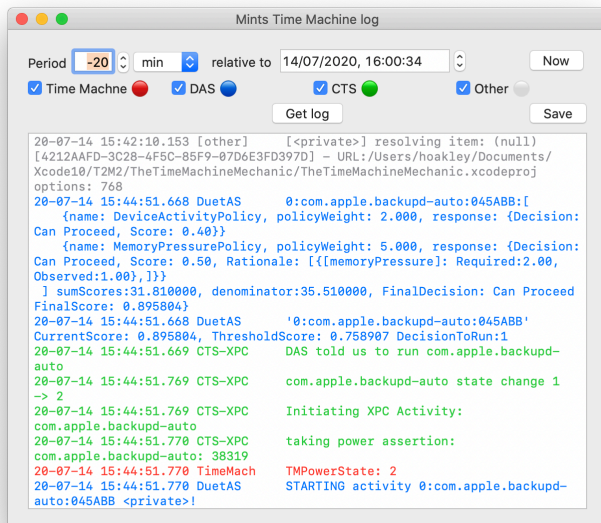


# Start

Mints is a growing collection of tools to provide deeper insight into macOS, to help you understand how its features work, and to help diagnose its problems. In this release, 12 tools are provided: these provide log extracts covering iCloud, TCC privacy protection, Time Machine backups, and task scheduling, test Spotlight search, look for Universal binaries, and show information about the Mach timebase, hardware, process environment, sudo, mounted volumes and keychains.

Complex systems in macOS write dialogues to the log while they operate. Following these in a normal log extract can be difficult, because lines in each conversation are interspersed with other irrelevant entries. Mints' log browsers are different because they bring together entries from many different sub-systems and services, helping you to follow those dialogues. The use of colour to distinguish different sub-systems makes these conversations even more understandable.



→ [Start](#)

→ [Contents](#)

# Contents

→ [Main window](#)

→ [Log Windows](#)

→ [iCloud](#)

→ [TCC](#)

→ [Time Machine](#)

→ [DAS Scheduling](#)

## Spotlight

→ [Log Window](#)

→ [Create Test](#)

→ [Check Search](#)

→ [Delete Test](#)

## Information

→ [Mac](#)

→ [Volume](#)

→ [Keychain](#)

→ [Environment](#)

→ [Mach Absolute Time](#)

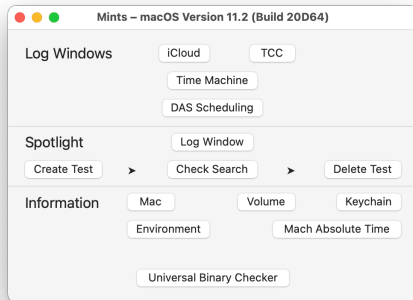
→ [Universal Binary Checker](#)

→ [Updates](#)

→ [Further information](#)

→ [Change list](#)

# Main window



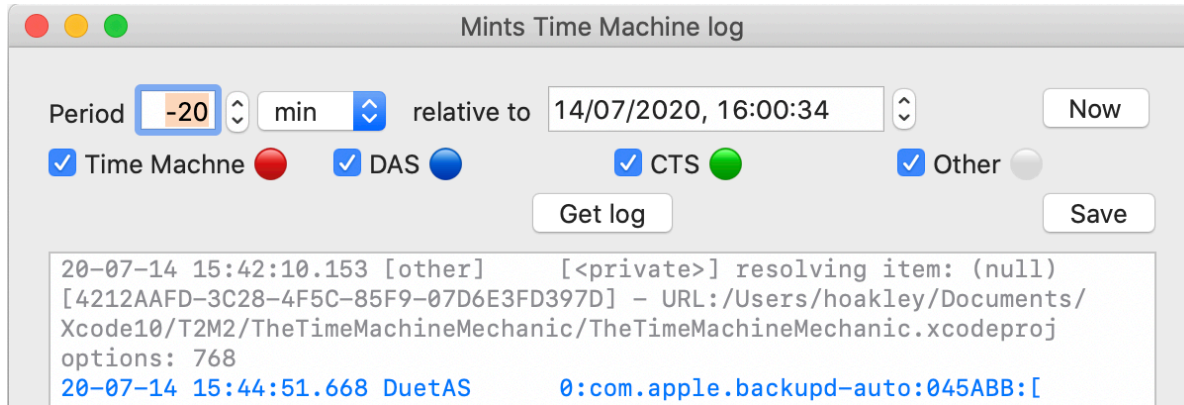
The main window currently offers 14 buttons to:

- open the → [iCloud Log Window](#), to explore iCloud messages in the log,
- open the → [TCC Log Window](#), to explore TCC and privacy system messages in the log,
- open the → [Time Machine Log Window](#), to explore Time Machine backup messages in the log,
- open the → [DAS Scheduling Log Window](#), to explore task scheduling messages in the log,
- open the → [Spotlight Log Window](#), the explore Spotlight searches in the log,
- → [create, run and delete](#) a test of Spotlight search,
- → [Get Mac Info](#),
- → [Get Volume Info](#),
- → [Get Keychain Info](#),
- → [Get Environment Info](#), about process execution environment,
- open → [Mach Absolute Time](#), to get information about clock rate and conversion factors,
- open the → [Universal Binary Checker](#), to discover which apps and other code runs native on Apple Silicon Macs (*available in Mojave and later only*).

When you close this window, Mints will quit.

