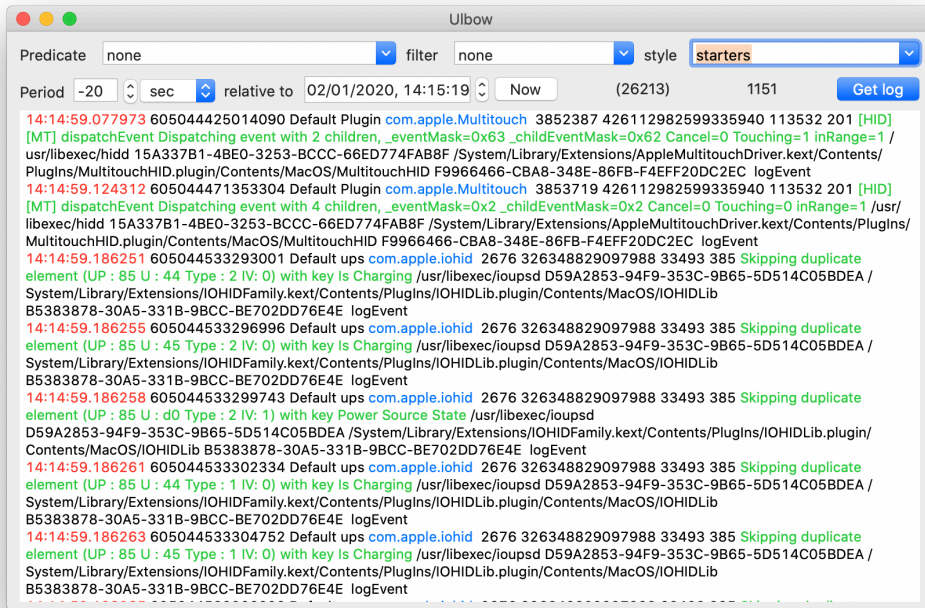


Start

Ulbow gives unrivalled access to the macOS unified log introduced in Sierra. Unlike Console, it analyses entries which have already been written to the log. Unlike the `log show` command in Terminal, Ulbow works with full log extracts in JSON format, showing you exactly what you want to know, and exporting in styled Rich Text. Its graphical interface makes the log accessible to all users, and makes intensive log analysis quick and simple – easier and less forbidding than even Consolation 3, with the power of its charts too.

[→ Contents](#)[→ Main window layout](#)[→ Check Time Machine](#)[→ Inspect a crash, fault or event](#)

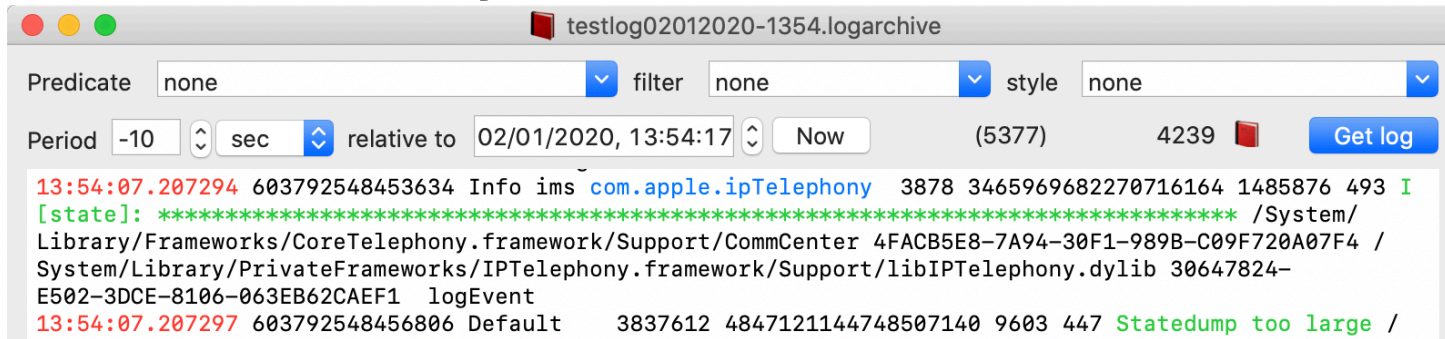
→ [Start](#)

