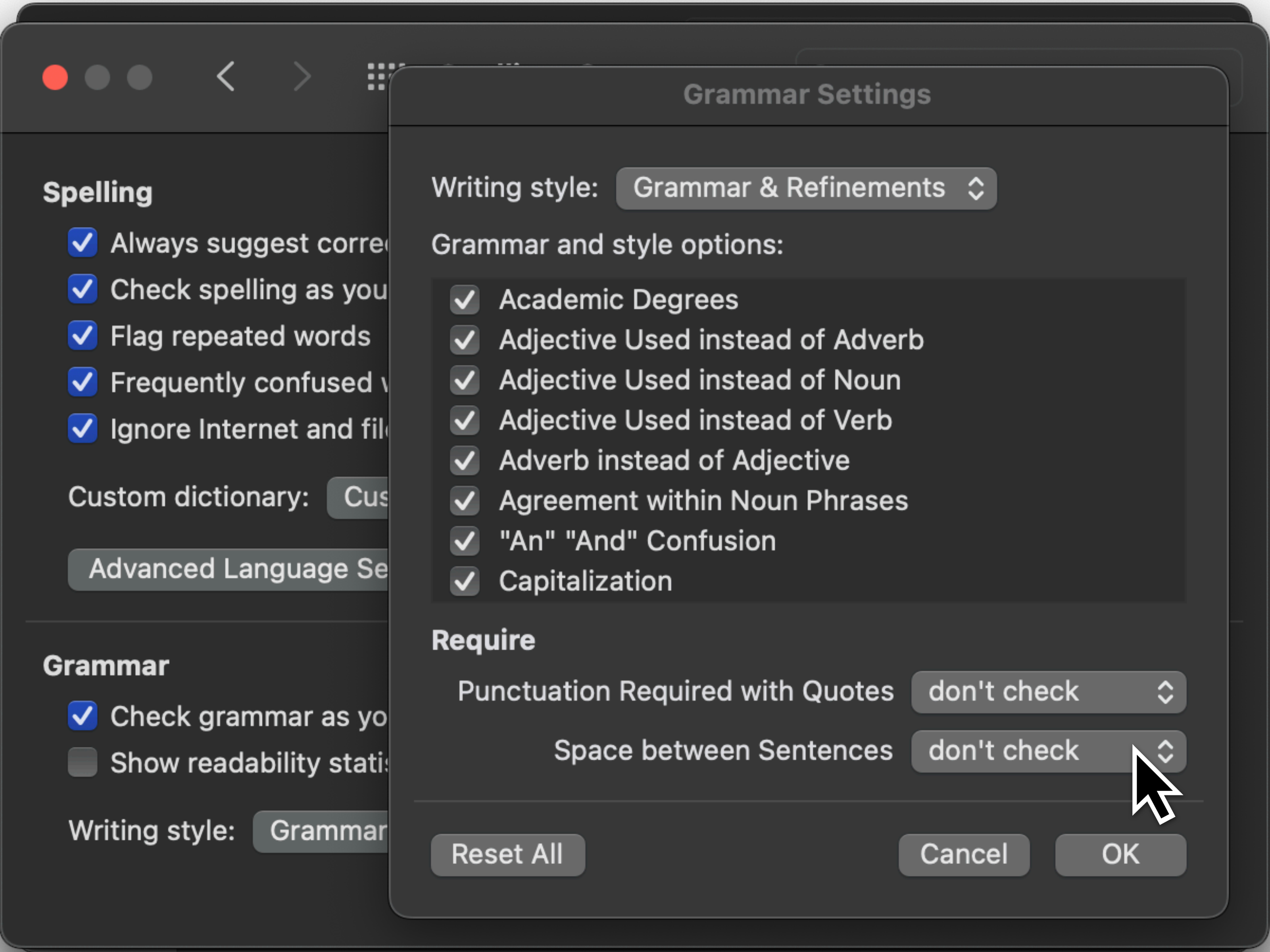
Set grammar to expect two spaces after a period instead of one.

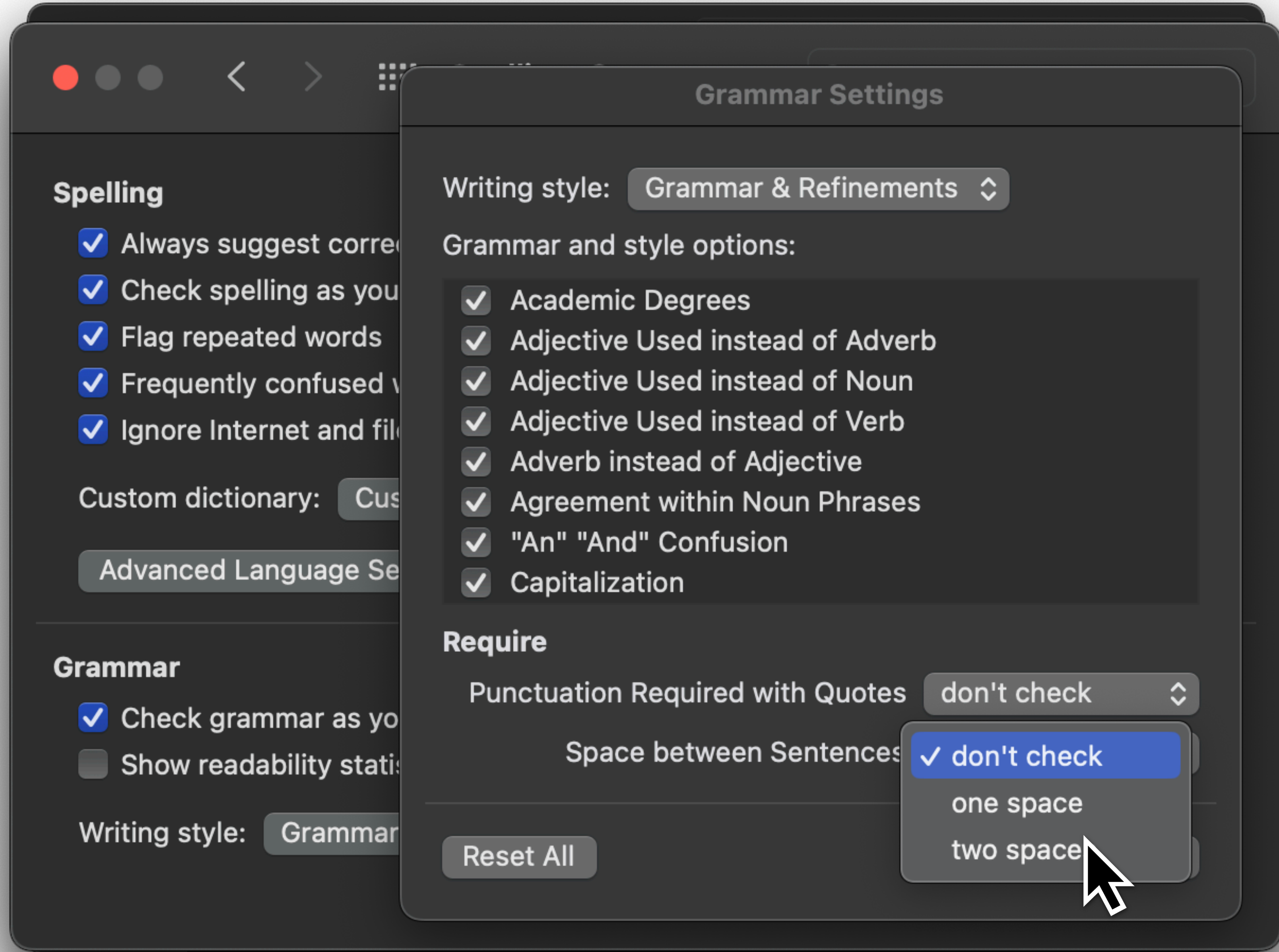
SCRIPTING THE UNSCRIPTABLE

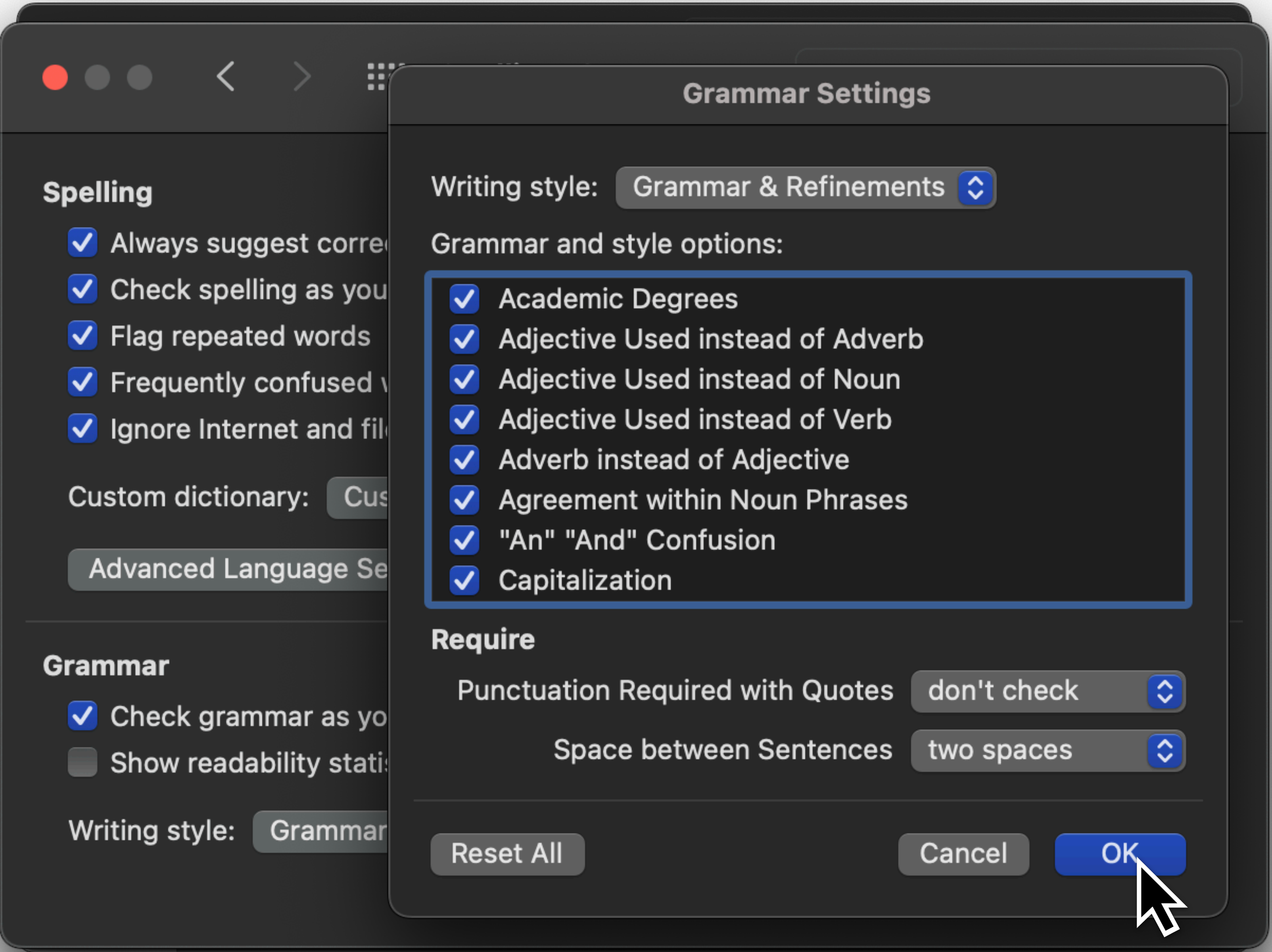


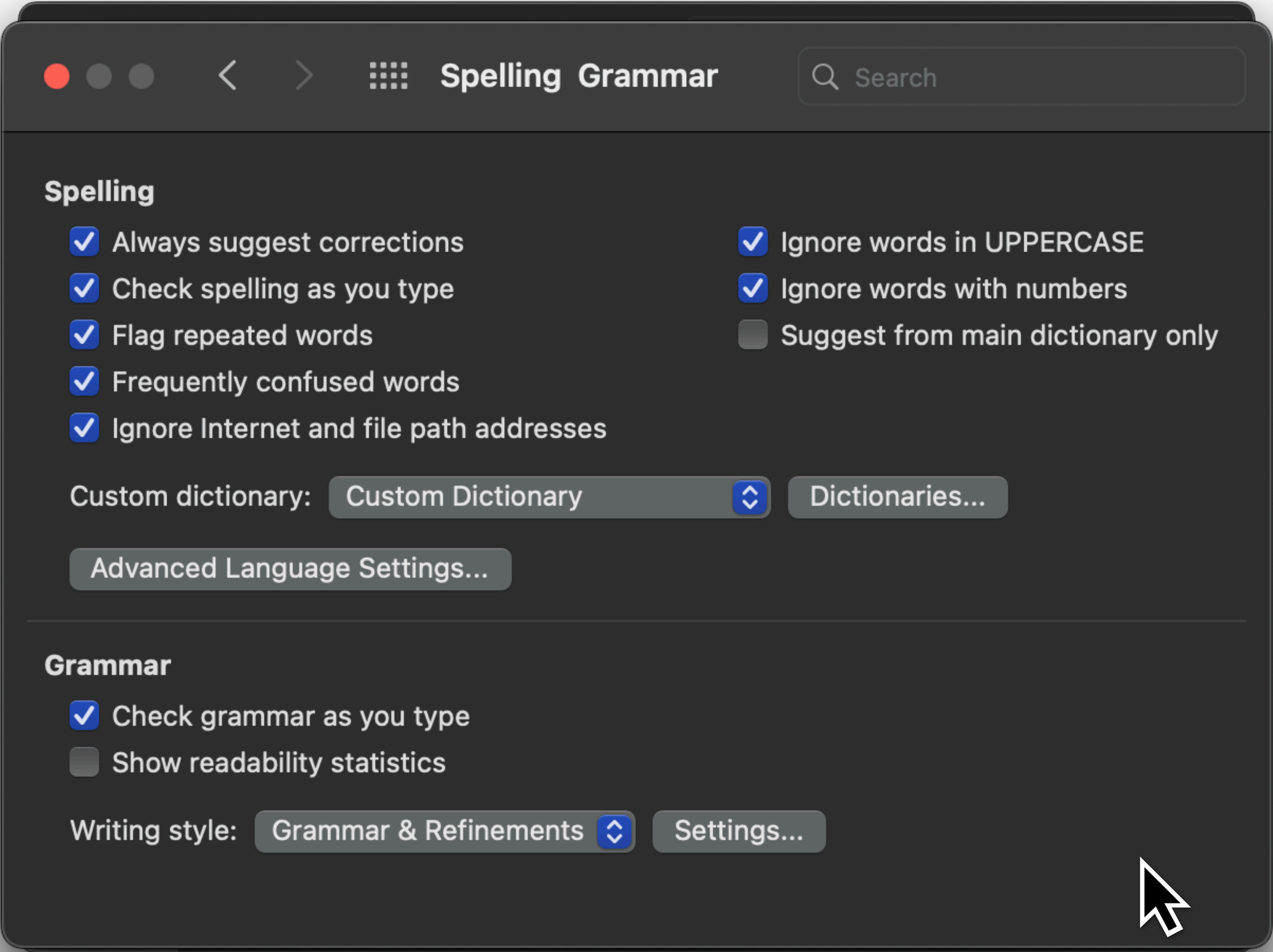












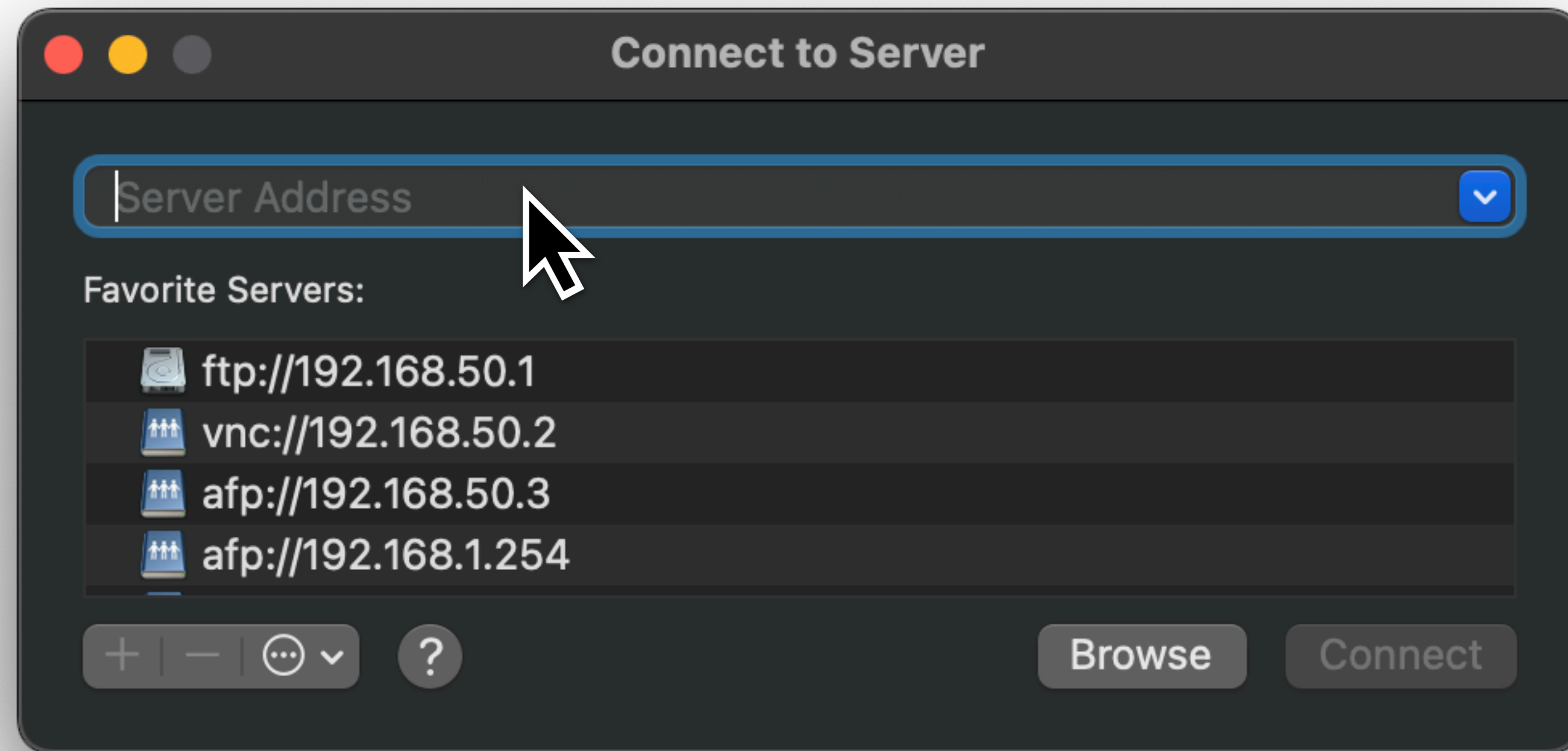
What is GUI scripting?

Example 2

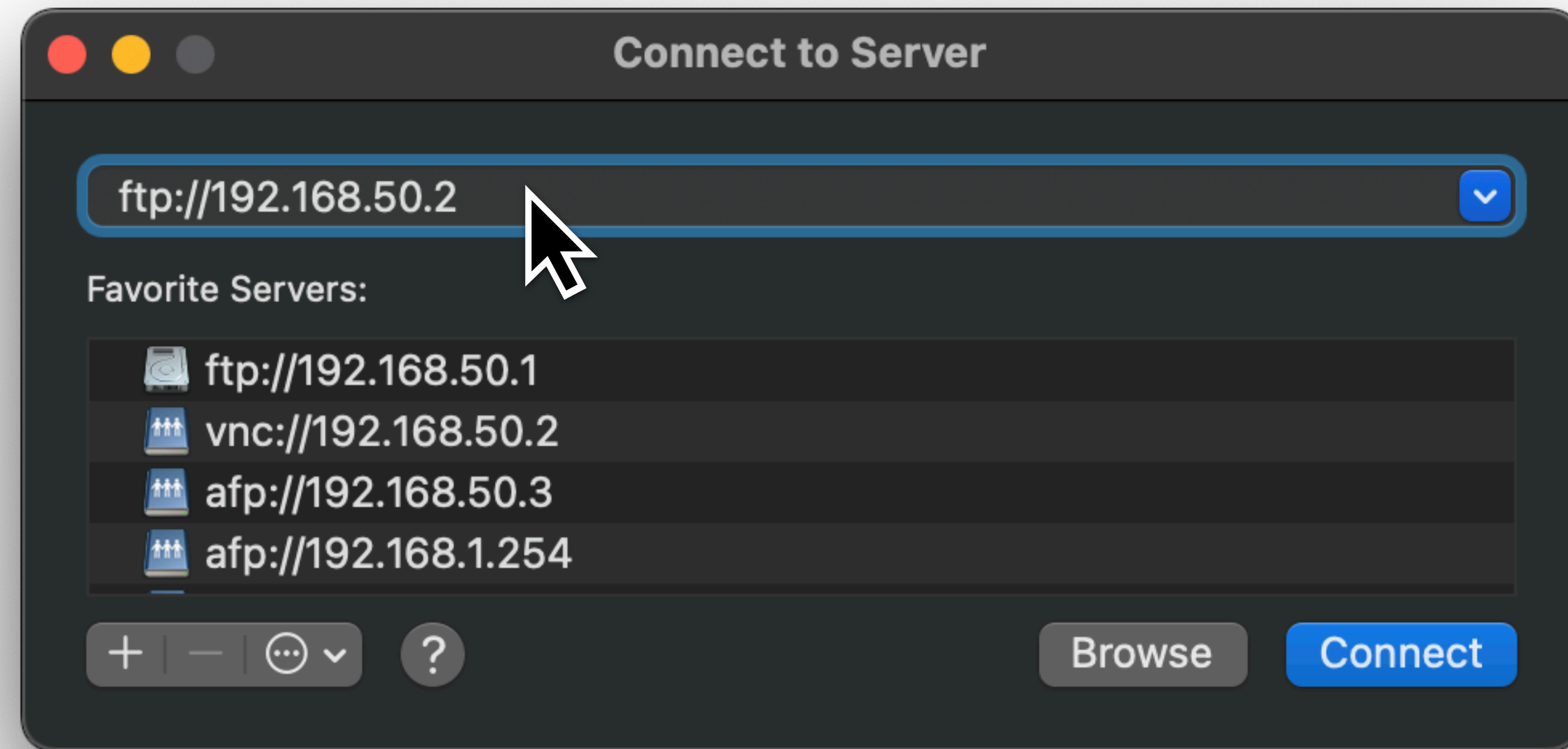
Finder:

Connect to Server Bookmark

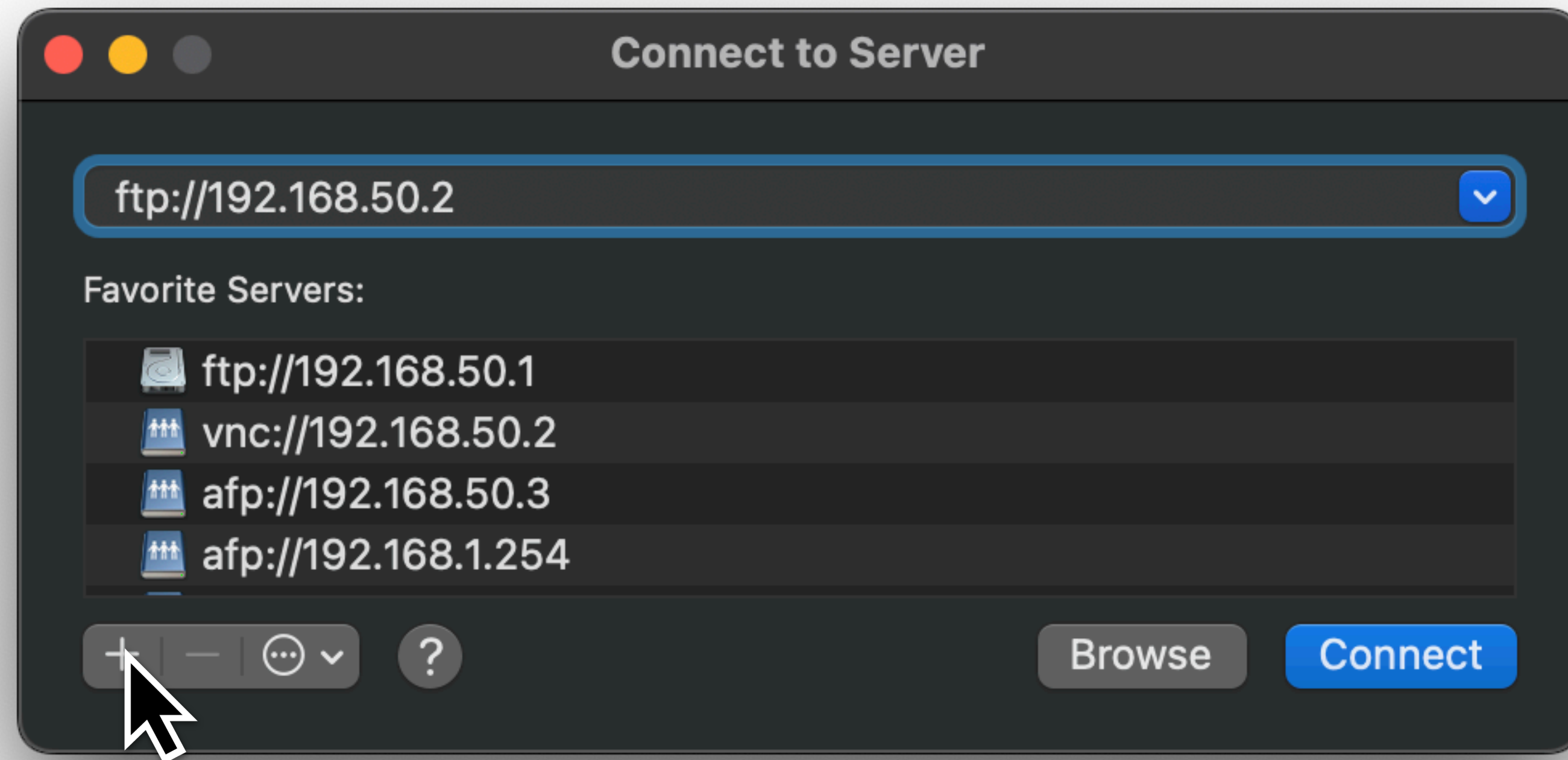
What is GUI scripting?



