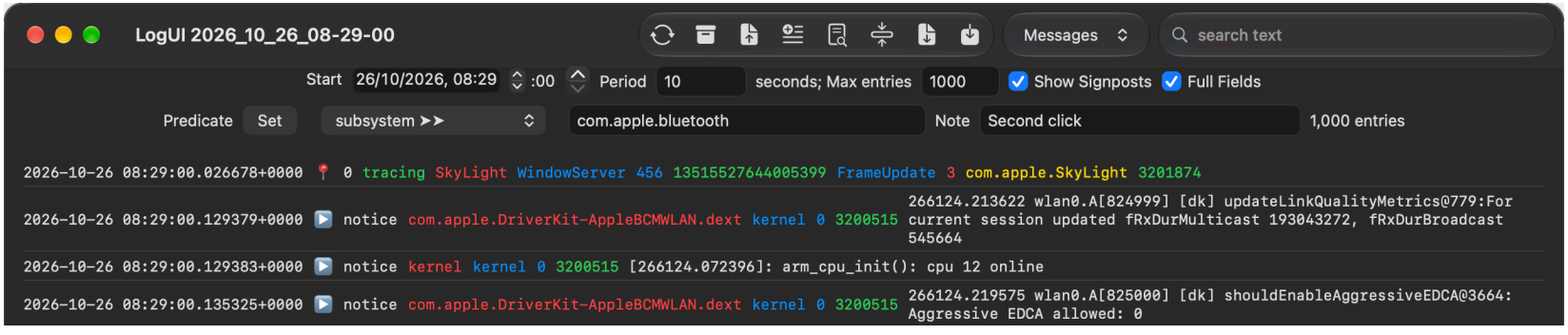
Start

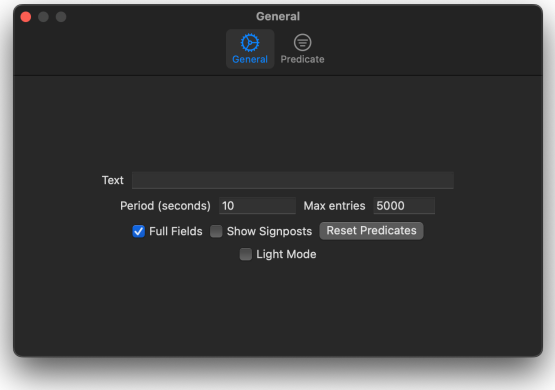


LogUI is a Unified log browser for macOS 14.6 Sonoma and later. Compared with Ulbow, Mints and my other log browsers, it features two major changes: its log engine uses API calls rather than the `log` command tool, and its front end is written in SwiftUI rather than AppKit. These make it a radical departure, and hopefully a refreshing and productive environment.

Support for writing your own filter predicates is still being improved. This version adds a tool to write log entries on the fly, and improves main window controls.

➜ Settings General
➜ Predicates
➜ Search
➜ Writing Tools
➜ Technical Information
➜ Settings Predicate
➜ One-off Predicate Editor
➜ Notes
➜ Diagnostics
➜ Tools
➜ Log display
➜ Log writer
➜ Logarchives
➜ Log settings
➜ Log entries
➜ Gloss
➜ Errors

Settings General

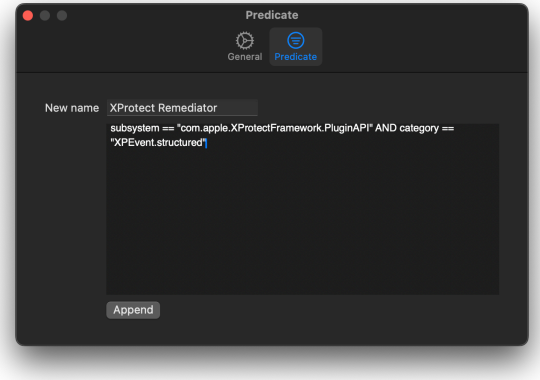


Set defaults for each log window here. Each of the five entries can be overridden in each window individually.

- **Text** is the default *subsystem, eventMessage* or *processImagePage* name used to filter log entries; this is case-insensitive.
- **Period** is the time in seconds for the log excerpt to include entries. This can be a decimal number, such as 2.5, but will always be used as a positive value, going forward in time from the start time.
- **Max entries** is the maximum number of log entries to be fetched and displayed.
- **Full Fields** displays all fields in each entry.
- **Show Signposts** fetches and displays Signpost entries as well as other types of log entry.
- The **Reset Predicates** button will discard all custom predicates and revert to those built into LogUI.

Light Mode sets all windows to appear in Light Mode. After changing this you must quit and re-open the app for the change to take effect.

Settings Predicate



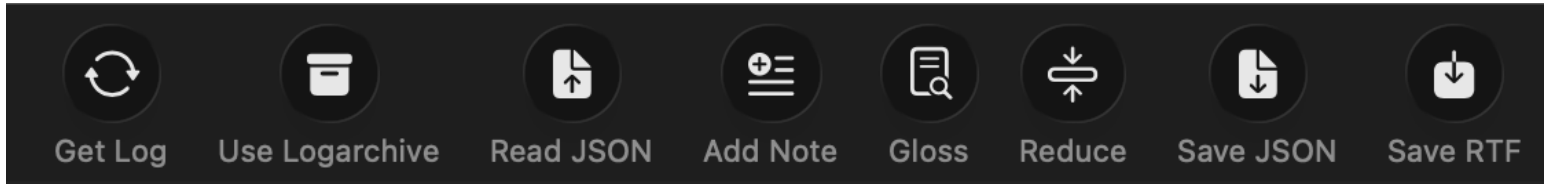
This provides an early and basic editor for adding your own custom predicate filters to LogUI's predicate menu. To do this:

1. enter the name you want to use for that predicate in the **New name** text box;
2. type or paste in the predicate text into the large text box;
3. click on the **Append** button to add the predicate to the app's preferences.