→ [Contents](#)

Contents

- [Main window layout](#)
- [Check Time Machine](#)
- [Set the source of the log](#)
- [Set the filter to determine which entries to show](#)
- [Set the period of log collection](#)
- [Chart views](#)
- [Custom predicates](#)
- [Custom styles](#)
 - [Style definitions](#)
 - [Available fields](#)
- [Custom filters](#)
- [Save log extracts](#)
- [Signposts](#)
- [Cause codes](#)
- [Check log](#)
- [Create logarchives](#)
- [Updates](#)
- [Change list](#)
- [First time setup](#)
- [Inspect a crash, fault or event](#)
- [Set the entries to be extracted](#)
- [Set the style of log content](#)
- [Get log](#)
- [Preferences and settings](#)
- [Save as CSV for a spreadsheet, etc.](#)
- [Navigation](#)
- [Controlling log size](#)
- [Analyse logarchives](#)
- [Further information](#)

Main window layout



Ulbow can open many windows, each containing different log extracts, and with different settings and options. The top of each window is structured into sections:

- at the top left, → [set up the predicate](#) to determine which log entries will be included
- at the top centre, → [set the filter](#) to determine which log entries to show from the extract
- at the top right, → [set the style for display](#) of the log entries
- at the left of the second row, → [set the time period](#)
- at the far right of the second row, click the **Get log** button → to [get and view](#) that log extract.

Below those, in the main part of the window, is the log extract obtained with those settings.

The top row 'Combo' controls are a combination of popup menu and text box: select an item from the popup menu, or type in custom text as you wish.

Two numbers are shown in the right side of the second row. These give first the total number of log entries in the current extract, and the number of lines in the excerpt below. When you have **Limit entries shown** (in the **View** menu) enabled, the number of log entries is shown in parentheses (), and the number of lines should be close to the limit you have set.

At any time, you can change the size of the font used in the log extract view: ⌘+ makes it larger up to a maximum of 24 points, and ⌘– makes it smaller down to a minimum of 4 points. This setting is remembered for when you open the next window, and when opening Ulbow next. The font used is the system monospace font, which is even better in Catalina.

First time setup

You no longer need to set up Ulbow's preferences before opening it for the first time on any Mac, as it automatically detects whether there's an existing set of preferences available. If there isn't, Ulbow creates a default set for you to use.

If you already have custom preferences on another Mac, you can export those to a property list file, copy that to the Mac on which you want to add those preferences, then import them to the Mac on which you are now using Ulbow.

If you ever want to completely reset all Ulbow's preference settings, including those for font size and auto-updating, quit Ulbow and use the `defaults delete co.electiclight.Ulbow` command to remove its current preferences:

```
defaults delete co.electiclight.Ulbow
```

When you open Ulbow next, it will have returned to its default preference settings. This is much more reliable than trying to delete its preferences file.

Each that you open Ulbow, it checks that Mac's logs. If there are any problems, it will warn you of them, and when possible explain how to fix them.

→ [Preferences and settings](#)

Check Time Machine

To browse Time Machine entries for the last 2 hours:

- set the *Predicate* to **TimeMachineBasic**
- leave the *Filter* set to **none**
- leave the *Style* set to **none** or a style of your choice
- set the *Period* to **-2 hour**
- then click on the **Get log** button.

This is simpler in my free utility **The Time Machine Mechanic** (T2M2), of course, which even interprets the log entries for you. To view all the entries in that period of 2 hours, you may need to disable **Limit entries shown** in the **View** menu if the number of log entries exceeds that limit.

→ [Inspect a crash, fault or event](#)