→ [Log window controls](#)

# Log window controls



Each log window in Mints offers a simple method to specify the precise period over which log entries are to be extracted. This is based on a reference time, entered in the **relative to** box using its standard controls, and a **Period** offset from that reference, set using its stepper arrows or by editing the text directly. You can also copy and paste times in the **relative to** box: simply select an item in the box and use the normal commands.

The offset entered in the **Period** box has a popup menu to determine its units. There is also a button **Now** which will set the reference time to the immediate present. Use that with a period of  $-20$  sec to get log entries over the last 20 seconds.

To get a log extract for the 1 minute *before* a reference time, simply set **Period** to  $-1$  and the popup menu to **min**. To obtain an extract for 10 seconds each side of a time in the past, the total period wanted is 20 seconds. You can either specify that using the reference time at the start of that period and 20 sec, or the time at the end of the period and  $-20$  sec.

→ [Log window controls \(concluded\)](#)

## Log window controls (*concluded*)

The shortest period which can be used here depends on the version of macOS. In Sierra, periods less than about 30 seconds are unreliable, and entries can be missed. In Mojave and Catalina, shorter periods can be used, even below 10 seconds if you wish.

Times in the **relative to** box are in the current time zone for the set date. On days when clocks change, they use the second of the two zones. For example, when the change to summer time occurs on a day, times entered here for that day use summer time throughout.

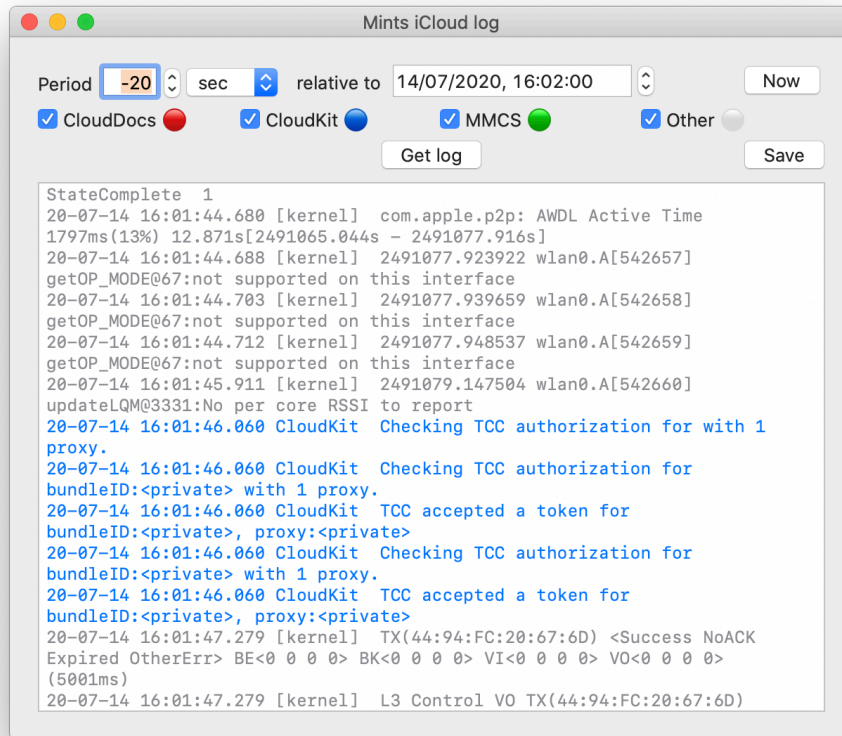
The row of controls below the time settings set which sub-systems are shown in the log view in the main part of the window. Each uses an emoji to indicate the colour in which those log entries are displayed.

The last controls are two buttons: **Get log**, which runs the log show command to fetch a fresh log extract, parse and display it, and **Save**, which saves what's currently displayed of the log in Rich Text format, which you can also do through the **Save** command in the File menu.

To enlarge the size of the text in any log extract, press ⌘+; to reduce the size, press ⌘- .

→ [iCloud log](#)

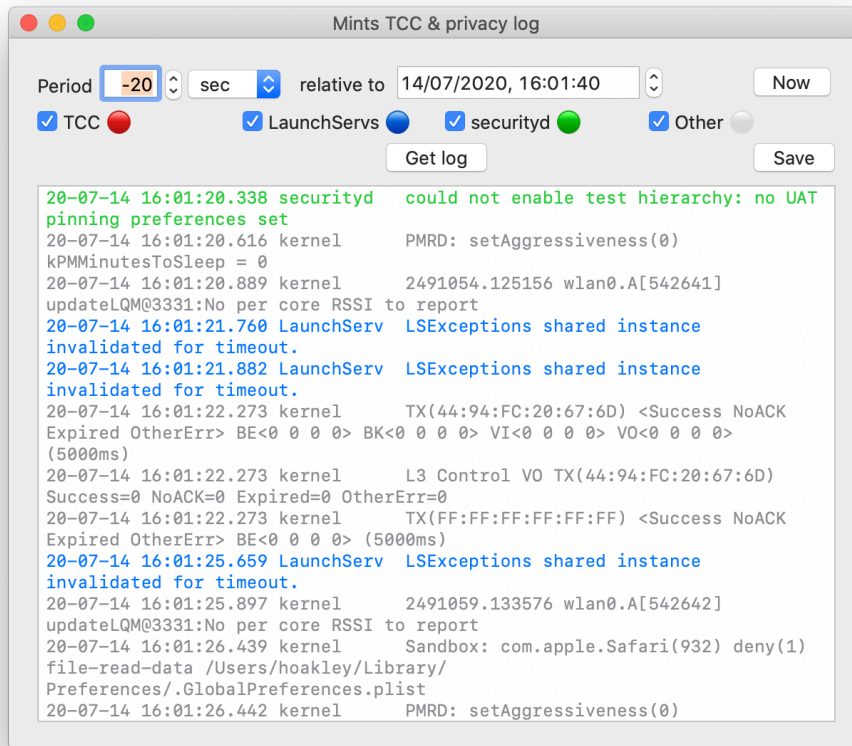
# iCloud log



The iCloud log extract shows entries from the three major sub-systems involved in iCloud transactions, and various related services and the kernel in the Other category. These are the same as shown in my free iCloud utility Cirrus.

