

The macOS Process Journey

Version 4.0
January - 2024

By Dr. Shlomi Boutnaru



Created using [Craivon, AI Image Generator](#)

Table of Contents

Table of Contents	2
Introduction	3
Launchd (System/Per-user Daemon Manager)	4
UsageTrackingAgent	6
timed (Time Sync Daemon)	7
tccd (Transparency, Consent, and Control Daemon)	8
nfcd (Near Field Communication Daemon)	9
nearbyd (The Proximity Daemon)	10
Installd (Software Installation Daemon)	11
locationd (Location Services Daemon)	12
searchpartyd (Search Party Daemon)	13
apfsd (APFS Volume Manager)	14
coreaudiod (Core Audio Daemon)	15
hidd (Human Interface Device Daemon)	16
homed (Home Services Daemon)	17
powerd (Power Daemon)	18
sandboxd (Sandbox Daemon)	19
fontd (Font Daemon)	20
gamecontrollerd (Game Controller Daemon)	21
auditd (Audit Log Management Daemon)	22

Introduction

When starting to learn OS internals I believe that we must understand the default processes executing (roles, tasks, etc). Because of that I have decided to write a series of short writeups named "Process ID Card" (aimed at providing the OS vocabulary).

Overall, I wanted to create something that will improve the overall knowledge of MacOS in writeups that can be read in 1-3 mins. I hope you are going to enjoy the ride.

Lastly, you can follow me on twitter - @boutnaru (<https://twitter.com/boutnaru>). Also, you can read my other writeups on medium - <https://medium.com/@boutnaru>. Lastly, You can find my free eBooks at <https://TheLearningJourneyEbooks.com>.

Lets GO!!!!!!

Launchd (System/Per-user Daemon Manager)

When starting to learn OS internals we must understand the default processes executing (roles, tasks, etc). Because of that I am going to start a series of posts named "Process ID Card" (aimed at providing the OS vocabulary). We are going to start our macOS journey with "launchd".

Overall, "launchd" is the first user mode application which is started by the kernel of macOS. The pid of "launchd" is 1, we can think about it as similar to "init (pid 1)" in Linux systems (more information about that you can read in the following link <https://medium.com/@boutnaru/the-linux-process-journey-pid-1-init-60765a069f17>).

Based on man (yes we also have man in macOS ;-) "launchd" manages processes for both the system as well as for specific users ("man 8 launchd"). Also, "launchd" manages XPC services (more on that in future writeups) which are included in applications and frameworks on macOS. By the way, we can't execute "launchd" directly.

One of the differences between "init" (Linux) and "launchd" (macOS) is the fact that "launchd" has a concept of both daemons and agents. Daemons are system-wide services which have one instance for each system (like daemons on Linux systems). Agents are services which run on for specific users. The reason for the separation between daemons and agents is that only the agents should interact with users and display a UI (User Interface).

In order to configure "launchd" we should use "launchctl" ("man 1 launchctl"). The relevant files which store the configurations of "launchd" are stored in the following directories: "~/Library/LaunchAgents"¹, "/Library/LaunchAgents"², "/Library/LaunchDaemons"³, "/System/Library/LaunchAgents"⁴ and "/System/Library/LaunchDaemons"⁵. You can see an example of "/Library/LaunchDaemon" in the screenshot below⁶. All the configurations are based on "*.plist" files (Property List), I will explain about them in a future writeup.

For more information I suggest reading the following links from Apple's developer docs <https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/BPSystemStartup/Chapters/Introduction.html> and <https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/BPSystemStartup/Chapters/CreatingLaunchdJobs.html>.

¹Agents for a specific user configured by the user.

²Agents for a user configured by the administrator.

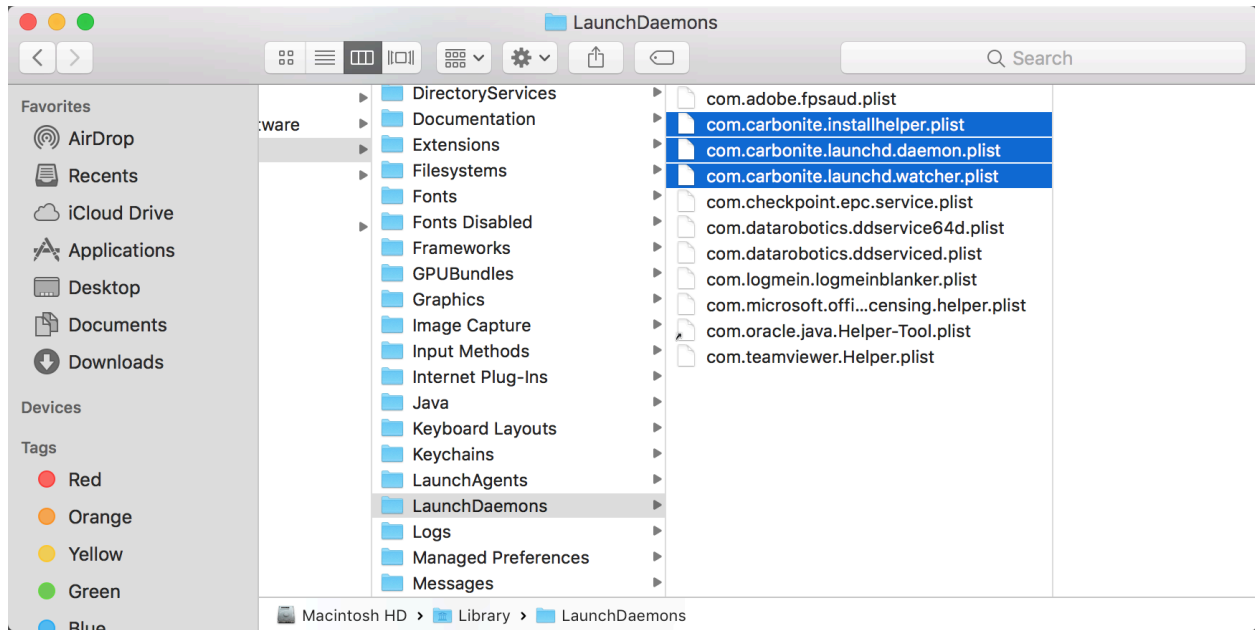
³Daemons configured by the administrator.

⁴Agents for a user provided by apple.

⁵Daemons provided by apple.

⁶ https://carbonite.azureedge.net/images/Mac/mac20_manuninstall_ldaemons_carboniteplists.png

Important note: init isn't just Linux (It's UNIX). Today, the majority of Linux systems have moved to systemd. Systemd has similar concepts as launchd, being able to run services as root and as a specific user/group. In the writeup I am focused on “old-fashioned” init (before the move to systemd).