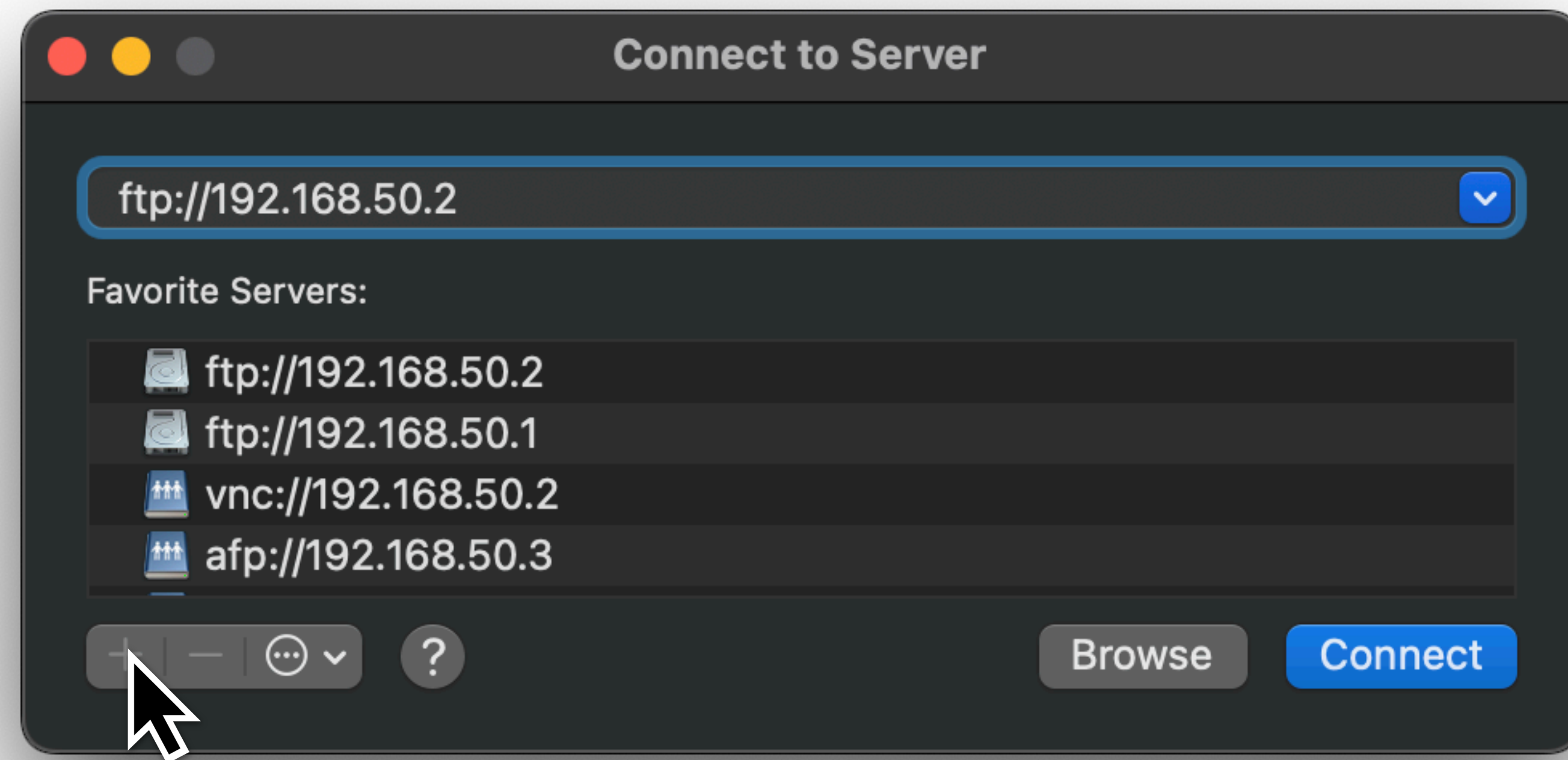
What is GUI scripting?



What is GUI scripting?

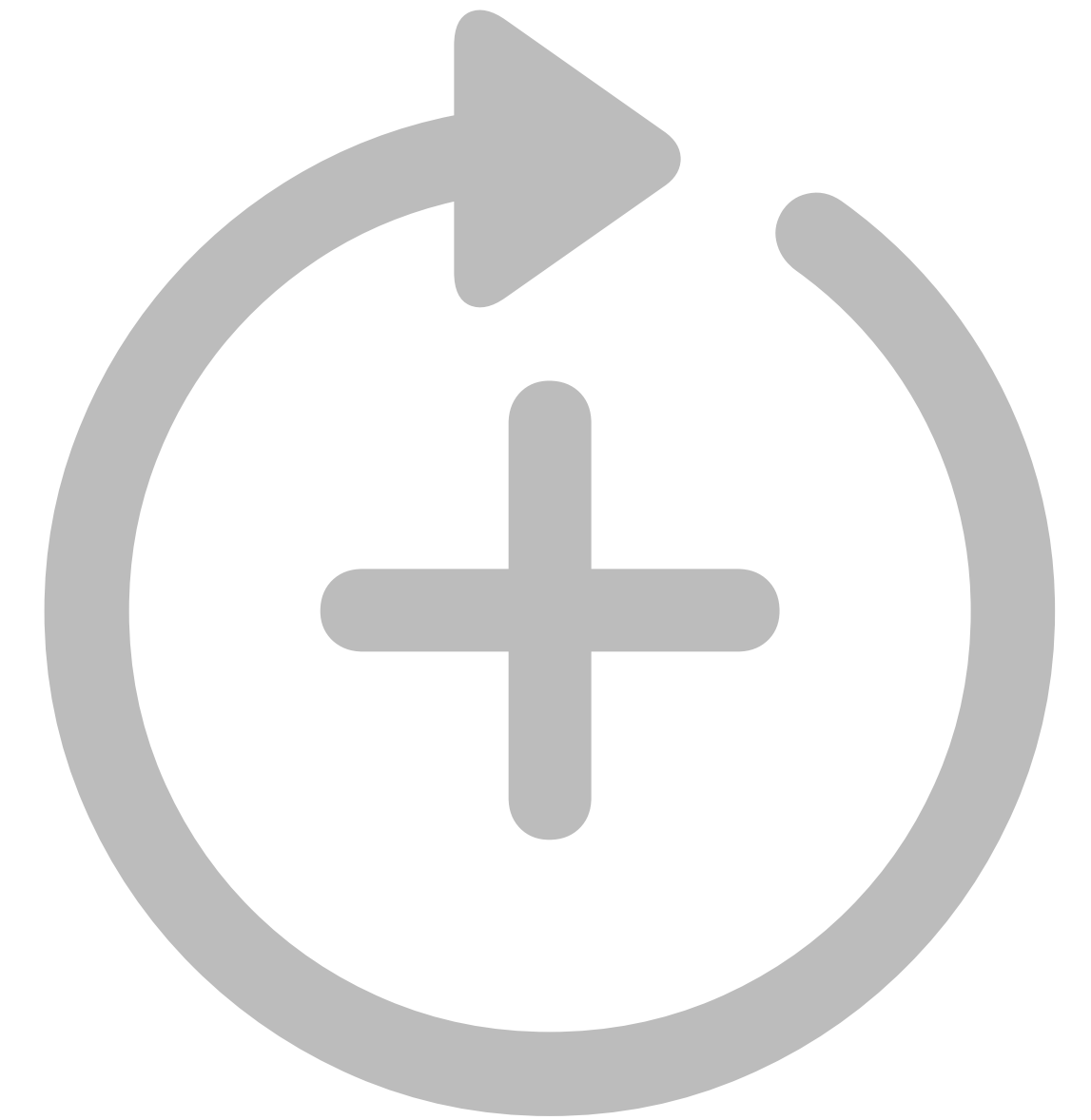


What is GUI scripting?



Pros

- You can automate things that don't have traditional CLI or MDM configurations
- Support desk time savings
- Makes you look like a spooky ghost



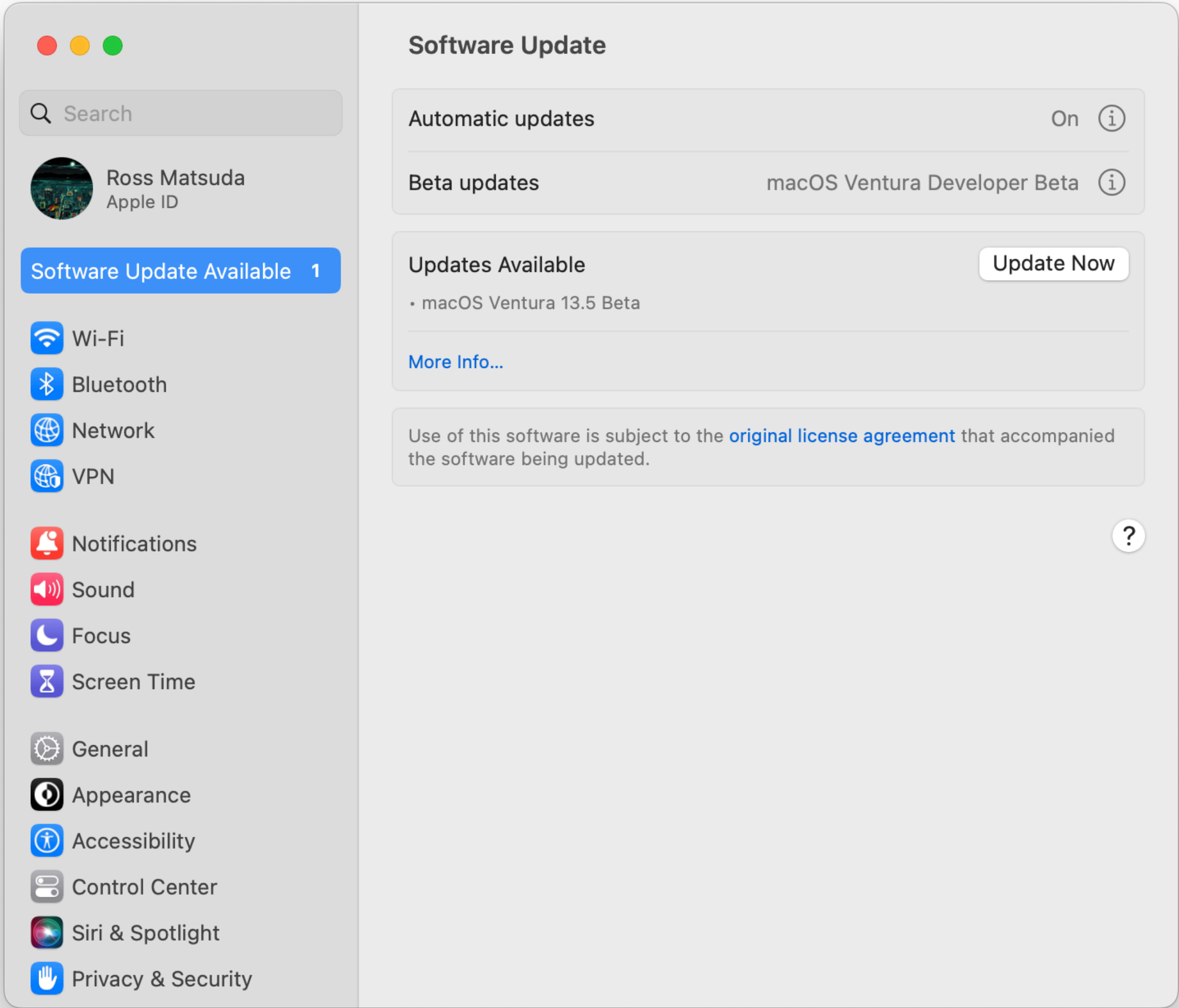
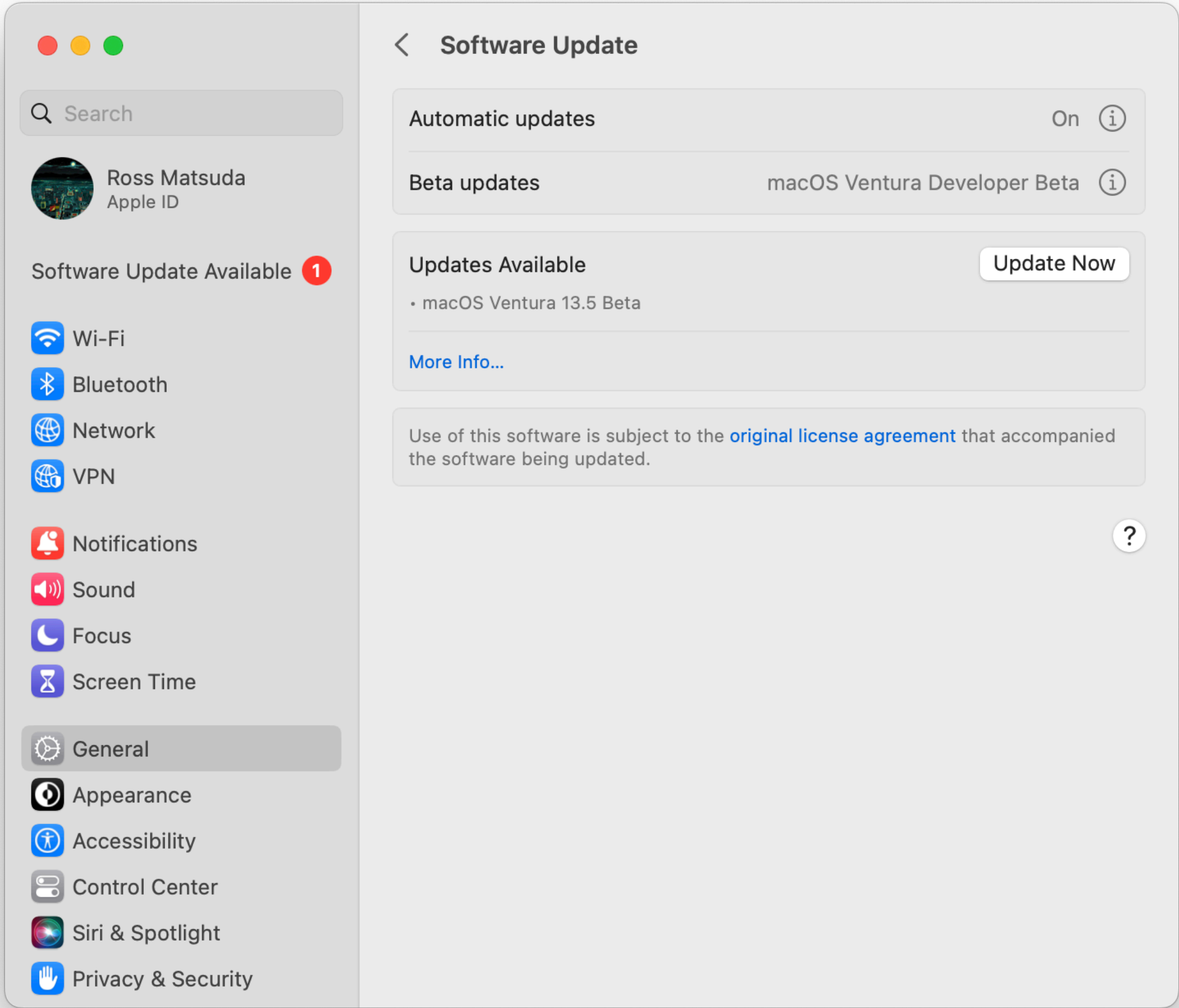
That's it.
That's the slide.

cons

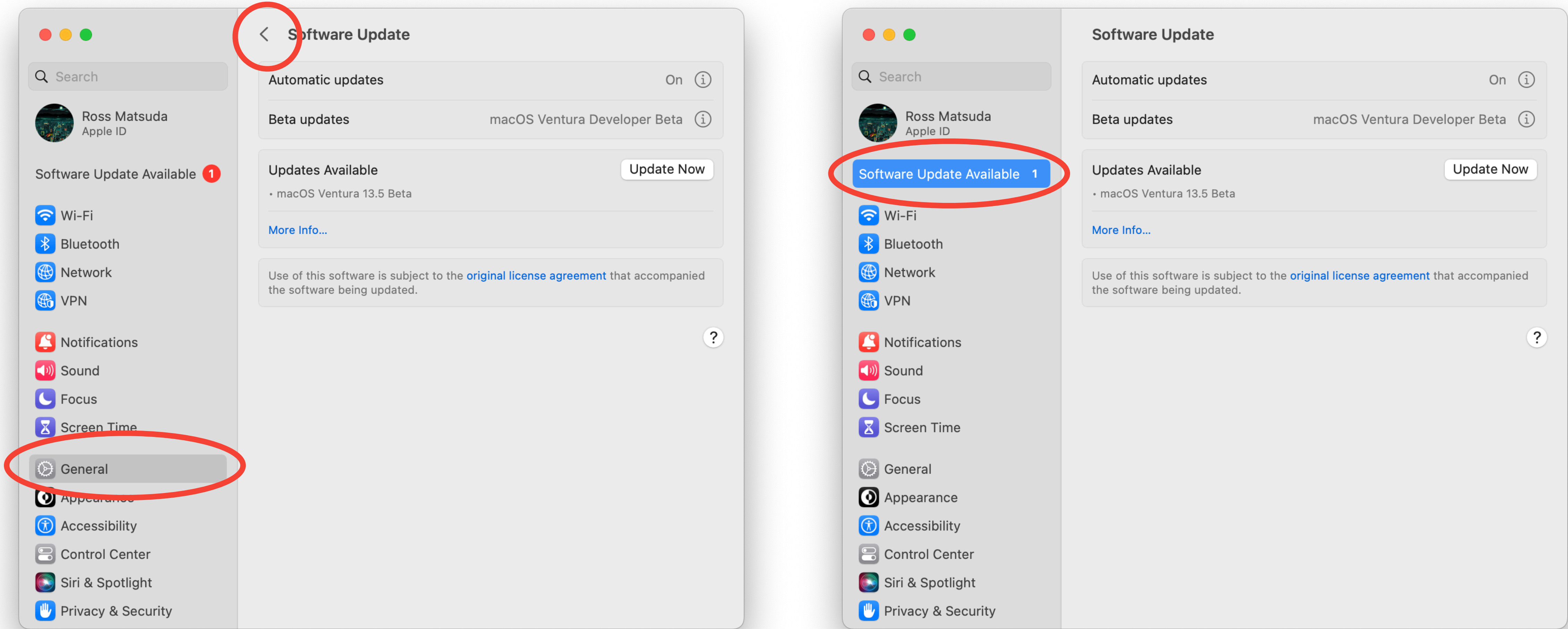
Cons

- Fragile
- OS-specific
- Software version-specific
- Specific-specific

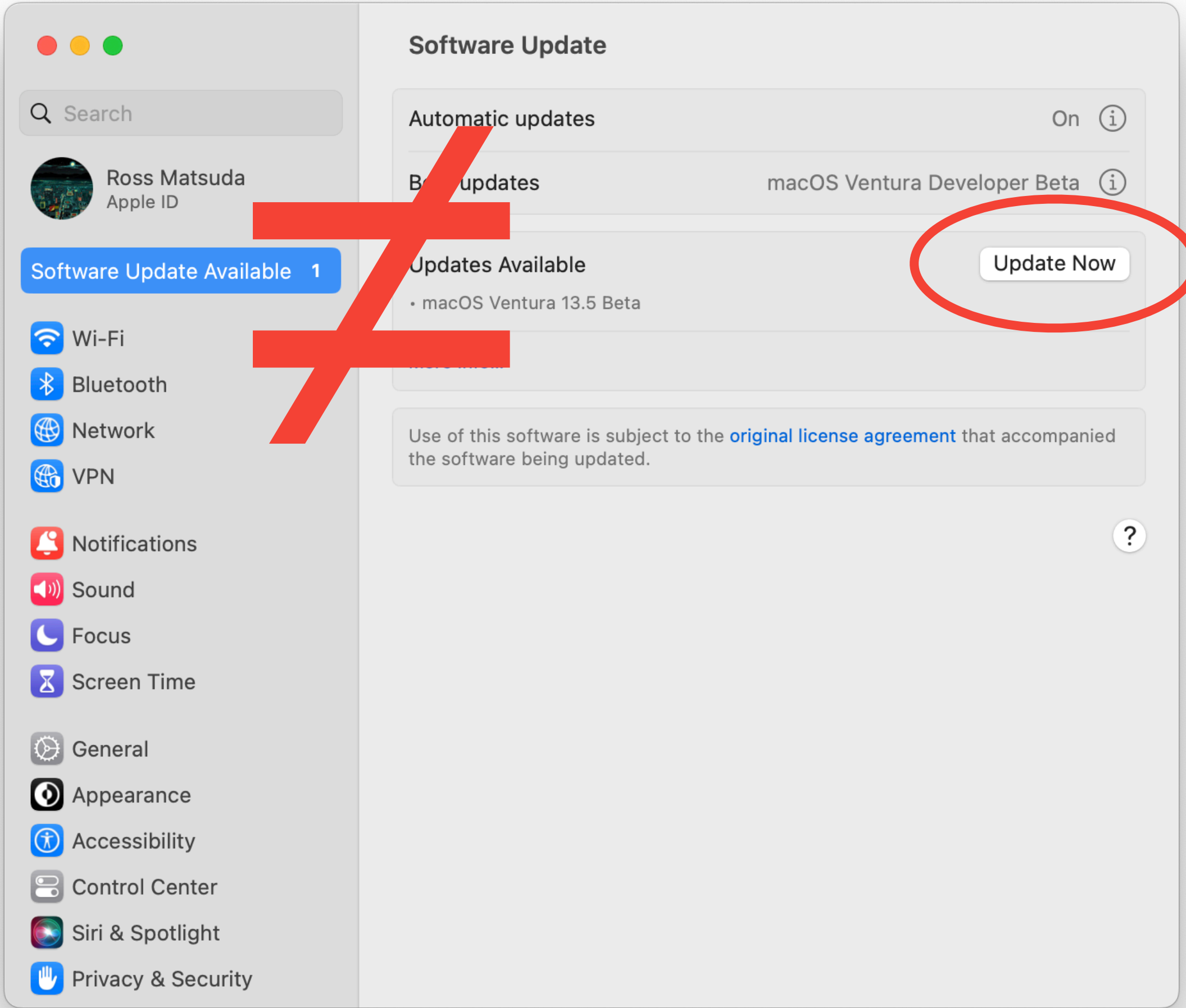
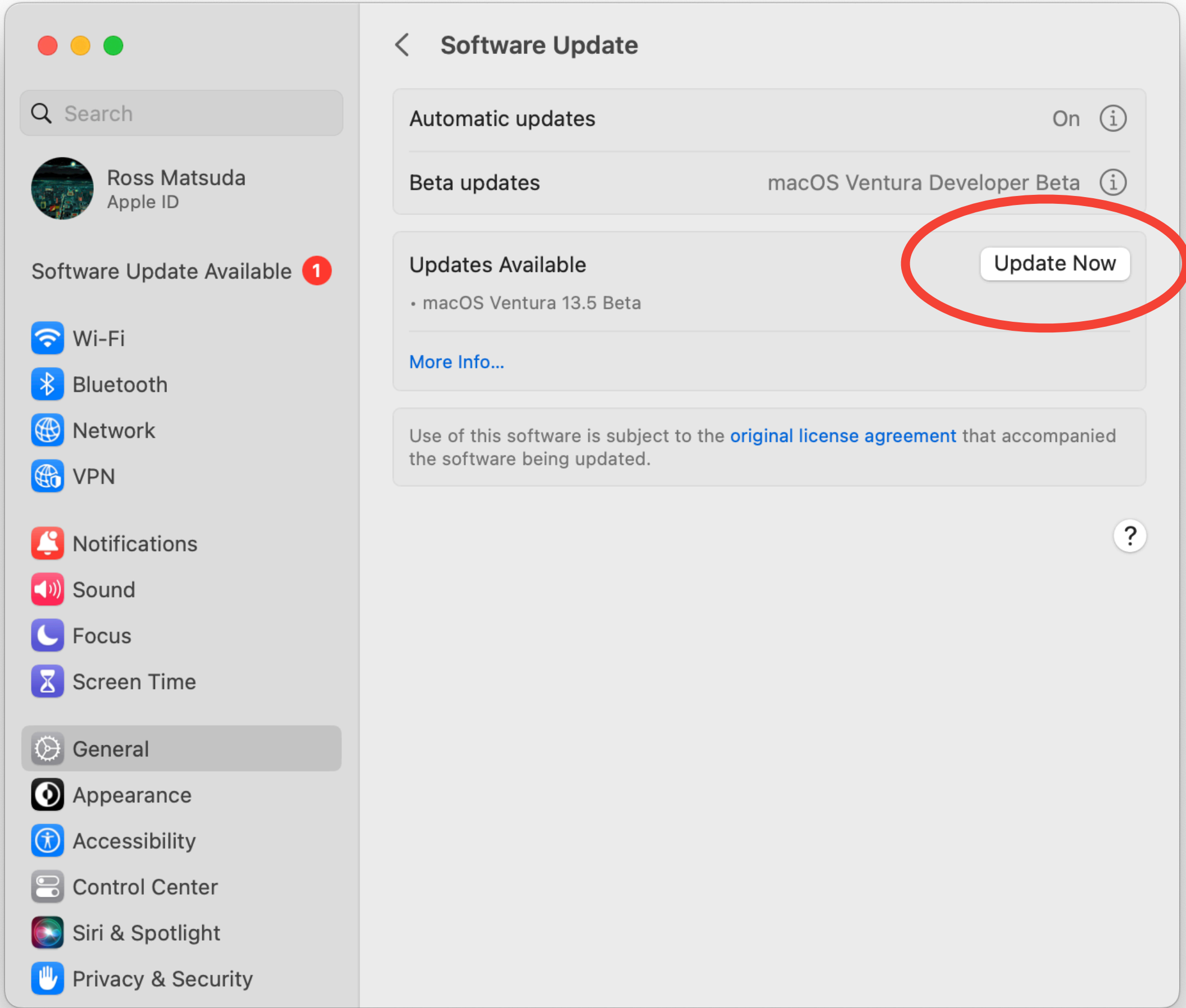
Cons



Cons



Cons



**And that's why this is
a terrible idea.**

So let's do it.