Currently, only one new predicate can be added at a time. For this to be saved correctly, you should then quit LogUI, and open it again before adding another predicate here. Although tedious, this enables you to build your own library of predicates in LogUI's preferences.

SwiftUI has a tendency to convert plain double-quotation marks " to styled marks “” but LogUI automatically corrects those for you when saving and handling predicates.

Tools



The toolbar at the top of the window offers seven action buttons, and search tools. From the left:

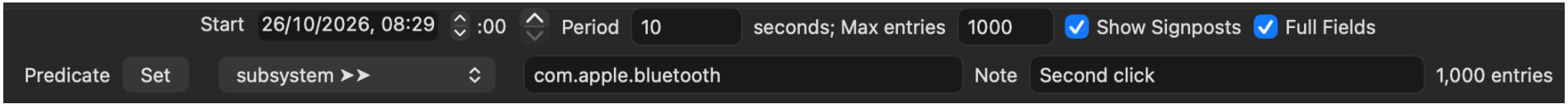
- **Get Log** fetches and displays log entries specified by the current log settings.
- **Use Logarchive** opens a dialog for you to select a logarchive to access, instead of the active log.
- **Read JSON** opens a dialog for you select a LogUI JSON file, and loads its log entries into the window, replacing its current contents.
- **Add Note** adds the text in the Note box below to all selected log entries, as a Note.
- **Gloss** opens the Gloss window with message text from selected log entries.
- **Reduce** applies the current search results to current log entries. This removes all those that are excluded by that search, so reducing the number of log entries. ⚠ If you want to keep a copy of the unreduced log before reducing it, save it as a JSON file before reduction, as there's *no undo.*
- **Save JSON** opens a dialog for you to save the current log records to a new LogUI JSON file. This has the type `co.eclecticlight.loguilog`, and the extension `.logui`. and can be used as a regular JSON export with other apps capable of importing JSON format.
- **Save RTF** saves the current log excerpt as a Rich Text Format file, using the same colour conventions.

To the right of those are the search menu and search box, and below the toolbar are settings for **Get Log**. Toolbar settings can be changed in its contextual menu (Control-click), to show icons and/or text.
⚠ Currently, if you double-click a LogUI JSON file or drop one on the app icon, a new window will be opened but the file's contents aren't opened in that.

➜ Settings General ➜ Settings Predicate ➜ Log settings ➜ Predicates ➜ One-off Predicate Editor
➜ Log display ➜ Log entries ➜ Search ➜ Notes ➜ Log writer ➜ Gloss ➜ Diagnostics ➜ Logarchives

Log settings

For the next log excerpt to be fetched using the **Get Log** tool, set:



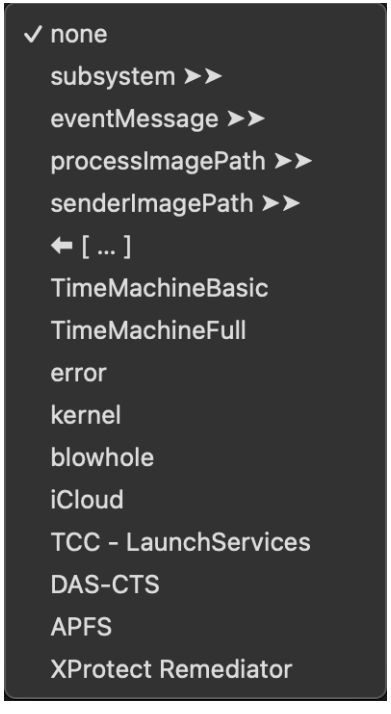
- **Start** to the date and time for the start of the log excerpt, with seconds in the separate stepper.
- **Period** to the time in seconds for the log excerpt. This can be a decimal number, such as 2.5, but will always be used as a positive value, going forward from the start time.
- **Max entries** to the maximum number of log entries to be fetched and displayed.
- **Show Signposts** to fetch and display all Signpost entries as well as other log entries.
- **Full Fields** to display all the fields in each log entry, otherwise only datestamp, sender, subsystem and message will be displayed.
- **Predicate** and its settings together determine the filter predicate to be used to select log entries.
- **Note** contains text to be added as a Note by the Add Note tool.
- **entries** at the far right displays the number of log entries in the current excerpt.

Values set here override defaults in Settings. When set to access a logarchive rather than the active log, a red warning is shown to the left of the **Start** time, as all dates and times must be set within those available in that logarchive.