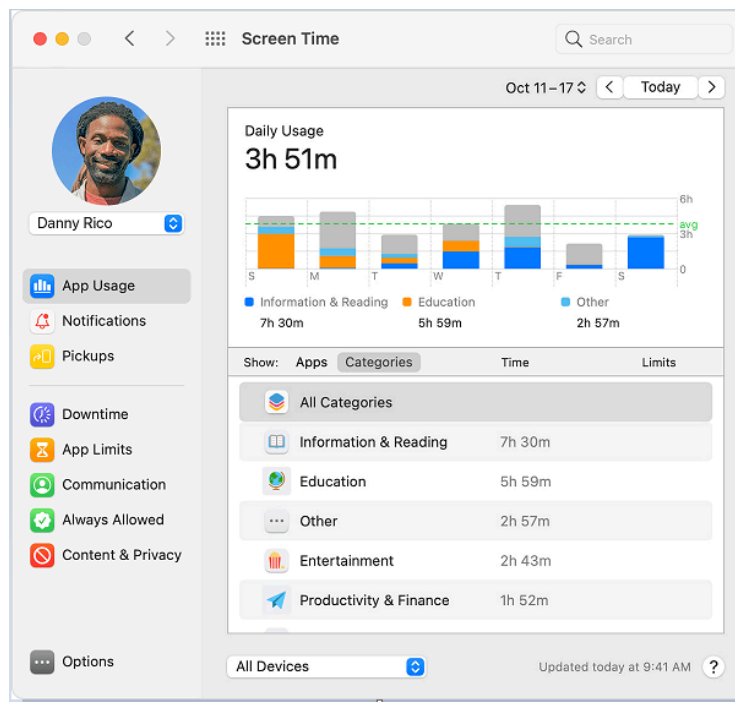
UsageTrackingAgent

“UsageTrackingAgent” is responsible for monitoring and reporting limits that have been set by “Screen Time”⁷. You can see the default UI in the image below⁸.

The “UsageTrackingAgent” is a Mach-O binary which is executed using permissions of the logged on user and it is signed by Apple’s software signing key. Also, it is being started by “launchd”⁹.

If you want to read more about “Screen Time” on Mac I suggest reading the following article <https://support.apple.com/en-us/HT210387>. By the way, due to the nature of the information that is collected by “Screen Time” it can be a great data resource for forensics analysis and incident response¹⁰.

Moreover, “Screen Time” is also relevant for iOS as can be seen in the following link <https://celebrite.com/en/data-quality-and-quantity-how-to-get-the-best-of-both-worlds-part-2-ex-aminig-screen-time-artifacts/>.



⁷ <https://manp.gs/mac/8/UsageTrackingAgent>

⁸ https://support.apple.com/library/content/dam/edam/applecare/images/en_US/macOS/Big-Sur/macOS-big-sur-system-prefs-screen-time-app-usage.png

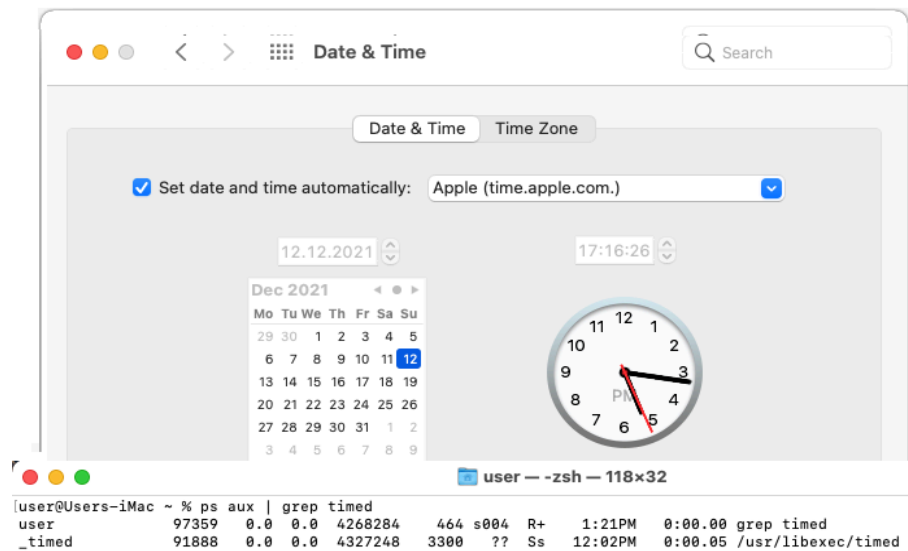
⁹ <https://medium.com/@boutnaru/macOS-launchd-a6628195f6e7>

¹⁰ <https://www.sans.org/cyber-security-courses/mac-and-ios-forensic-analysis-and-incident-response/>

timed (Time Sync Daemon)

The goal of “timed” is to maintain the accuracy of the system clock automatically, it is done by reference clocks (like by using NTP) - see the screenshot below¹¹.

Since 2017 (macOS High Sierra) the roles of “ntpd” (Network Time Protocol Daemon) have been merged into “timed”¹².



The relevant config files for “timed” are: “/etc/ntp.conf” (NTP server configuration), “/var/db/timed/com.apple.timed.plist” (cached state of “timed”) and “/System/Library/LaunchDaemons/com.apple.timed.plist” (the “time” service’s plist file for launchd). Moreover, “timed” started by launchd¹³ and users should not start it manually¹⁴.

“timed” reaches out to the NTP server every 15 mins and using the syscall `settimeofday`¹⁵ it sets the system clock¹⁶. Also, “timed” is running under the `_timed` user - as shown in the screenshot below¹⁷. `_timed` is part of the `_timed` and `_sntpd`¹⁸ groups. By the way, those are not the only groups that `_timed` is a member of.

¹¹ <https://malykh.blogspot.com/2021/12/macOS.html>

¹² <https://eclecticlight.co/2017/10/27/has-anyone-got-the-time-how-high-sierra-has-changed-time-synchronisation/>

¹³ <https://medium.com/@boutnaru/macOS-launchd-a6628195f6e7>

¹⁴ <https://keith.github.io/xcode-man-pages/timed.8.html>

¹⁵

https://developer.apple.com/library/archive/documentation/System/Conceptual/ManPages_iPhoneOS/man2/settimeofday.2.html

¹⁶ <https://developer.apple.com/forums/thread/83240>

¹⁷ <https://malykh.blogspot.com/2021/12/macOS.html>

¹⁸ <https://keith.github.io/xcode-man-pages/sntpd.8.html>

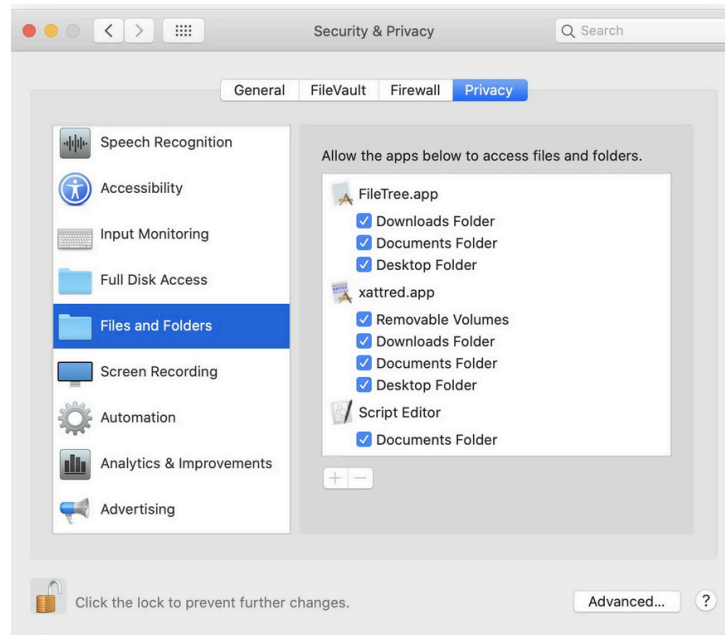
tccd (Transparency, Consent, and Control Daemon)

TCC is a macOS mechanism that is used in order to control/limit access of applications to specific features of the operating system. Among those privacy related permissions are: full disk access, location services, contacts, camera, accessibility, microphone and more¹⁹. It was introduced in OSX Mavericks (10.9) that was released on 22-October-2013.

