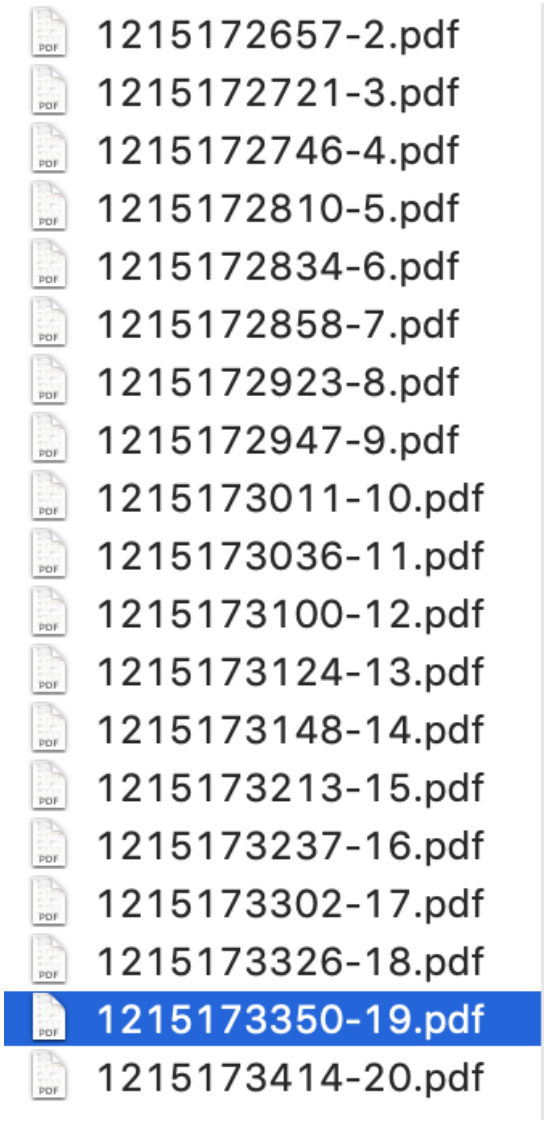
Scripts to collect browser content or Panachrome repeatedly over time