➜ Settings General ➜ Settings Predicate ➜ Tools ➜ Predicates ➜ One-off Predicate Editor
➜ Log display ➜ Log entries ➜ Search ➜ Notes ➜ Log writer ➜ Gloss ➜ Diagnostics ➜ Logarchives

Predicates

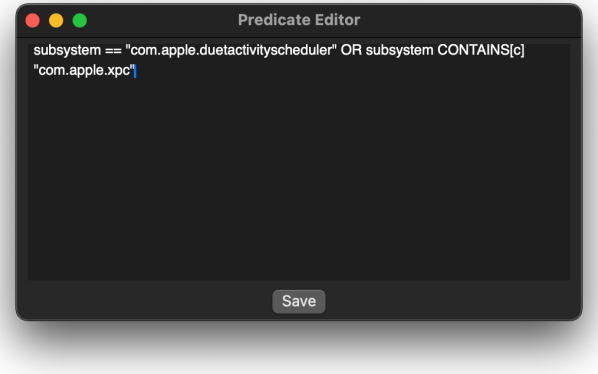
The Predicate menu filters log entries according to their contents. This offers the following options:



- **none** uses no filter, to display all log entries;
- **subsystem** obtains only those entries with the subsystem entered in the text box;
- **eventMessage** obtains only those entries with the text entered in the text box included in their eventMessage;
- **processImagePath** obtains only those entries with a processImagePath containing the text entered in the text box. Use this to filter entries from a named process such as the kernel.
- **senderImagePath** obtains only those entries with a senderImagePath containing the text entered in the text box.
- **[...]** obtains only those entries that match the filter entered in the one-off predicate editor using the **Set** button;
- five built-in predicates from **TimeMachineBasic** to **blowhole**, followed by those you have added to LogUI's preferences using Settings Predicate.

Text entered is not case-sensitive, so a subsystem entered as `com.apple.timemachine` will include entries with the subsystem `com.apple.TimeMachine`.

One-off Predicate Editor



To open this editor and edit the one-off predicate used by the **[…]** item in the predicate menu, click on the **Set** button to the left of the predicate menu. Enter the text of the predicate you want to apply in this window, then click on its **Save** button.

Once saved, this persists as it's saved to LogUI's preferences. This makes it a quick and simple way to develop and use a custom predicate before adding it to the predicate menu. Check its text carefully, though, as broken predicates can result in unexpected errors at present.

SwiftUI has a tendency to convert plain double-quotation marks " to styled marks “ ” but LogUI automatically corrects those for you when saving and handling this one-off predicate.

➜ Settings General ➜ Settings Predicate ➜ Tools ➜ Log settings ➜ Predicates ➜ Log display
➜ Log entries ➜ Search ➜ Notes ➜ Log writer ➜ Gloss ➜ Diagnostics ➜ Logarchives

Log display

```
2026-07-23 22:06:00.712925+0100 ▶️ client info CoreAnalytics Finder 606 com.apple.CoreAnalytics 3450 Dropping "com.apple.appkit.app_config" as it isn't used in a
2026-07-23 22:06:00.713198+0100 ⚾ 44072 0 AppKit Finder 606 3450 sendAction: 📝 First click
2026-07-23 22:06:00.760160+0100 ▶️ HCHearing notice HearingUtilities heard 669 com.apple.Hearing 4011 HeardController: BT not yet ready, delaying shutdown check
```

When **Full Fields** are displayed, fields shown for log entries are:
datestamp, type, **activity ID**, **category**, **level**, **parent activity ID**, **sender**, **process**, **process ID**, **subsystem**, **thread ID**, **message**, **note**

Types are coded as: ▶️ = regular; ⚾ = Activity; 🎬 = Boundary; 📍 = Signpost.

Signposts give three additional fields after the process ID: **signpost ID**, **signpost name**, **signpost type**.

When short fields are displayed, only the following fields are shown:
datestamp, **sender**, **subsystem**, **message**, **note**

Datestamps give: year–month–day hour:minute:second.microsecond±timezone
Times and time zones given are those current when the extract is obtained, not when that entry was written.

Notes are displayed after the note emoji, which is for display purposes only, and not part of the note.

The level *always* includes any Debug entries.

Fields that are empty aren't shown, and most entries don't use all fields available to them.

Log entries

Each log entry is shown in a single line in the log excerpt. Where the message field is too long to display, or is formatted in lines, it's fitted into the right end of the line as well as possible.

Select entries by clicking or tapping in their line in the window. Select *continuous* series of entries with the ⇧ **Shift** key modifier. Select *discontinuous* entries with the ⌘ **Command** key modifier, just as you would in the Finder.

Copy text content from selected entries using the **Copy** command in the Edit menu, shortcut ⌘-**C**.
Because of the number of fields and their length, only the following will be copied, with | separating them:

date | level | category | sender | process | subsystem | message *or* signpostName

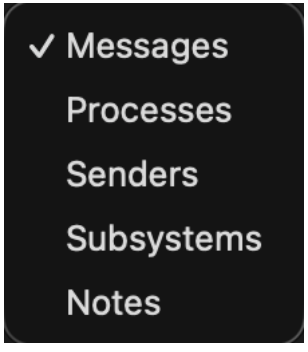
Copied entries can be pasted into any app accepting plain text from the clipboard.

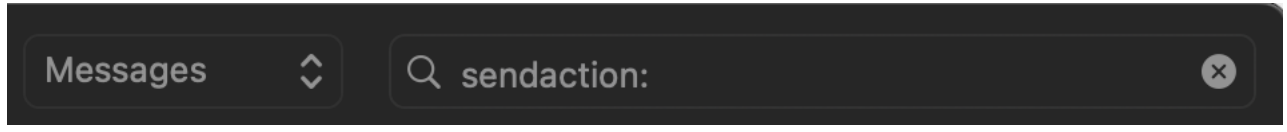
To view, edit, analyse and copy just the message contents from selected entries, click on the **Gloss** button in the toolbar.

If you want fuller or formatted log entries, save the log excerpt as an RTF file and copy from that.