Thus, we can say TCC is used in macOS for preventing applications from accessing data that might be sensitive without any user consent. For example in case of an access by an app to the “Downloads“ folder the operating system will display an alert²⁰. It is executed multiple times by launchd, once under the root user and the others using the permissions of the logged-on users

We can use the “Privacy” tab in the “Security & Privacy” pane in order to manage the configuration of TCC - you can see that in the screenshot below²¹. Also, can manage the privacy database using the “tccutil” cli command²².

All the information of TCC is stored using SQLite databases that can be located in the following folders: “/Library/Application Support/com.apple.TCC” and “~/Library/Application Support/com.apple.TCC”²³. The name of the database is TCC.db²⁴.



¹⁹ <https://www.rainforestqa.com/blog/mac-os-tcc-db-deep-dive>

²⁰ <https://www.malwarebytes.com/blog/news/2022/11/mac-os-ventura-bug-disables-security-software>

²¹ <https://eclecticlight.co/2020/01/28/a-guide-to-catalinas-privacy-protection-4-tccutil/>

²² <https://ss64.com/osx/tccutil.html>

²³ <https://i.blackhat.com/USA21/Wednesday-Handouts/US-21-Regula-20-Plus-Ways-to-Bypass-Your-macOS-Privacy-Mechanism-s.pdf>

²⁴ <https://eclecticlight.co/2018/11/20/what-does-the-tcc-compatibility-database-do/>

nfcd (Near Field Communication Daemon)

“nfcd” is responsible for managing the NFC (Near-Field Communication) controller²⁵. “nfcd” is a Mach-O binary file which is located at “/usr/libexec/nfcd”. Moreover, it is executed under the “_applepay” by launchd²⁶. It is needed by “ApplePay”, due to the fact it means making payments using NFC²⁷.

When running, “nfcd” exposes an XPC service called “com.apple.nfcd”. XPC service is an IPC mechanism which allows applications to send data and messages between each other²⁸. Also, iOS apps can also use NFC to read information from different electronic devices like: toys, products (for inventory tracking), in-store sign-up and more. When an app is active it can let the user know that there is an expectation for a scan to be conducted - as shown in the image below²⁹.



²⁵ <https://keith.github.io/xcode-man-pages/nfcd.8.html>

²⁶ <https://medium.com/@boutnaru/macos-launchd-a6628195f6e7>

²⁷ <https://www.macrumors.com/roundup/apple-pay/>

²⁸ <https://developer.apple.com/documentation/xpc>

²⁹ <https://developer.apple.com/design/human-interface-guidelines/technologies/nfc/>

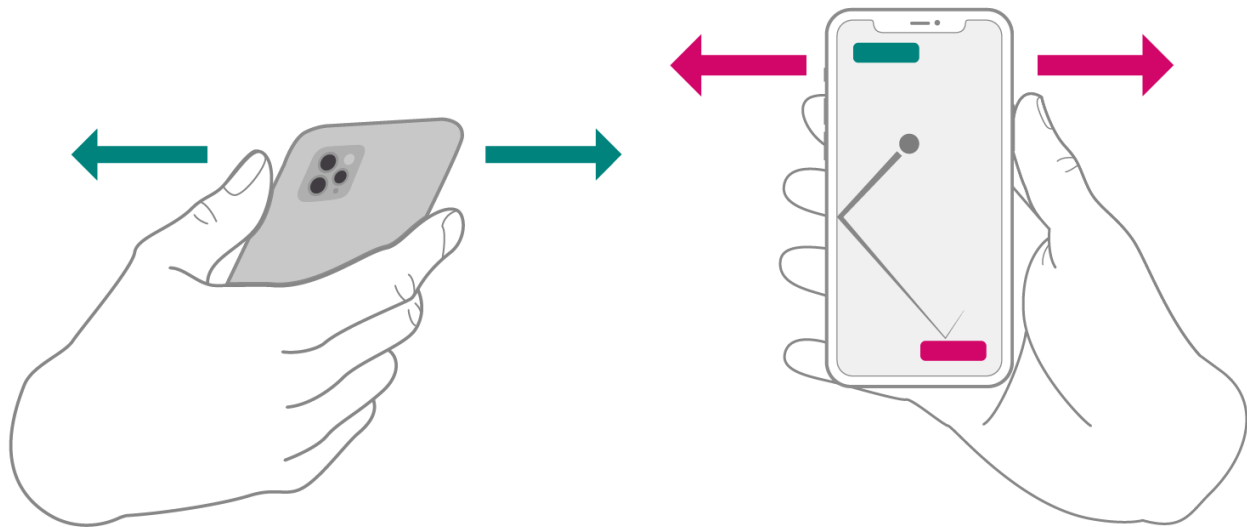
nearbyd (The Proximity Daemon)

“nearbyd” is the “Proximity Daemon” which is responsible for powering the spatial interaction between devices. “nearbyd” uses ultra-wideband and other wireless technologies. It is relevant since macOS 10.14/iOS 12.0³⁰.

Moreover, the Mac-O binary is executed from “/usr/libexec/nearbyd” by launchd³¹. The created process is running with the permission of the “_nearbyd” user.

Overall, by using “nearbyd” applications can interact with accessories by simply being close to them³². In order to participate, devices in a physical proximity that share their position and device token which uniquely identifies them. To achieve that Apple devices can use the high frequency capabilities of the U1 chip³³.

Thus, we can think about a game that allow users to control a paddle by moving their devices and playing near each other - as seen in the diagram below³⁴.