Inspect a crash, fault or event

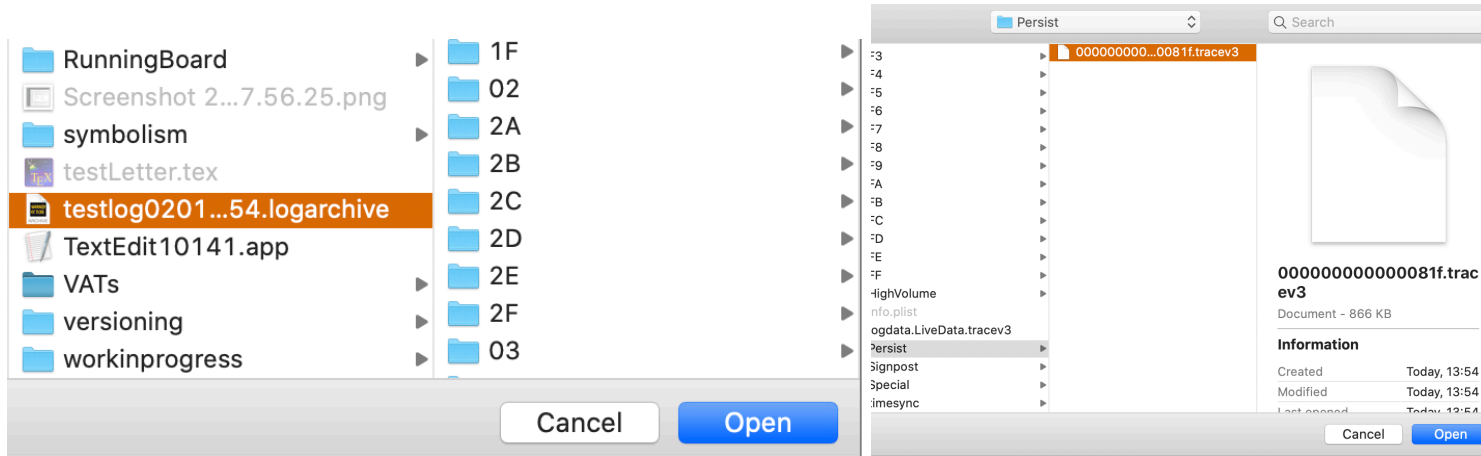
Make a note of the clock time (on your Mac) at which the event occurred. In Ulbow:

- set the *Predicate* to **none**
- leave the *Filter* set to **none**
- leave the *Style* set to **none**, or use a style such as **starters+** supplied in the additional Property List
- set the *Period* to **–20 sec**, then in the **relative to** box enter a date and time about 10 seconds *after* the event occurred
- then click on the **Get log** button.

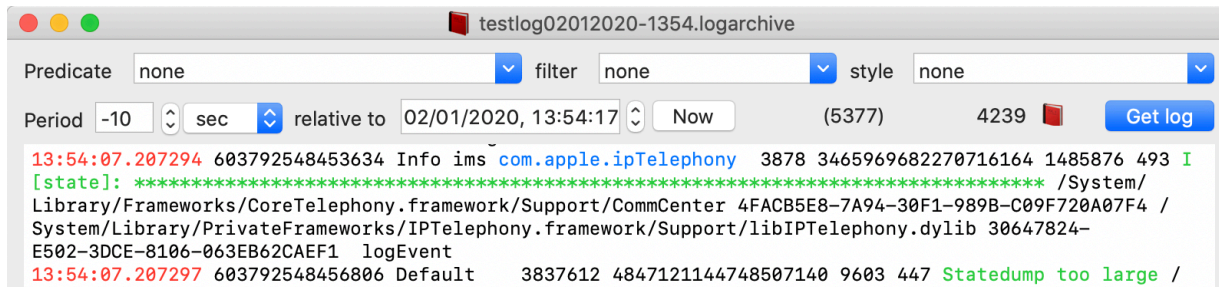
Scroll through the entries, which include everything written to the log over that 20 seconds, and you should be able to see that event occurring, whatever led up to it, and its consequences. To view all the entries in that period, you may need to disable **Limit entries shown** in the **View** menu. If the total number of log entries is more than 10,000, the displayed log extract may then be excessively large and make performance sluggish; consider using a **Predicate** and/or **filter** to reduce the number displayed.


→ [Check Time Machine](#)

Set the source of the log



When you open a **New** window, Ulbow defaults to accessing the active system log. To examine a previously-saved logarchive in an open window, use the **Open logarchive...** command in the **File** menu, then select the logarchive, or an individual log file, with the extension `.tracev3`, within a logarchive bundle.



When a window is browsing a logarchive or log file, a  red book emoji is shown to the left of the Get log button, and the title of the window shows the name of the log source, with the same emoji prefixed to it.