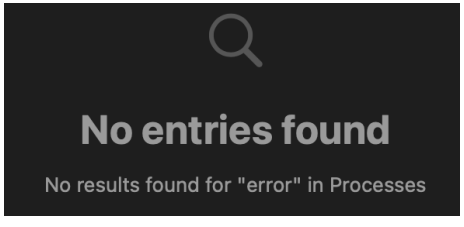
➜ Settings General ➜ Settings Predicate ➜ Tools ➜ Log settings ➜ Predicates ➜ One-off Predicate Editor
➜ Log display ➜ Search ➜ Notes ➜ Log writer ➜ Gloss ➜ Diagnostics ➜ Logarchives

Search





To search the current log excerpt, set the field to be searched in the popup menu in the toolbar, enter the text to be searched for in the Search box to the right, and press Return. If the search is unsuccessful and can't find any log entries with that text in the selected field, the window will report that.

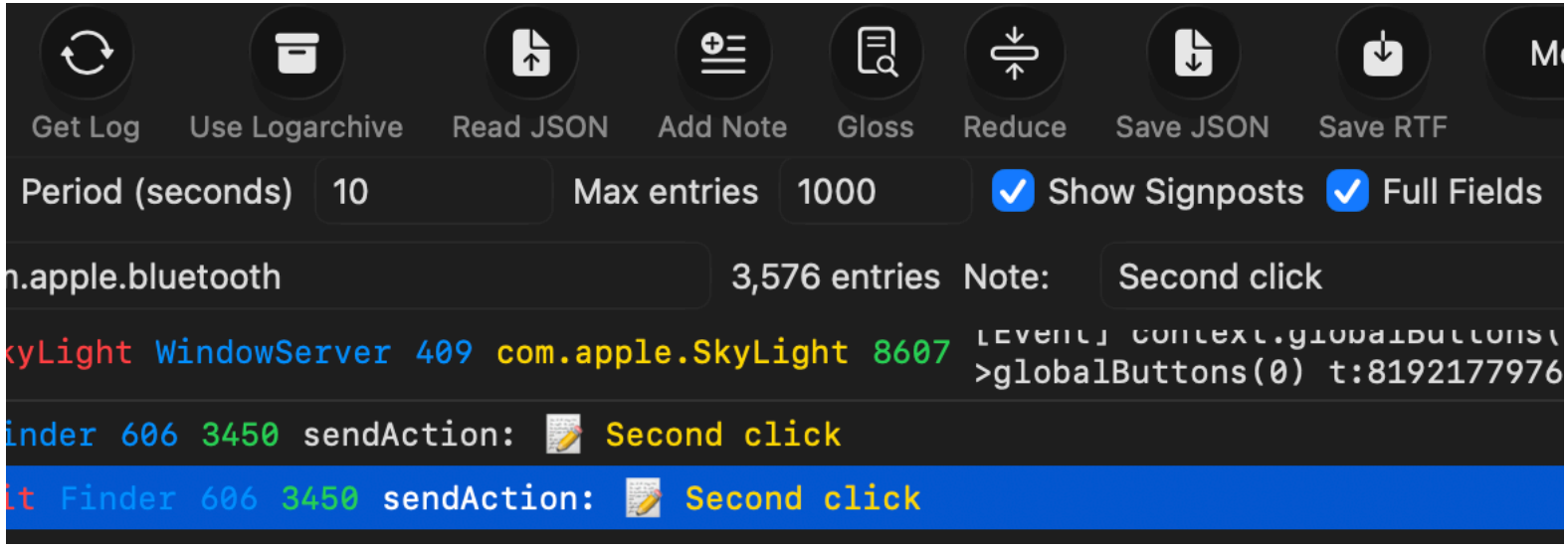


Search is *case-insensitive,* and will only search again when the search box is in focus (active) and the Return key is pressed. Live incremental search, performed as you type in characters, isn't used as it would be overwhelming when searching tens of thousands of log entries. If pressing Return results in a warning sound, click in the Search box and press Return again.

To return to the full log excerpt, clear the text from the Search box and press Return.

➜ Settings General ➜ Settings Predicate ➜ Tools ➜ Log settings ➜ Predicates ➜ One-off Predicate Editor
➜ Log display ➜ Log entries ➜ Notes ➜ Log writer ➜ Gloss ➜ Diagnostics ➜ Logarchives

Notes



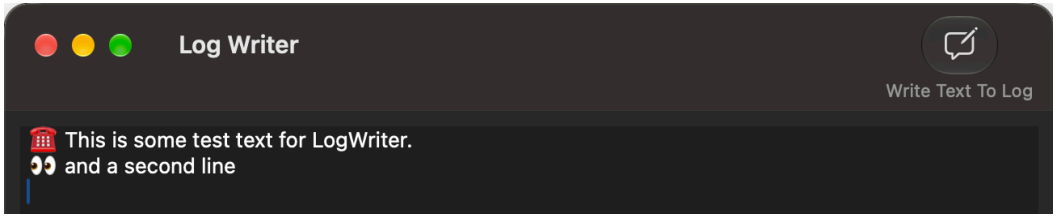
As with other log browsers, LogUI doesn't let you change the contents of log entries. However, it now supports adding a note to any log entries you wish:

- to add a note, enter the text for that note in the **Note** text entry box at the lower right of the window header, select the log entries to add that note to, and click on the **Add Note** tool;
- to remove a note, remove all text (including spaces) from the **Note** text entry box, select the log entries to remove notes from, and click on the **Add Note** tool. If any entries don't have a note, they should remain noteless.

Combine this with **Search**, for instance, to add multiple copies of the same note to all log entries selected in search output. Notes are saved to both JSON and RTF files, but aren't included in text copied from log entries.