³⁰ <https://keith.github.io/xcode-man-pages/nearbyd.8.html>

³¹ <https://medium.com/@boutnaru/macos-launchd-a6628195f6e7>

³² <https://developer.apple.com/nearby-interaction/>

³³ <https://developer.apple.com/documentation/nearbyinteraction>

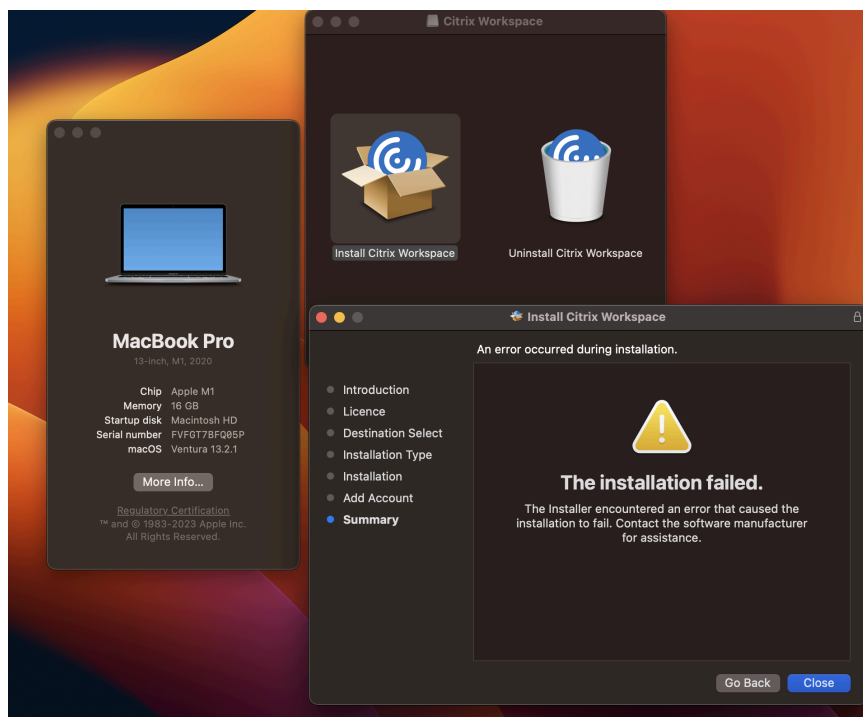
³⁴ <https://docs-assets.developer.apple.com/published/437d909426/rendered2x-1630937497.png>

Installd (Software Installation Daemon)

“installd” is the macOS software installation daemon. It is used while packages are being installed³⁵. Moreover, “installd” is not meant to be launched manually by users. It is started by launchd³⁶ with the permissions of the root user. For installing software using cli we can execute the “installer” tool³⁷. In order to query/manipulate macOS installer packages/receipts we can use “pkgutil”³⁸.

Thus, we can say that “installd” handles installations/updates of applications from the “App Store” or update of the OS itself³⁹. You can see an example of an error while installing an application in the screenshot below⁴⁰.

Lastly, “installd” is a component of Apple’s framework called PackageKit which is used for managing programs and updates⁴¹. By the way, the “installd” Mach-O executable is located in the “/System/Library/PrivateFrameworks/PackageKit.framework/Resources/installd”.



³⁵ <https://keith.github.io/xcode-man-pages/installd.8.html>

³⁶ <https://medium.com/@boutnaru/macos-launchd-a6628195f6e7>

³⁷ <https://keith.github.io/xcode-man-pages/installer.8.html>

³⁸ <https://keith.github.io/xcode-man-pages/pkgutil.1.html>

³⁹ <https://tw.begin-it.com/2134-what-is-the-installd-process-in-os-x-and-why-is-it-using-cpu-on-my-mac>

⁴⁰ <https://discussions.citrix.com/topic/418505-the-installer-encountered-an-error-that-caused-the-installation-to-fail-contact-the-software-manufacturer-for-assistance-macos-to-ventura-1321-citrix-workspace-app-23011/>

⁴¹ <https://iboysoft.com/news/macos-installd-high-cpu.html>

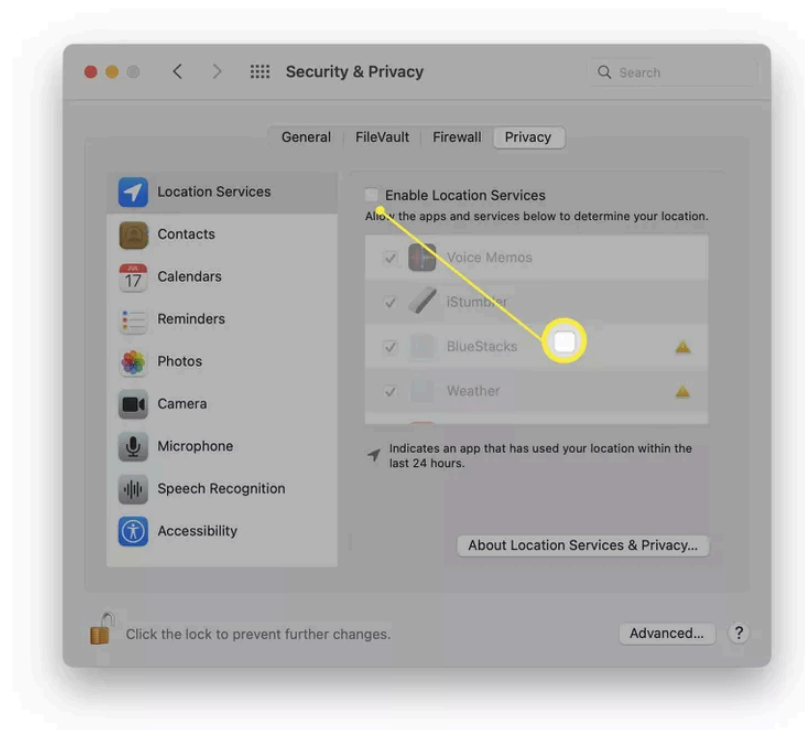
locationd (Location Services Daemon)

“locationd” is the “Location Services Daemon”, which is responsible for handling the geographical location of the Mac system. It is used to provide location information for specific Mac applications⁴². The “locationd” Mach-O binary is located at “/usr/libexec/locationd”.

Also, “locationd” manages the authorization for widgets, daemons and applications that request location updates⁴³. The process is executed with the permissions of the “_locationd” user (it is not the only process running with those permissions).

Moreover, macOS uses Wi-Fi network information in order to provide location data. One nice use case is for changing the system’s timezone automatically based on its location⁴⁴. “locationd” is started by “launchd”⁴⁵, the plist that is stored at the following location “/System/Library/LaunchDaemons/com.apple.locationd.plist”.

Lastly, in order to enable/disable “Location Services” we can go to “System Preferences->Security & Privacy->Privacy” and check/uncheck the “Enable Location Services” - as shown in the screenshot below⁴⁶.



⁴² <https://discussions.apple.com/thread/2510219>

⁴³ <https://www.manpagez.com/man/8/locationd/>

⁴⁴ https://www.tenable.com/audits/items/CIS_Apple_macOS_10.12_v1.2.0_Level_2.audit:313be81c537fdd9cc19caf953d550daf

⁴⁵ <https://medium.com/@boutnaru/macos-launchd-a6628195f6e7>

⁴⁶ <https://www.lifewire.com/turn-on-location-services-on-mac-6665787>

searchpartyd (Search Party Daemon)

“searchpartyd” is the “Search Party Daemon” which allows the discovery of remote devices and services⁴⁷. “searchpartd” is a Mach-O binary located at “/usr/libexec/searchpartyd” which is started by launchd() with the permissions of the root user.

Overall, “Search Party” is part of the “Find My” service for offline devices. Apple devices emit the public part of a rotating key pair using BLE (Bluetooth Low Energy). In parallel, receiving devices which are online encrypt the current location with the key and are sending that to Apple. The private key is shared by CloudKit⁴⁸.