→ [Set the source of the log](#) (concluded)

Set the source of the log (concluded)

Ulbow cannot examine isolated tracev3 files, because of limitations imposed on the `log show` command.

Ulbow also opens and browses logarchives written from iOS devices, Apple Watch, and Apple TV, and those obtained from macOS using the `log collect` command in Terminal.

⚠ Ulbow uses the `log` command provided by the version of macOS on which it is running, and is therefore limited by that command's abilities to access logarchives and tracev3 files. The version of `log` provided with Sierra *cannot* open logarchives or tracev3 files created by Mojave. To analyse logs from Mojave, use Ulbow running on Mojave or later. However, Ulbow running on Mojave can access logs written by Sierra and High Sierra, as well as Mojave itself.

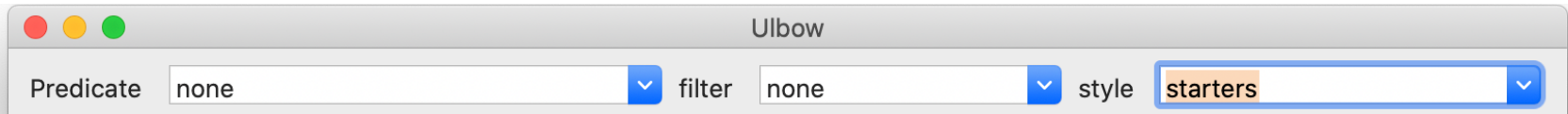
To save the current active log on that Mac as a logarchive, first set the **Period** of log to be captured, including the units of time, then select the **Write logarchive...** command in the File menu. You will be prompted to give the name and location of the logarchive to be written, and finally to authenticate as an admin user in order to do so, as the command has to be run as root. ⚠ The filename and path *mustn't* contain single or double quotes, a limitation of using AppleScript to call the command.

⚠ Catalina's new privacy protection limits where you can save the resulting logarchive, which can be larger than 1 GB in size. You shouldn't have any problems saving the logarchive to your ~/Documents folder, or folders within that. You should also be able to save to other folders to which you have appropriate permissions on your startup volume. Normally, you can't save direct to any external volumes, but can of course copy the logarchive to them using the Finder.

Additional features for creating and working with logarchives are now available in the → [Logarchive Tool](#)

→ [Set the entries to be extracted](#)

Set the entries to be extracted



Ulbow offers three controls to specify which entries in the log will be obtained in a log extract:

- select a **Predicate** to browse only entries which satisfy its conditions, or type a predicate into the text box to use that instead
- in the **View** menu, the **Get info messages** command determines whether entries of messageType **Info** will be included
- in the **View** menu, the **Get Signposts** command determines whether entries of messageType **Signpost** will be included (*not macOS Sierra*)
- in the **View** menu, the **Get Debug** command determines whether entries of messageType **Debug** will be included.

When you have changed any of these controls, you'll need to click on the **Get log** button again so that Ulbow can extract those entries afresh from the log.

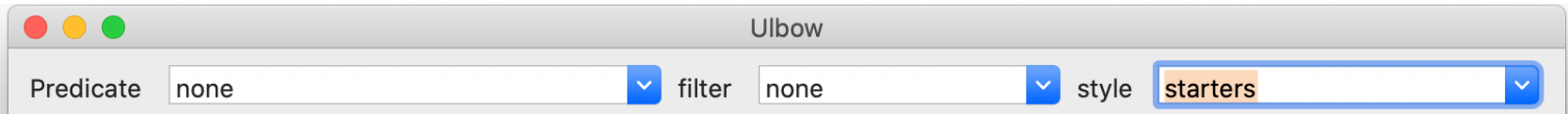
The **Predicate** 'Combo' works as both a popup menu to select from the available predicates, and as a text box into which you can type a custom predicate. If you opt for the latter, your text must give the predicate to be used in a form similar to the predicate box in the Preferences window, e.g.
`eventMessage CONTAINS[c] "error"`

→ [Custom predicates](#)

→ [Set the filter to determine which entries to show](#)

→ [Set the style of log content](#)

Set the filter to determine which entries to show



Ulbow offers five ways to specify which entries in the log extract will be shown in the view below:

- select a **filter** to browse only entries which contain the specified text in the eventMessage field, or type a filter string into the text box to use that instead
- in the **View** menu, the **Show log entries**, **Show Signposts** and **Show Activities** commands set whether to show log entries. *Show Signposts isn't available in macOS Sierra.*

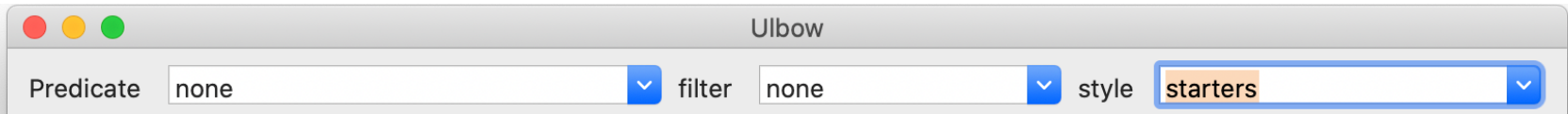
When you change any of these controls, they don't require you or Ulbow to obtain a fresh log extract, as they're applied immediately to the current log entries. When you have relatively few entries in your current log extract, or **Limit entries shown** enabled, your change should be reflected in the log extract displayed almost immediately. With larger numbers of log entries, over 10,000, then this will take longer, and with more than 50,000 it will take many seconds or even minutes.

To try out a filter on the current log extract, select any item already in the popup menu, or type the filter string without quotation marks into the **filter** text box and press **Tab** or **Enter**.

→ [Custom filters](#)

→ [Set the style of log content](#)

Set the style of log content



Ulbow offers one basic style which is the default, named **none**, and as many **custom styles** as you want. In those you choose which of the available fields to display, in any order you wish, with the addition of three colours plus black/white displayed according to the Light/Dark Mode setting. If you install the custom Property List supplied, this offers minimal styles like **basic**, and the more inclusive **starters+**.

The default style is not dissimilar to traditional syslogs. Each log entry is displayed as
the timestamp without any date
the processUniqueID as a number
the whole contents of the eventMessage.

At the other extreme, starters+ contains the full date and timestamp in black/white, the messageType in green, the subsystem in blue, process and sender image paths truncated, another three fields, and the whole eventMessage.

To try out a style on the current log extract, select any item already in the popup menu, or type its format string without quotation marks into the **style** text box and press **Tab** or **Enter**.

→ [Set the period of collection](#)

→ [Custom styles](#)

Set the period of collection

Period relative to (26213) 1151

Ulbow offers a simple and efficient method of specifying 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.

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 later, shorter periods can be used, even below 10 seconds if you wish.

If you may a terrible mistake and accidentally **Get log** for all entries over a period of 2 hours, for example, which could be millions of entries, just close the window and you should be able to try again in a new window.

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.

→ [Get log](#)

Get log

When you've set the sections above, click on the **Get log** button to run the `log show` command, obtain and display the log extract. If that command returns an error, that should be displayed. If you can't get any log entries when you expect them, or see an error, use the → [Check log](#) feature to discover what's wrong.

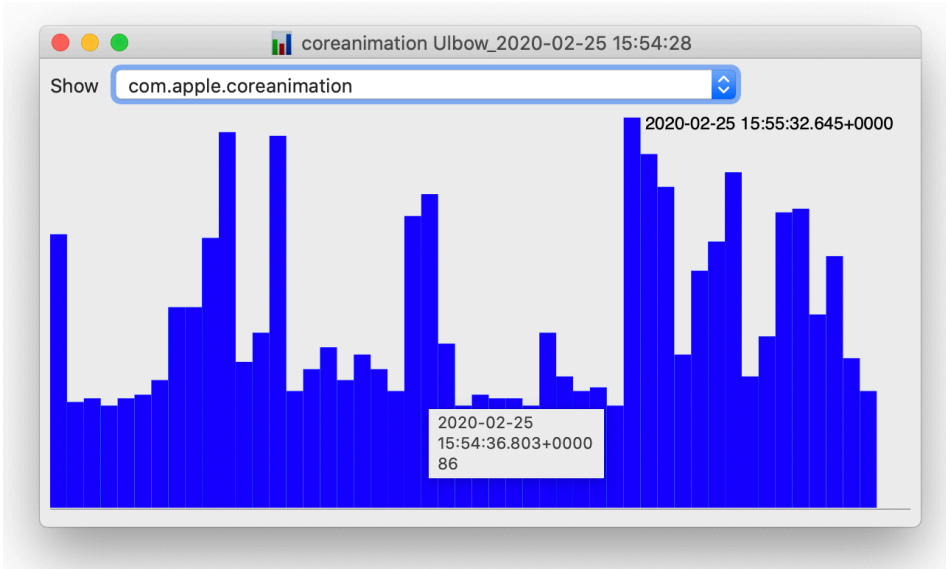
You can search the log extract shown in the scrolling view at the bottom using the normal **Find** command in the **Edit** menu. Note that that doesn't make any distinction between the fields of each entry, but searches the whole text on display. The filter setting is limited to the contents of the message field in the log extract.