➜ Settings General ➜ Settings Predicate ➜ Tools ➜ Log settings ➜ Predicates ➜ One-off Predicate Editor
➜ Log display ➜ Log entries ➜ Search ➜ Log writer ➜ Gloss ➜ Diagnostics ➜ Logarchives

Log writer



It can be useful to write an entry into the log. To do so in LogUI, open the **Log Writer** window using its command in the **Window** menu. Compose its text message in its editor, and when you want to write that as the message in a log entry, click on the **Write Text To Log** tool.

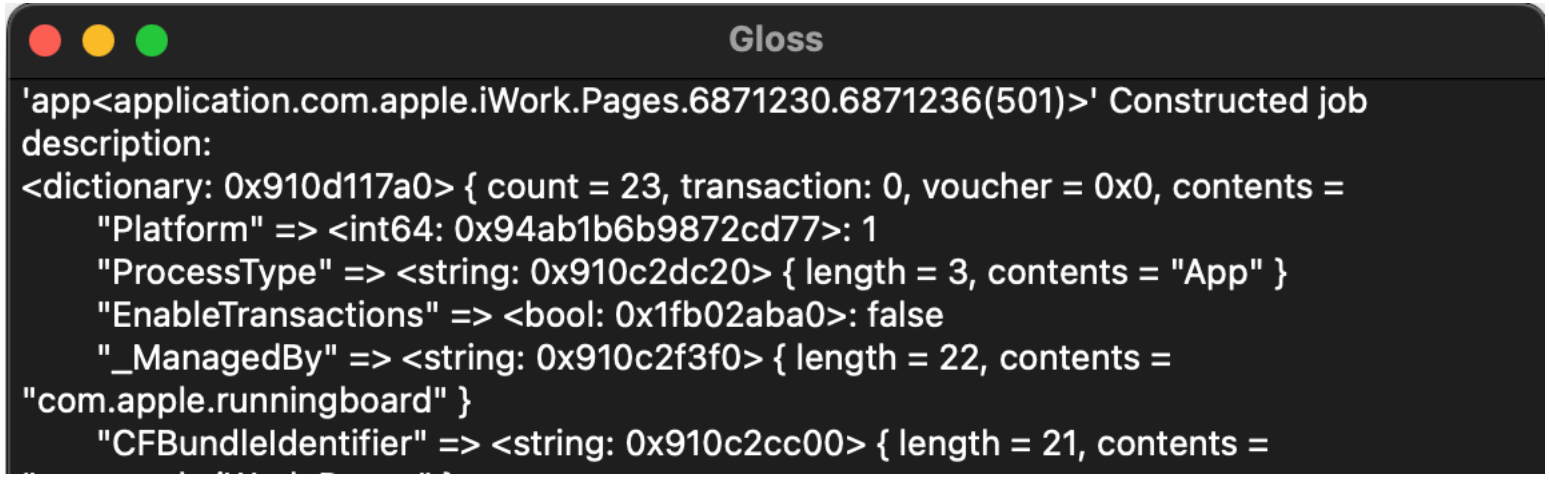
```
2026-10-26 08:35:16.057963+0000 20106859 0 AppKit LogUI 12443 3203281 sendAction:
2026-10-26 08:35:16.057975+0000 20106860 0 AppKit LogUI 12443 3203281 sendAction:
2026-10-26 08:35:16.057986+0000 20106861 0 AppKit LogUI 12443 3203281 sendAction:
2026-10-26 08:35:16.058179+0000 20106861 LogNote info LogUI.debug.dylib LogUI 12443 co.eclecticlight.LogUI 3203281 This is some test text for LogWriter.
and a second line
```

That entry should follow a series of three *sendAction:* messages from AppKit, and is marked as a *LogNote* from the *co.eclecticlight.LogUI* subsystem, with the text as its message field.

Text in the Log Writer window is saved in LogUI's preferences, so persists.

➜ Settings General ➜ Settings Predicate ➜ Tools ➜ Log settings ➜ Predicates ➜ One-off Predicate Editor
➜ Log display ➜ Log entries ➜ Search ➜ Notes ➜ Gloss ➜ Diagnostics ➜ Logarchives

Gloss



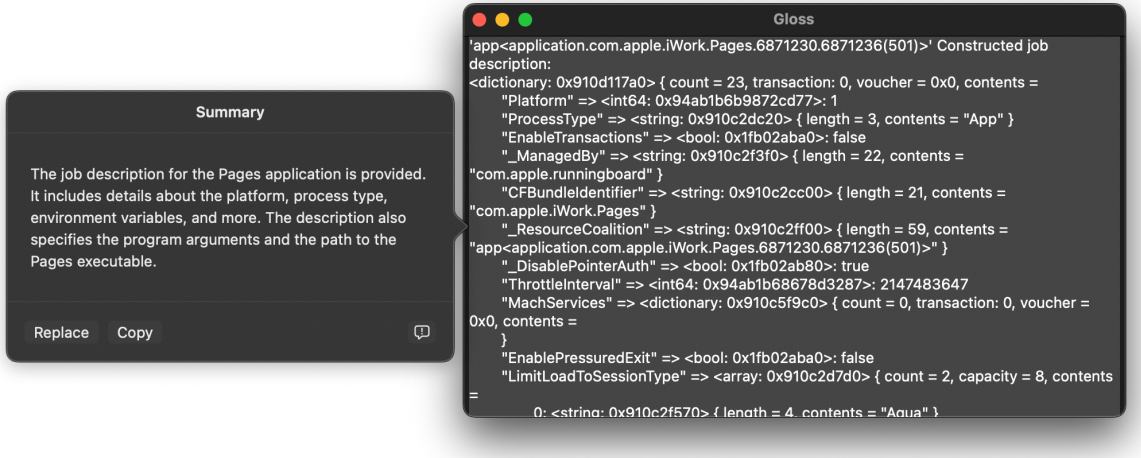
To view, edit, analyse and copy just the message contents from selected entries, click on the **Gloss** button in the toolbar.