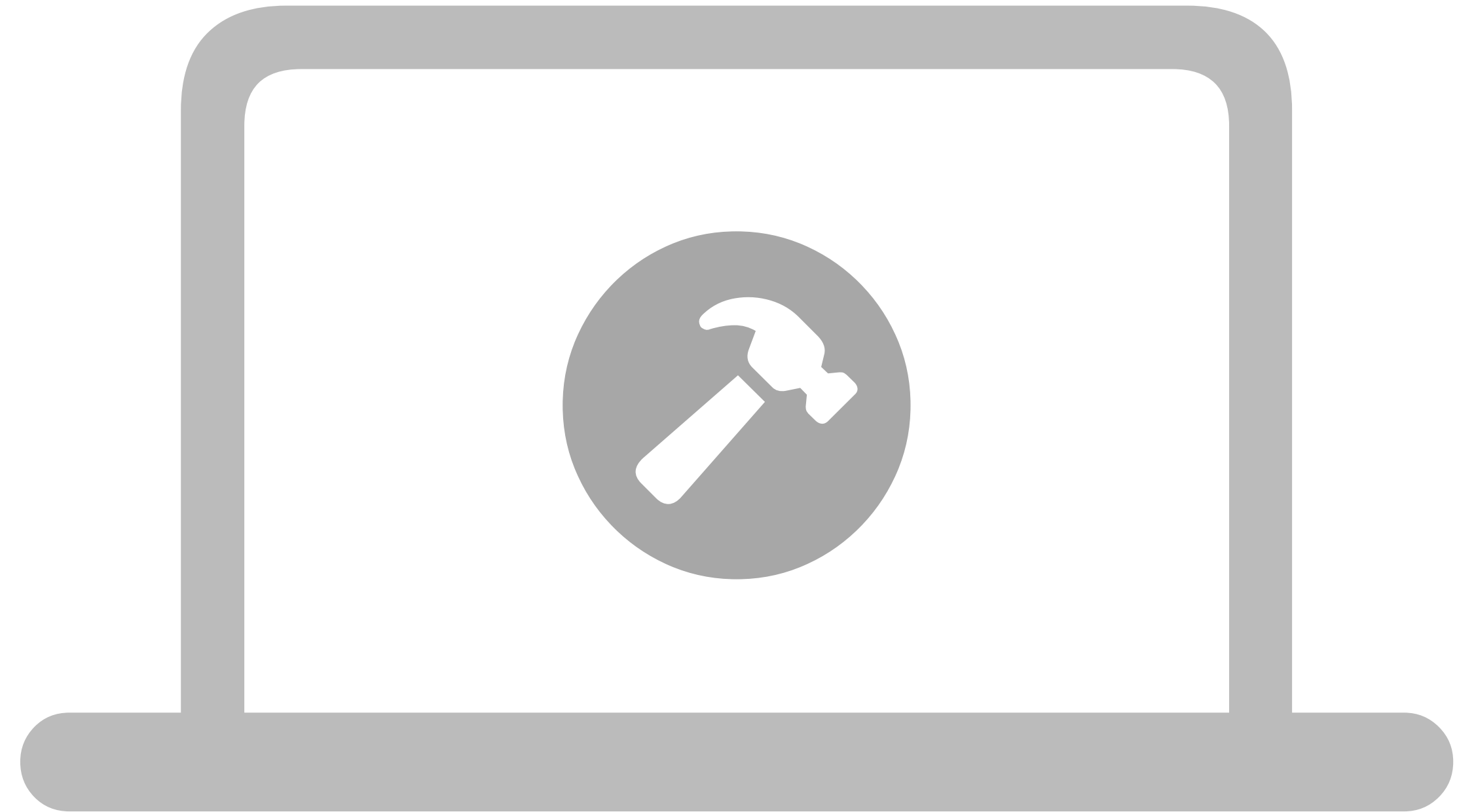


**Become ungovernable.
Write GUI scripts.**

Prerequisites

1 - Test Environment

- Hardware-required or VM?
- OS targets (major/minor)
- App targets
- User account settings
- Constraints



Prerequisites

2 - Delivery method

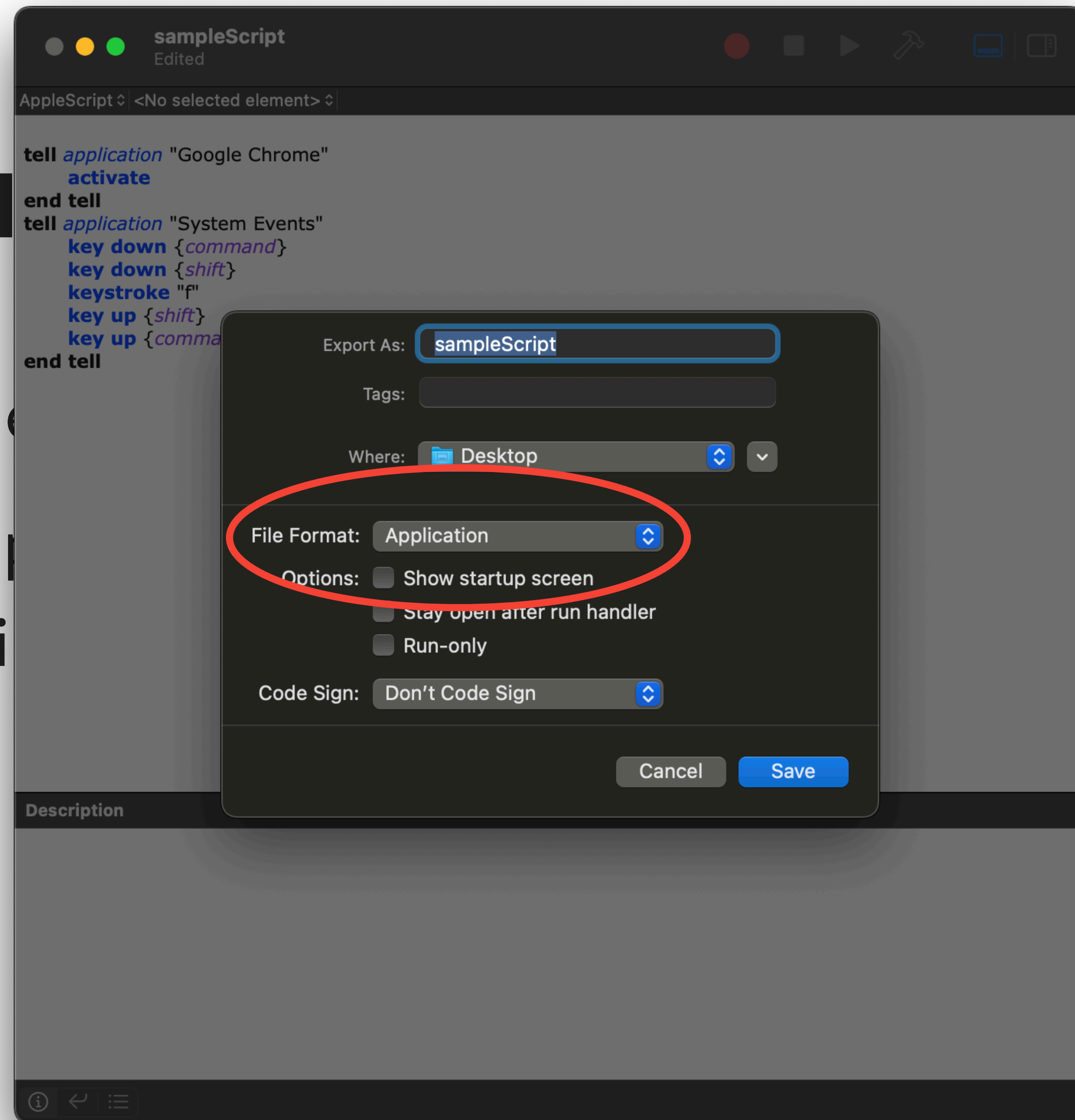
- AppleScript App
- `osascript`



Prerequisite

2 - Delivery method

- AppleScript
- osascript




```
#!/bin/bash
osascript <<EOD
    tell application "Google Chrome"
        activate
    end tell
    tell application "System Events"
        key down {command}
        key down {shift}
        keystroke "f"
        key up {shift}
        key up {command}
    end tell
EOD
```

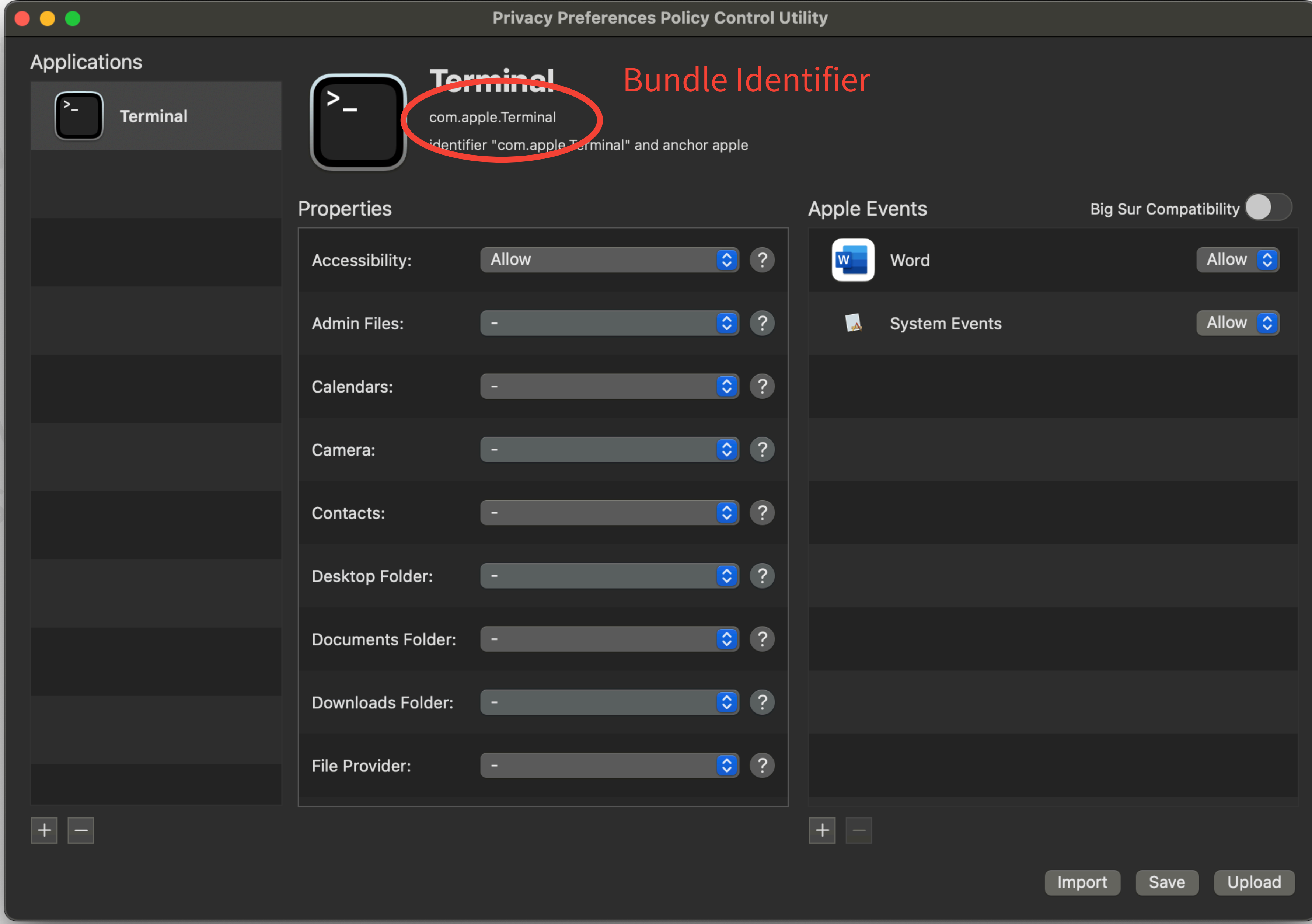
Prerequisites

3 - PPPCs

- Accessibility
- System Events




```
<key>Accessibility</key>
<array>
  <dict>
    <key>IdentifierType</key>
    <string>bundleID</string>
    <key>Identifier</key>
    <string>com.apple.ScriptEditor.id.Loading-Available-Updates</string>
    <key>CodeRequirement</key>
    <string>identifier "com.apple.ScriptEditor.id.Loading-Available-
Updates" and anchor apple generic and certificate leaf[subject.CN] =
"Apple Development: yourNameHere (XXXXXXXXXX)" and certificate
1[field.2.4.680.135792.468.0.1.3] /* exists */</string>
    <key>Comment</key>
    <string></string>
    <key>Allowed</key>
    <true/>
  </dict>
</array>
```

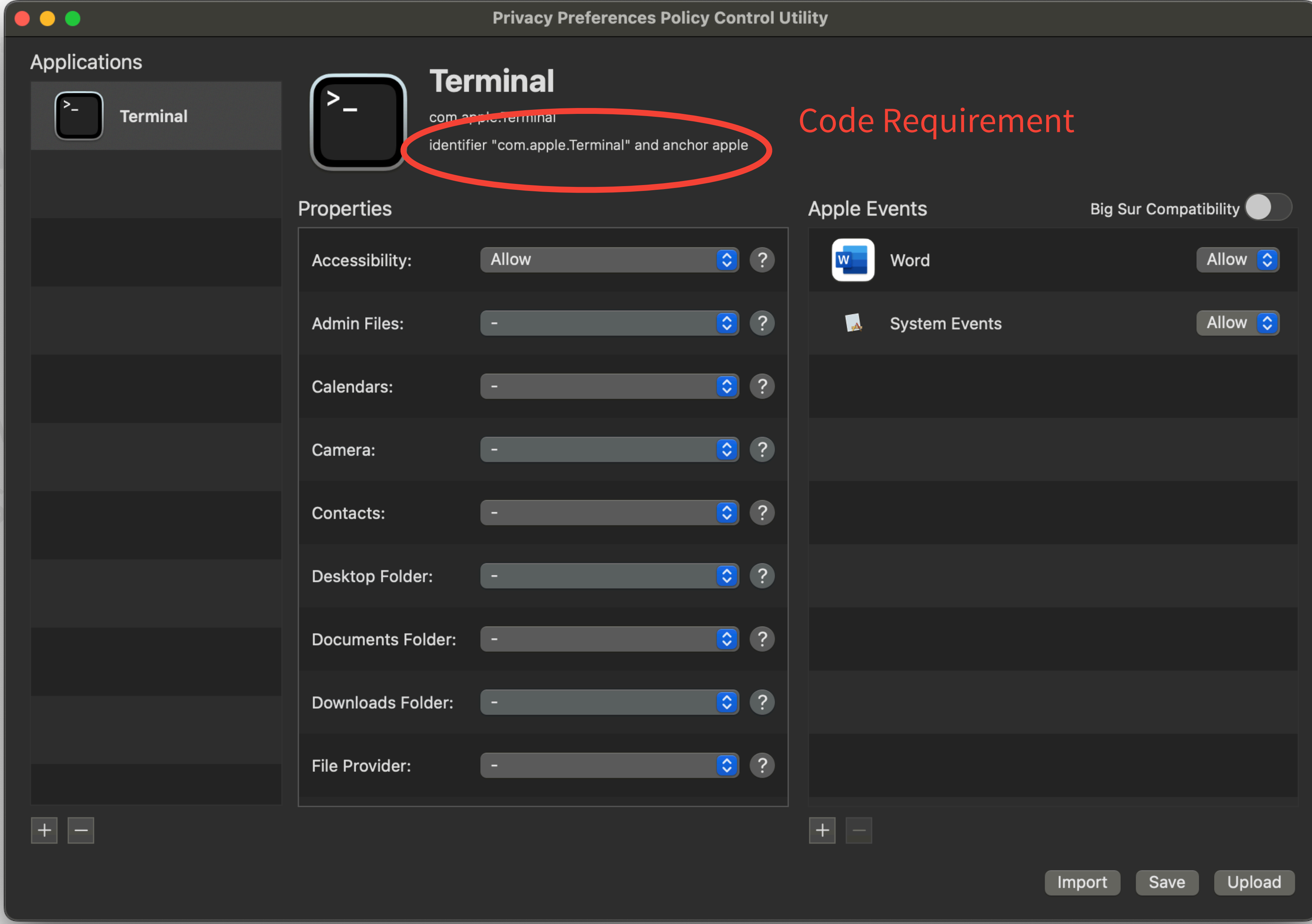


SCRIPTING T

Pre

3 - P

- A
- S



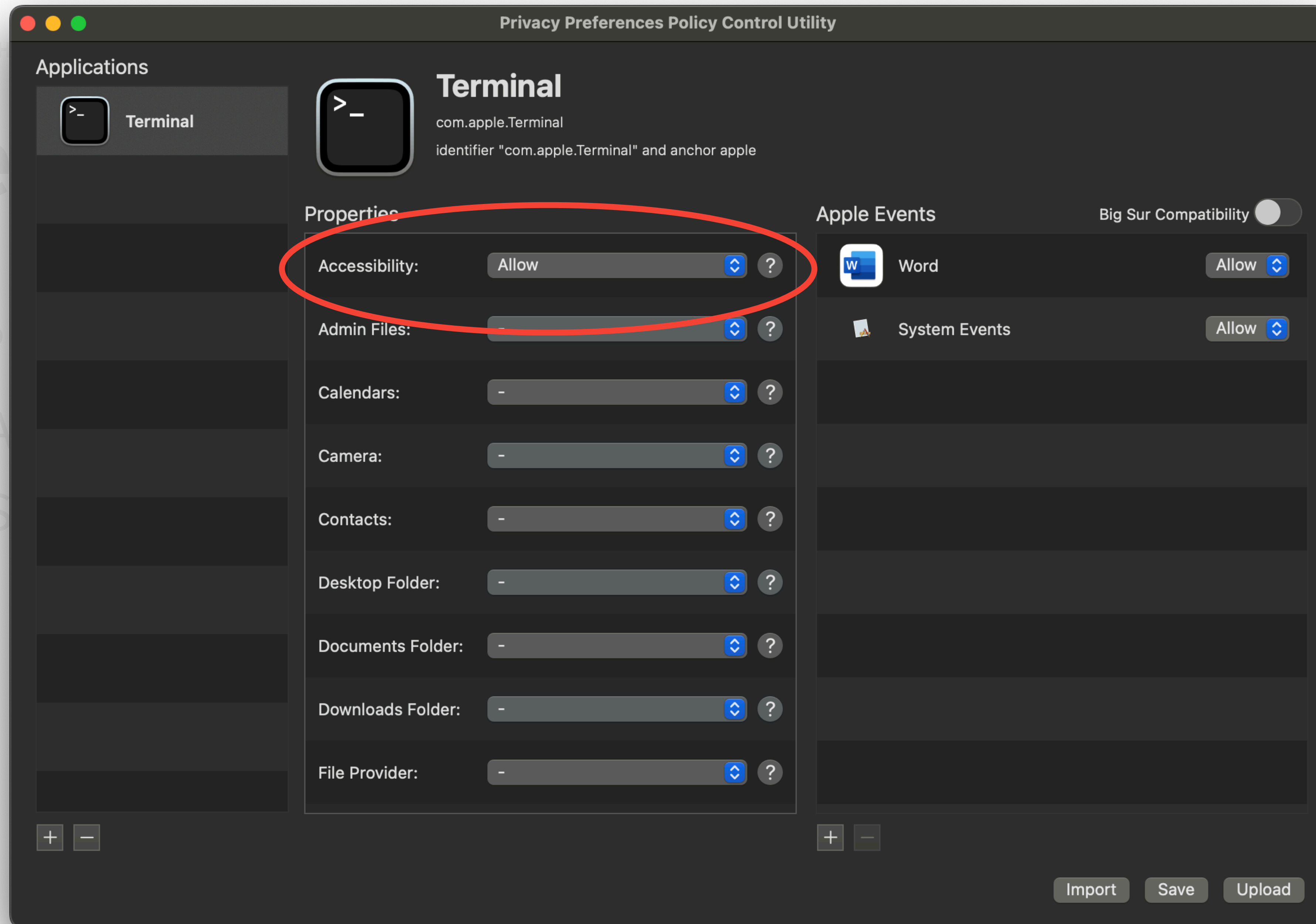
Code Requirement

SCRIPTING T

Pre

3 - P

- A
- S

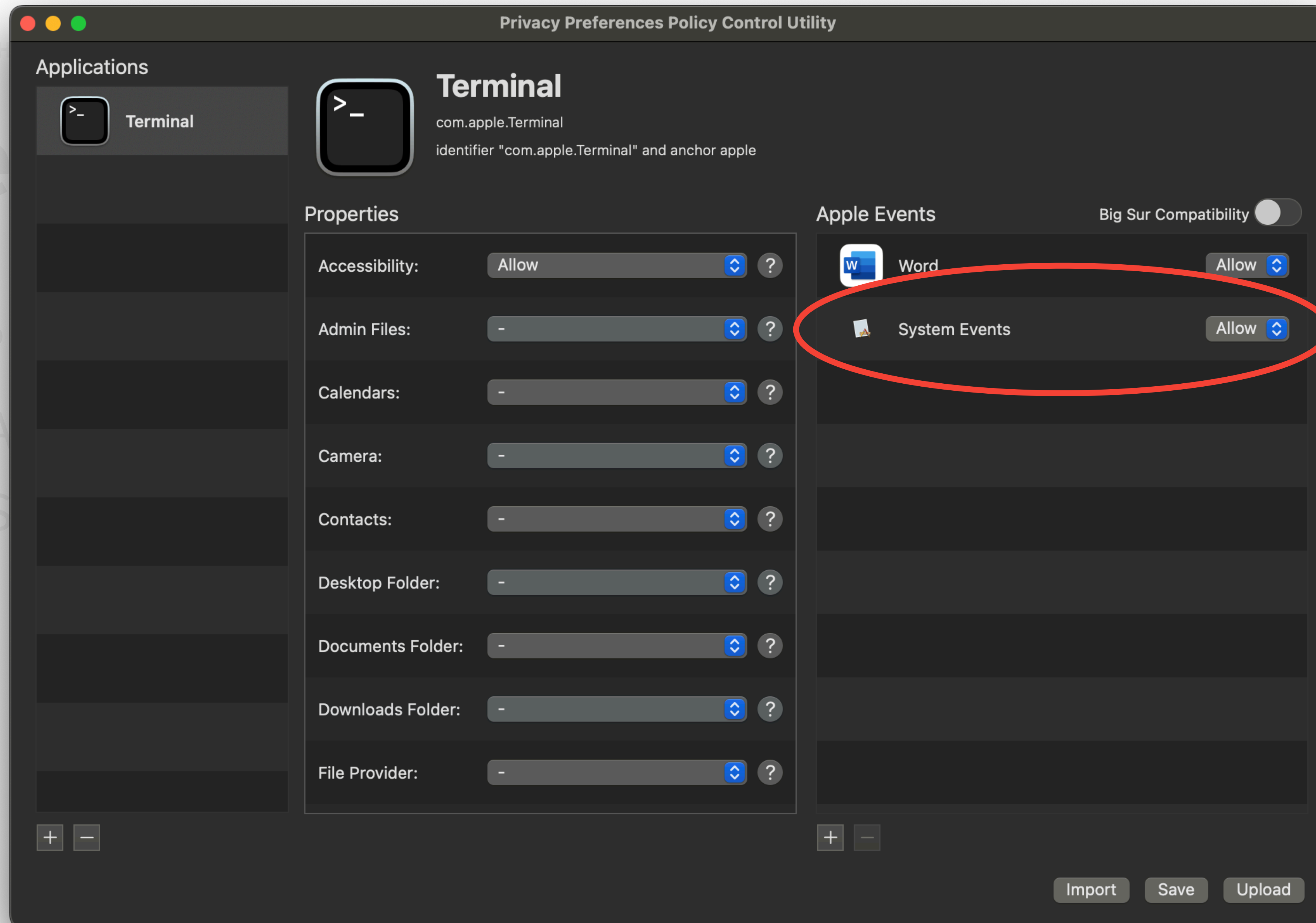


SCRIPTING T

Pre

3 - P

- A
- S



Prerequisites

3 - PPPCs

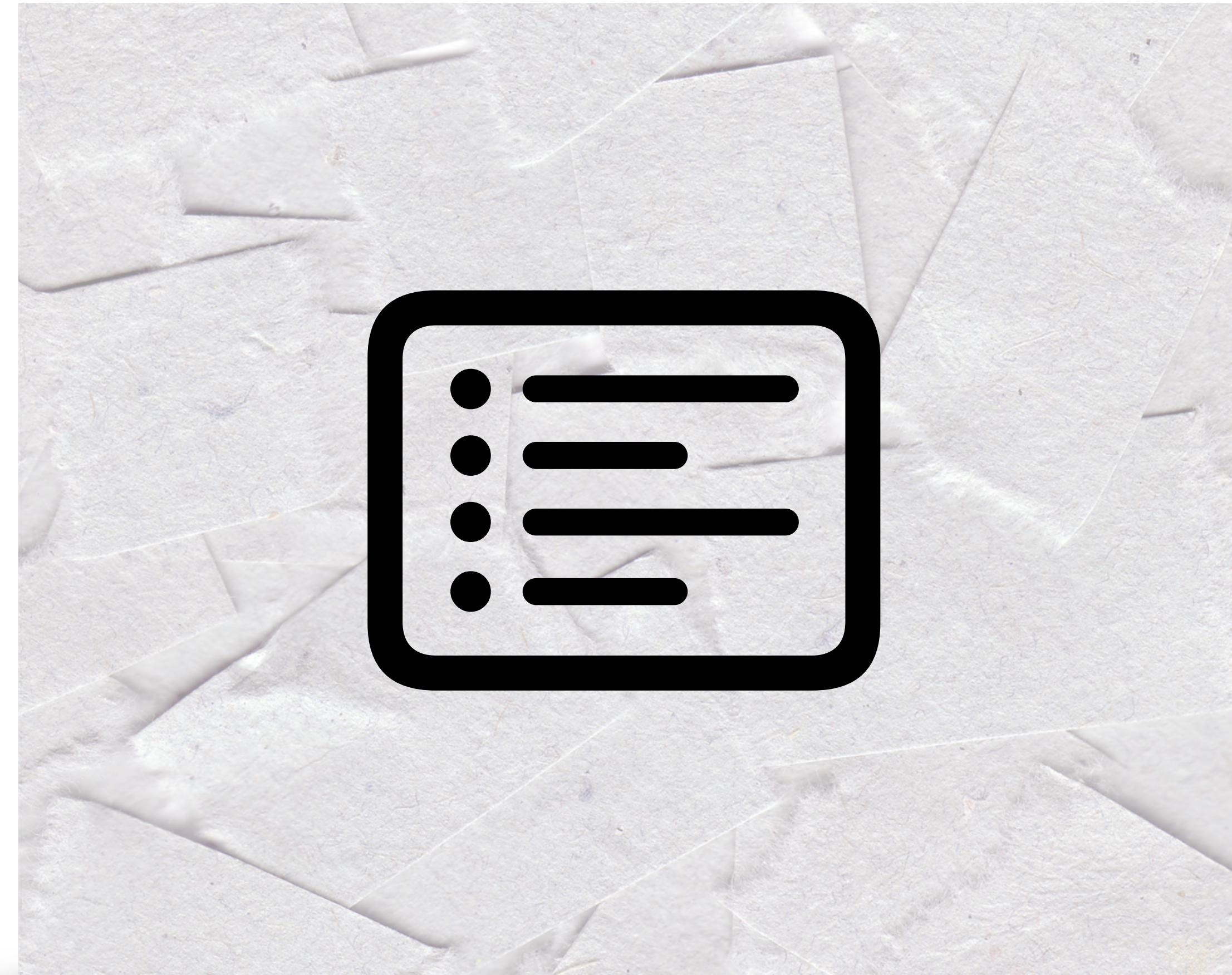
- Accessibility
- System Events




```
<dict>
  <key>IdentifierType</key>
  <string>bundleID</string>
  <key>Identifier</key>
  <string>com.apple.ScriptEditor.id.Loading-Available-Updates</string>
  <key>Comment</key>
  <string></string>
  <key>CodeRequirement</key>
  <string>identifier "com.apple.ScriptEditor.id.Loading-Available-Updates"
and anchor apple generic and certificate leaf[subject.CN] = "Apple
Development: yourNameHere (XXXXXXXXXX)" and certificate
1[field.2.4.680.135792.468.0.1.3] /* exists */</string>
  <key>Allowed</key>
  <true/>
  <key>AEReceiverIdentifierType</key>
  <string>bundleID</string>
  <key>AEReceiverIdentifier</key>
  <string>com.apple.systempreferences</string>
  <key>AEReceiverCodeRequirement</key>
  <string>identifier "com.apple.systempreferences" and anchor
apple</string>
</dict>
```