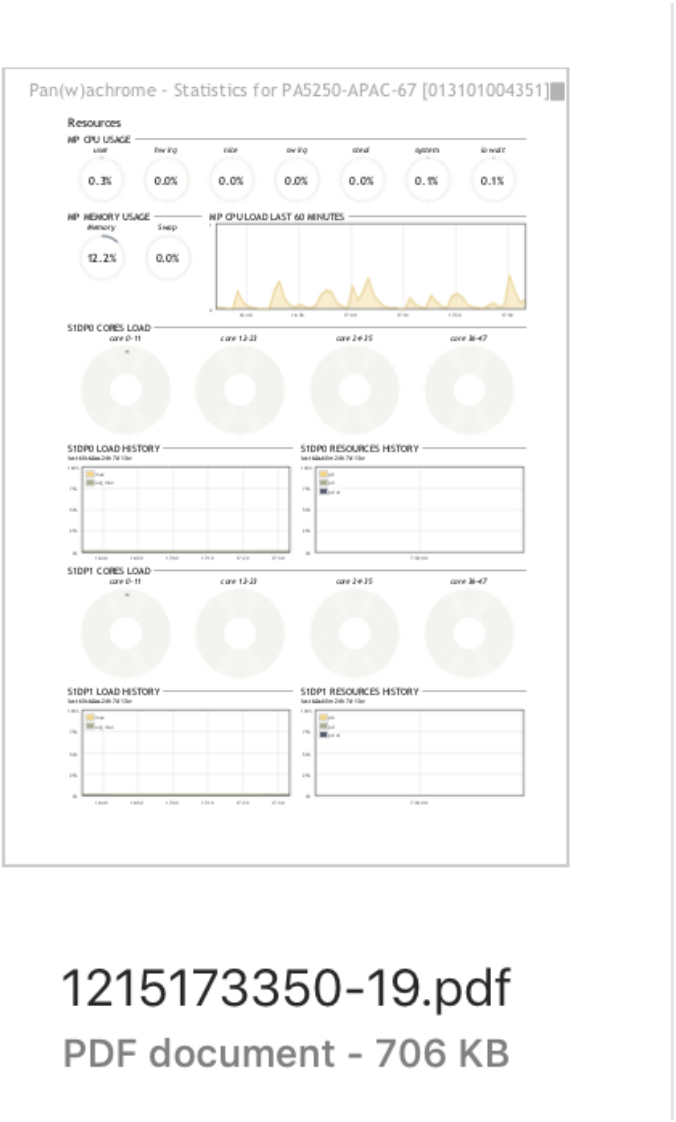
Task requirement

There are situations where you need to save browser content regularly over a time period.

For example, if you run a test for hours or days, you may be required to capture dataplane CPU or session table utilization on Panachrome. Some may take screen shots or save the page as a PDF file.

We automate this task Save as PDF with a script (video attached).





The scripts

There are 2 versions of the scripts (attached). You may either run it on Windows or MacOS.

Script on Windows (*.ahk)

AutoHotKey is required for this script. Install this software so that the script becomes executable.

Scripts to collect browser content or Panachrome repeatedly over time

Since the application window must be focused (clicked as an active window), user interaction should be avoided in the script environment.

It is recommended that the script run in a virtual machine so its execution will not be interfered.

When it finishes, a dialog box will pop up.

```
;
; Script to automate Panachrome Save as PDF [2019121401]
;
; Documentation of AutoHotkey
;
; https://www.autohotkey.com/docs/AutoHotkey.htm
;

timeDelayInSeconds := 300
timeIntervalInSeconds := 1800
nCaptures := 10

EnvGet, homePath, HOMEPATH
targetFolder := homePath . "\Desktop\Panachrome\"
; targetFolder := homePath . "\Desktop\tmp\"

smallDelay := 3000

;
; perform Save as PDF repeatedly
;

Sleep, timeDelayInSeconds * 1000

i := 1
n := nCaptures

Loop, %n% {
    targetFile := targetFolder . A_MM . A_DD . A_Hour . A_Min . A_Sec . "-" . i . ".pdf"

    ; MsgBox, %targetFile%

    Send ^p
    Sleep, %smallDelay%
    Send {Enter}
    Sleep, %smallDelay%
```

```
    Send %targetFile%
    Send {Enter}

    if (i < n) {
        Sleep, timeIntervalInSeconds * 1000
    }

    i += 1
}

MsgBox
```

Script on MacOS (*.scpt)

This is an AppleScript, so no additional software is required. Just run it with Script Editor (in Utilities).

The script will determine the browser tab of the application (this case Pan(w)achrome on Google Chrome).

When it comes the time to perform Save as PDF, the browser window will get focused by itself so interruption can be minimized.

That means it is less blocking compared with that on Windows. Just let it finish each scheduled capture task then you may continue your work with the Mac machine.

When it finishes, a dialog box will pop up.

There are limitations with this script. The script may not work or time out when the screen is locked.

For testing purpose, I put a control parameter to avoid keystroke on a locked screen. **This is important because it may cause account lockout.**

```
set noKeystroke to true # keystroke on a login screen may cause account lockout
```