Thus, we can say “Search Party” is the “offline finding” system of the “Find My” app. It is responsible for sending communication to Apple: sync crypto keys, sending location reports as a finder device, obtaining reports for owned devices⁴⁹. By the way, information sent from the device to Apple over HTTP uses a UserAgent with the substring “searchpartyd”⁵⁰.

Lastly, “searchpartyd” is used with other components as part of the “Find My” app as seen in the diagram below⁵¹.

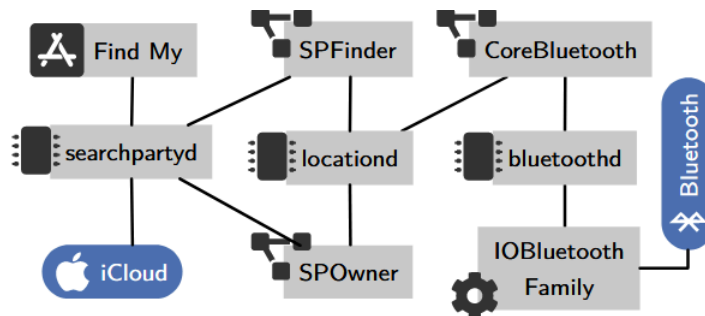


Fig. 3. Simplified view of components and their interactions such as apps (📱), daemons (👤), frameworks (⚙️), and drivers (🔌) that are used by OF on iOS. We highlight the two involved external communication interfaces in blue.

⁴⁷ <https://keith.github.io/xcode-man-pages/searchpartyd.8.html>

⁴⁸ <https://mroi.github.io/apple-internals/>

⁴⁹ <https://iboysoft.com/wiki/searchpartyuseragent.html>

⁵⁰ <https://arxiv.org/pdf/2103.02282.pdf>

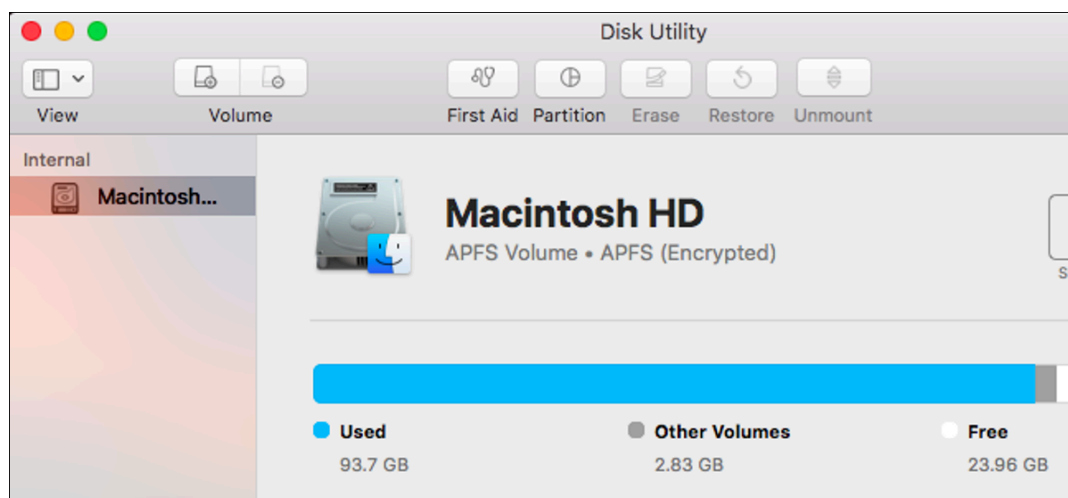
⁵¹ <https://arxiv.org/pdf/2103.02282.pdf>

apfsd (APFS Volume Manager)

“apfsd” is the “APFS Volume Manager” which is responsible for different tasks such as encryption/decryption, automatic file fragmentation and more⁵².

Overall, “apfsd” is a Mach-O binary located at “/usr/libexec/apfsd”, which is running with the permissions of the root user. It is started by “launchd”⁵³ in case an APFS volume is identified⁵⁴. You can see an example of an APFS volume in the screenshot below that was taken from “Disk Utility”⁵⁵.

Moreover, APFS stands for “Apple File System”, which is a proprietary file system that replaced HFS+ (Hierarchical File System) that was used by MacOS⁵⁶. APFS was introduced since MacOS Sierra (10.12.4)/iOS 10.3/tvOS 10.2⁵⁷. Lastly, APFS has multiple features like snapshots, encryption, data integrity, crash protection, compression and space sharing⁵⁸.



⁵² <https://www.unix.com/man-page/mojave/8/apfsd>

⁵³ <https://medium.com/@boutnaru/macos-launchd-a6628195f6e7>

⁵⁴ <https://www.unix.com/man-page/mojave/8/apfsd>

⁵⁵ <https://www.howtogeek.com/327328/apfs-explained-what-you-need-to-know-apples-new-file-system/>

⁵⁶ <https://www.lifewire.com/apple-apfs-file-system-4117093>

⁵⁷ https://developer.apple.com/library/archive/releasenotes/General/WhatsNewinTVOS/Articles/tvOS10_2.html

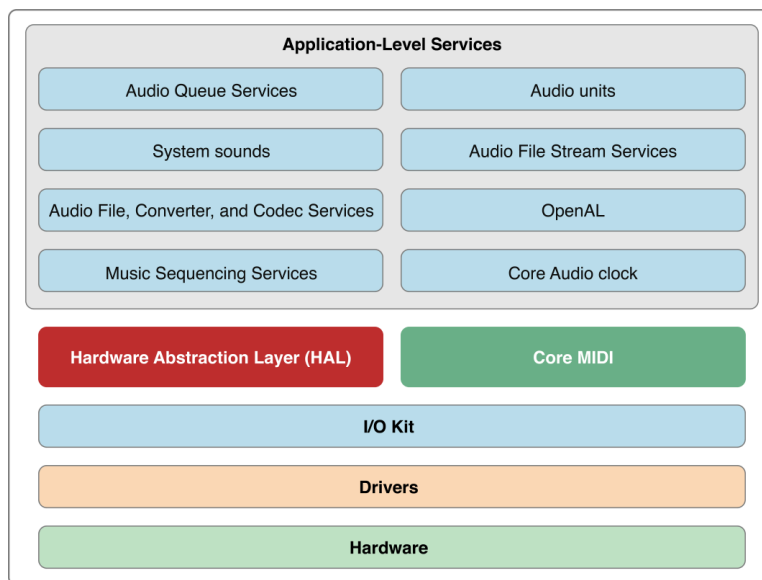
⁵⁸ https://en.wikipedia.org/wiki/Apple_File_System

coreaudiod (Core Audio Daemon)

“coreaudiod” is the “Core Audio Daemon” which is responsible for audio management. It was introduced as part of OS X 10.4⁵⁹. Overall, “Core Audio” includes everything related to editing/recording/playing/compressing/decompressing/MIDI (Musical Instrument Digital Interface)/signal processing/file stream parsing/audio synthesis⁶⁰.

Also, “coreaudiod” is a Mach-O binary started by launchd⁶¹ with the permissions of the “_coreaudiod” user. The binary is located at “/usr/sbin/coreaudiod”.