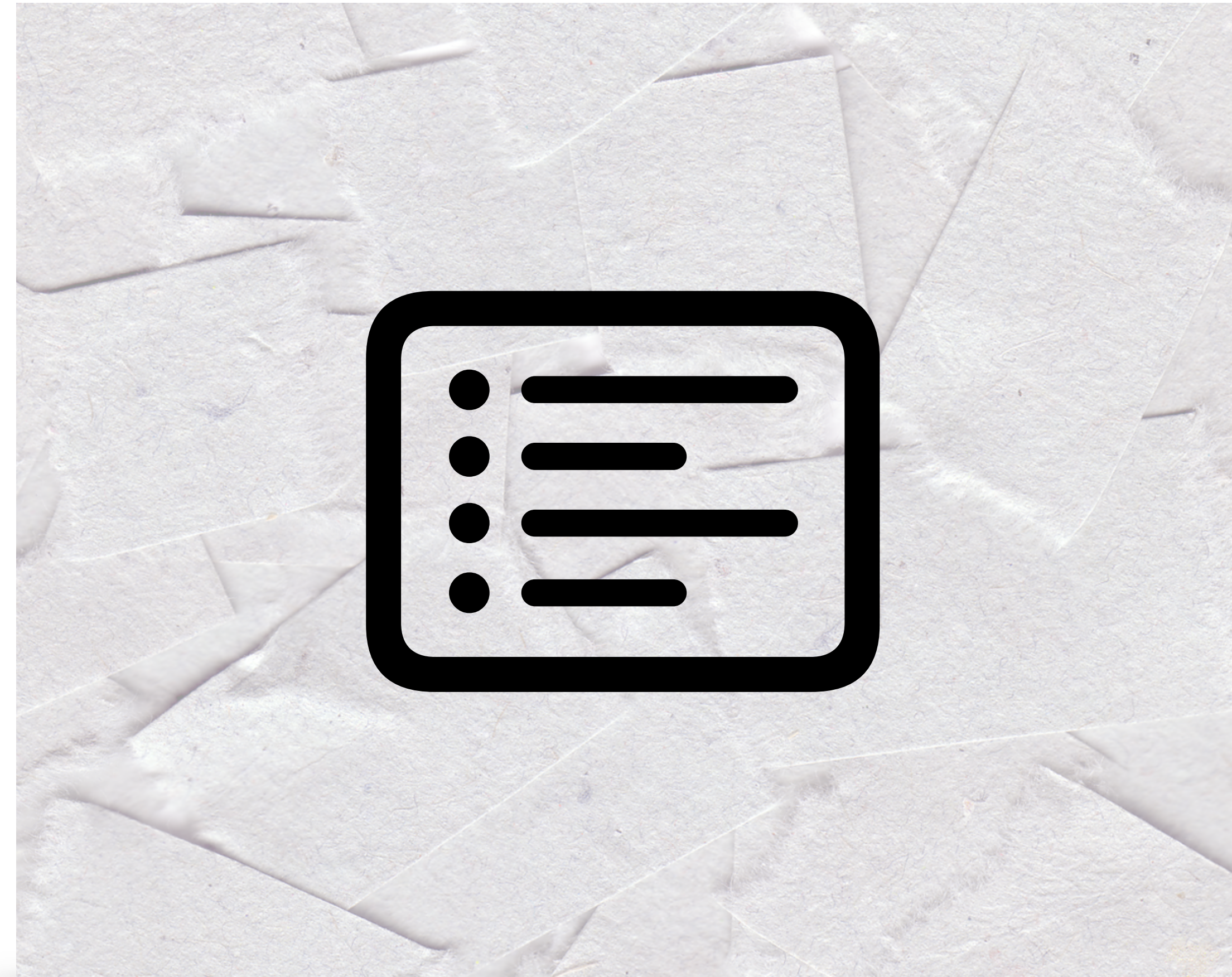

1 - Set your workflow

- Pretend you're helping a user
- Start with device state
- Detailed instructions
- Identify your final interactive step



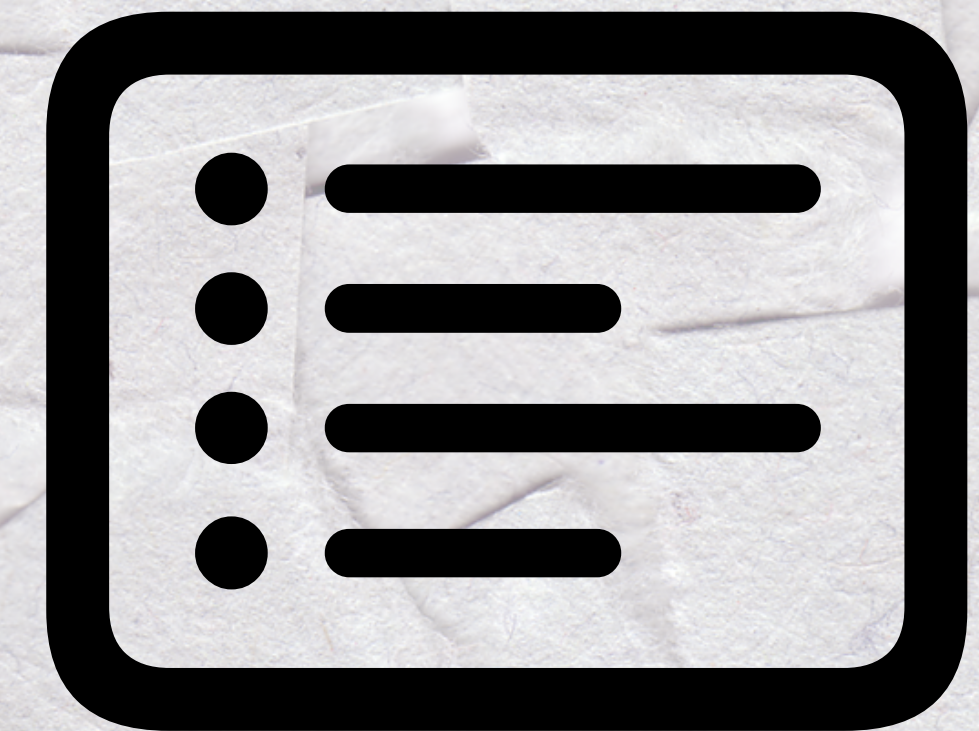
1 - Set your workflow

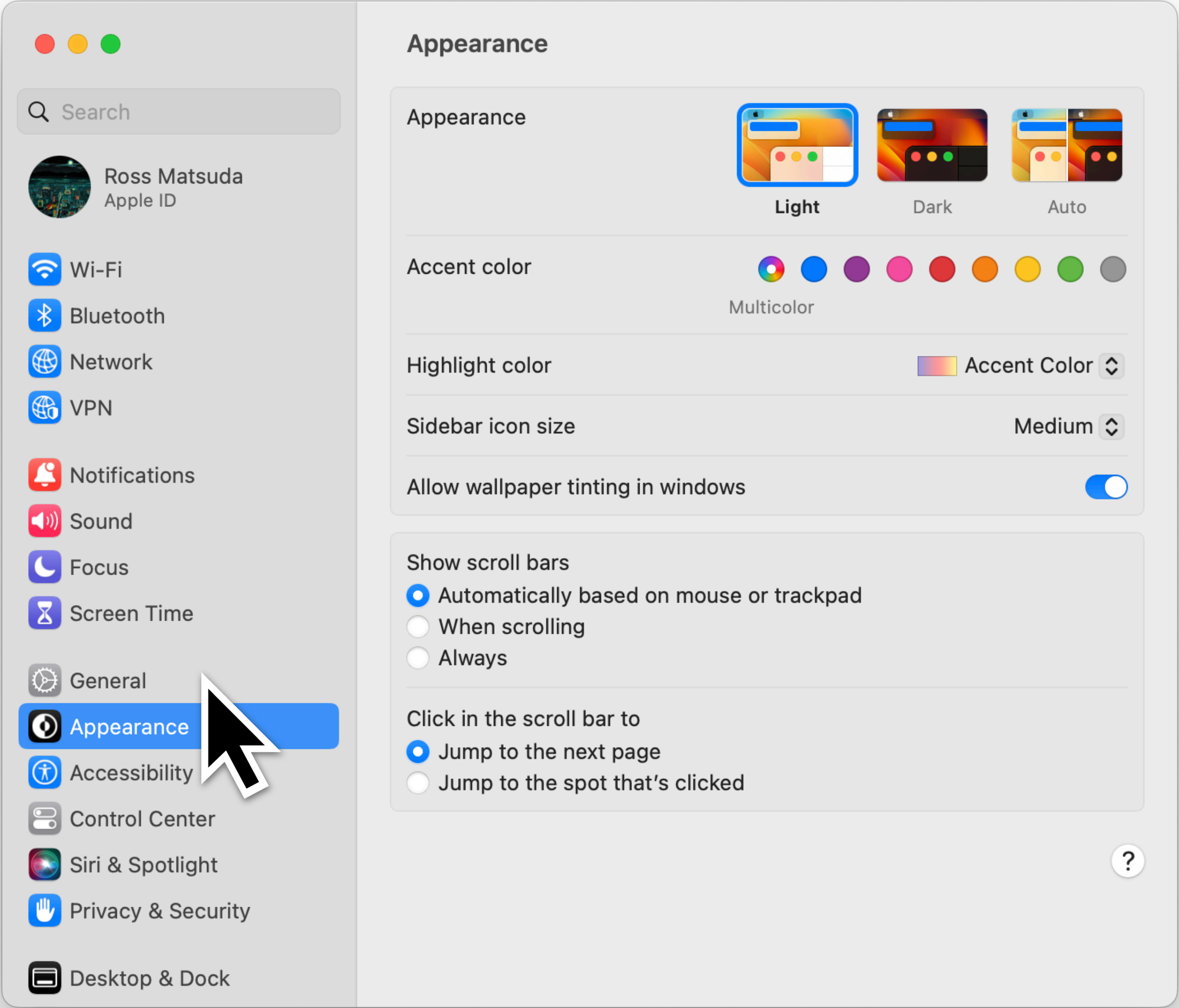
- Any steps to remove for savvy users?
- What steps can be handled with CLI?
- Use GUI scripting for as few steps as possible

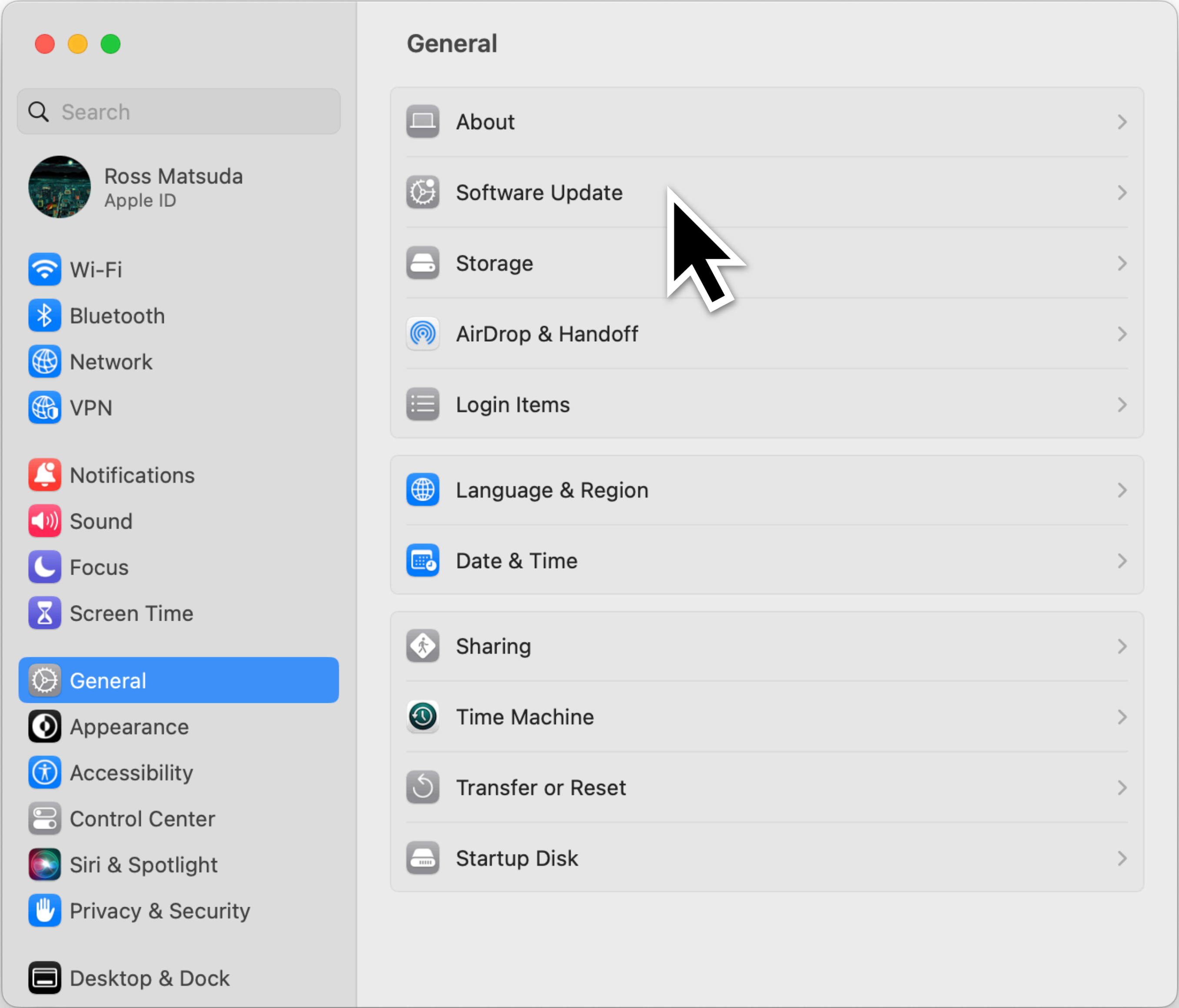


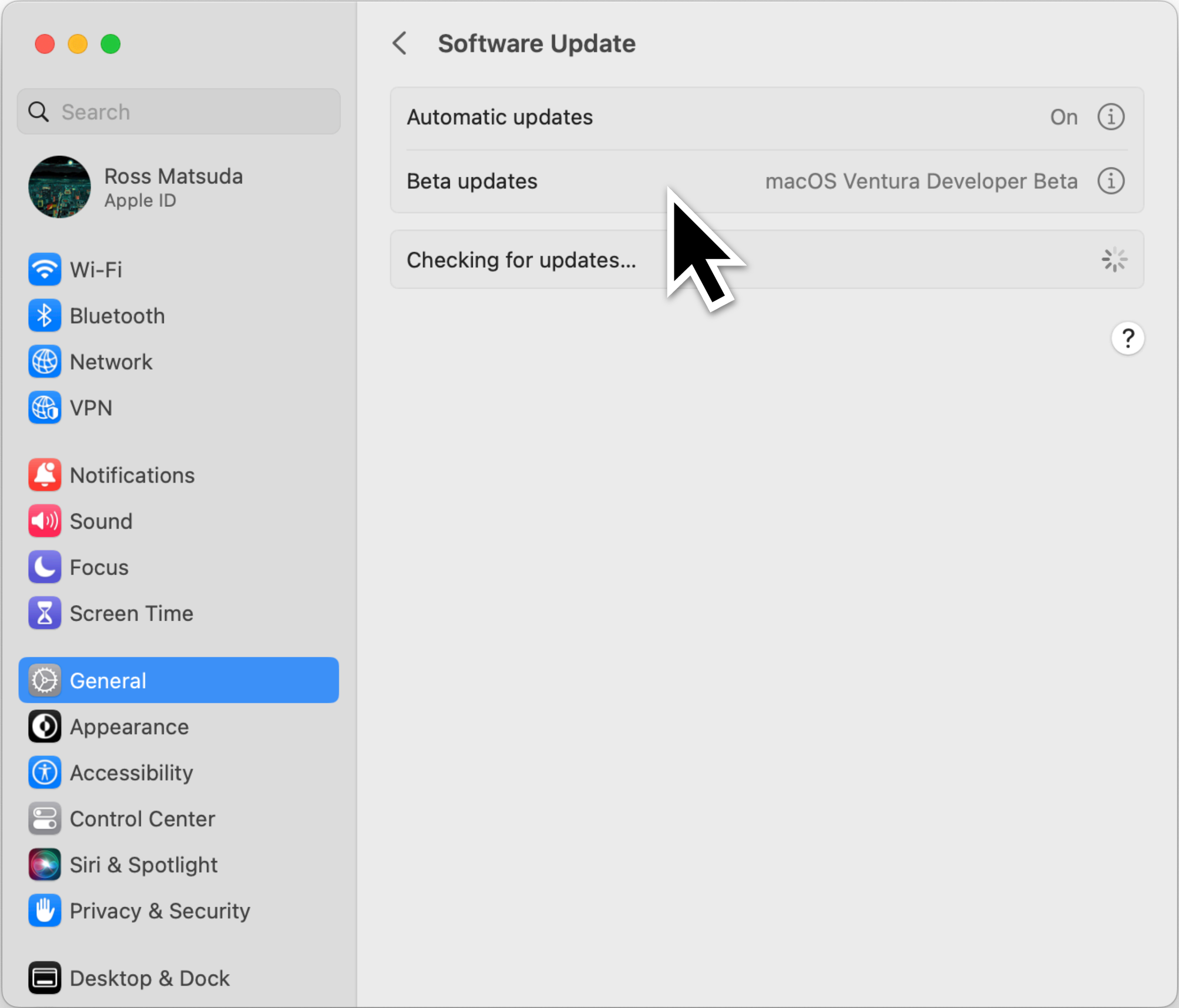
1 - Set your workflow

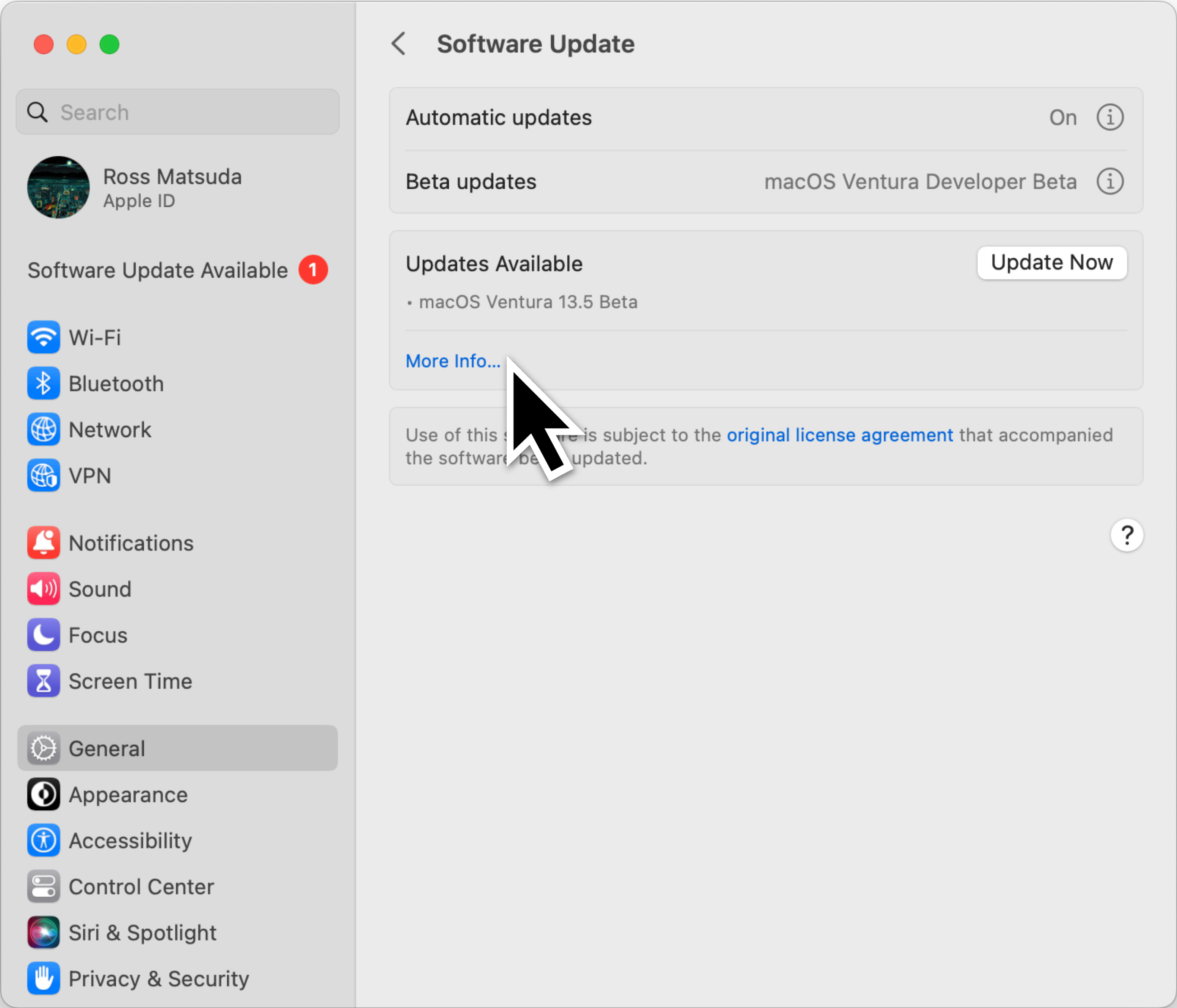
For today's nonsense, let's look at clicking the "More Info" button in System Settings, Software Update

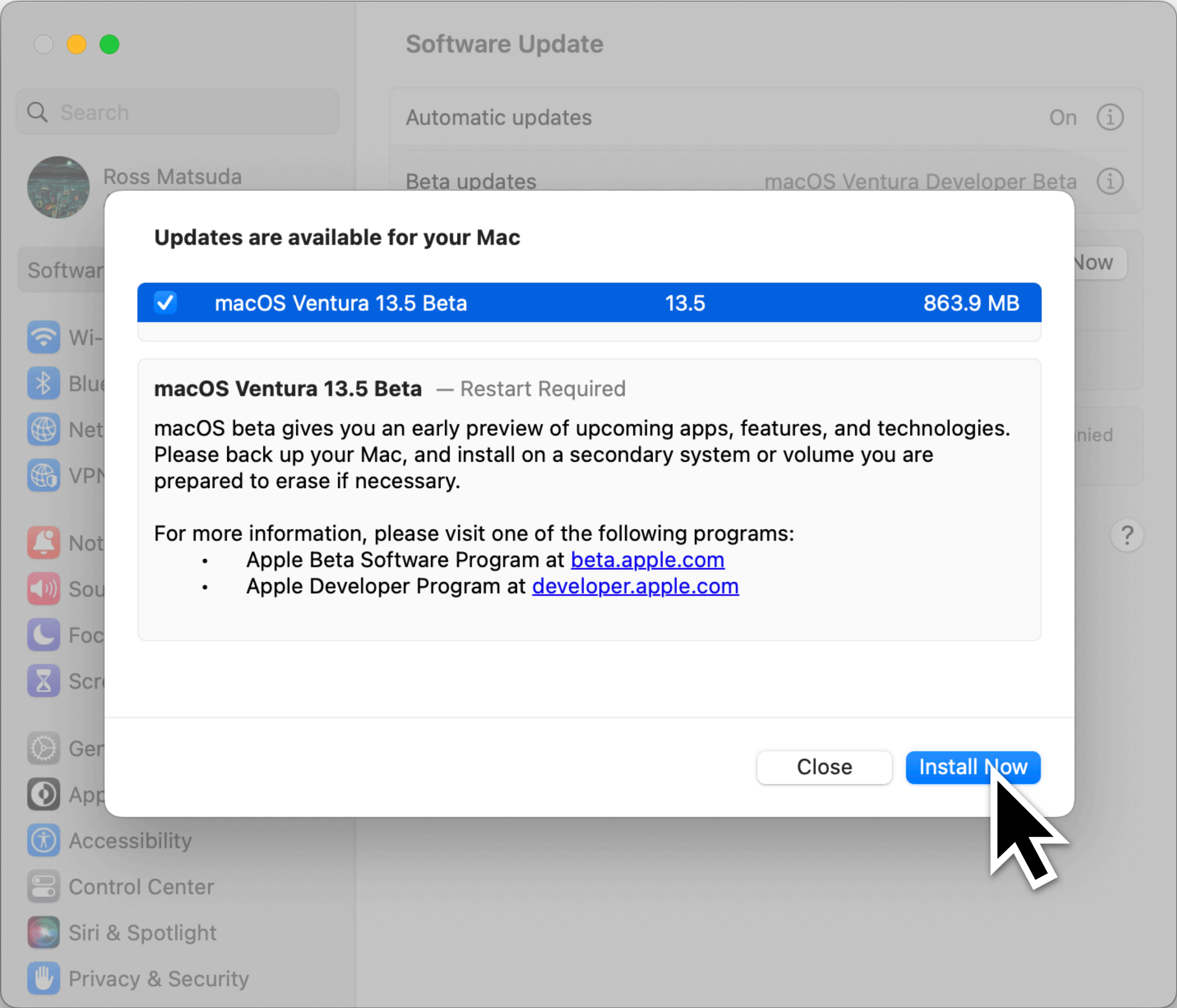










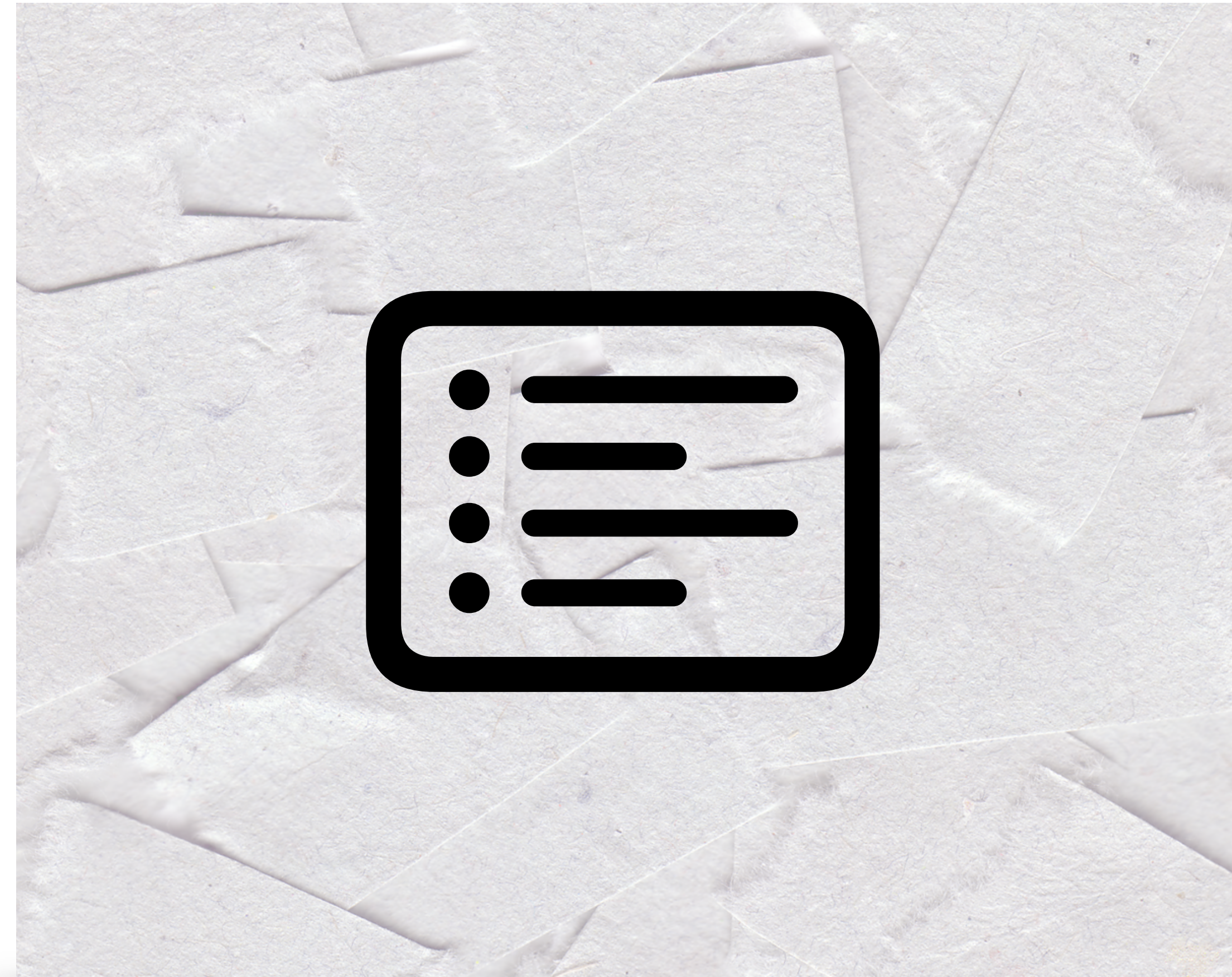


**Regarding the "More Info" target in macOS 13+
FB11749823**

2 - Define element names

Start from the first step you can't handle with CLI

- Open System Preferences
- Click on General
- Click on Software Update



2 - Define element names

Start from the first step you

```
open x-apple.systempreferences:  
com.apple.Software-Update-Settings.extension
```