With that said, no matter whether your screen will be locked automatically, there will still be issues with security. It is recommended that this script run with a lab computer or you keep it safe physically.

```
#
# Script to automate Panachrome Save as PDF [2019121401]
#
```

```
set timeDelayInSeconds to 300
set timeIntervalInSeconds to 1800
set nCaptures to 20

set appBrowser to "Google Chrome"
set appTitle to "Pan(w)achrome - Statistics"

# set targetFolder to "~/Desktop/Panachrome/"
set targetFolder to "~/Desktop/tmp/"
set filenameScript to "date '+%m%d%H%M%S'"

set smallDelay to 2

set noKeystroke to true # keystroke on a login screen may cause account lockout

# display dialog targetFolder

#
# locate the window of the application
#

tell application "Google Chrome"
    set found to false
    set windowList to every window
    repeat with theWindow in windowList
        set tabList to every tab in theWindow
        repeat with theTab in tabList
            set theURL to the URL of theTab
            set theTitle to the title of theTab
            if theTitle is equal to appTitle then
                set found to true
                set appWindow to theWindow
                exit repeat
                # display dialog theURL & " ||| " & theTitle
            end if
        end repeat
        if found then
            exit repeat
        end if
    end repeat
end tell
```

```
#
# the app is not loaded
#

if not found then
    error appTitle & ": not loaded"
end if

delay timeDelayInSeconds

#
# perform "Save as PDF" repeatedly
#

# do shell script "caffeinate -di"

set i to 1
set n to nCaptures

repeat n times
    set targetFile to (do shell script filenameScript) & "-" & i & ".pdf"

    activate application appBrowser

    tell appWindow
        set visible to false
        set visible to true
        set index to 1

        tell application "System Events" to tell process appBrowser
            if not noKeystroke then
                keystroke "p" using command down
                delay smallDelay
                #
                # assuming "Save as PDF" was selected and remembered
                #
                keystroke return
                delay smallDelay
                keystroke targetFolder
                keystroke return
                delay smallDelay
                keystroke targetFile
                keystroke return
```

```
            end if
        end tell
    end tell

    log targetFile

    if i < n then
        delay timeIntervalInSeconds
    end if

    set i to (i + 1)
end repeat

display dialog "DONE"

#
# End
#
```

Before you start

First of all, customize the scripts to suit your needs. Change these parameters at the beginning.

timeDelayInSeconds - initial delay (in seconds) before it starts the first capture
timeIntervalInSeconds - time (in seconds) between captures
nCaptures - total number of captures
targetFolder - destination folder (by default a folder on Desktop called Panachrome)
noKeystoke - MacOS only, set it true first for testing as keystroke on a login screen may cause account lockout, as mentioned above.

For example, if test duration is 120 mins or 2 hours, you may want to capture 60m dataplane CPU (on Panachrome) for at least 3 times so that all moments are covered.

Set the parameters as follows.

timeDelayInSeconds := 1800 (wait for 30 mins before the first capture)
timeIntervalInSeconds := 3600 (60 minutes)
nCaptures := 3

Scripts to collect browser content or Panachrome repeatedly over time

The script will eventually run for totally 150 minutes or 2.5 hours (30 + 60 x (3-1))

Saved files all have this naming convention.

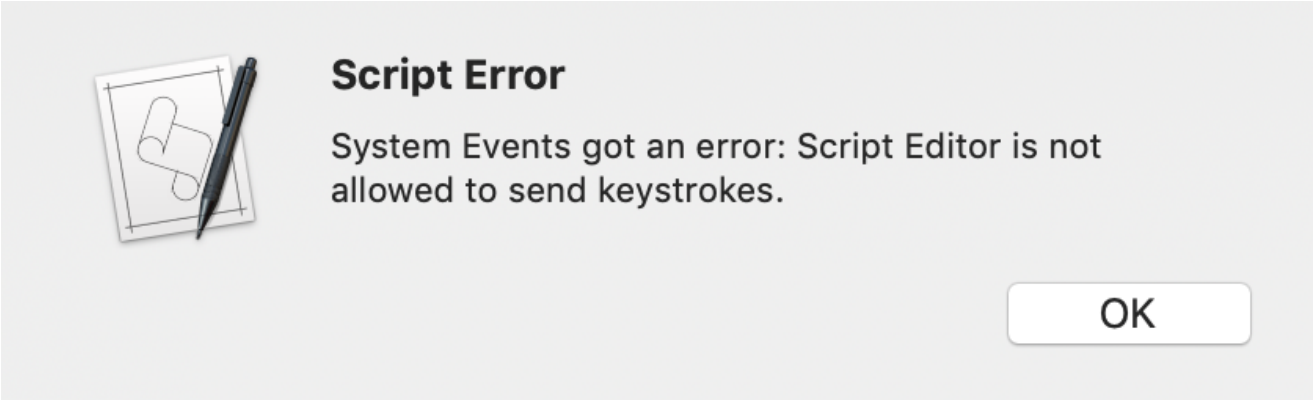
MMDDhhmmss-i.pdf

MM - current month, DD - current day, hh - current hour, mm - current minute, ss - current second, i - the i th capture.