The text near the **Get log** button provides you with information about your log extract: the total number of log entries, and the number of lines of text. It is easy to think you are only working with a few thousand lines, when there can be over 100,000. Use the **Limit entries shown** command in the **View** menu to start small and refine your settings until you're getting a more reasonable number of log entries displayed.

Once you've got a log extract, use the **Open Chart** command in the **Window** menu to look at the frequency and subsystems of log entries in the whole of your log extract.

→ [Chart views](#)

Chart views



Once you've got a log extract to browse, you can open a Chart view using the **Open Chart** command in the **Window** menu, which shows the frequency of log entries from different sub-systems, including the kernel, over the period covered by the extract. This *isn't* restricted to the normal limit on entries, so its analysis covers the whole period. When you're working with very large extracts, and don't need charts, turn them off using the **Enable Chart View** command in the main app menu.

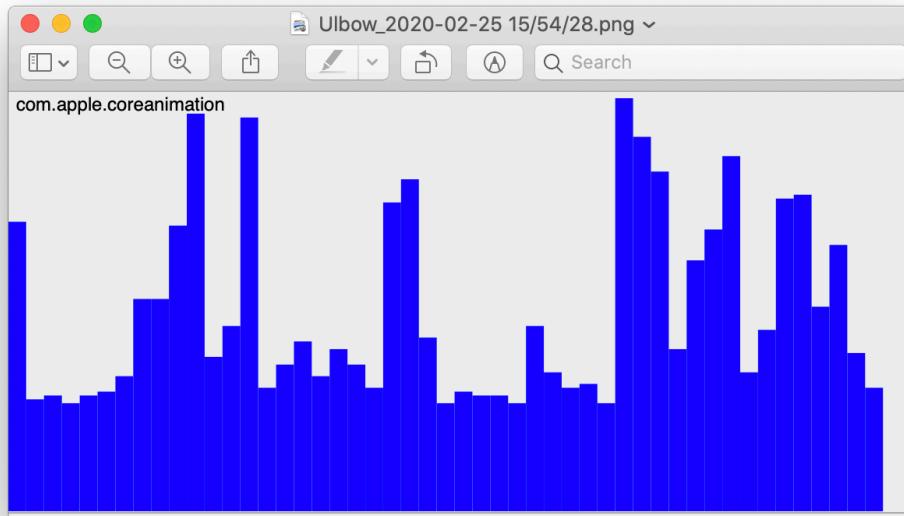
The **Show** popup menu at the top lets you view entries by subsystem, and hovering the pointer over a bar reveals **tooltips** which give the date and time of the start of that period, followed by the number of entries for that subsystem. Split the frequencies into finer periods by **clicking** on the view, or make them coarser by **right-clicking** instead.