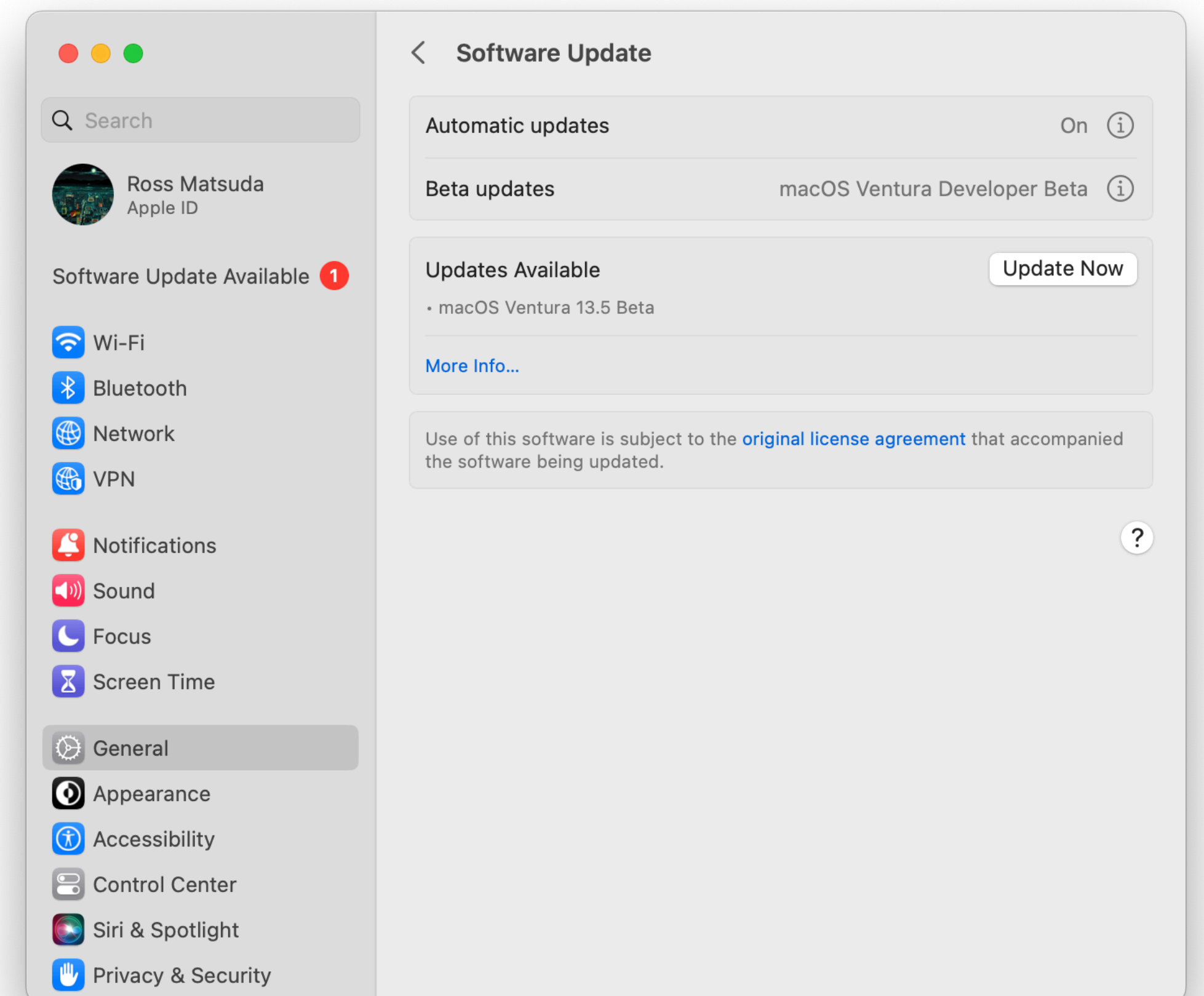
- Open System Preferences
- Click on General
- Click on Software Update



2 - Define element names

```
tell application "System Events"  
  tell front window of process "System Settings"  
    set uiElems to entire contents  
  end tell  
end tell
```

- Customize the second line to change targets





Untitled

Edited



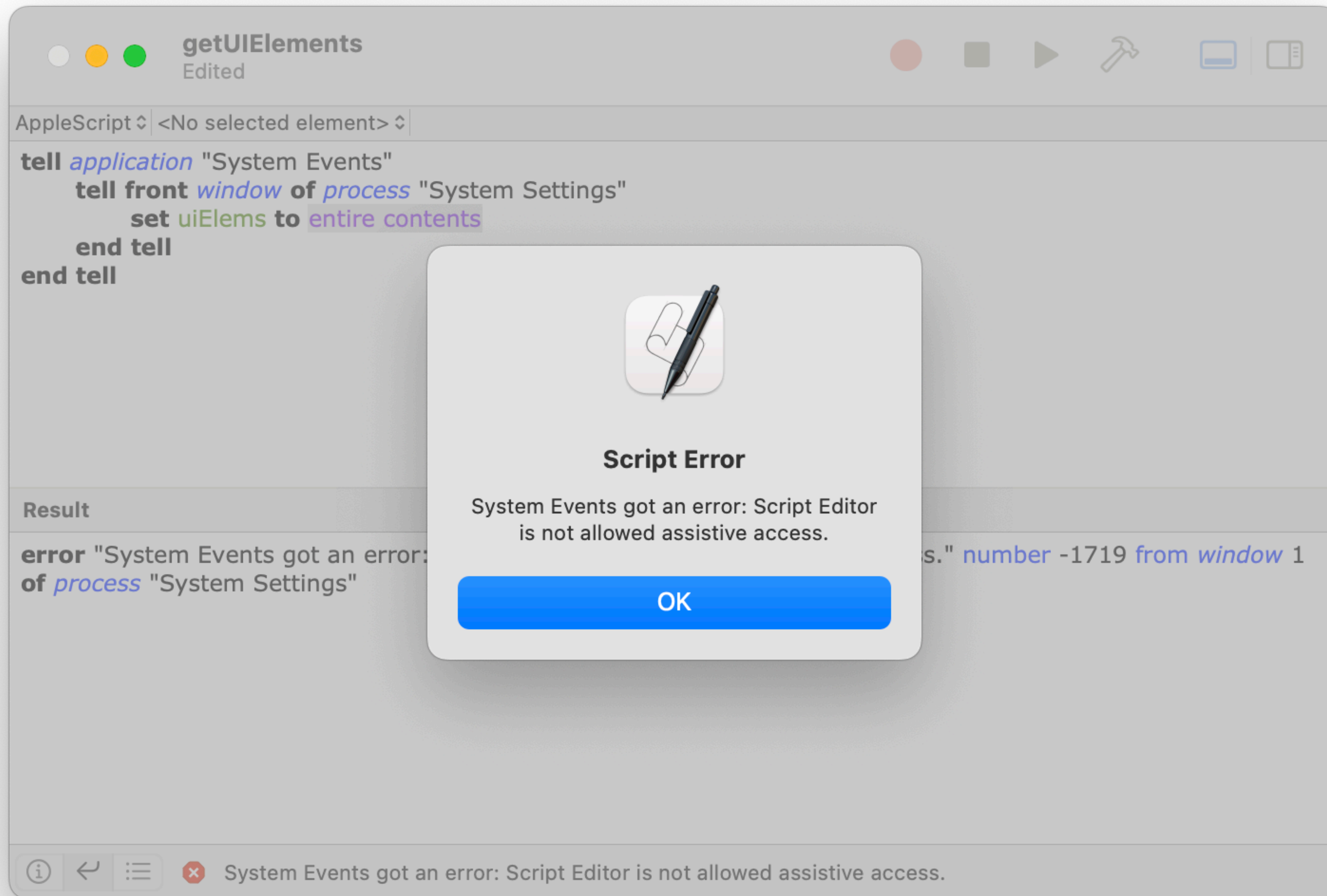
AppleScript ⌵ <No selected element> ⌵

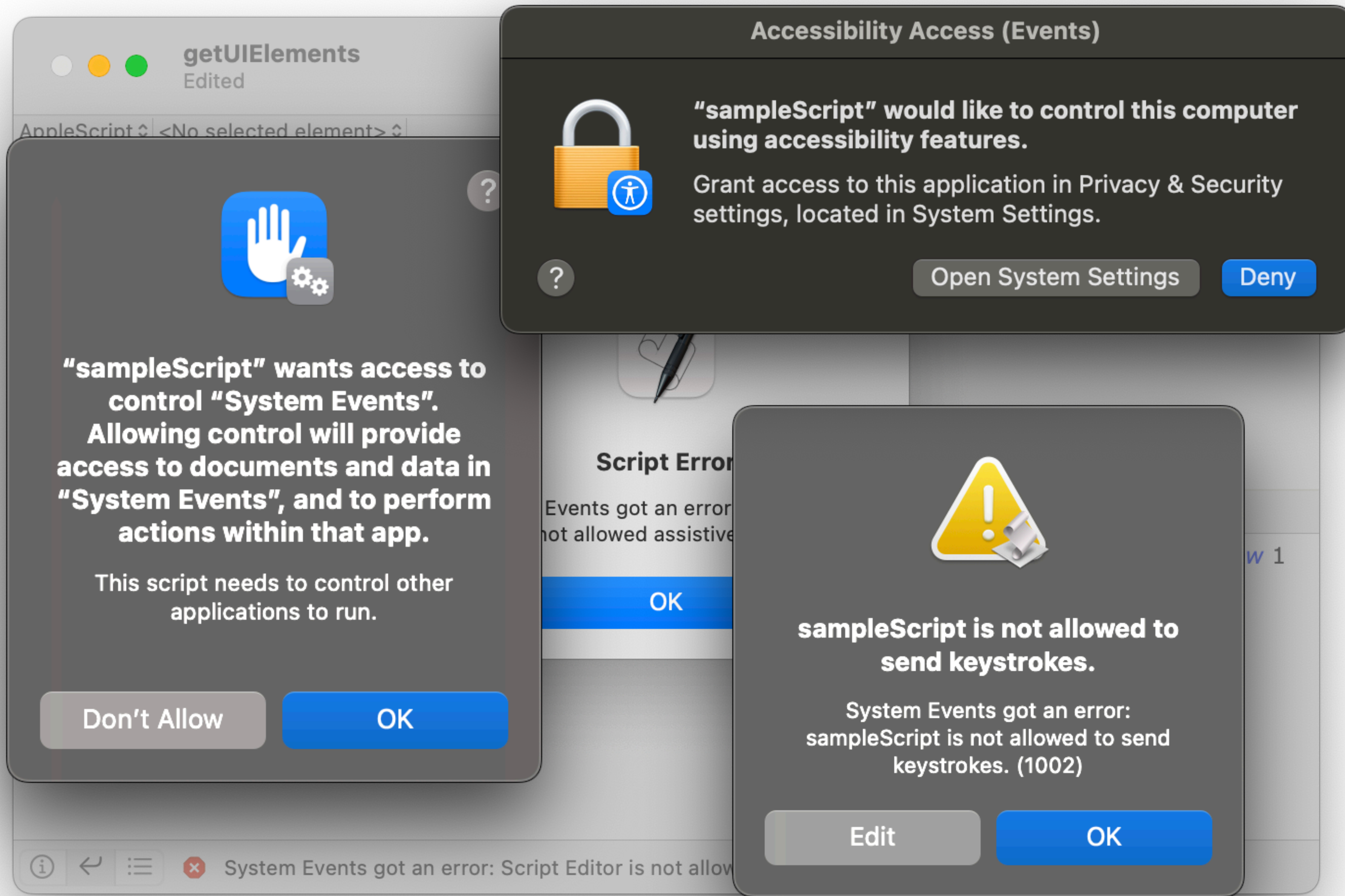
```
tell application "System Events"  
  tell front window of process "System Settings"  
    set uiElems to entire contents  
  end tell  
end tell
```

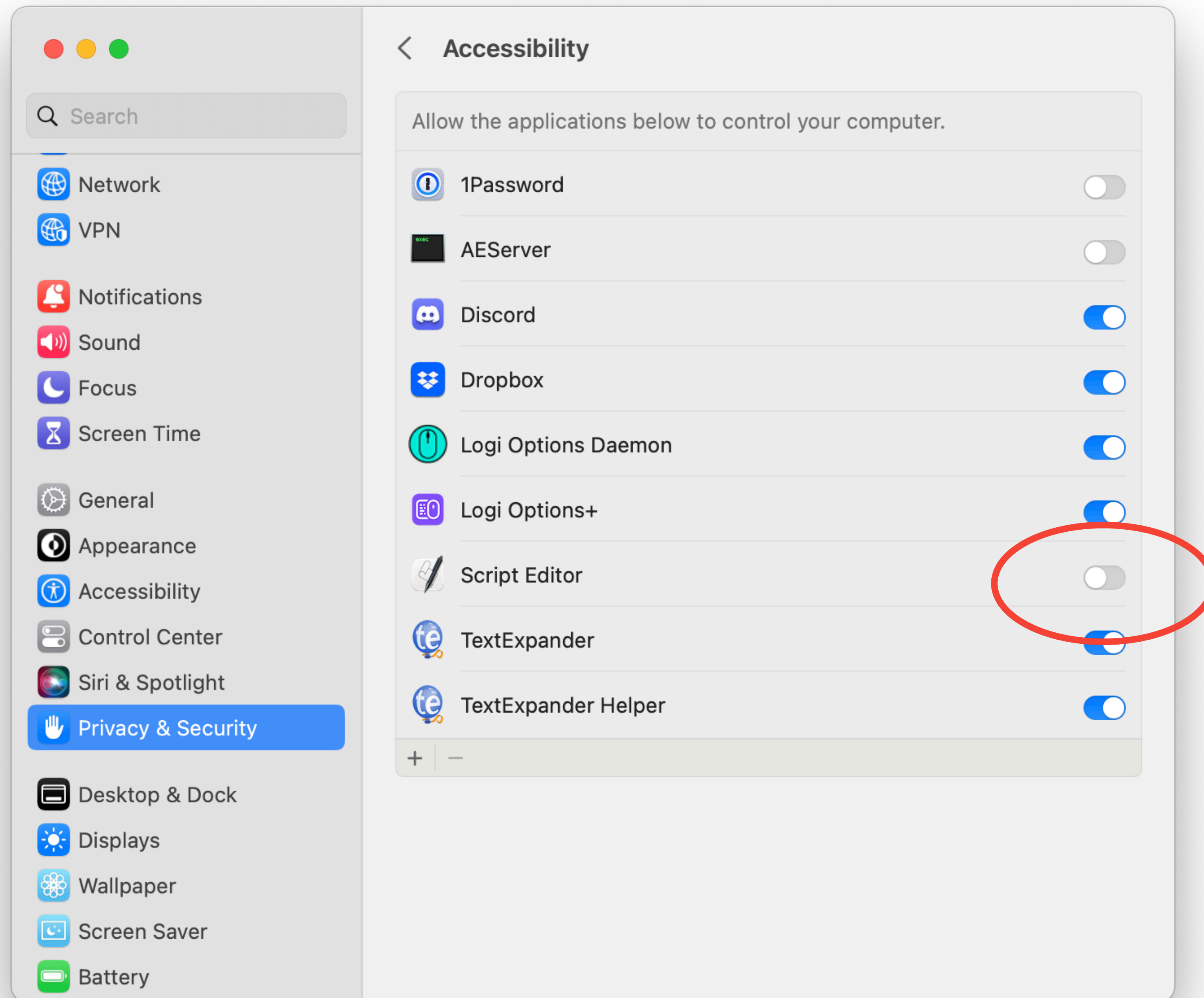
Result

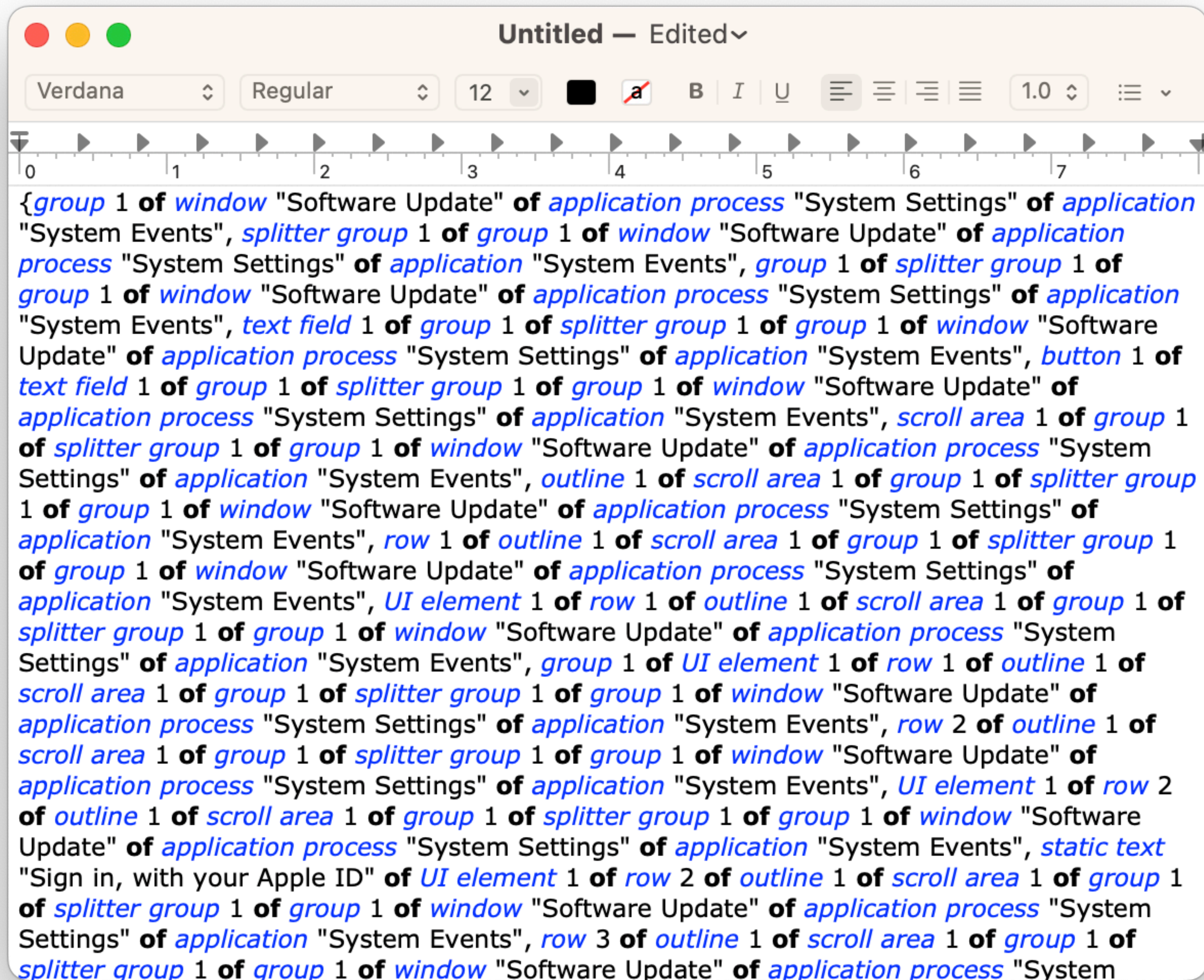
```
{group 1 of window "Appearance" of application process "System Settings" of application "System Events",  
splitter group 1 of group 1 of window "Appearance" of application process "System Settings" of application  
"System Events", group 1 of splitter group 1 of group 1 of window "Appearance" of application process "System  
Settings" of application "System Events", text field 1 of group 1 of splitter group 1 of group 1 of window  
"Appearance" of application process "System Settings" of application "System Events", button 1 of text field 1  
of group 1 of splitter group 1 of group 1 of window "Appearance" of application process "System Settings" of  
application "System Events", scroll area 1 of group 1 of splitter group 1 of group 1 of window "Appearance" of  
application process "System Settings" of application "System Events", outline 1 of scroll area 1 of group 1 of  
splitter group 1 of group 1 of window "Appearance" of application process "System Settings" of application  
"System Events", row 1 of outline 1 of scroll area 1 of group 1 of splitter group 1 of group 1 of window  
"Appearance" of application process "System Settings" of application "System Events", UI element 1 of row 1 of
```