→ [TCC log](#)

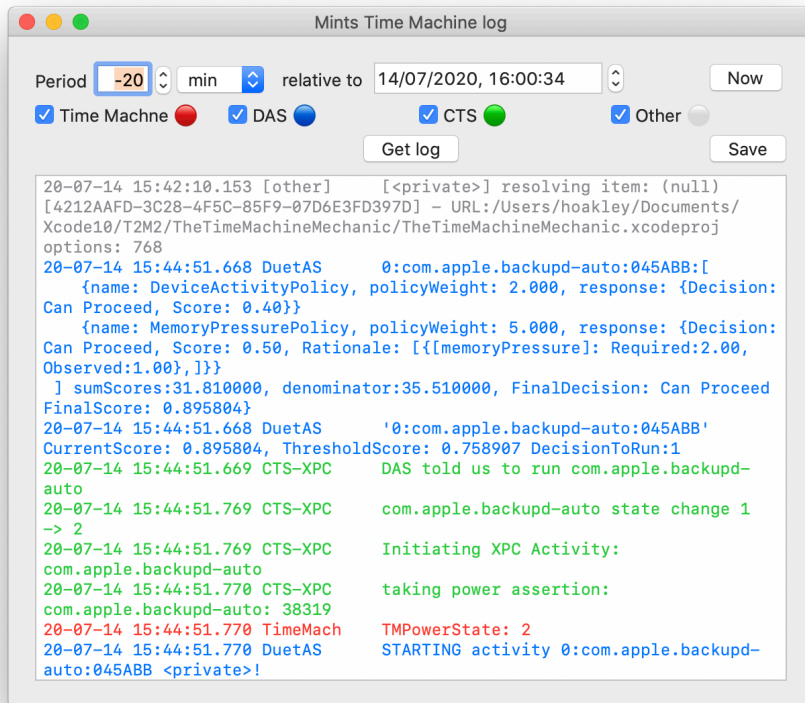
# TCC log



The TCC log extract shows entries from the three major sub-systems involved in controlling privacy and controlled access in Mojave and later. The Other category here includes additional services and the kernel. These are the same as shown in my free utility Taccy.

→ [Time Machine log](#)

# Time Machine log

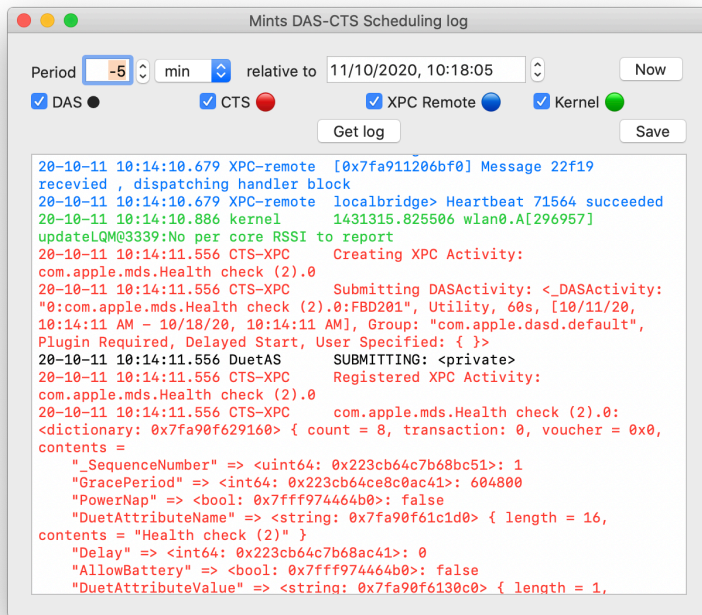


The Time Machine log extract shows entries from the three major sub-systems involved in scheduling, despatching and making Time Machine backups. The Other category here shows related services, but doesn't include the kernel. Although based on the extracts analysed by my free utility The Time Machine Mechanic, these extracts are new for Mints.

→ [DAS Scheduling log](#)