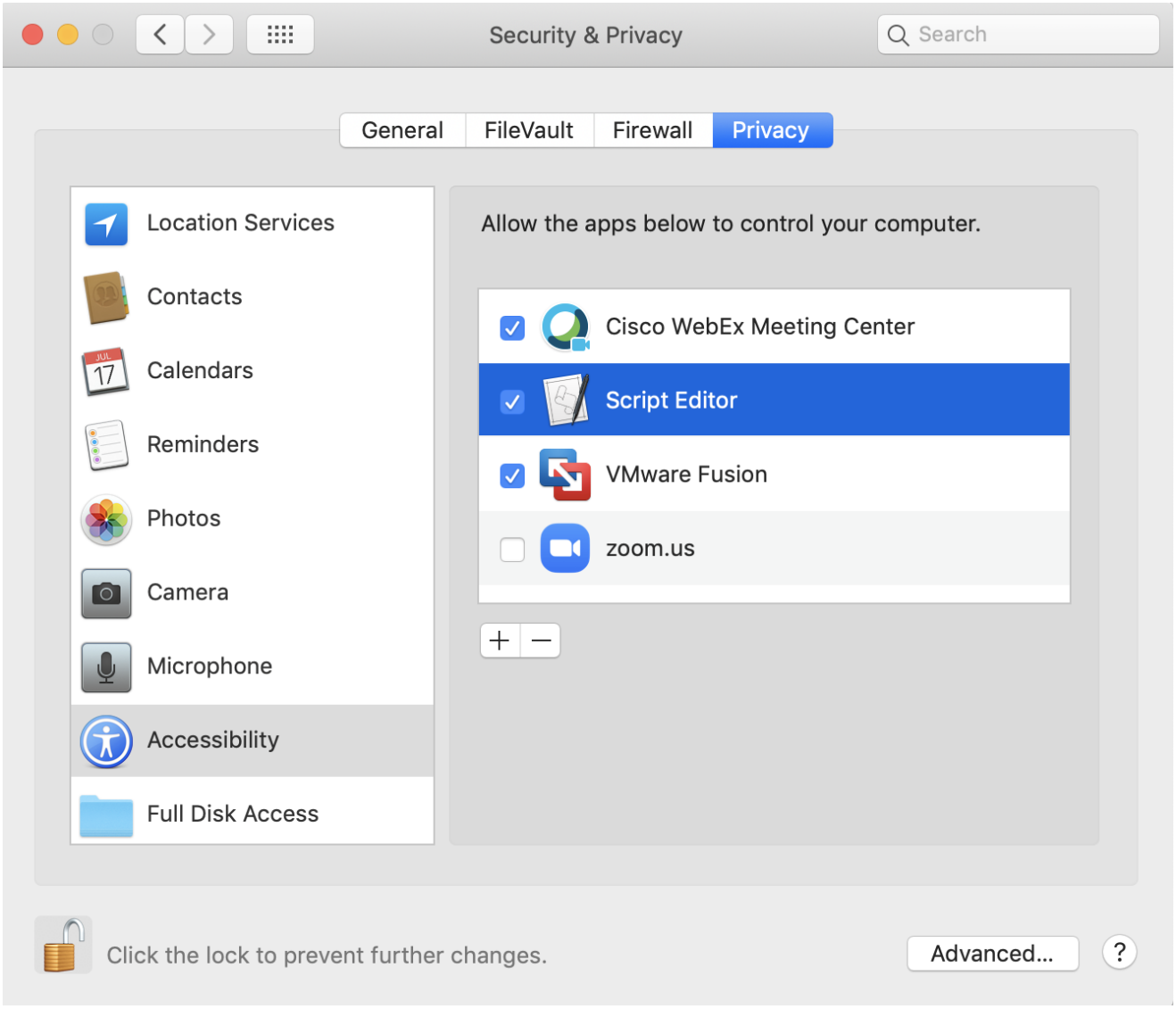
This is a lazy design as file name conflict will not happen. The counter variable i is added to ease tracking.