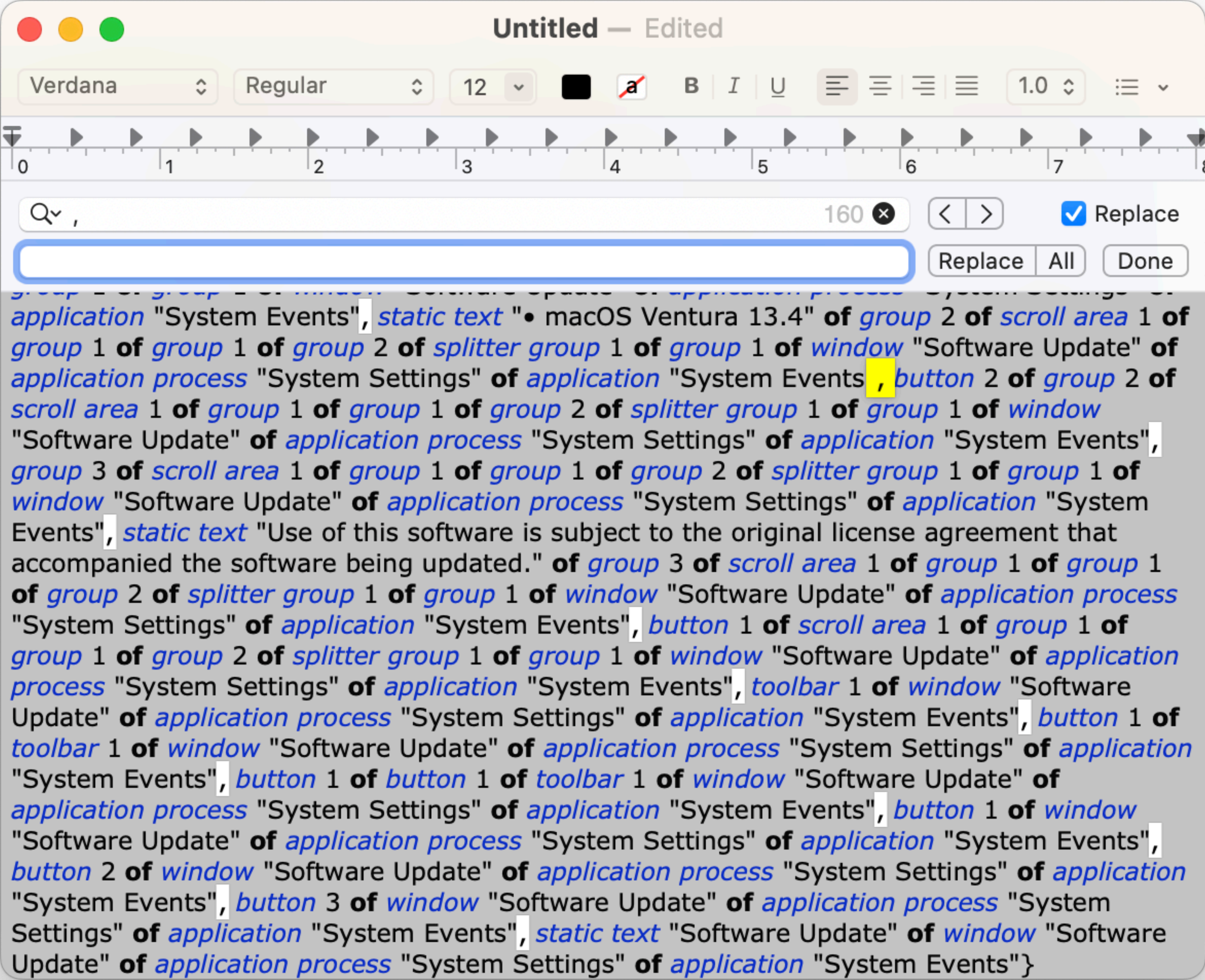


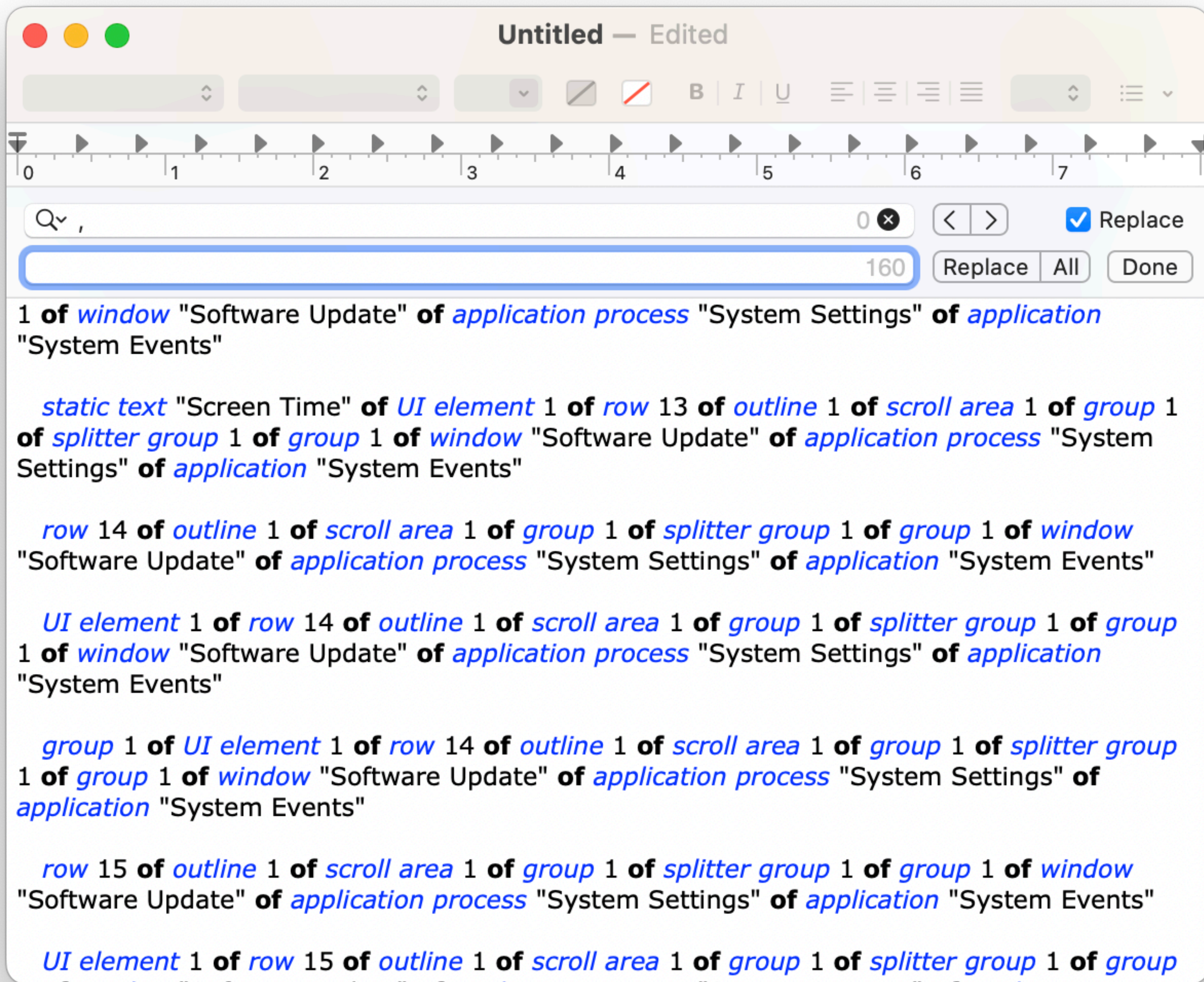




CARRIAGE RETURN

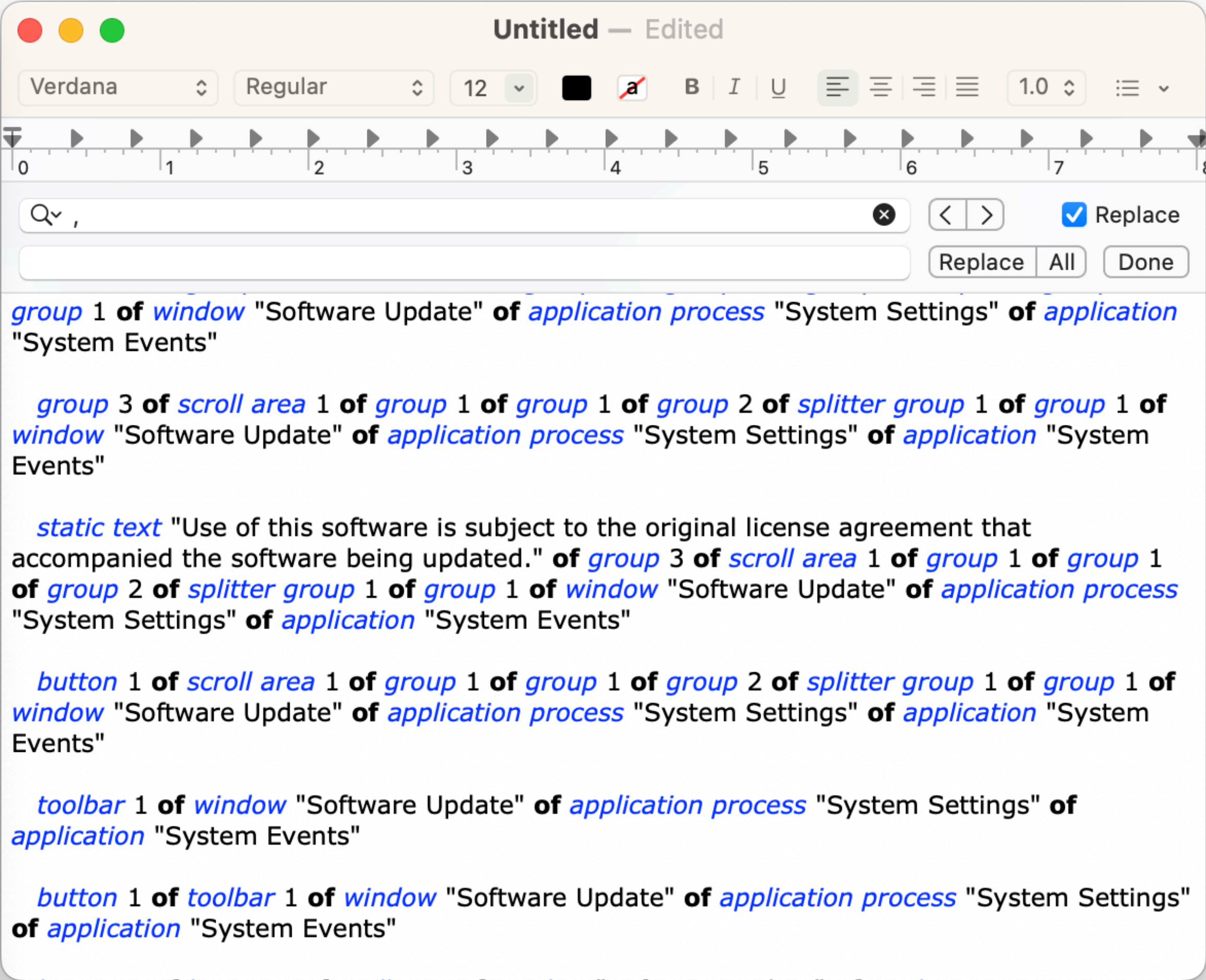
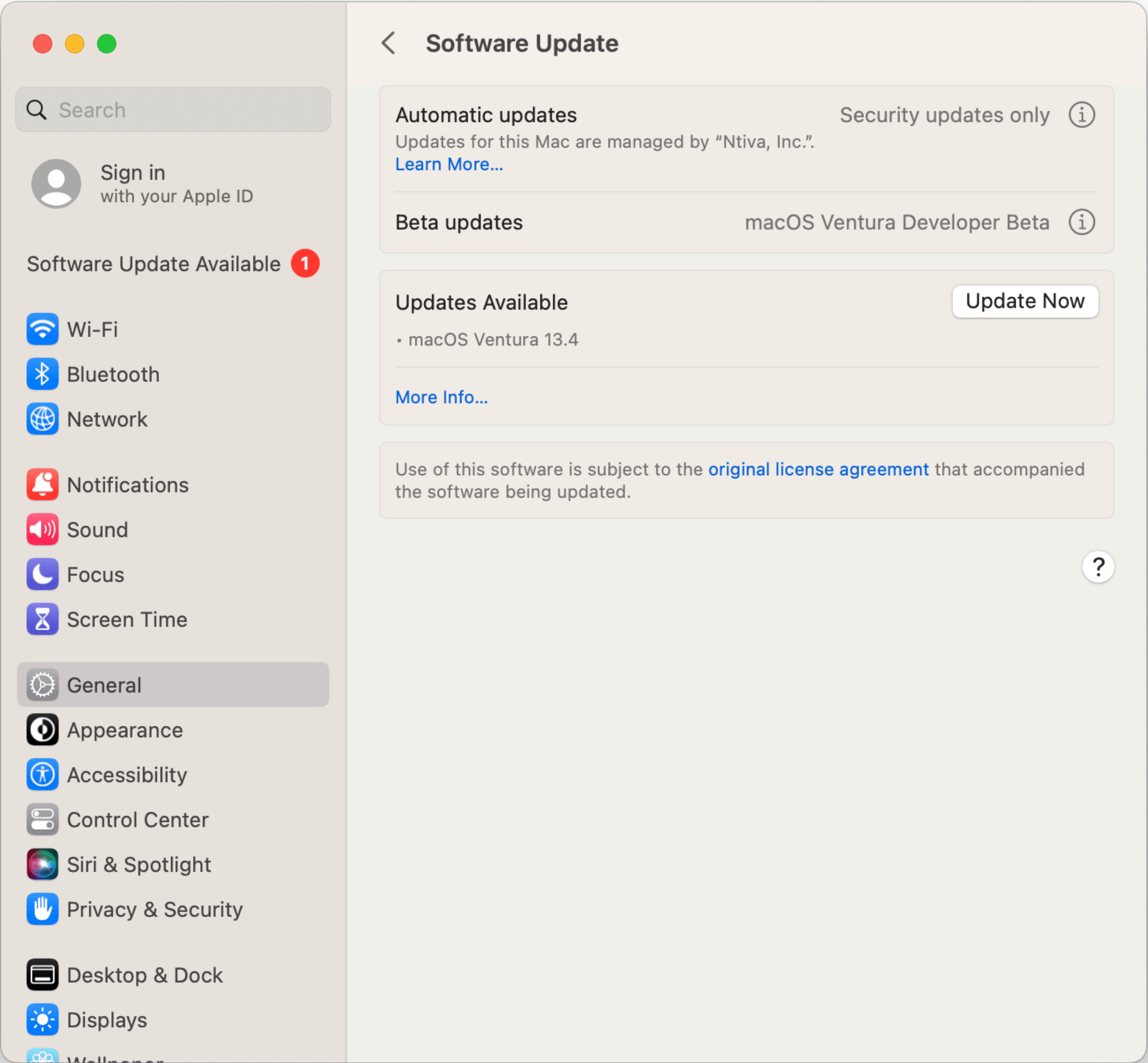
\-return



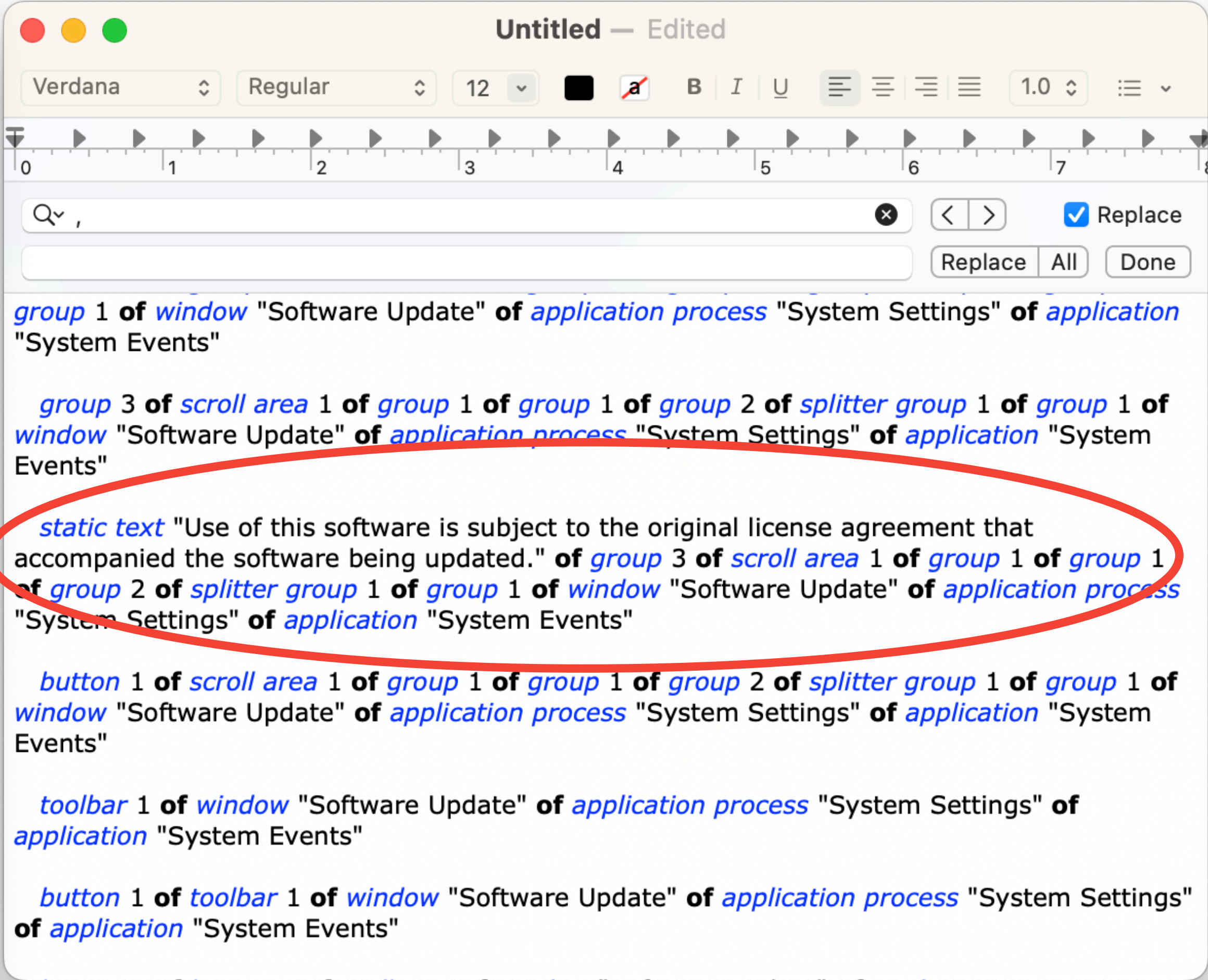
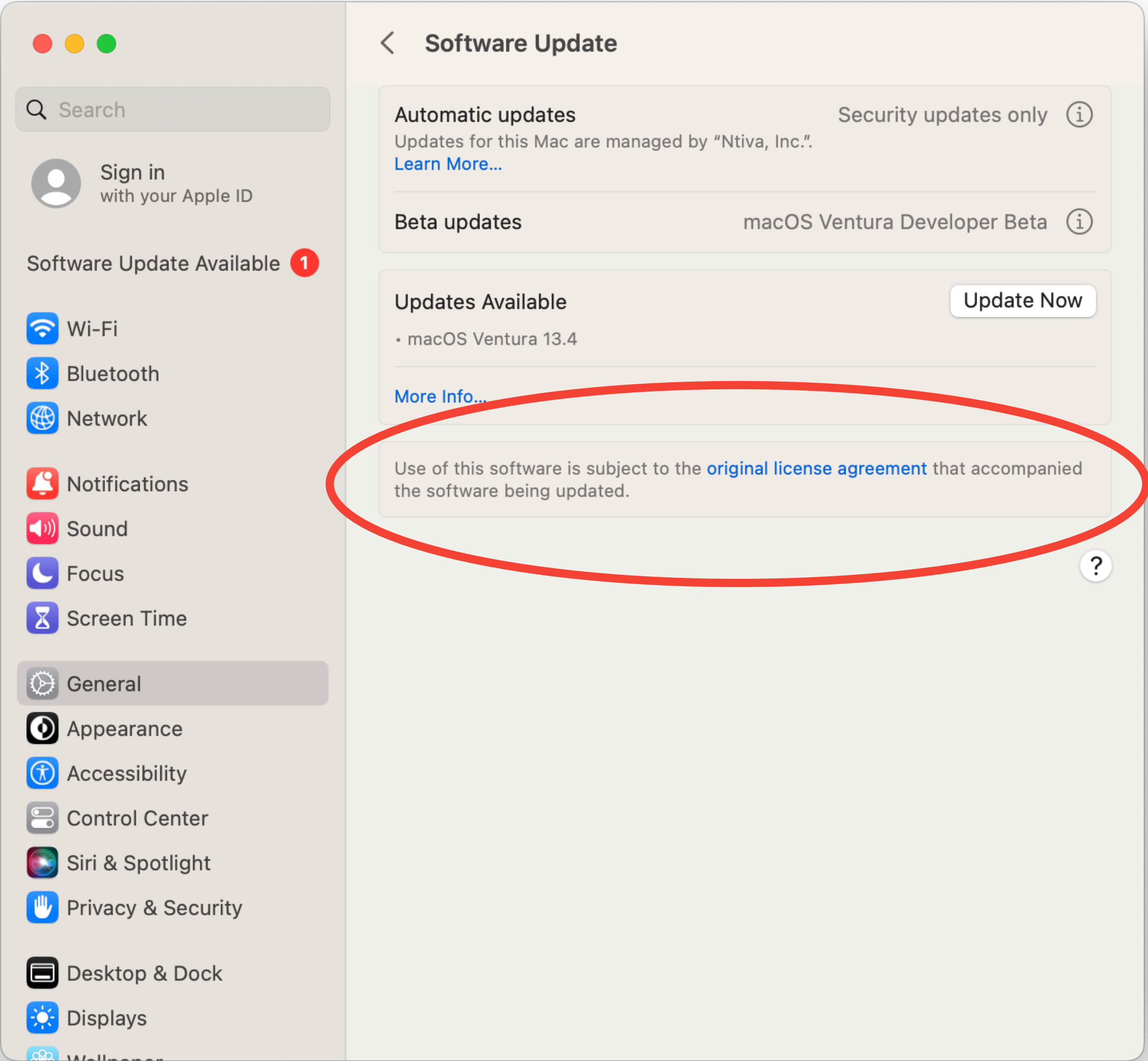


static text "Use of this software is subject to the original license agreement that accompanied the software being updated." **of** *group* 3 **of** *scroll area* 1 **of** *group* 1 **of** *group* 1 **of** *group* 2 **of** *splitter group* 1 **of** *group* 1 **of** *window* "Software Update" **of** *application process* "System Settings" **of** *application* "System Events"

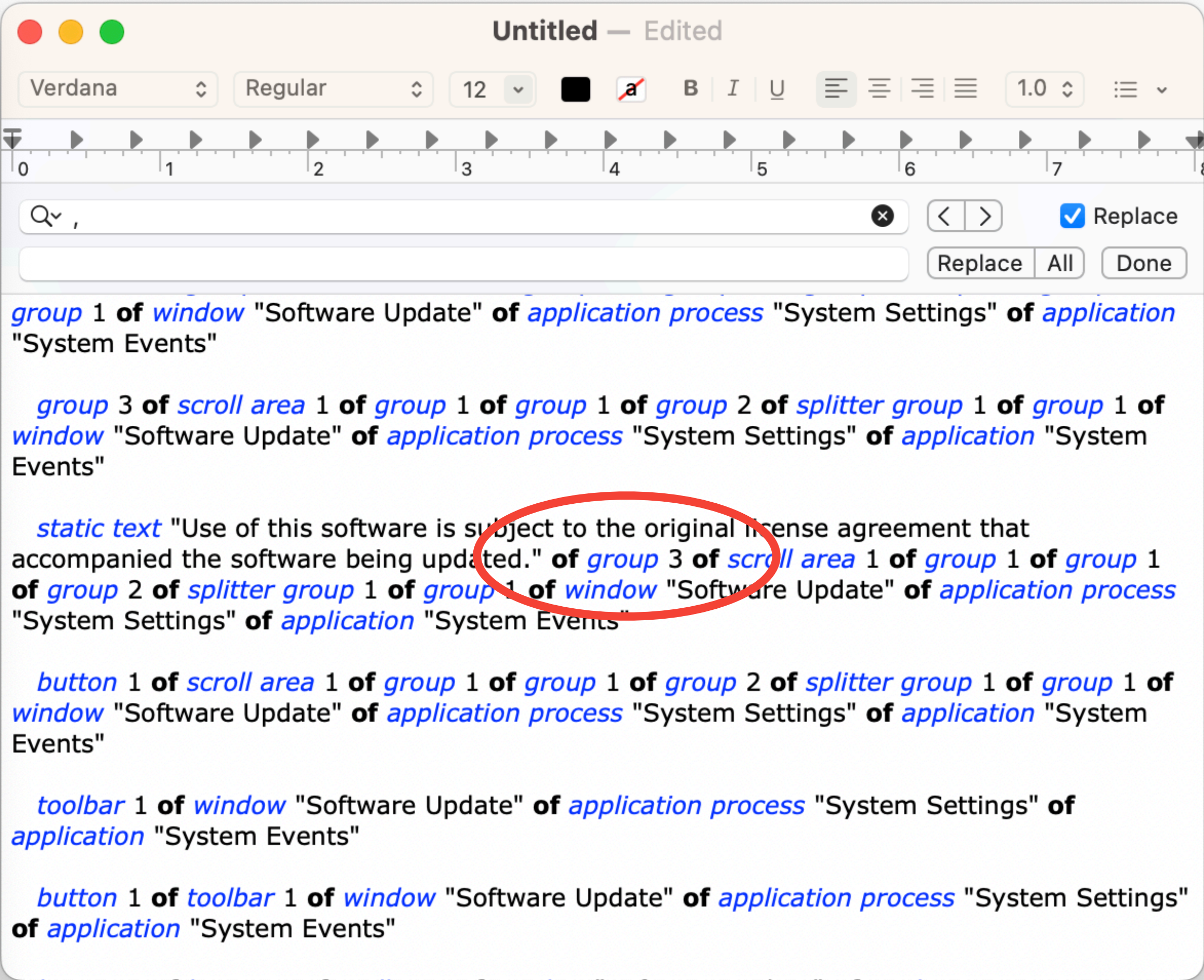
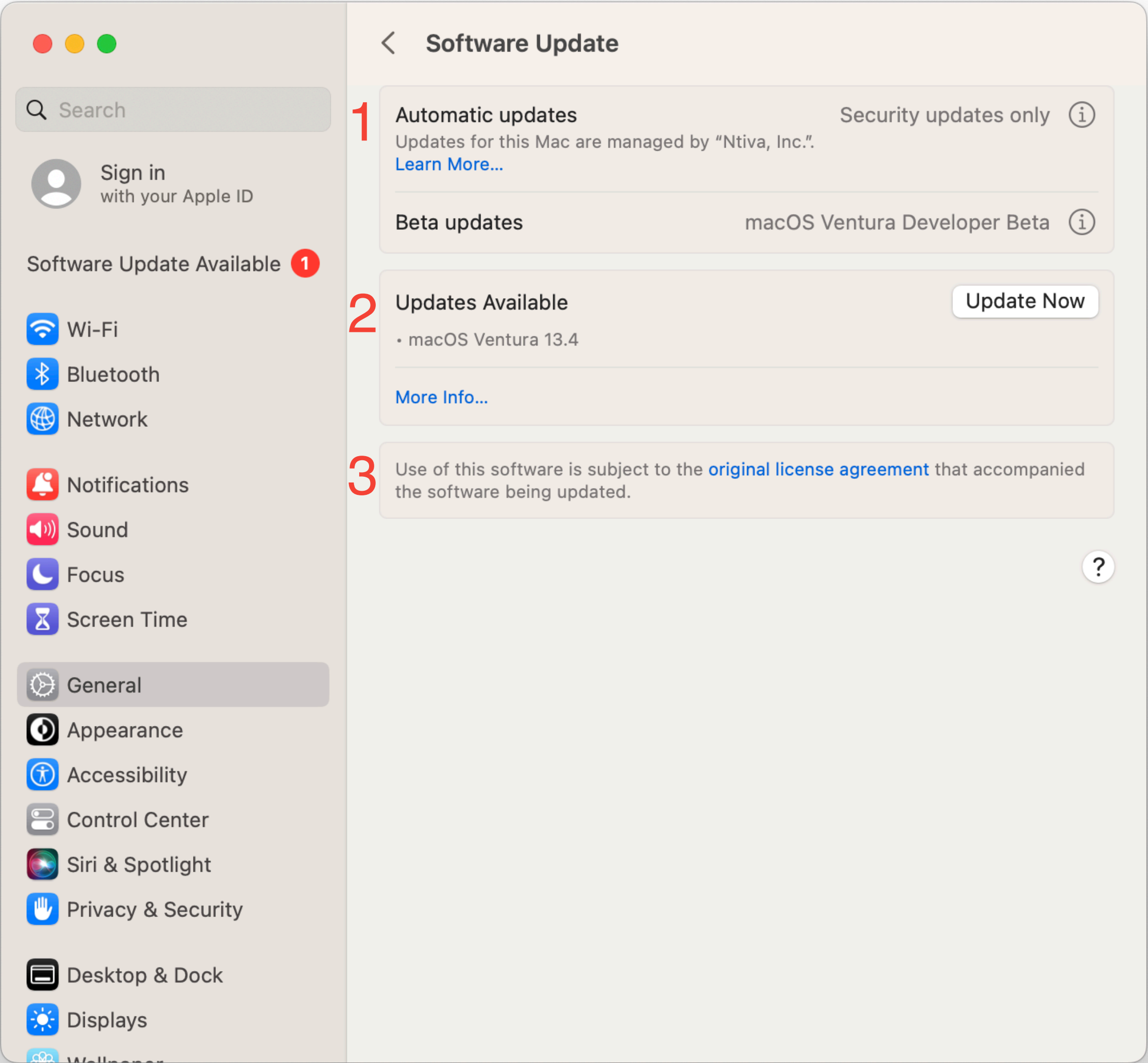
2 - Define element