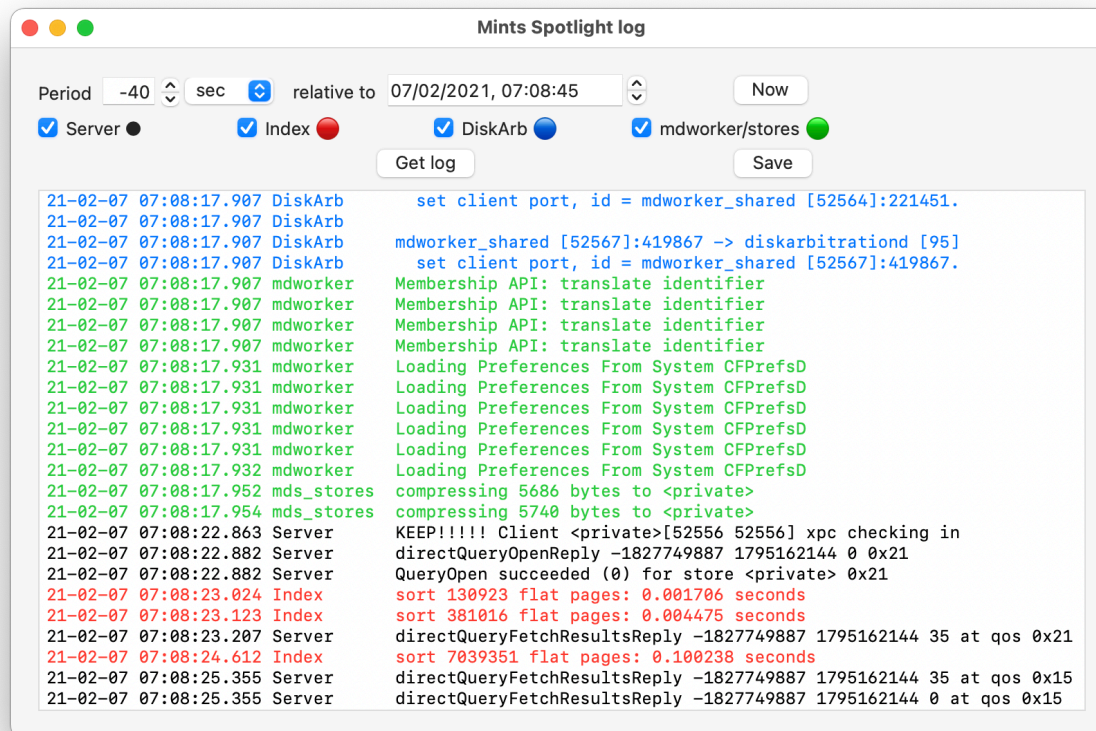
# DAS Scheduling log



The DAS scheduling log extract shows entries from the two major sub-systems involved in macOS background scheduling, which despatch and execute tasks, **DAS** (Duet Activity Scheduler) and **CTS** (XPC, Centralized Task Scheduling). The **XPC Remote** category here shows entries for XPC External, and **Kernel** shows messages from the kernel.

Routine background tasks in macOS often use this system for their scheduling. DAS determines when they can be run, according to priorities and load, then calls on CTS to call them using XPC. This log extract lets you watch the dialog between DAS and CTS, supplemented by information about external XPC activity, and that of the kernel.

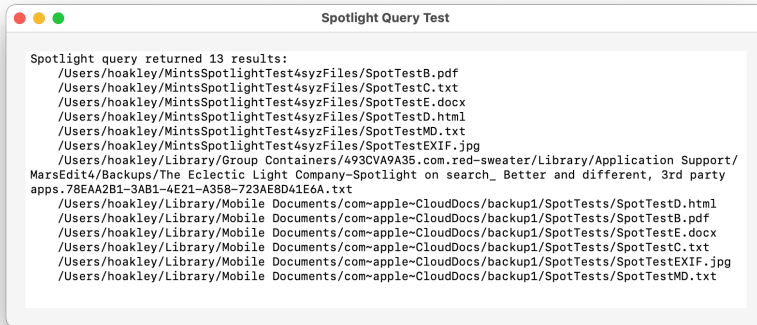
# Spotlight log



The Spotlight log extract shows entries from the three sub-systems involved in maintaining and accessing the Spotlight indexes: Spotlight Server, Index, and Disk Arbitration. The **mdworkers/stores** category here shows entries for mdworker\_shared and mds\_stores.

→ [Spotlight check and test](#)

# Spotlight check and test



Three buttons let you perform tests on Spotlight indexing and search, which are conveniently linked with inspecting its entries in the log.

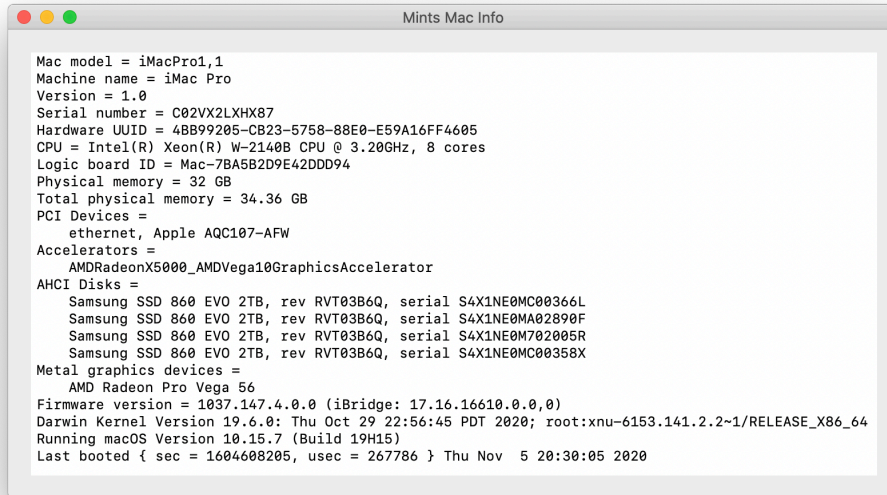
**Create Test** creates a new folder in your Home folder named MintsSpotlightTest4syzyFiles to contain the test files. Mints then copies 9 files there for testing purposes. To delete that folder and its contents, use the **Delete Test** button.