Thus, “coreaudiod” is part of the “Core Audio”, a low level API for working with sound as part of macOS/iOS⁶². It is also relevant for tvOS, watchOS and iPadOS⁶³. The OS X “Core Audio” architecture is shown in the diagram below⁶⁴. Lastly, For more information I suggest reading Apple’s documentation about “Core Audio”⁶⁵.



⁵⁹ <https://keith.github.io/xcode-man-pages/coreaudiod.8.html>

⁶⁰ <https://www.howtogeek.com/321905/what-is-coreaudiod-and-why-is-it-running-on-my-mac/>

⁶¹ <https://medium.com/@boutnaru/macos-launchd-a6628195f6e7>

⁶² https://en.wikipedia.org/wiki/Core_Audio

⁶³ <https://developer.apple.com/documentation/coreaudio>

⁶⁴ <https://developer.apple.com/library/archive/documentation/MusicAudio/Conceptual/CoreAudioOverview/WhatIsCoreAudio/WhatisCoreAudio.html>

⁶⁵ <https://developer.apple.com/library/archive/documentation/MusicAudio/Conceptual/CoreAudioOverview/Introduction/Introduction.html>

hid (Human Interface Device Daemon)

“hid” is the “HID Library Userland Daemon”⁶⁶. HID stands for “Human Interface Device”. Thus, “hid” intercepts all keyboard taps/mouse movement/trackpad gestures and more. Also, input from other devices like tablets are managed by “hid”. An example of such HID devices are shown in the image below⁶⁷.

Overall, if “hid” is not running most of the peripheral devices might be useless⁶⁸. “hid” is a Mach-O binary started by launchd⁶⁹ with the permission of the “_hid” user. The binary is located in “/usr/libexec/hid”.

Thus, we can summarize that the “hid” process purpose is to respond to input devices. In case of termination of “hid” it can block the system from responding to mouse/keyboard. However, the OS will automatically restart it⁷⁰.



⁶⁶ <http://www.manpagez.com/man/8/hidd/>

⁶⁷ <https://www.howtogeek.com/310378/what-is-hidd-and-why-is-it-running-on-my-mac/>

⁶⁸ <https://www.cyclonis.com/what-is-the-hidd-process-on-mac/>

⁶⁹ <https://medium.com/@boutnaru/macos-launchd-a6628195f6e7>

⁷⁰ <https://macmyths.com/activity-monitor-mac-what-to-quit/>

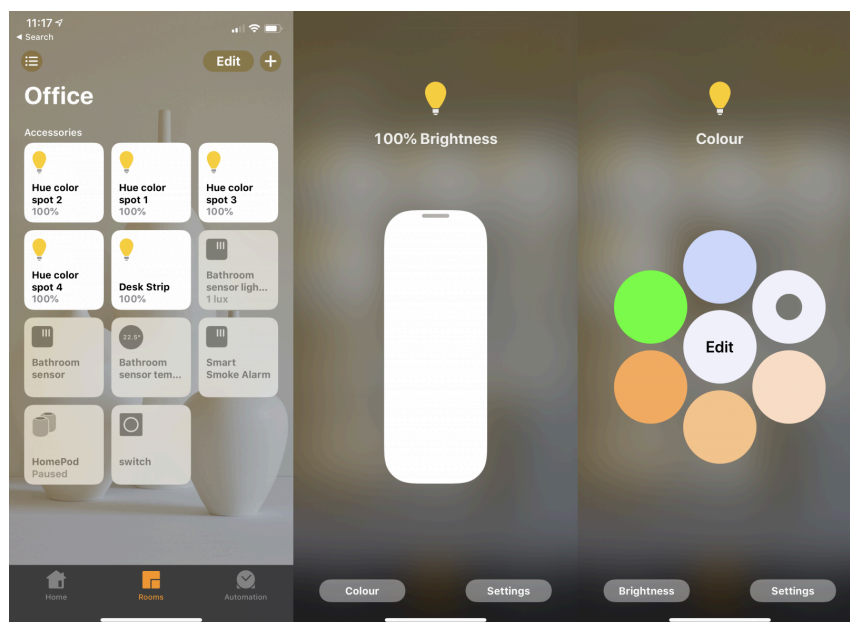
homed (Home Services Daemon)

“homed” is the “Home Services Daemon” which is responsible for managing the home state and controls of HomeKit accessories⁷¹.

Overall, HomeKit is Apple’s smart home platform. It allows users to control smart home products using Siri voice commands or apps running on iPhone/MacOS/iPad all at once using scenes and even set automations. HomeKit devices connect to an HomeKit setup using WiFi/Thread/hub⁷².

Moreover, HomeKits support different types of devices such as: video doorbells, security systems, locks, lights, fans, garage door openers, IP cameras, smoke alarms, sprinklers and more⁷³.

We can see an example of controlling a light bulb from an iOS, in which a user can adjust the brightness and the coloring scheme⁷⁴.



⁷¹ <https://keith.github.io/xcode-man-pages/homed.8.html>

⁷² <https://www.macrumors.com/guide/homekit/>

⁷³ <https://en.wikipedia.org/wiki/HomeKit>

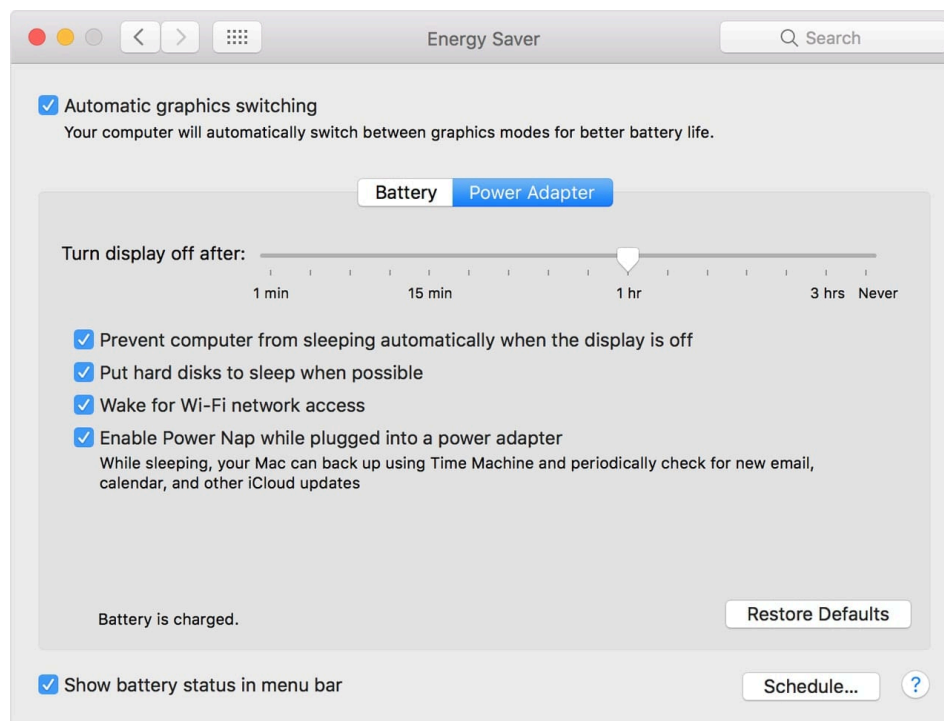
⁷⁴ <https://www.trustedreviews.com/reviews/apple-home-and-homekit>

powerd (Power Daemon)

“powerd” is the “Power Daemon” which is responsible for managing energy preferences⁷⁵. “powerd” is a Mach-O binary located at “/System/Library/CoreServices/power.bundle/powerd”. It is executed by “launchd”⁷⁶ with the permissions of the root user.