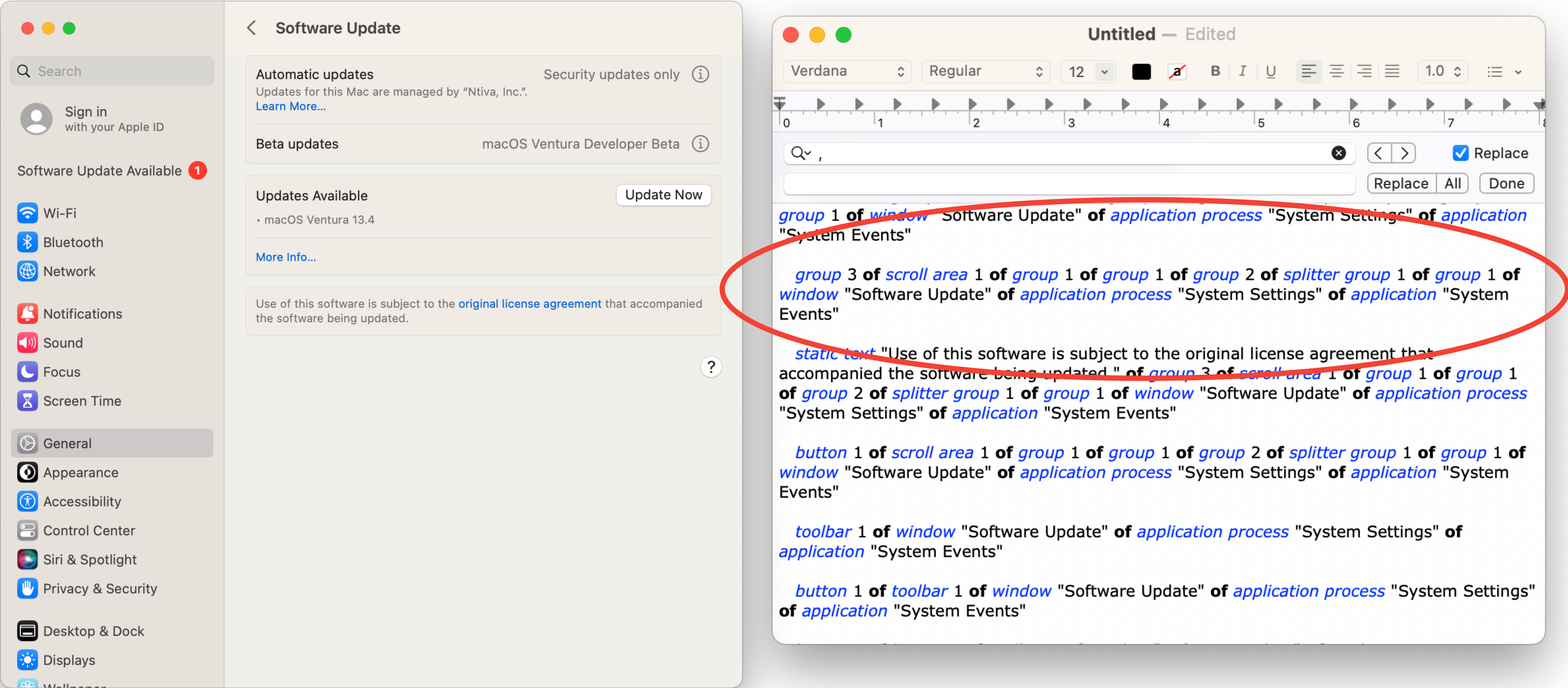
2 - Define element



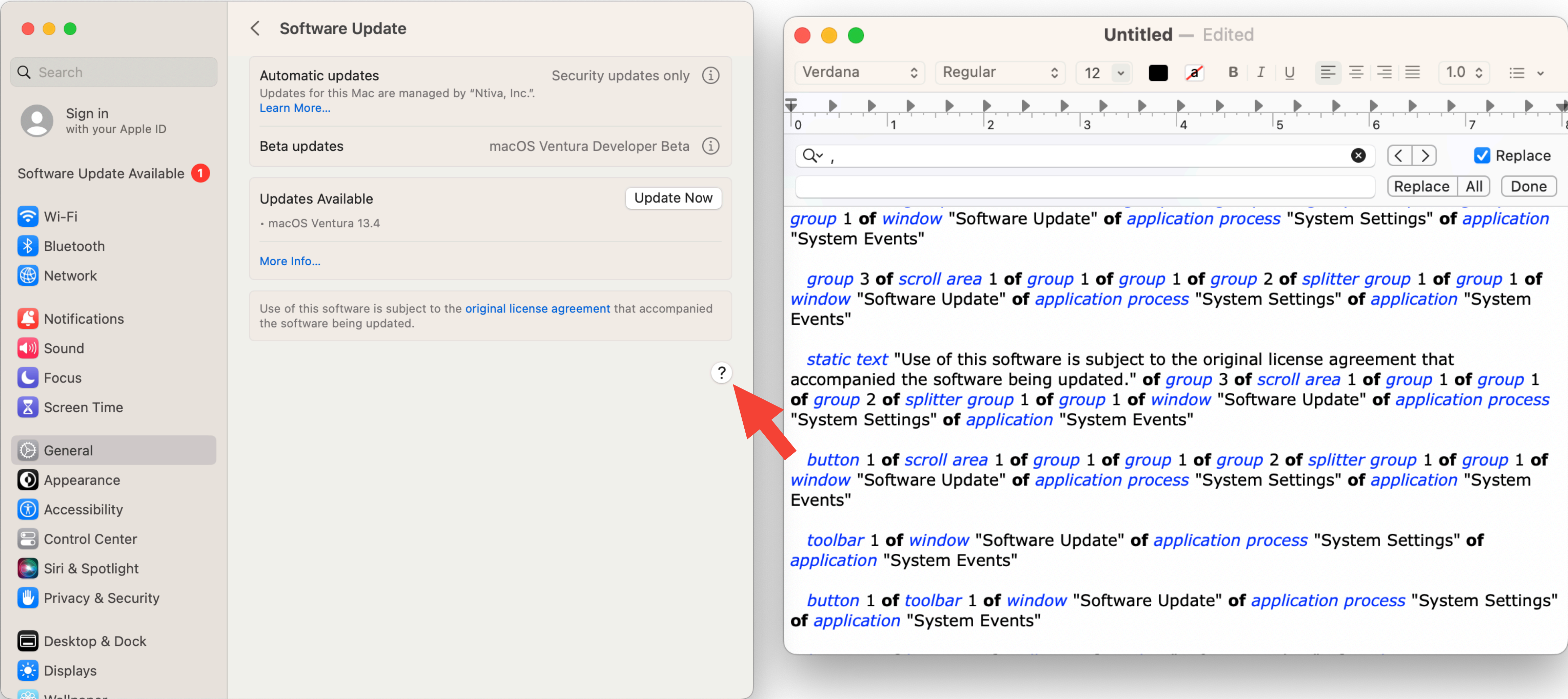
2 - Define element



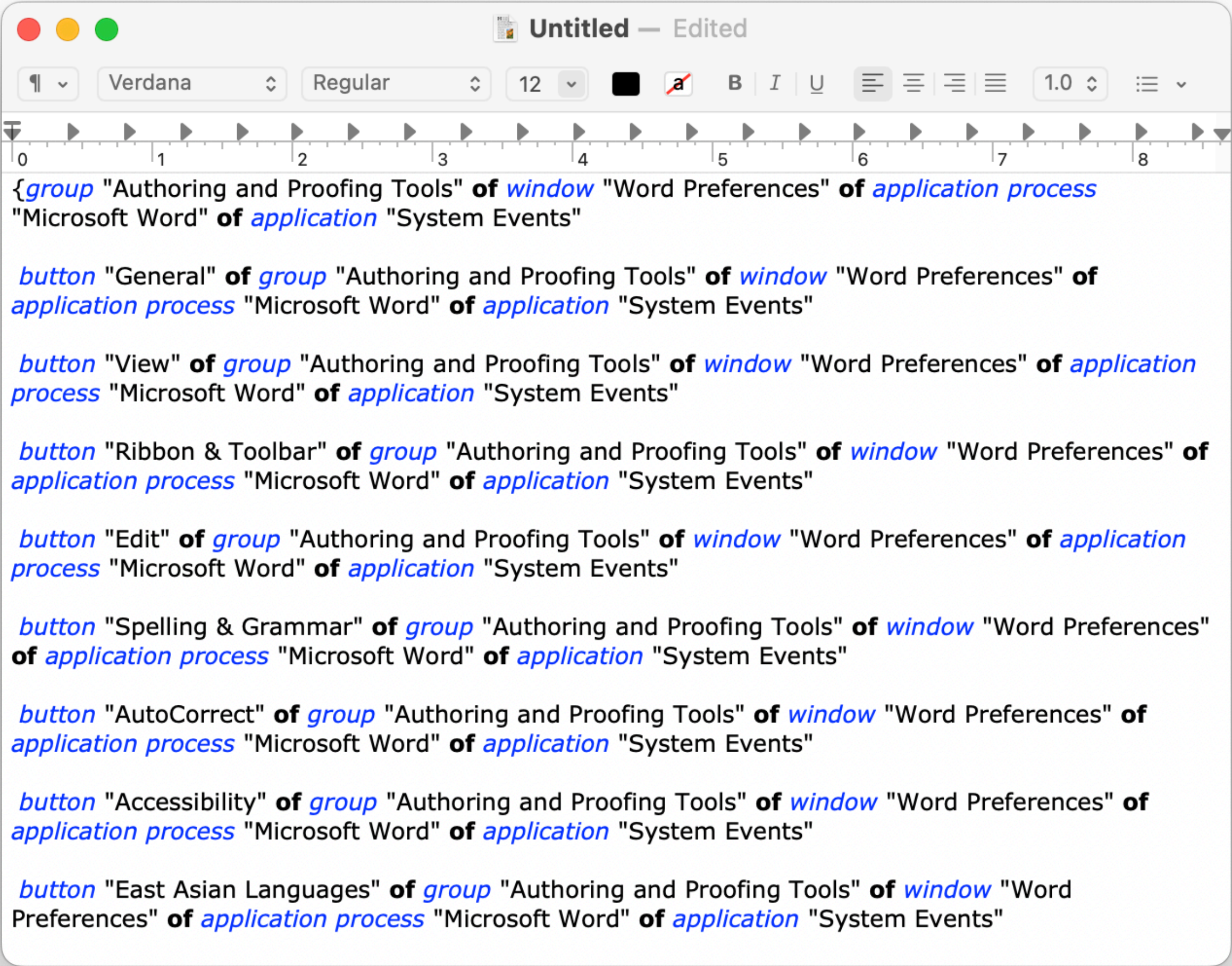
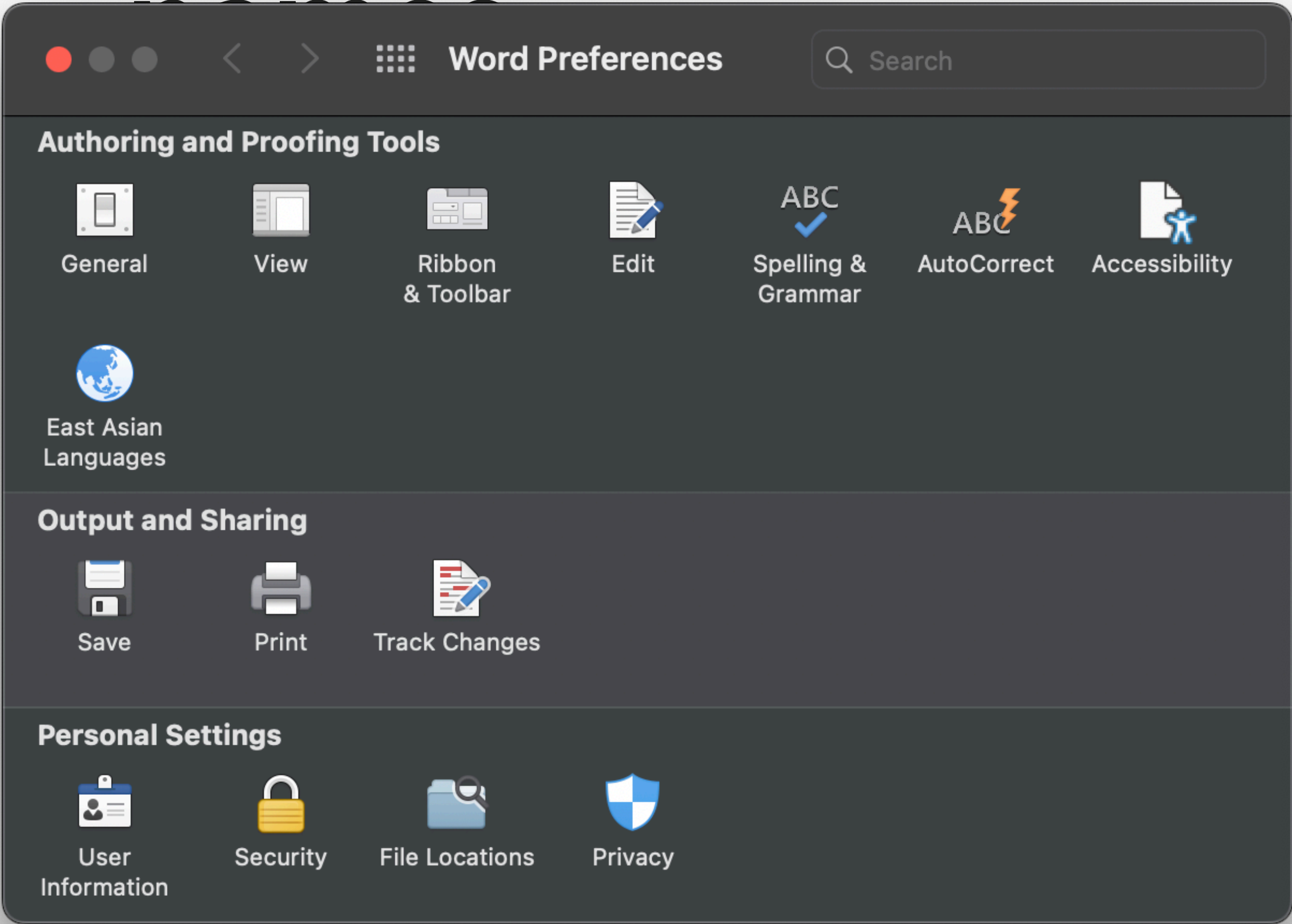
2 - Define element



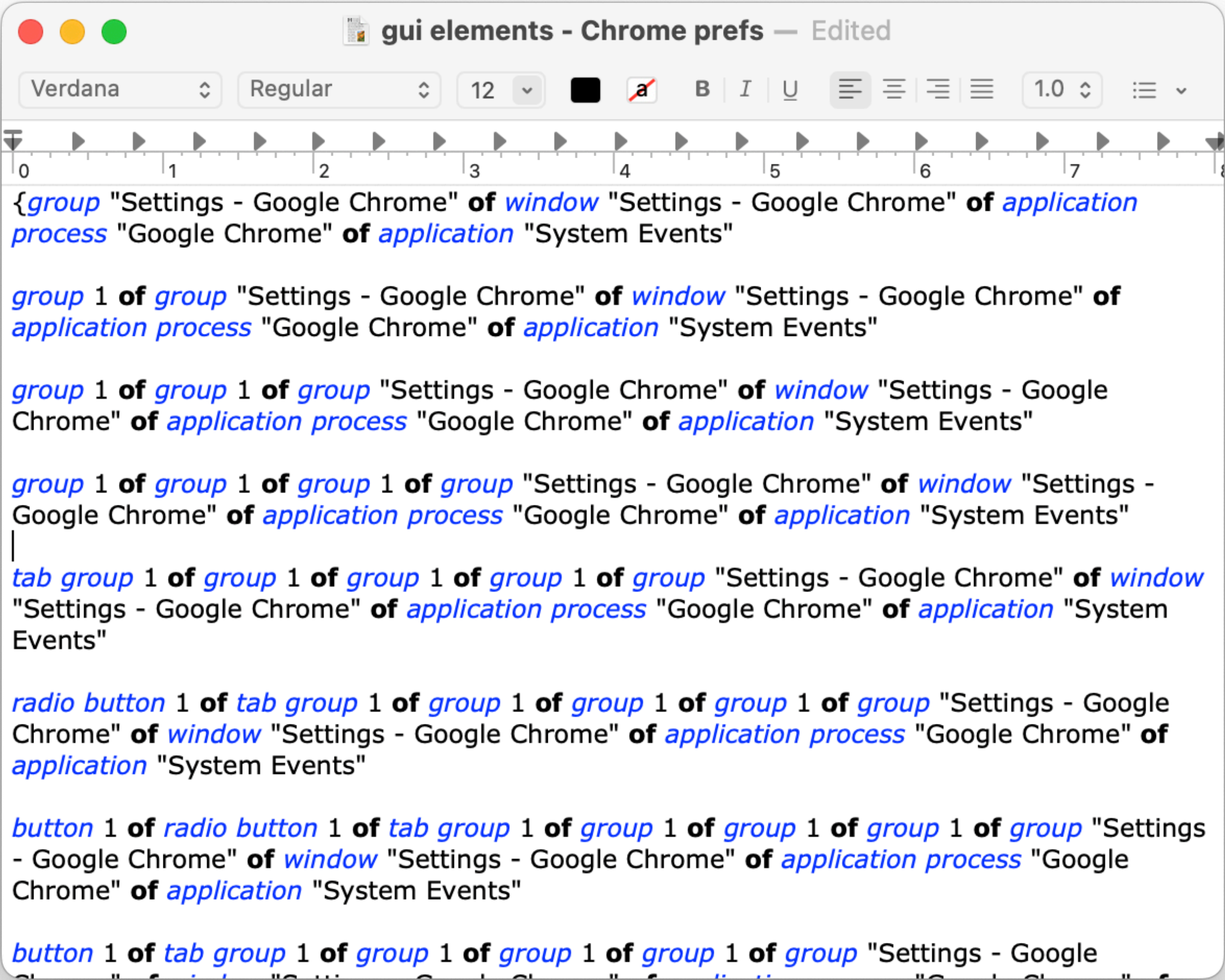
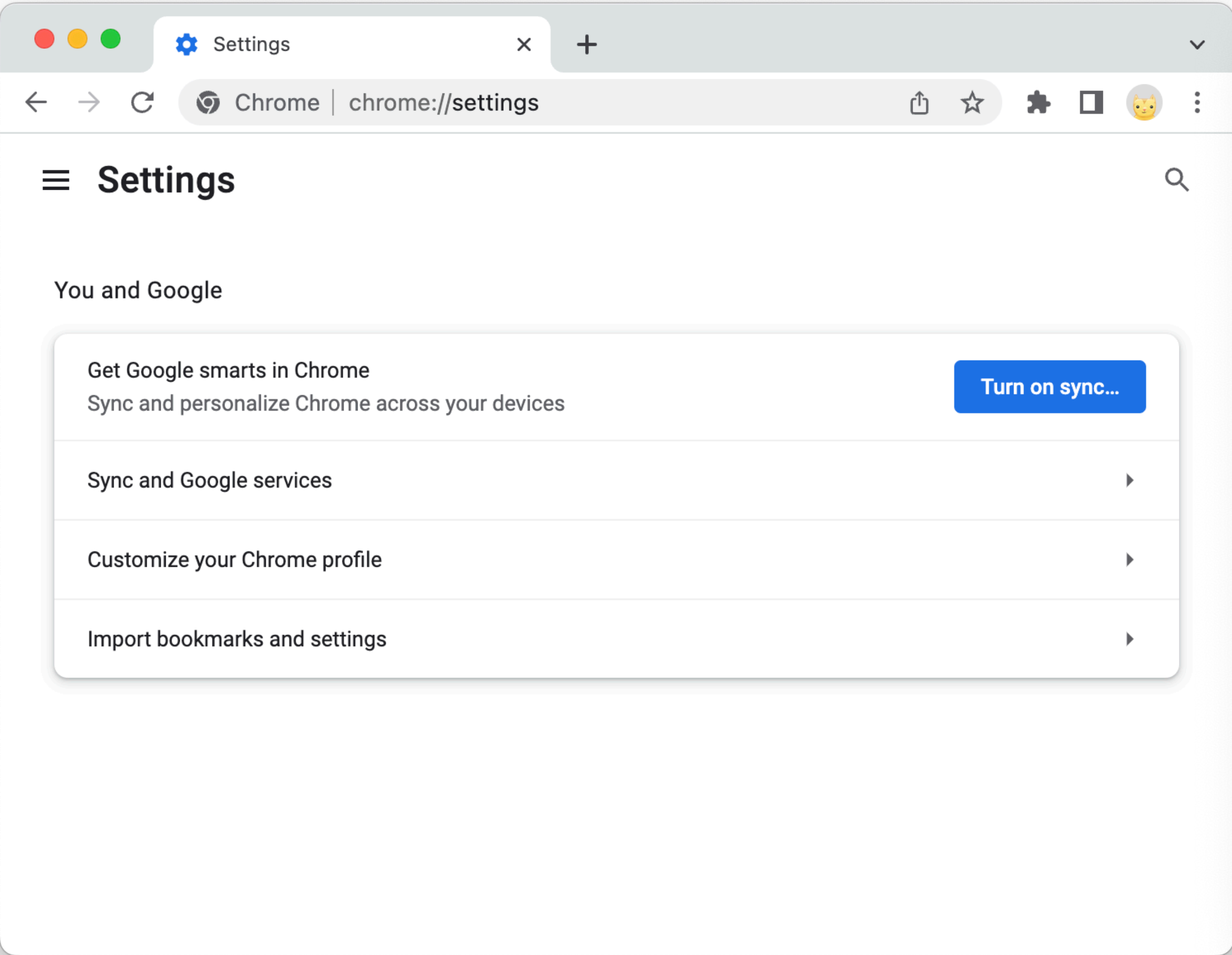
2 - Define element



2 - Define element



2 - Define element



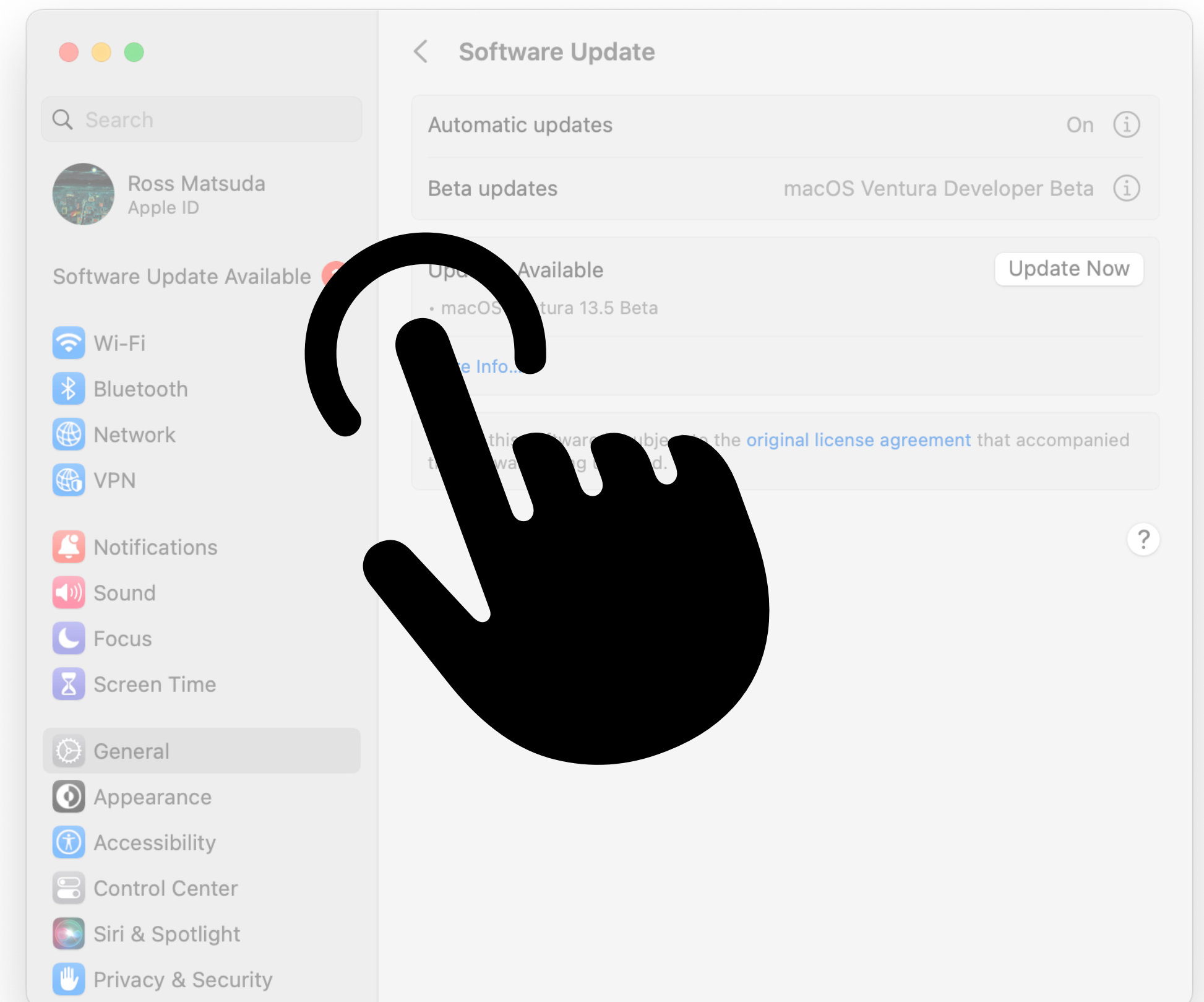
3 - Act on your targets

Click element

tell *application* "System Events"

tell *process* "System Settings"

click *button* 1 **of** *group* 2 **of** *scroll*
area 1 **of** *group* 1 **of** *group* 1 **of** *group*
2 **of** *splitter group* 1 **of** *group* 1 **of**
window "Software Update" **of**
application process "System Settings"
of *application* "System Events"



3 - Act on your targets

Click element

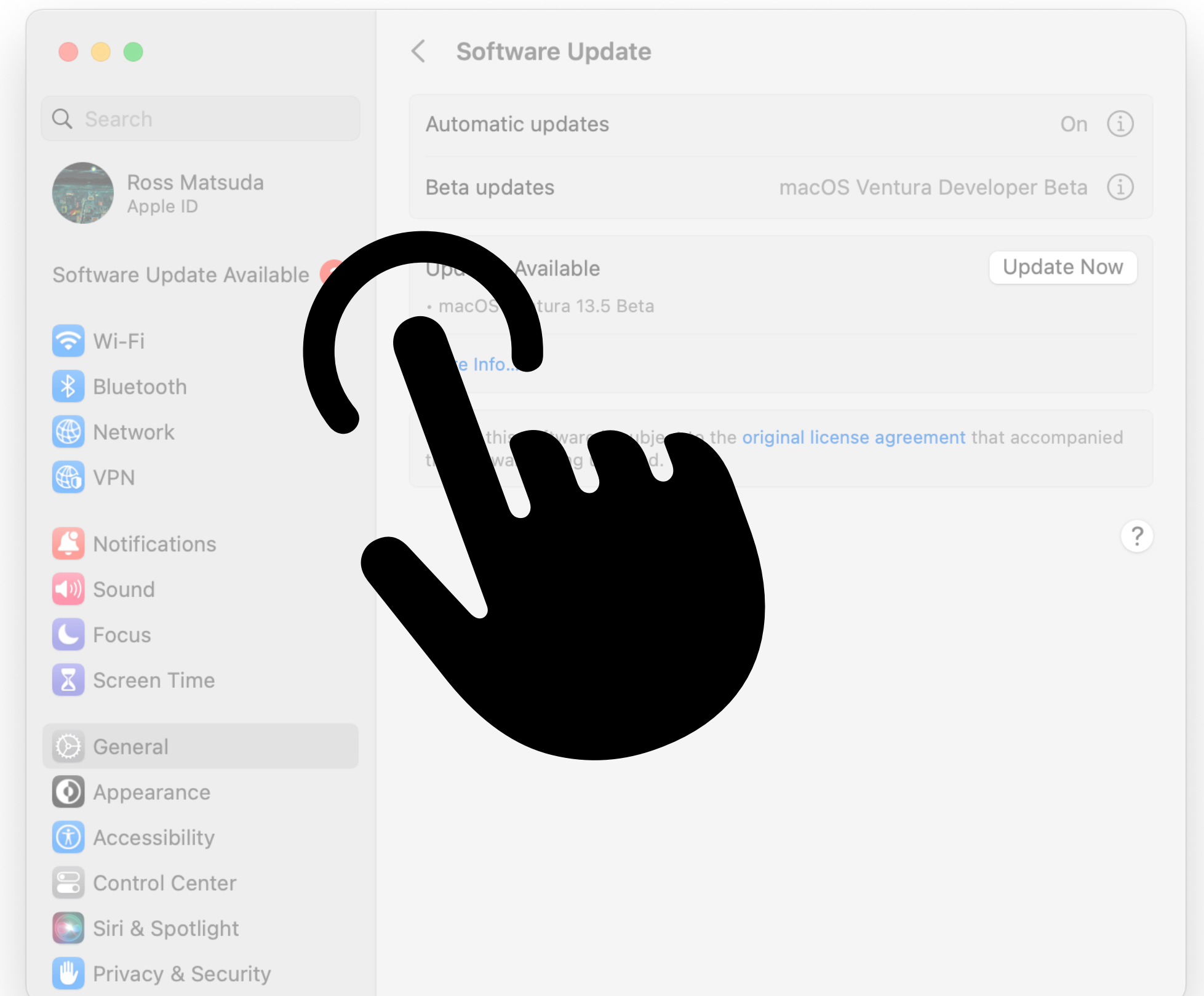
tell *application* "System Events"

tell *process* "System Settings"