**Check Search** performs a Spotlight search for the text syzygy999 in files in the whole of your Home folder, including iCloud. It then opens a new window and displays the results, which should include the test files SpotTestA.rtf (Rich Text), SpotTestB.pdf (PDF), SpotTestC.txt (plain text), SpotTestD.html (HTML), SpotTestE.docx (Microsoft Word), SpotTestF.numbers (Numbers), SpotTestG.pages (Pages), SpotTestEXIF.jpg (EXIF metadata in JPEG), and SpotTestMD.txt (Keywords extended attribute). You can also add your own containing the search text to the folder. In addition to reporting which files were found, Mints also reports UTI and Spotlight Importer information for each test file in that folder, and gives the time taken to perform the search.

A standard check might start with **Create Test** at a clock time of, say, 01:00:00, followed by **Check Search** at 01:00:10. At 01:00:30 open the Spotlight log and set the period to -30 seconds to capture all entries since 01:00:00. When finished, clean up using **Delete Test** to delete the test folder and its entire contents, *including any custom files* you may have put there.

→ [Spotlight log](#)

# Get Mac Info

A screenshot of a macOS window titled "Mints Mac Info". The window contains a list of system information in a monospaced font. The information includes Mac model, machine name, version, serial number, hardware UUID, CPU details, logic board ID, physical memory, total physical memory, PCI devices, accelerators, AHCI disks, metal graphics devices, firmware version, Darwin kernel version, running macOS version, and the last boot time.

```
Mac model = iMacPro1,1
Machine name = iMac Pro
Version = 1.0
Serial number = C02VX2LXH87
Hardware UUID = 4BB99205-CB23-5758-88E0-E59A16FF4605
CPU = Intel(R) Xeon(R) W-2140B CPU @ 3.20GHz, 8 cores
Logic board ID = Mac-7BA5B2D9E42DDD94
Physical memory = 32 GB
Total physical memory = 34.36 GB
PCI Devices =
  ethernet, Apple AQC107-AFW
Accelerators =
  AMDRadeonX5000_AMDVega10GraphicsAccelerator
AHCI Disks =
  Samsung SSD 860 EVO 2TB, rev RVT03B6Q, serial S4X1NE0MC00366L
  Samsung SSD 860 EVO 2TB, rev RVT03B6Q, serial S4X1NE0MA02890F
  Samsung SSD 860 EVO 2TB, rev RVT03B6Q, serial S4X1NE0M702005R
  Samsung SSD 860 EVO 2TB, rev RVT03B6Q, serial S4X1NE0MC00358X
Metal graphics devices =
  AMD Radeon Pro Vega 56
Firmware version = 1037.147.4.0.0 (iBridge: 17.16.16610.0.0.0)
Darwin Kernel Version 19.0.0: Thu Oct 29 22:56:45 PDT 2020; root:xnu-6153.141.2.2~1/RELEASE_ARM_T8020
Running macOS Version 10.15.7 (Build 19H15)
Last booted { sec = 1604608205, usec = 267786 } Thu Nov  5 20:30:05 2020
```

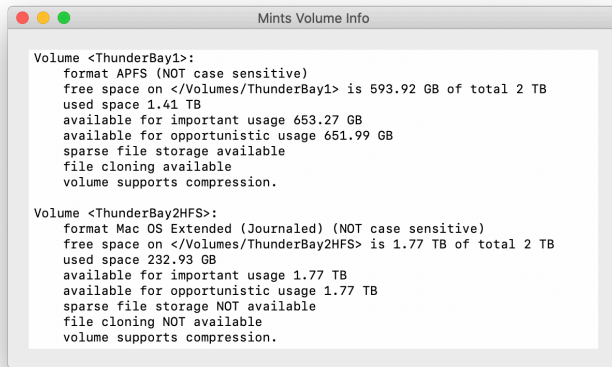
This window lists information about your Mac's hardware. This includes:

- the model designator, e.g. iMac17,1, machine name and version
- the serial number of that specific Mac, its hardware UUID
- the CPU type and details, and the ID for its logic board
- physical memory installed (two units), PCI devices (Ethernet card)
- graphics accelerators, AHCI disks, Metal graphics devices
- firmware version, kernel version, macOS version, and time of last boot.

Further items will be added to this list in the future.

Increase the size of the text used with `⌘+`; to reduce the size, press `⌘-`. Save the text to a text file using the **Save...** command in the **File** menu.

# Get Volume Info



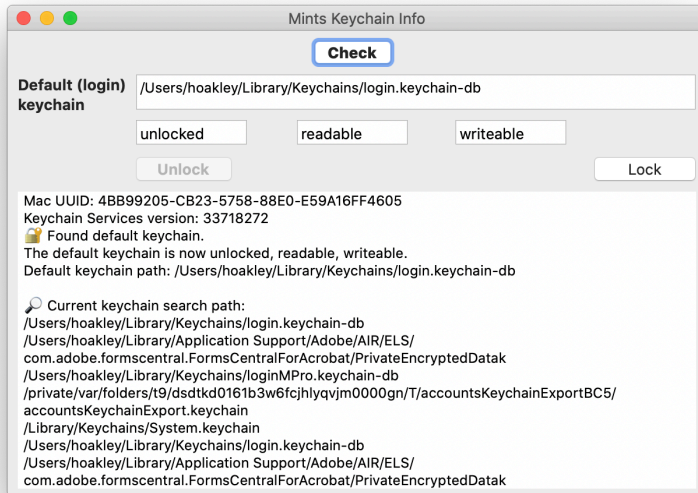
This window lists capacity information for all volumes currently mounted in the path /Volumes. This includes:

- volume name, format and case sensitivity
- total free space and total capacity
- the derived used space
- the amount of space available for 'important' and 'opportunistic' use
- whether sparse files, file cloning, and compression are supported.

On APFS volumes, this should include mounted snapshots, but doesn't include the current Data volume in the startup Volume Group (as that isn't mounted in /Volumes).