Moreover, “powerd” is the one responsible for putting the system to sleep after it is idle. The same thing is also relevant when the display shuts off/disks spin now⁷⁷..

Lastly, you can configure the setting of “powerd” in the “Energy Saver” section as part of the “System Preferences” - as seen in the screenshot below⁷⁸.



⁷⁵ <https://keith.github.io/xcode-man-pages/powerd.1.html>

⁷⁶ <https://medium.com/@boutnaru/macos-launchd-a6628195f6e7>

⁷⁷ <https://www.howtogeek.com/326965/what-is-the-powerd-process-and-why-is-it-running-on-my-mac/>

⁷⁸ <https://eshop.macsales.com/blog/57756-use-the-macs-energy-saver-preference-pane/>

sandboxd (Sandbox Daemon)

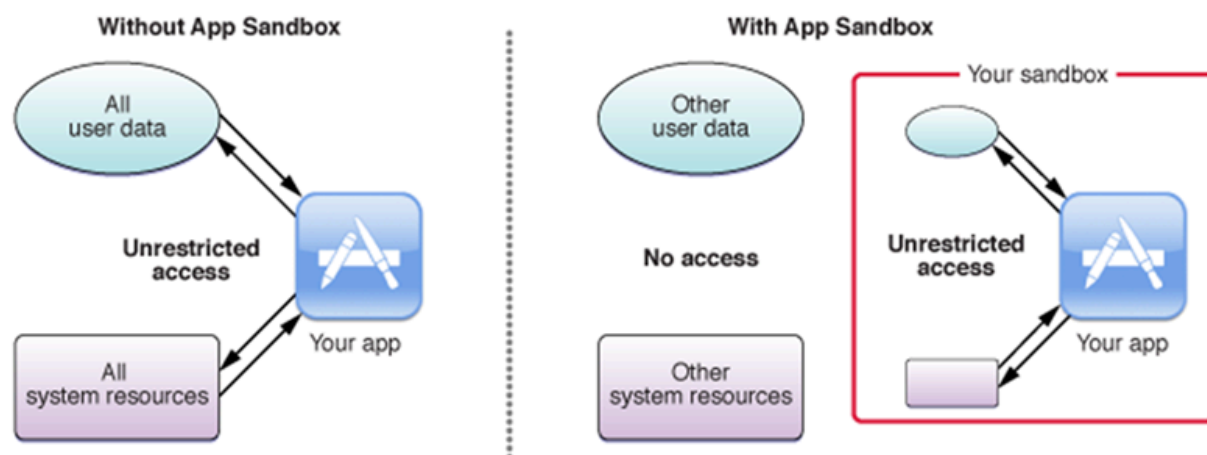
“sandboxd” is the “Sandbox Daemon” which is responsible for performing services on behalf of the Sandbox kernel extension⁷⁹. It is a Mach-O binary located at “/usr/libexec/sandboxd” which is started by “launchd”⁸⁰ with the permissions of the “root” user.

Overall, the sandbox facility in MacOS allows applications to restrict access to operating system resources. By doing so it helps in limiting the exposure in case of an exploitation of a vulnerability. New processes inherit the sandbox from their parent⁸¹.

Thus, we can say that an app sandbox allows the restriction of access to user data/system resources in order to contain damage if a MacOS application is compromised. This is achieved by limiting the app access by entitlements⁸².

We can split the entitlements to different categories such as: hardware, network, file operations, Application data and more. Let see two examples, one from the hardware category and one from the network category. “com.apple.security.device.microphone” which indicates if the application can use the microphone⁸³. “com.apple.security.network.client” which indicates if the application can open outgoing network connections⁸⁴.

Lastly, if we want we can use the cli command “sandbox-exec” to execute an application/command inside a sandbox⁸⁵. A diagram showcasing the sandboxing concept is shown below⁸⁶.



⁷⁹ <https://www.manpagez.com/man/8/sandboxd/>

⁸⁰ <https://medium.com/@boutnaru/mac-os-launchd-a6628195f6e7>

⁸¹ <https://www.manpagez.com/man/7/sandbox/>

⁸² https://developer.apple.com/documentation/security/app_sandbox

⁸³ https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_device_microphone

⁸⁴ https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_network_client

⁸⁵ <https://www.manpagez.com/man/1/sandbox-exec/>

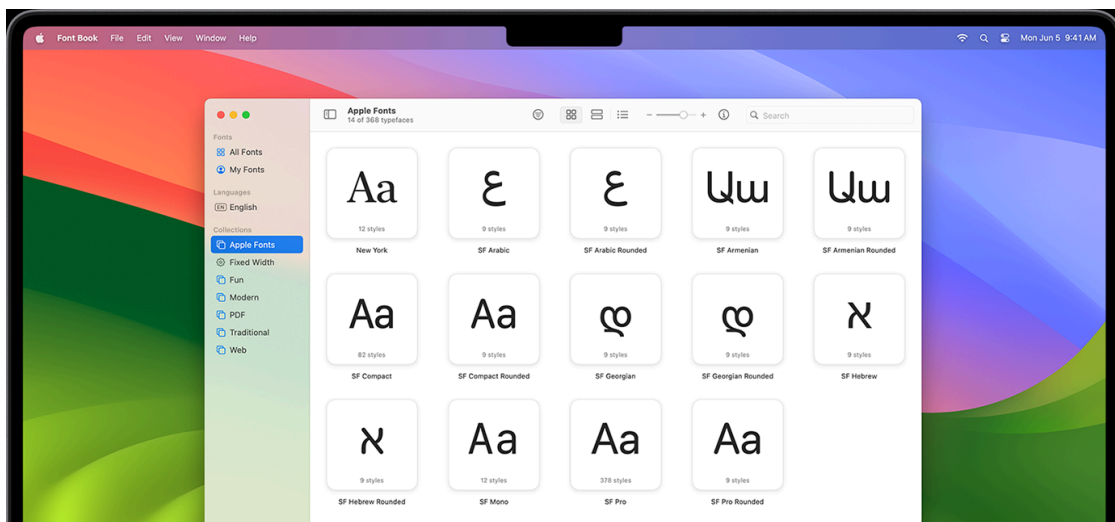
⁸⁶ <https://cyberhoot.com/cybrary/sandboxing/>

fontd (Font Daemon)

“fontd” is the “Font Daemon” which is responsible for managing font registration on MacOS. It makes fonts available to the system - as shown in the screenshot below⁸⁷. “fontd” replaced ATSServer from MacOS 10.6⁸⁸. By the way, AST stands for “Apple Type Services”.

Also, we can use “astutil” in order to register fonts using a system utility. Using the utility we can query the status of “fontd” or even shutting it down. Also, we can remove the “fontd” system/user database, which can cause the loss of fonts⁸⁹.