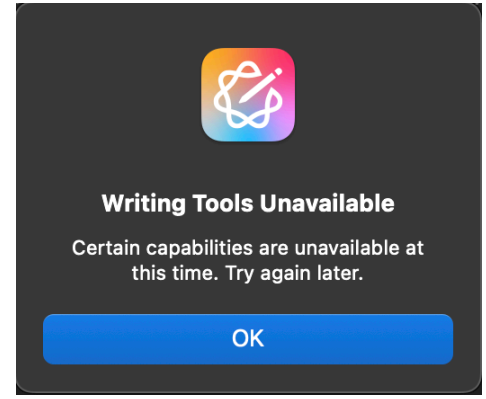
Those messages will then be displayed as plain text in the Gloss window, where you can edit them, copy and paste them into any text editor, and use **Writing Tools** to summarise them (Apple silicon Macs only). Gloss text persists, and is saved into LogUI's preferences.

Only one Gloss window is available, and it contains the text from those entries selected the last time you clicked on the Gloss button. When you do that, the new gloss replaces the old. If you want to save the previous gloss, copy and paste it before clicking on the **Gloss** button.

➜ Settings General ➜ Settings Predicate ➜ Tools ➜ Log settings ➜ Predicates ➜ One-off Predicate Editor
➜ Log display ➜ Log entries ➜ Search ➜ Notes ➜ Log writer ➜ Writing Tools

Writing Tools

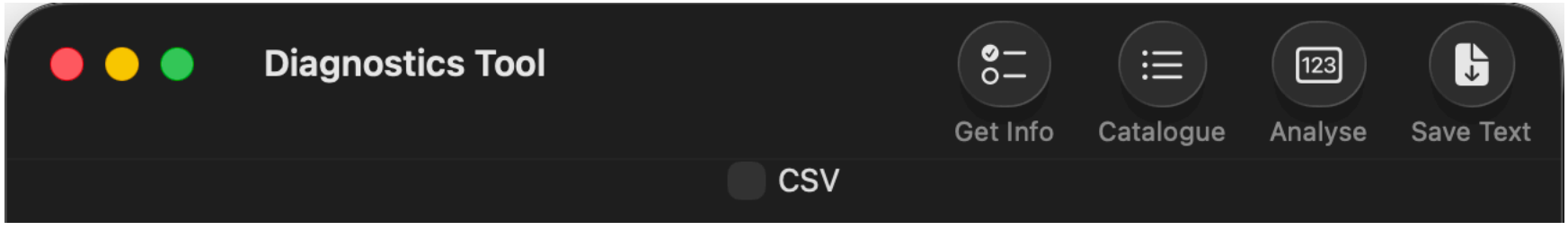




On Apple silicon Macs, you can use Writing Tools to summarise messages in the **Gloss** window. Select all the text there (Command-A) and either use the Writing Tools blue icon or the contextual menu to apply the tool of your choice. Most useful are **Summarise**, **Create Key Points** and **Make List**. If one of the tools can't perform its task, it may report that Writing Tools are Unavailable; try a different tool instead.

Writing Tools can't interpret log entries or diagnose problems from the log, but its summaries can make complex lists more comprehensible, and sometimes provide additional information that can be helpful. Always check its output against the original, as it does make mistakes.

Diagnostics Tool



Live logs are stored in the /private/var/db/diagnostics folder of the Data volume and backups made of it. This window, opened using the **Diagnostics Tool** command in the **Window** menu, provides a suite of tools for checking and analysing the diagnostics folder on any volume.

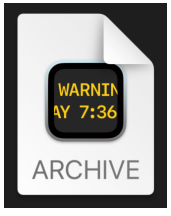
Tools provided are:

- **Get Info** provides basic information about how many log files there are in each folder, and how far the Persist (general) log records go back in time;
- **Catalogue** lists all the tracev3 log files in each of the folders inside diagnostics;
- **Analyse** provides statistical analyses of the contents of all log files;
- **Save Text** saves the current contents of the window as a text file, or in CSV format if that is ticked.

These are similar in function and contents to their equivalents for logarchives.

➜ Settings General ➜ Settings Predicate ➜ Tools ➜ Log settings ➜ Predicates
➜ One-off Predicate Editor ➜ Log display ➜ Log entries ➜ Gloss ➜ Logarchives

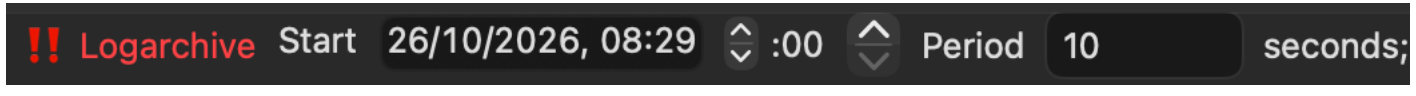
Logarchives



Because the log contains many files, there's a standard way to save them all in a package termed a *logarchive.* Once saved, you can access the whole log for that moment in time, at any time in the future. To make a logarchive, use the `log collect` command, as in

```
log collect --output ~/Documents/my.logarchive --last 5m
```