Script environment

For the script running on MacOS, give Script Editor control of your computer or the error will show up.



This is done by checking Script Editor in Accessibility, Security & Privacy under Settings



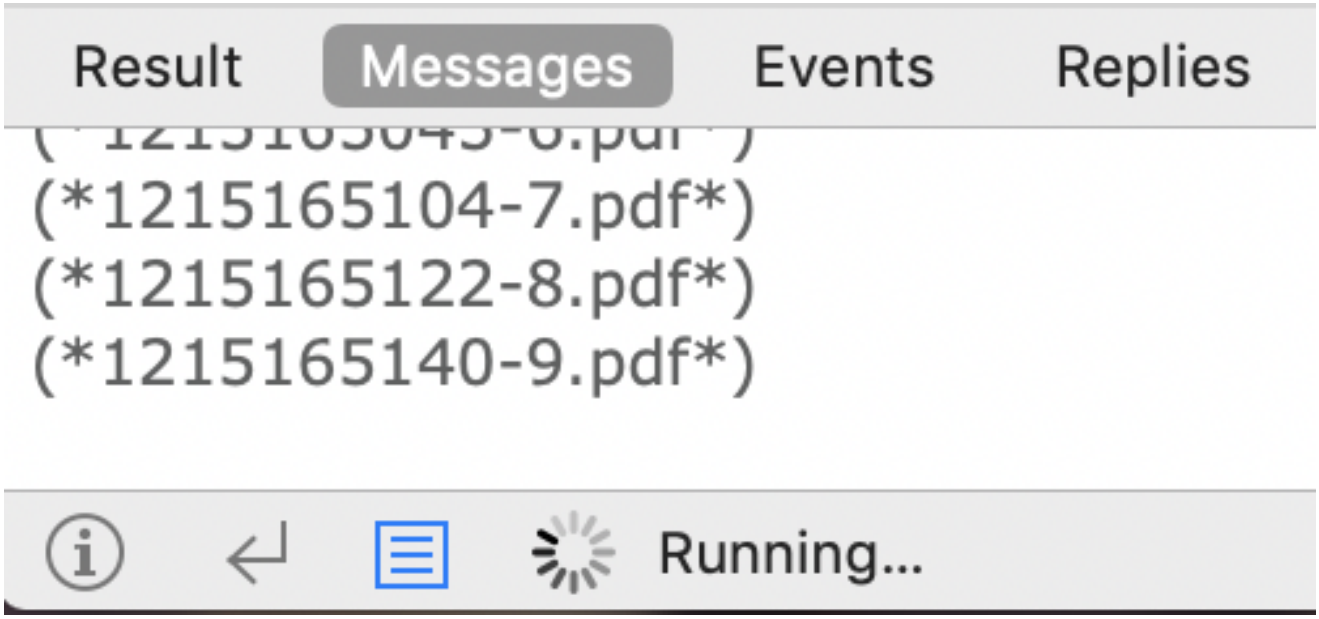
Besides, the scripts will not touch print settings. They just save files with minimum key stokes (lazy again to lower chance of failure).

Thus, please check the following for your needs.

1. Save as PDF is always selected on the print dialog when Ctrl-P (Windows) or Command-P (MacOS) is pressed.
2. Layout, Paper size and Scale are properly set so that PDF files are formatted as desired (e.g. Portrait +A4+65% for PA-5250 1-pager)
3. On Panachrome, 60s/60m/24h/7d/13w average is selected accordingly at ALL dataplanes, so that every test moment is covered with the complete set of captures.

Notes

- The AppleScript generates logs at the end of each iteration. You may track when it stops by reviewing "Messages" at the bottom of Script Editor.



- Use the Windows script (ahk) if the test will run long and a VM can be reserved for it.
- Use the MacOS script (scpt) otherwise.

References

- Quick Reference | AutoHotkey
- References For Learning & Using Applescript · GitHub
- Script to execute CLI commands at short regular intervals
- Script to perform CLI `commit` against a list of saved config
- Script to generate PA XML config for objects, policy rules and routes