Overall, “fontd” is a Mach-O binary located at “/System/Library/Frameworks/ApplicationServices.framework/Frameworks/ATS.framework/Support/fontd”, which is started by “launchd”⁹⁰ using the permissions of the logged on user. Moreover, “fontd” hosts two services: “com.apple.FontObjectServer” and “com.apple.FontServer”⁹¹.



⁸⁷ <https://developer.apple.com/fonts/>

⁸⁸ <https://www.manpagez.com/man/8/fontd/>

⁸⁹ <https://www.manpagez.com/man/8/atsutil/>

⁹⁰ <https://medium.com/@boutnaru/macos-launchd-a6628195f6e7>

⁹¹ https://theycyberwire.com/events/docs/IanBeer_JSS_Slides.pdf

gamecontrollerd (Game Controller Daemon)

“gamecontrollerd” is the “Game Controller Daemon” which is responsible for arbitrating access to hardware controllers between applications. It is being done by leveraging the GameController framework. Also, it was introduced in OS X version 10.9⁹².

Overall, by leveraging the GameController framework users can interact with apps using virtual/physical game controllers. Among game controllers we can include Siri remote, DualSense, keyboards, XBox and even racing wheels⁹³.

Lastly, it is a Mach-O binary located at “/usr/libexec/gamecontrollerd”. It is started by “launchd”⁹⁴ with the permissions of the “_gamecontrollerd” user. We can find the user's game controller settings in the following file ~/Library/Preferences/com.apple.GameController.plist (you should not edit this file directly).



⁹² <https://keith.github.io/xcode-man-pages/gamecontrollerd.8.html>

⁹³ <https://developer.apple.com/documentation/gamecontroller>

⁹⁴ <https://medium.com/@boutnaru/macos-launchd-a6628195f6e7>

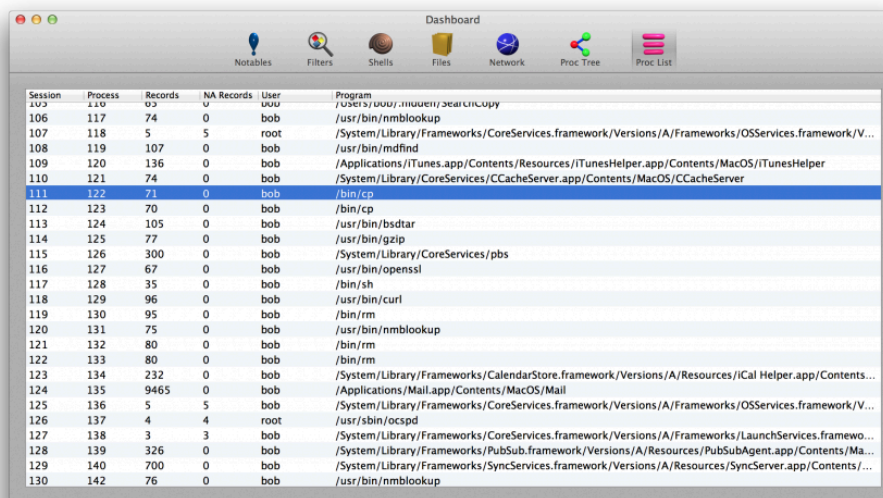
auditd (Audit Log Management Daemon)

“auditd” is the audit log management daemon . It is responsible for managing the resulting audit logs (like file integrity monitoring and access). “auditd” used the asl⁹⁵ API for writing to the log messages, thus only administrators and members of the audit review group can see them⁹⁶.

Moreover, the default location for storing the audit logs is “/var/audit” while the configuration files are stored by default at “/var/security”: audit_class (event class description - “man audit_class”), audit_control (audit system parameters -”man audit_control”), audit_event (event description - “man audit_event”) and audit_warn (issues and warnings - man “audit_warn”).

Also, “auditd” is a Mach-O binary file located at “/usr/sbin/auditd”. It is started by “launchd” and executes with the permission of the “root” user. The full command line used is “auditd -l”. It is based on OpenBSM. Which is an implementation of Sun’s “Basic Security Module” (BSM) audit API and file format. It defines a set of system calls and library interfaces in order to manage audit logs. It was created originally for apple computers and now maintained by volunteers (TrustedBSD). It has both user space/cli tools and a kernel portion⁹⁷.

Apple relicensed OpenBSM under a BSD license to allow integration with FreeBSD and other systems⁹⁸. We can also go over the source code of the kernel portion as part of the XNU⁹⁹. An example of an audit log of executed commands in macOS is shown in the screenshot below¹⁰⁰.



Session	Process	Records	NA Records	User	Program
103	110	83	0	root	/usr/bin/ls
106	117	74	0	bob	/usr/bin/nmblookup
107	118	5	5	root	/System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/OSServices.framework/V...
108	119	107	0	bob	/usr/bin/mdfind
109	120	136	0	bob	/Applications/iTunes.app/Contents/Resources/iTunesHelper.app/Contents/MacOS/iTunesHelper
110	121	74	0	bob	/System/Library/CoreServices/CCacheServer.app/Contents/MacOS/CCacheServer
111	122	71	0	bob	/bin/cp
112	123	70	0	bob	/bin/cp
113	124	105	0	bob	/usr/bin/bsdtar
114	125	77	0	bob	/usr/bin/gzip
115	126	300	0	bob	/System/Library/CoreServices/pbs
116	127	67	0	bob	/usr/bin/openssl
117	128	35	0	bob	/bin/sh
118	129	96	0	bob	/usr/bin/curl
119	130	95	0	bob	/bin/rm
120	131	75	0	bob	/usr/bin/nmblookup
121	132	80	0	bob	/bin/rm
122	133	80	0	bob	/bin/rm
123	134	232	0	bob	/System/Library/Frameworks/CalendarStore.framework/Versions/A/Resources/iCal Helper.app/Contents...
124	135	9465	0	bob	/Applications/Mail.app/Contents/MacOS/Mail
125	136	5	5	bob	/System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/OSServices.framework/V...
126	137	4	4	root	/usr/sbin/ocspd
127	138	3	3	bob	/System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/LaunchServices.framewo...
128	139	326	0	bob	/System/Library/Frameworks/PubSub.framework/Versions/A/Resources/PubSubAgent.app/Contents/Ma...
129	140	700	0	bob	/System/Library/Frameworks/SyncServices.framework/Versions/A/Resources/SyncServer.app/Contents/...
130	142	76	0	bob	/usr/bin/nmblookup

⁹⁵ https://developer.apple.com/library/archive/documentation/System/Conceptual/ManPages_iPhoneOS/man3/asl.3.html

⁹⁶ <https://nxmnpng.lemoda.net/8/auditd>

⁹⁷ <https://github.com/openbsm/openbsm>

⁹⁸ <http://www.trustedbsd.org/openbsm.html>

⁹⁹ <https://github.com/apple-oss-distributions/xnu/blob/main/bsd/bsm/>

¹⁰⁰ <https://halilozturkci.com/mac-os-x-sistemlerde-audit-log-analizi/>