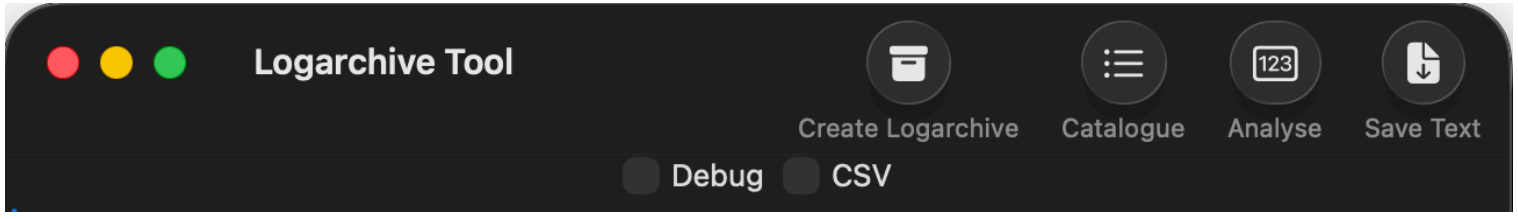
or run a `sysdiagnose`, or use LogUI's Create Logarchive tool. LogUI can read and use logarchives through its **Use Logarchive** tool. Click on that and select the logarchive, and that will be used as the source of all its log extracts. This also works with logarchives obtained from iOS and iPadOS (and others).



To remind you that window is set to access a logarchive, rather than the active log, and requires appropriate date and time settings for its entries, a red warning is displayed to the left of the **Start** time settings. To resume accessing the active log in the same window, click on the **Use Logarchive** tool again, and then click its **Cancel** button.

In the Window menu, open the **Logarchive Tool** to create and explore logarchives.

Create Logarchives



If you have access to the folders containing log files from a Mac or Apple device, you can use the **Create Logarchive** tool in the Logarchive Tool to turn those into a logarchive. First create a new folder somewhere like ~/Documents, and copy the whole folder at /var/db/diagnostics into that, together with that at /var/db/uuidtext. Click on **Create Logarchive** and select that folder, and LogUI will do its best to turn that into a logarchive. If you do this with the **Debug** checkbox ticked, you'll get a detailed diagnostic account in the Logarchive Tool window, similar to that below.

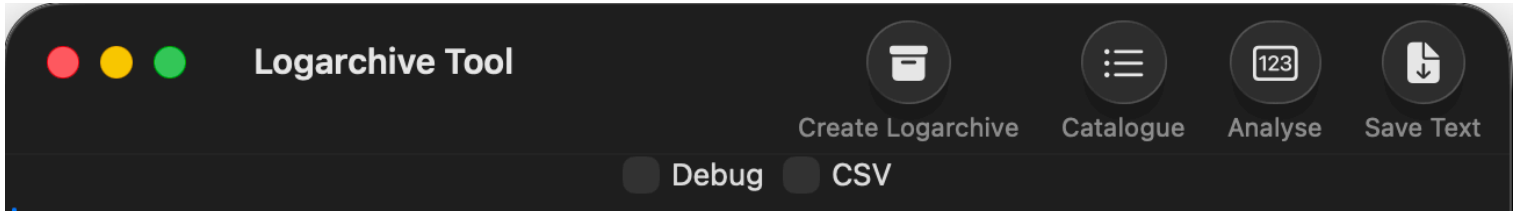
```
Source: /Users/hoakley/Documents/4logarchive
Destination:/Users/hoakley/Documents/4logarchive/4logarchive4.logarchive
copying from /Users/hoakley/Documents/4logarchive/uuidtext to /Users/hoakley/Documents/4logarchive/4logarchive4.logarchive
copying from /Users/hoakley/Documents/4logarchive/diagnostics/Persist to /Users/hoakley/Documents/4logarchive/4logarchive4.logarchive/Persist
copying from /Users/hoakley/Documents/4logarchive/diagnostics/Special to /Users/hoakley/Documents/4logarchive/4logarchive4.logarchive/Special
copying from /Users/hoakley/Documents/4logarchive/diagnostics/Signpost to /Users/hoakley/Documents/4logarchive/4logarchive4.logarchive/Signpost
copying from /Users/hoakley/Documents/4logarchive/diagnostics/HighVolume to /Users/hoakley/Documents/4logarchive/4logarchive4.logarchive/HighVolume
```

Although this is generally reliable, as the logarchive format is undocumented, it could fail in some cases, hence the option for Debug output. You can save that to a text file using the **Save Text** tool.

Check and analyse the new logarchive using Catalogue and Analyse tools.

➜ Settings General ➜ Settings Predicate ➜ Tools ➜ Log settings ➜ Predicates
➜ One-off Predicate Editor ➜ Log display ➜ Log entries ➜ Gloss ➜ Diagnostics ➜ Logarchives

Catalogue and Analyse Logarchives



To help you get their dates and times right, and provide other useful information about a logarchive, use the **Catalogue** and **Analyse** tools in the Logarchive Tool. **Catalogue** lists all the tracev3 log files in the logarchive, giving you their dates and times, file sizes and approximate periods in minutes.