click *[the thing]*

end tell

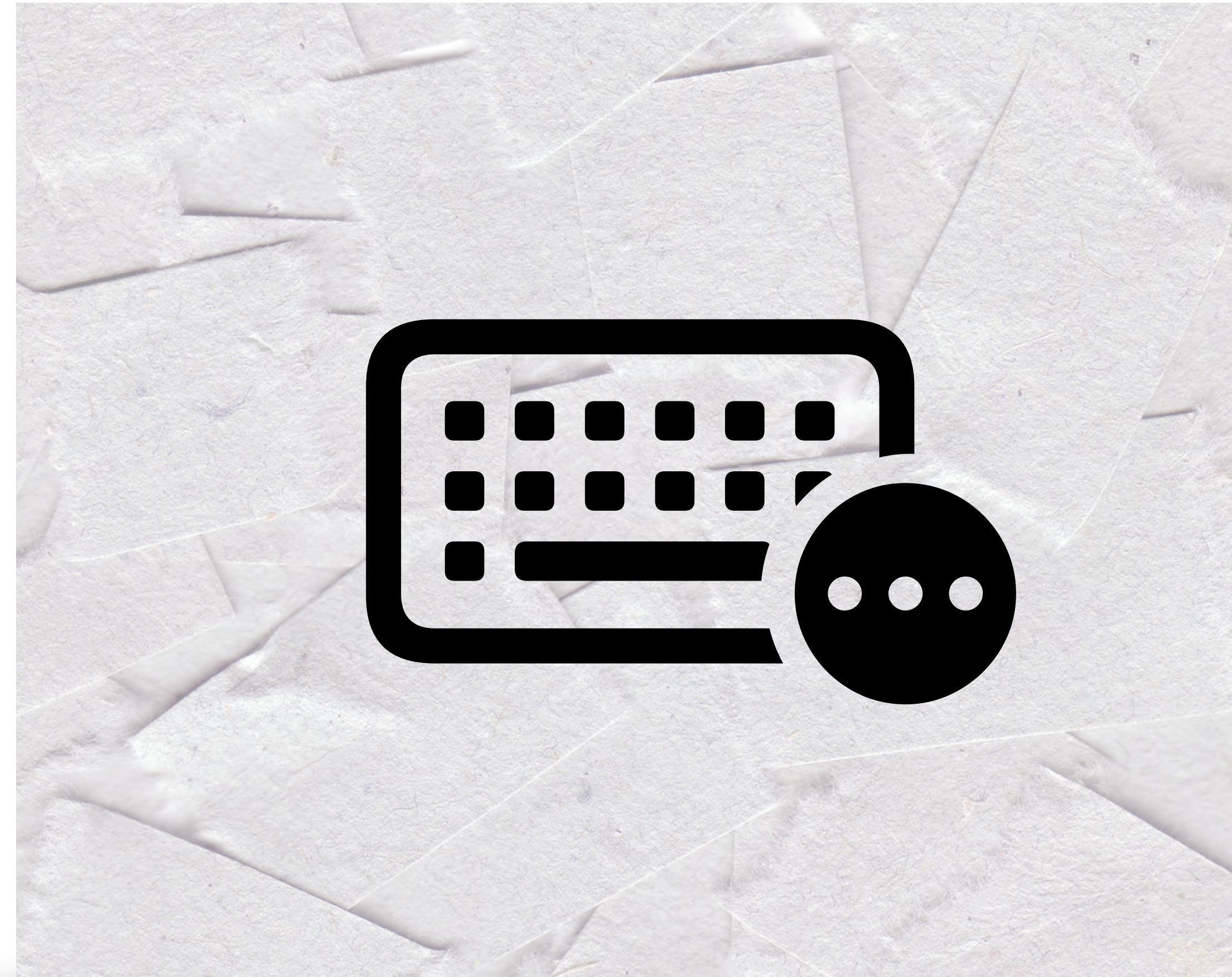
end tell



3 - Act on your targets

Sending keystrokes

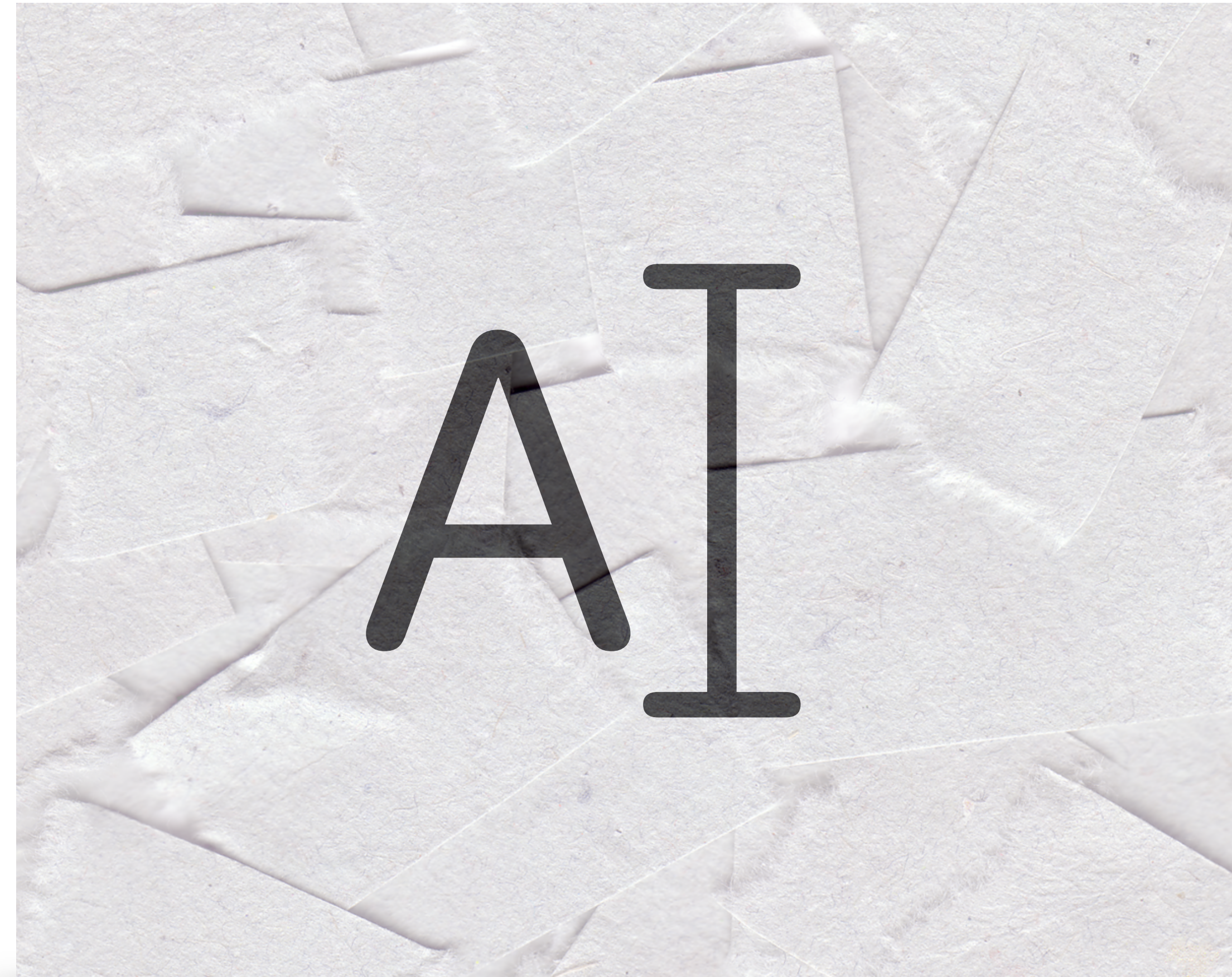
```
tell application "Microsoft Word"  
  activate  
end tell  
tell application "System Events"  
  key down {command}  
  keystroke ","  
  key up {command}  
end tell
```



3 - Act on your targets

Selecting and filling text fields

```
tell application "Microsoft Word"  
  activate  
end tell  
tell application "System Events"  
  set focused of text field 1 of group 2 of  
    toolbar 1 of window "Word Preferences"  
    of application process "Microsoft Word"  
    of application "System Events" to true  
  keystroke "spelling"  
  key code 125  
  key down {return}  
  key up {return}  
end tell
```



3 - Act on your targets

Selecting and filling text fields

```
tell application "Microsoft Word"  
  activate  
end tell  
tell application "System Events"  
  set focused of [the thing] to true  
  keystroke "spelling"  
  key code 125  
  key down {return}  
  key up {return}  
end tell
```

ARROW KEYS

LEFT	key code 123
RIGHT	key code 124
DOWN	key code 125
UP	key code 126

3 - Act on your targets

Selecting and filling text fields

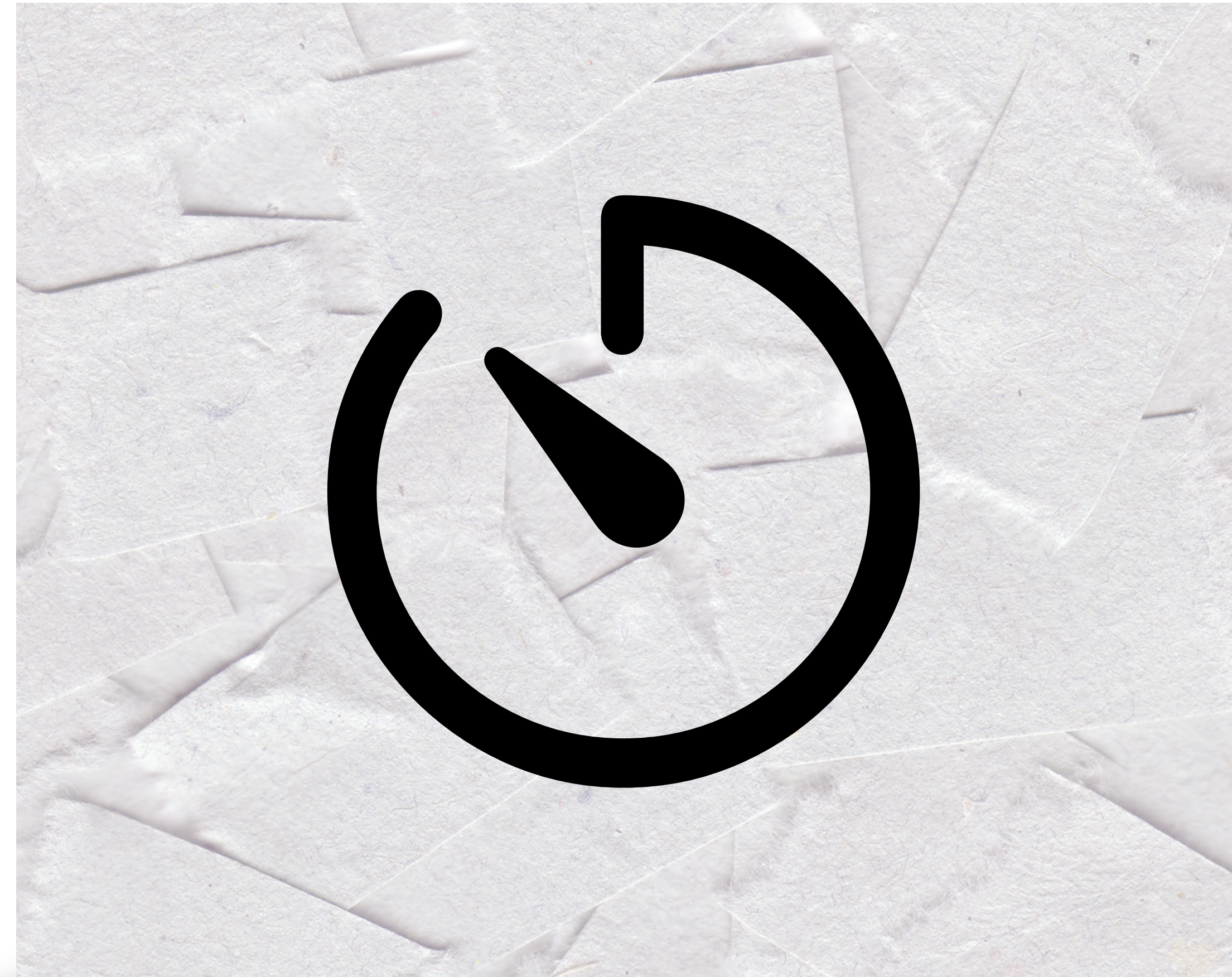
```
tell application "Microsoft Word"  
    activate  
end tell  
tell application "System Events"  
    key code 123 using {shift down,  
                           option down, command  
                           down}  
end tell
```

ARROW KEYS

LEFT	key code 123
RIGHT	key code 124
DOWN	key code 125
UP	key code 126

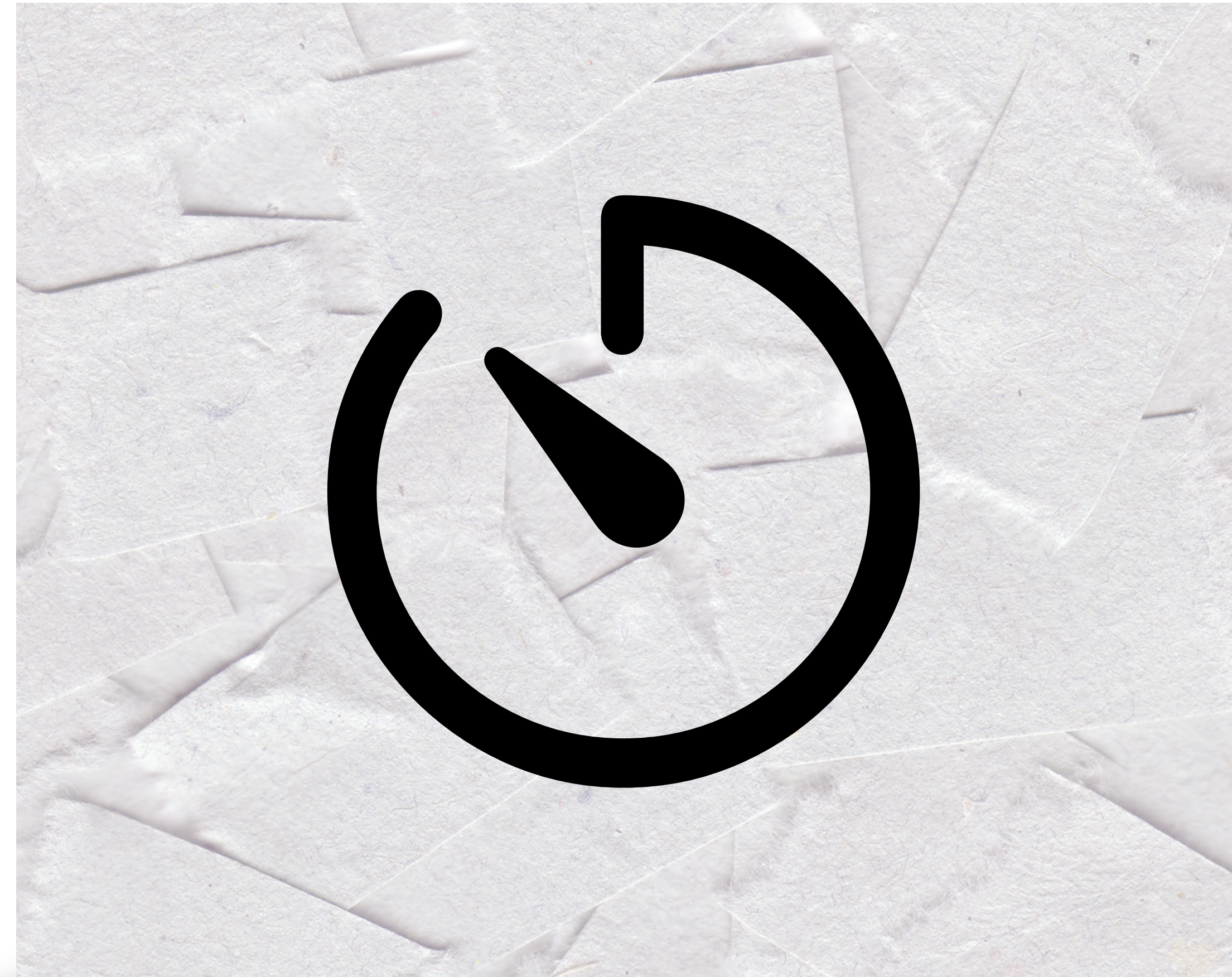
4 - Timing

```
if exists (button 1 of group 2 of scroll area 1 of group 1 of  
           group 1 of group 2 of splitter group 1  
           of group 1 of window "Software  
           Update" of application process  
           "System Settings" of application  
           "System Events") then  
    click button 1 of group 2 of scroll area 1 of group 1 of  
           group 1 of group 2 of splitter  
           group 1 of group 1 of window  
           "Software Update" of application  
           process "System Settings" of  
           application "System Events"  
  
end if
```



4 - Timing

if exists (*[the thing]*) **then**
click *[the thing]*
end if



4 - Timing

```
tell application "System Events"  
  repeat 60 times  
    if exists (window 1 of process "System Settings") then  
      delay 3  
      exit repeat  
    else  
      delay 1  
    end if  
  end repeat  
  
  if not (exists (window 1 of process "System Settings")) then  
    error number -128  
  end if  
end tell
```


4. Timing

```
tell application "System Events"  
  tell process "System Settings"  
    repeat 60 times  
      if exists (button 1 of group 2 of scroll area 1 of group 1 of group 1 of group 2 of splitter group 1 of group 1 of window "Software  
        Update" of application process "System Settings" of application "System Events") then  
          click button 1 of group 2 of scroll area 1 of group 1 of group 1 of group 2 of splitter group 1 of group 1 of window "Software  
            Update" of application process "System Settings" of application "System Events"  
          exit repeat  
        end if  
  
      tell application "System Events"  
        if application process "System Settings" exists then  
          delay 0.5  
        else  
          exit repeat  
        end if  
      end tell  
  
    delay 1  
  end repeat  
end tell  
end tell
```



tell *application* "System Events"
 tell *process* "System Settings"
 repeat 60 **times**
 if exists (*button* 1 ...) **then**
 click *button* 1 ...
 exit repeat
 end if

tell *application* "System Events"
 if *application process* "System Settings" **exists** **then**
 delay 0.5
 else
 exit repeat
 end if
 end tell

delay 1
 end repeat
 end tell
end tell



4 - Timing

Define your timeouts

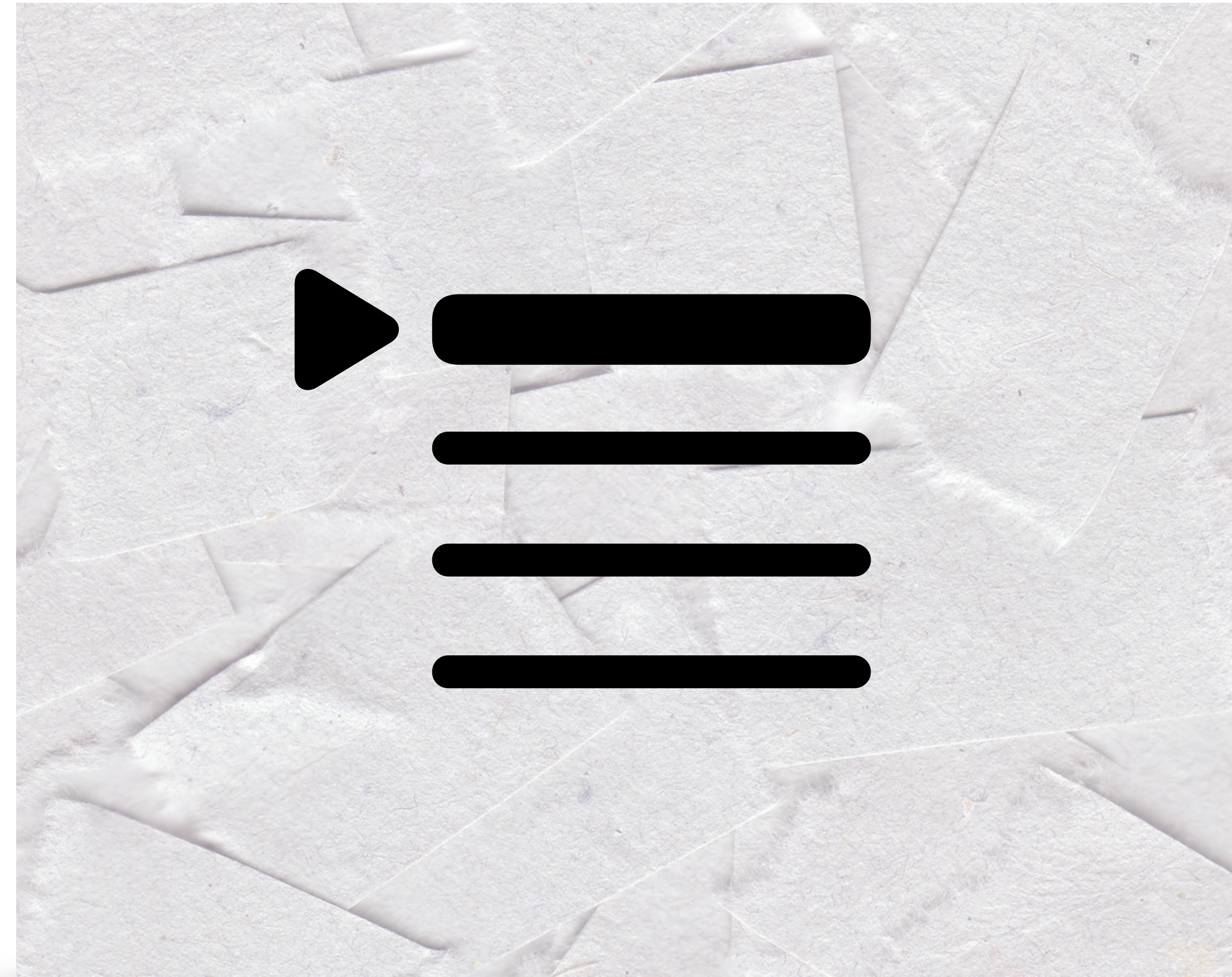
Clean up your script/app to
avoid duplicate processes



error number -128

5 - Versioning

macOS version commands
Switch statement or ifs




```
# Get Major OS (10, 11, 12, etc.)  
set _major to system attribute "sys1"  
  
# Bailout if old version  
if _major < 11 then  
    log "Catalina or earlier detected"  
    error number -128  
end if  
  
# Monterey and Big Sur  
if _major < 13 then  
    log "Monterey or Big Sur detected"  
end if  
  
# Ventura and Sonoma  
if (_major > 12) then  
    log "Ventura or Sonoma detected"  
end if
```

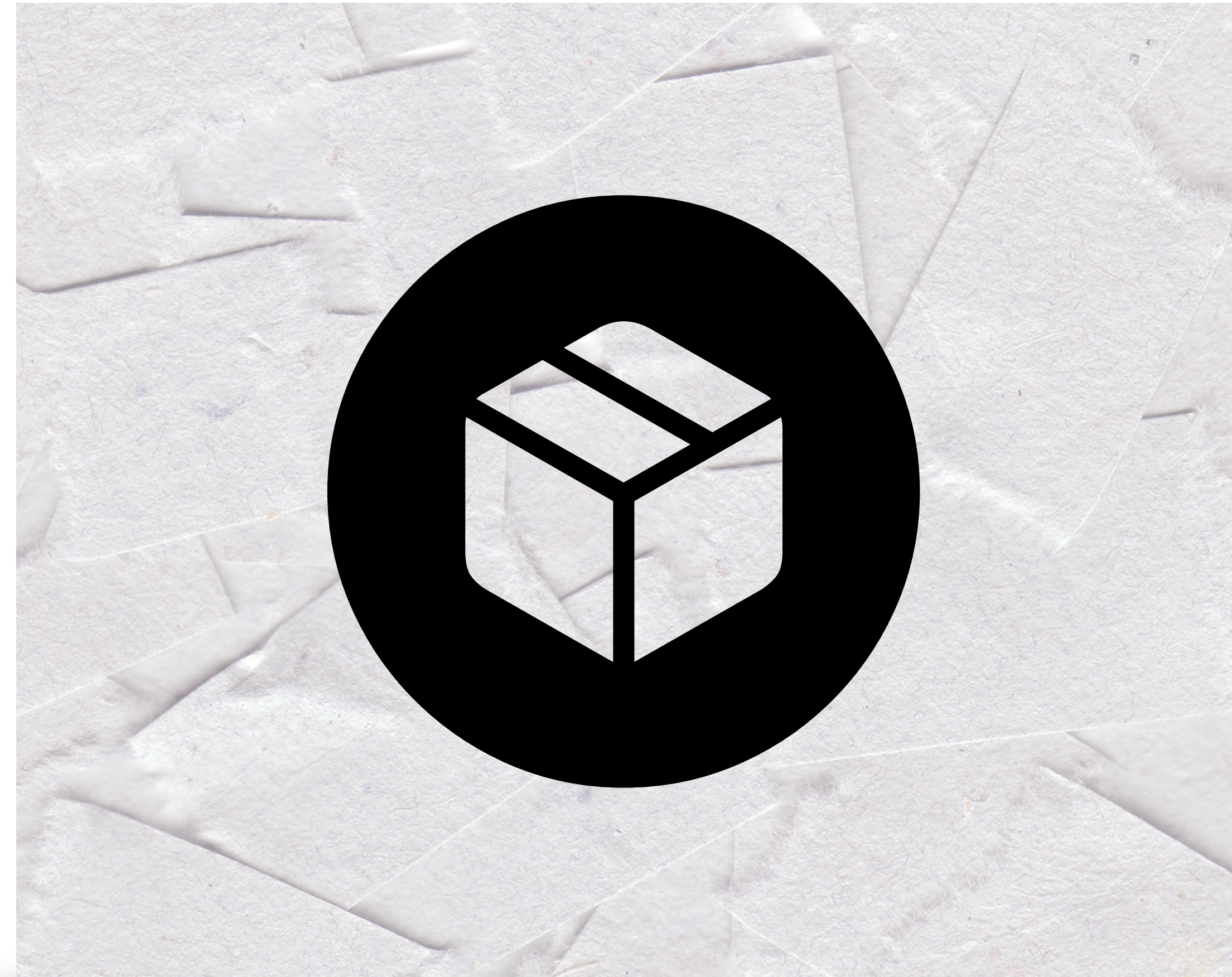


```
set ReadAppVersion to version of application "Keynote"  
log ReadAppVersion
```

(*13.0*)

6 - Packaging

- osascript
- Making an app



6 - Packaging

Hiding a Dock Icon and signing:

1. Export unsigned
2. Edit info.plist
3. Sign the app

HIDE DOCK ICON

Add this to your info.plist:

```
<key>LSUIElement</key>  
<true/>
```


Sign your AppleScript app

```
codesign --force --sign "Apple Development:  
John Doe (1234567890)" -v /path/to/bundle.app
```

Run this to clear errors:

```
xattr -cr /path/to/bundle.app
```


7 - Best practices

Make GUI Script as small a part
of your workflow as possible



7 - Best practices

- Named elements > numbered elements
- Clicks > keystrokes

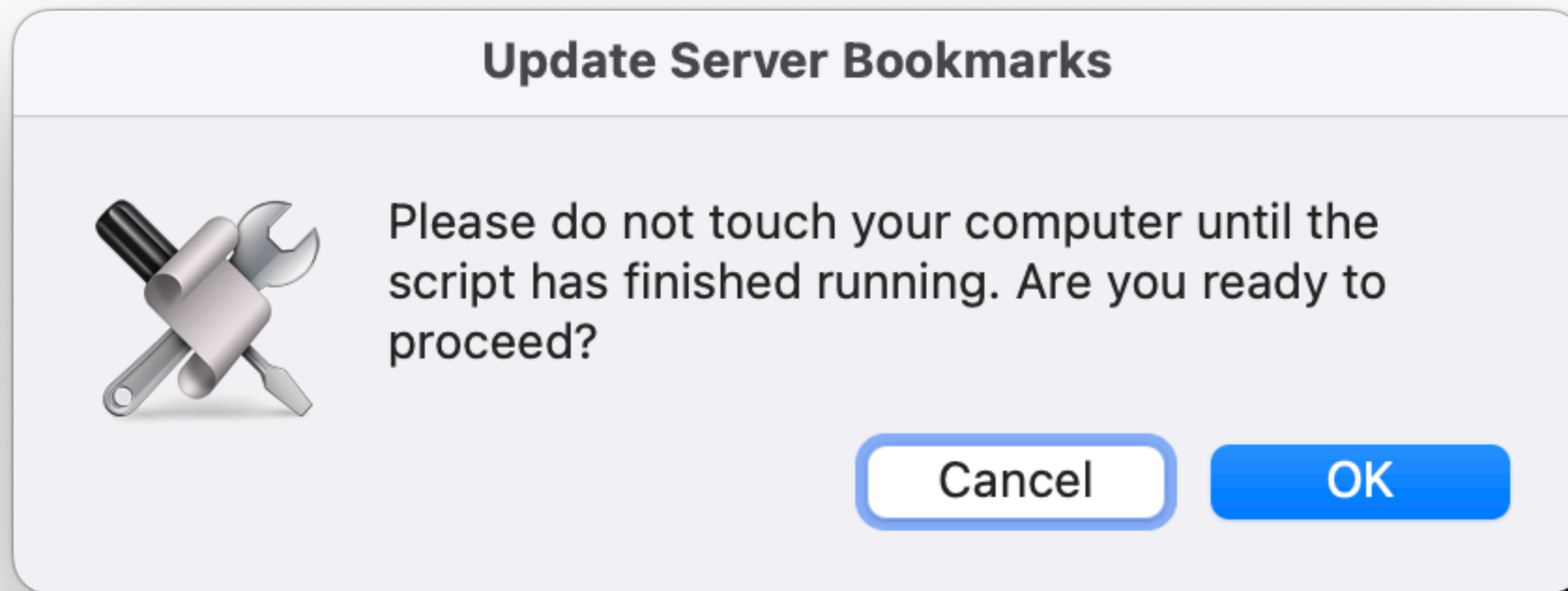


7 - Best practices

Consider user communication



7 - Best practices



7 - Best practices

Consistency checks:

- If process exists, activate
- Get window name
- If window name \neq desired name, act



7 - Best practices

Participate in betas for:

- Compatibility tests
- Obsolescence



7 - Best practices

- Document everything, especially version compatibility
- Limit use to builds you've tested



Resources

PPPC Utility: <https://github.com/jamf/PPPC-Utility>

SF Symbols: <https://developer.apple.com/sf-symbols/>

HIG: <https://developer.apple.com/design/human-interface-guidelines>

Get your Developer cert: developer.apple.com

Sudoade | GUI Scripting with AppleScript:

<https://www.sudoade.com/gui-scripting-with-applescript/>

Q&A

Feedback
<https://bit.ly/psumac2023-119>