Increase the size of the text used with ⌘+; to reduce the size, press ⌘- . Save the text to a text file using the **Save...** command in the **File** menu.

# Get Keychain Info



This window has three simple button controls:

- **Check**, which runs its checks and updates the information in its text boxes;
- **Unlock**, which unlocks your default (login) keychain;
- **Lock**, which locks your default (login) keychain.

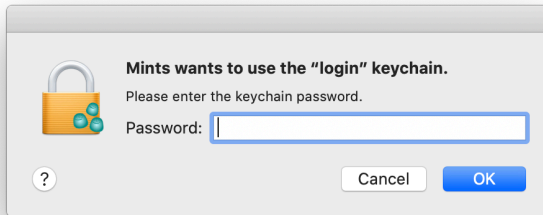
When you click on the **Check** button, Mints runs its full battery of tests, and completes the text boxes as shown below.

Displayed in the **Default (login) keychain** box is the full path to the keychain which macOS is using as your main user keychain at present. Even if you are using iCloud Keychain, this should be a path to the file named login.keychain-db in the Library/Keychains folder of your Home folder. If it isn't, then you need to establish why.

→ [Get Keychain Info](#) (concluded)

## Get Keychain Info (*concluded*)

Below that box are three smaller boxes, which tell you whether that default keychain is **locked**, **readable**, and **writeable**. You can change two of those by clicking on the **Unlock** or **Lock** buttons nearby. When you click on **Unlock**, a dialog will appear requiring you to enter the password for your default keychain (which should be the same as your normal user login password) in order to unlock that keychain, although in Catalina that may only appear when you click on **Lock**.



The bottom scrolling box contains the hardware UUID number of your Mac, its version number of Keychain Services, all the information from the other boxes, and a lot more too.

You can save that to a text file using the **Save...** command in the **File** menu. You can also select, copy and paste any of the text, but cannot modify it within this app.

You can resize text in the lower scrolling window between 4 and 24 points using ⌘+ to enlarge and ⌘– to reduce the size.

→ [Keychain Details](#)



# Keychain Details

**Mac UUID:** this should correspond with the Hardware UUID given in System Information.

**Keychain Services version:** this should be 33718272 for all versions of macOS from 10.12 to 10.15.

**Default keychain path:** should be `/Users/[username]/Library/Keychains/login.keychain-db` where [username] is the short user name of the current user.

**Current keychain search path:** should include the default keychain and `/Library/Keychains/System.keychain`.

**Keychain system search path:** should include `/Library/Keychains/System.keychain`.

**Keychain common search path:** should include `/Library/Keychains/System.keychain`.

**Keychain dynamic search path:** most probably empty.

**Keychain user search path:** should include the default keychain.

**Contents of `~/Library/Keychains`:** should include `login.keychain-db`, and is likely to include `metadata.keychain-db` too. For each file found, the date and time of last modification is given first, then that of the file's creation, then the size in bytes. The default keychain should not be empty.

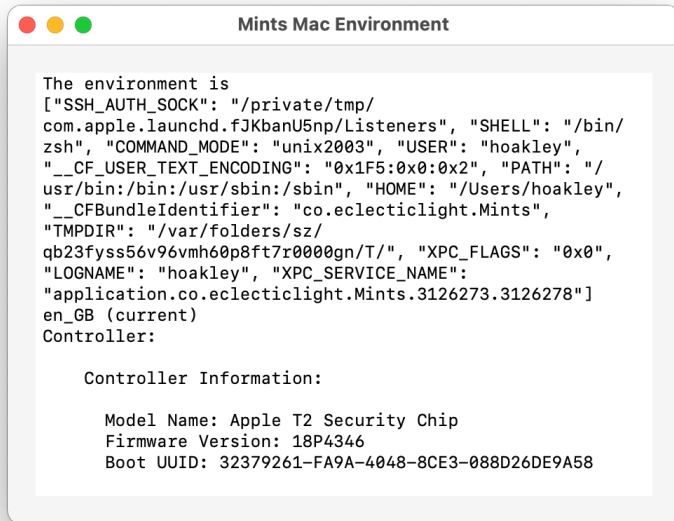
**Contents of `~/Library/Keychains/[UUID]`:** this is the folder used for iCloud Keychain, or for the Local Items listed in Keychain Access. It may not exist, but if it does it should include `keychain-2.db` at least, and sometimes much more.

**Contents of `/Library/Keychains`:** should include `System.keychain`, and may well also include the Apple Push Services keychain `apsd.keychain`

**Contents of `/System/Library/Keychains`:** should include `SystemRootCertificates.keychain` and is likely to contain `EVRoots.plist`, `X509Anchors`, and `SystemTrustSettings.plist`.