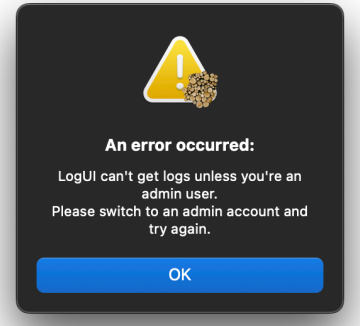
```
Catalogue listing of /Users/hoakley/Documents/4logarchive/4logarchive2.logarchive :

Persist/0000000000000001.tracev3 2024-11-01 07:50:42 +0000  2024-11-01 07:53:23 +0000  10436856 bytes, period 2.7 min
Persist/0000000000000002.tracev3      2024-11-01 07:53:23 +0000  2024-11-01 07:53:28 +0000  10396936 bytes, period 0.1 min
Persist/0000000000000003.tracev3      2024-11-01 07:53:28 +0000  2024-11-01 07:57:23 +0000  2342760 bytes, period 3.9 min
Persist/0000000000000004.tracev3      2024-11-08 15:52:27 +0000  2024-11-08 15:53:57 +0000  10411424 bytes, period 1.5 min
Persist/0000000000000005.tracev32024-11-08 15:53:58 +0000  2024-11-08 15:58:25 +0000  10473288 bytes, period 4.4 min
```

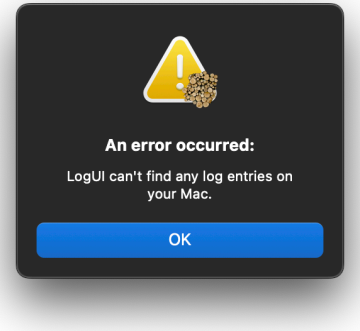
Analysis additionally provides the percentage of entries from the most common processes in each persist log file. This is provided in plain text, or when the CSV checkbox is ticked, in CSV format for importing into a spreadsheet. Save this as a text file using the **Save Text** tool. See also how to create a logarchive.

```
date,time,totalsize,filename,size,percent,process
2024-11-01,00:53:23,59778620,0000000000000001.tracev3,51923352,86.9,audiomxd
2024-11-01,00:53:23,59778620,0000000000000001.tracev3,1500932,2.5,airportd
2024-11-01,00:53:23,59778620,0000000000000001.tracev3,1132616,1.9,kernel
2024-11-01,00:53:23,59778620,0000000000000001.tracev3,1029554,1.7,launchd
2024-11-01,00:53:23,59778620,0000000000000001.tracev3,472560,0.8,trustd
```

Errors



When LogUI launches, it performs some basic checks to confirm that it can obtain log extracts. These require that it's run from an **admin user account**; if you try running it as a normal user, you'll see the alert above and the app will quit when you close that. It also checks there are log files that can be accessed; if there aren't any, or there are problems with them, you'll see the alert below before it quits.



Technical Information

The log engine uses OSLog's `OSLogStore.getEntries()` to fetch log entries for parsing into fields according to OSLogEntry types. See this article for further details.

The array of parsed log entries is then displayed in a SwiftUI List of coloured Text fields. The requirement for Sonoma or later is to support the use of the log engine to be `Observable`; older equivalents are too clumsy and pose memory management issues, particularly with large log excerpts.

LogUI's JSON files are of type `co.eclecticlight.loguilog` and extension `.logui`. and can be used as a regular JSON export with other apps capable of importing JSON format.

All text entered as a predicate is checked to ensure that it only uses plain double quotation marks " as required in predicates. Because of the propensity of SwiftUI view to silently convert those to styled marks “ ”, which cause errors in predicates, those are all replaced in predicate strings using plain marks. This does have the side-effect that, should you want to include styled marks in a predicate, you can't. However I'm not aware of any circumstance where they might be required.

The currently primitive predicate editor in Settings Predicate is the result of limitations in SwiftUI and my failure to find a better workaround. When I have a better solution that provides full support for editing the contents of the predicate menu, I will release an update. Please bear with me.

Change list

1.3 (95):

- replaced seconds control with stepper
- adjusted layout of other controls
- added Log Writer tool to write log entries on the fly
- *failed* to fix time zone change bug.

1.2 (91):
- fixed macOS 27 focus problem with log list, thanks to Philip Savoulidis.

1.1 (90):
- cosmetic fix to search menu for macOS 27
- rebuilt using Xcode 27.0.

1.1 (88):
- added new user notes
- fixed tool icons in older versions of macOS.

1.0 (77):
- changed displayed type representation to emoji
- added signpostName field to copied data
- improved RTF export.

1.0 (74):
- added Diagnostics Tool
- updated Help book for Tahoe's new look.

1.0 (70):
- adjusted the width of the seconds field to allow for Tahoe's spaciousness.

1.0 (68):
- rebuilt for Tahoe, with new app icon and spruced views.

1.0 (65):
- added Logarchive Tool.

1.0 (60):
- added logarchive access.

1.0 (58):
- added Read and Save as JSON, and Reduce tools
- improved window and file titles.

1.0 (52):
- all log entry times are now given using microseconds, and in the same time zone.

1.0 (48):
- all log entry times are now given using nanoseconds.

1.0 (46):
- added Gloss and AI support
- moved action buttons to toolbar
- added active search to window title.

1.0 (42):
- added Search feature.

1.0 (37):
- augmented predicate menu
- added one-off predicate editor
- added basic preferences predicate editor, with tabbed Settings.

1.0 (31):
- added initial tests for presence of and access to logs
- added predicate menu with support for more filter predicates.

1.0 (27):
- added support for Signposts in RTF files
- changed the Settings dialog to use *Full Fields* for consistency
- updated technical info with link to original source code.

1.0 (25):

- added support for Signposts
- subsystem name is now case-insensitive
- added support for discontinuous selection of log entries
- added support for copying text from log entries
- window name changes to reflect the start of each log excerpt
- RTF saved file name changes to reflect the start of that log excerpt.

1.0 (20):

- first prototype release.

26 October 2026.