# Get Environment Info



```
The environment is
{"SSH_AUTH_SOCK": "/private/tmp/
com.apple.launchd.fJKbanU5np/Listeners", "SHELL": "/bin/
zsh", "COMMAND_MODE": "unix2003", "USER": "hoakley",
"__CF_USER_TEXT_ENCODING": "0x1F5:0x0:0x2", "PATH": "/
usr/bin:/bin:/usr/sbin:/sbin", "HOME": "/Users/hoakley",
"__CFBundleIdentifier": "co.electiclight.Mints",
"TMPDIR": "/var/folders/sz/
qb23fyss56v96vmh60p8ft7r0000gn/T/", "XPC_FLAGS": "0x0",
"LOGNAME": "hoakley", "XPC_SERVICE_NAME":
"application.co.electiclight.Mints.3126273.3126278"]
en_GB (current)
Controller:

Controller Information:

Model Name: Apple T2 Security Chip
Firmware Version: 18P4346
Boot UUID: 32379261-FA9A-4048-8CE3-088D26DE9A58
```

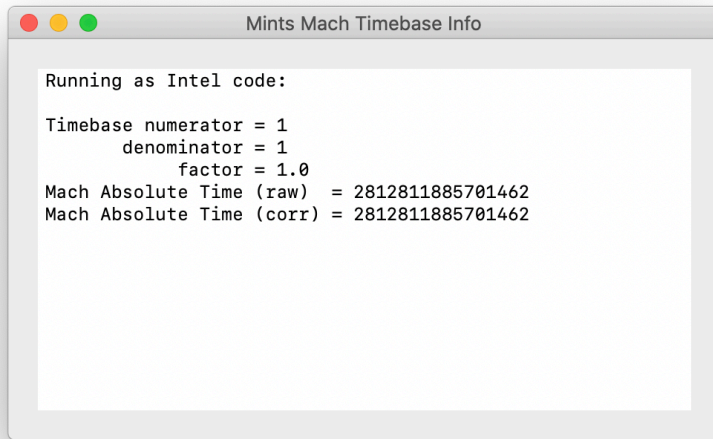
This window shows the process execution environment which processes launched by Mints would inherit. This is of particular interest to anyone experiencing problems which may be attributable to environment variables.

Although the environment shown is that for Mints, it should be typical of most apps run. Of particular interest are the shell, text encoding and any localisation, and TMPDIR variables.

Following that, an assessment is made as to whether the current version of macOS has been patched to fix the serious vulnerability in the sudo command.

The Controller information given at the end is specific to the type of Mac, differing between non-T2 and T2 Intel systems, and M1 Macs.

# Mach Absolute Time

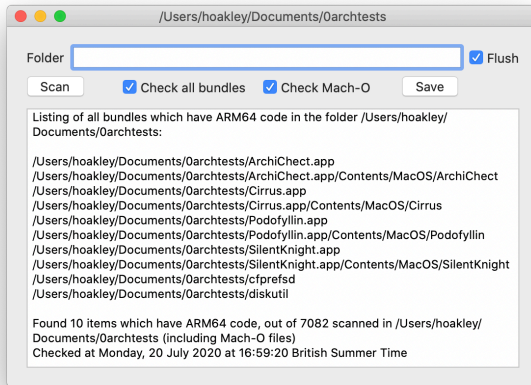


This window displays conversion factors and other information about Mach Absolute Time, which differ between Intel and Apple Silicon hardware, and between Intel code running using Rosetta 2 and native ARM code on Apple Silicon systems.

This first line states which architecture the code was run on, then gives the two integers for the Mach timebase correction, and the correction factor; multiply raw time values by that to scale them from ticks to nanoseconds. Finally, it gives the current raw time, and that time after correction.

To discover the factors applied to Intel apps running in Rosetta 2 translation on an Apple Silicon Mac, force Mints to **Open using Rosetta** on that Mac.

# Universal binary checker (Mojave and later)



Before scanning for Universal software, decide whether you just want apps scanned, or whether you want all code-containing bundles scanned. To scan apps only, leave the checkbox labelled **Check all bundles** unchecked; to scan all bundles, ensure it is ticked (the default).

Next, decide whether you want all executable (Mach-O) code files scanned too. These include dynamic libraries (dylibs), command tools, and other code which is not delivered in a bundle. To scan all executable code, ensure that the checkbox labelled **Check Mach-O** is ticked (the default); for a basic scan, leave that checkbox unchecked. Scanning all bundles and all Mach-O files will take significantly longer.

Then decide which folder you wish to scan. You can select that in either of two ways:

- leave the **Folder** text box empty, and click on the **Scan** button. You will then be prompted to select the folder using the normal Open File dialog;
- enter a standard path into the **Folder** text box. You can use the standard shortcut of ~ to indicate your Home folder. Once you have entered the path, click on the **Scan** button to start the scan. If the **Flush** checkbox is ticked when you do this, the path in the **Folder** text box will be cleared when the scan starts; if you uncheck the **Flush** checkbox, your path will be left for you to edit it for the next scan.

→ [Universal binary checker](#) (concluded)

## Universal binary checker *(concluded)*

Scanning takes time. If you put the root path / in the **Folder** text box, this can take many minutes, even an hour or so. You may wish to scan top-level folders like /Applications one at a time rather than attempting to scan your entire Mac. During this period, the window shows a busy ‘spinner’ next to the **Scan** button, and you can work normally with other apps. Give the app time to complete its scan, and once complete it will display the results.

Scanning takes plenty of memory. Because of the deep traversal which it uses, Mints consumes memory by the GB in order to check every nook and cranny in the folder it is scanning. This should work fine with virtual memory, though. Again, you can reduce this by performing scans on smaller folders.

When a scan is complete, the title of the window will change to show the folder which has just been scanned. Change the size of the text in the output using ⌘+ to make it larger, to a maximum of 60 points, or ⌘– to make it smaller, to a minimum of 4 points.

To save the results of a scan as a text file, click on the **Save** button, or use the **Save...** command in the **File** menu. You will then be prompted to select the name of the text file.

The scanner performs a deep traversal of the chosen folder, using `Bundle.executableArchitectures` to list executable architectures for each bundle. To check for Mach-O code files, it first inspects the ‘magic’ initial four bytes of every file. If it finds that they are `0xfeedface`, it then directly inspects the subsequent Fat Architecture slices to check whether the supported architectures include ARM64.

# Updates

Whenever you open Mints, it may check to see if an update is available. This *doesn't* use the popular Sparkle mechanism for updating in place, but works as detailed here.

Once Mints has successfully completed its integrity check, it checks whether update checking has been turned off in its preferences file. If that has, it abandons any attempt to check for updates. If checking is allowed, it then checks when it last checked for updates. If that was more than 12 hours ago, it continues to perform the check. It then connects to my GitHub server, from where it downloads a list of current versions of my apps. It doesn't upload any data to the GitHub server at all, and no statistics beyond GitHub normal connection figures are collected either: no personal identifiers are recorded. If there is an update available, Mints then checks that its location is on this WordPress blog, and posts a dialog which invites you to download the update.

If you click on the **Download** button, it then points your default browser at that update, which should trigger the update to be downloaded to your normal downloads folder. The update is received as a regular Zip archive, and is exactly the same as you would download from the Downloads page here. It also carries a quarantine flag, so that when you unZip it and install the app inside, it undergoes normal first run 'Gatekeeper' security checks. If you click on the **Ignore** button, Mints won't remind you about it again for another 12 hours.

An additional item at the end of the **Help** menu explains the update status. If no update check is performed, or the check fails, the last item reads **Update not checked**. If the check is performed and update information is obtained, even when no update is available or you decline to download it, that menu item reads **Checked for update** and is ticked (but still disabled).

You can customise this behaviour by changing Mint's preferences. The keys to use are:

- `noUpdateCheck`, a Boolean. When set to `true`, this disables all update checking. Default is `false`.
- `updateCheckInt`, a real number (Double). When set to a value greater than 1.0, the minimum time interval between checks, in seconds. Default is 43200, which is 12 hours. If you set it to any value less than 1, Ulbow will reset it automatically to that default.

To change either of these, use a Terminal command of the form

```
defaults write co.electiclight.Mints updateCheckInt '10'
```

which works properly through the preferences server `cfprefsd`.

# Further Information

The monospace font using in versions of macOS up to and including Mojave is `monospacedDigitSystemFont`; that used in Catalina and Big Sur is `monospacedSystemFont`, which is unfortunately not available in earlier versions of macOS. Both are used in their regular weight.

Predicates used to obtain log extracts include:

- `subsystem == "com.apple.cloudDocs" OR subsystem == "com.apple.CloudKit" OR subsystem == "com.apple.mmcS" OR processImagePath CONTAINS[c] "cloudD" OR processImagePath CONTAINS[c] "bird" OR processID = 0 (for iCloud)`
- `subsystem == "com.apple.TCC" OR subsystem == "com.apple.LaunchServices" OR subsystem == "com.apple.SecurityD" OR subsystem == "com.apple.Sandbox" OR processImagePath CONTAINS[c] "tccd" OR processImagePath CONTAINS[c] "sandboxD" OR processID = 0 (for TCC)`
- `subsystem == "com.apple.TimeMachine" OR (subsystem == "com.apple.DuetActivityScheduler" AND eventMessage CONTAINS[c] "Rescoring all") OR (subsystem == "com.apple.xpc.activity" AND eventMessage CONTAINS[c] "com.apple.backupd-auto") OR eventMessage CONTAINS[c] "backup" OR eventMessage CONTAINS[c] "Time Machine" OR eventMessage CONTAINS[c] "TimeMachine" (for Time Machine)`
- `subsystem == "com.apple.DuetActivityScheduler" OR subsystem CONTAINS "com.apple.xpc" OR processID = 0 (for DAS scheduling)`
- `subsystem == "com.apple.SpotlightServer" OR subsystem CONTAINS "com.apple.SpotlightIndex" OR subsystem CONTAINS "com.apple.DiskArbitration.DiskArbitrationD" OR processImagePath CONTAINS[c] "mdworker_shared" OR processImagePath CONTAINS[c] "mds_stores" (for Spotlight).`

Extensive information about Mints and the unified log is available from the [Eclectic Light Company blog](#), and the [product page](#) in particular. That can be accessed through the **Mints Support** command in the **Help** menu too.

# Change list

## 1.1:

- added busy spinner and search timing to Spotlight test
- added sudo test to the Environment feature.

## 1.0:

- additional files for Spotlight test
- added mdimporter info for Spotlight test
- first full release.

## 1.0b11:

- added Environment info window
- added Spotlight section, complete with test files
- laid out main window more systematically.

## 1.0b10:

- added many new items to the Mac Info feature, with support for Apple Silicon Macs too.

## 1.0b9:

- added DAS Scheduling log browser
- revised Volume Info and corrected its coverage and information
- fixed a bug in Mac Info which resulted in crashing if that Mac's logic board had no serial number.

## 1.0b8:

- added Mac Info feature

## 1.0b7:

- added Volume Info feature
- added Keychain Info feature
- added support for saving via menu command to main windows.

## 1.0b6:

- added Mach Absolute Time feature
- minor tweaks to the interface.

## 1.0b5:

- changed algorithm used by Universal Binary Checker to stop using lipo
- Universal App for for Intel and Apple Silicon Macs.

#### *1.0b4:*

- changed standard log text colours to black/white, red, blue, green
- added Universal Binary Checker for 10.14 and above.

#### *1.0b3:*

- tried to fix further bug opening log windows in Sierra.

#### *1.0b2:*

- fixed weird bugs in the log window nib.

#### *1.0b1:*

- first beta release, with iCloud, TCC and Time Machine log browsers.

20 February 2021.