→ [Chart views \(concluded\)](#)

Chart views (concluded)

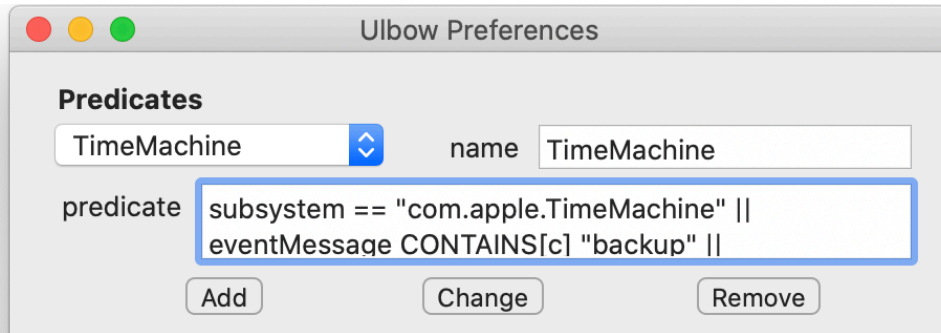
Displayed at the top right of the Chart is a date and time. Initially, it displays the time at the start of the chart, but **Command-click** on one of the bars and it shows the time at the start of that bar. It automatically copies that date and time, which can then be pasted into the **relative to** control in any of Ulbow's main windows. This provides a convenient way to examine a shorter period of time from within a given log excerpt:

- identify the period that you're interested in from the Chart, **⌘-click** on the first bar of that period, open a **New** document window and paste the time into that, set the **Period**, and **Get Log**.



You can also save a Chart in PNG format using the **Save PNG...** command in the **File** menu. The resulting document doesn't display the date and time from the upper right, but instead displays the selection in the **Show** popup menu in the upper left. The PNG is set to the same size as the window at the time the chart is saved.

Custom predicates



Ulbow can store custom predicates for you, which are ideal for those which you use often. To add and edit custom predicates, use the **Preferences...** command in the main app menu to open the Preferences window.

The first item in that menu should always be **none**. You can add any other predicates which you wish, to make them instantly accessible. They can also contain multiple terms which go beyond anything that you can currently achieve in Ulbow's own filters.

To add a new predicate, open the popup menu in Preferences and select the **New...** item at the end. In the **name** box, give that predicate a suitable name, which will appear in the popup menu. In the **predicate** box, enter the text of the predicate to be applied. Note that strings *must* be placed within plain double quotes `""`. You can enter any predicate which would be acceptable for use in the `log show` command. The following works, for example:

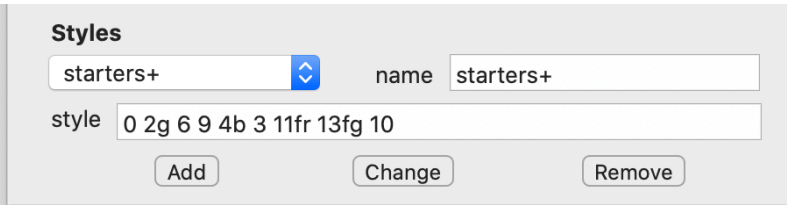
```
subsystem == "com.apple.duetactivityscheduler" || subsystem == "com.apple.xpc.activity" || (subsystem == "com.apple.TimeMachine" && eventMessage CONTAINS[c] "start")
```

When your new entry is ready, click on the **Add** button to add it to the popup menu, and continue to add the predicates which you want to store. To change a predicate already in the menu, select that menu item and its **name** and **predicate** will be loaded ready. Use the Tab or Enter key to finish editing, then click on the **Change** button. If you don't click on the button, your changes will be ignored. To remove a predicate from the menu, select it in the menu, then click on the **Remove** button.

Once you have finished editing your predicate library, simply close the Preferences window and those new items will be saved to Ulbow's preferences, and immediately available in all windows.

→ [Custom styles](#)

Custom styles



Ulbow can store custom styles for you, which are ideal for those which you use often. To add and edit custom styles, use the **Preferences...** command in the main app menu to open the Preferences window.

To add a new style, open the popup menu in Preferences and select the **New...** item at the end. In the **name** box, give that style a suitable name, which will appear in the popup menu. In the **style** box, enter the definition of the style to be applied.

When your new entry is ready, click on the **Add** button to add it to the popup menu, and continue to add the styles which you want to store. To change a style already in the menu, select that menu item and its **name** and **style** will be loaded ready. Use the Tab or Enter key to finish editing, then click on the **Change** button. If you don't click on the button, your changes will be ignored. To remove a style from the menu, select it in the menu, then click on the **Remove** button.

Once you have finished editing your style library, simply close the Preferences window and those new items will be saved to Ulbow's preferences, and immediately available in all windows.

When you have built your own library of styles, you will get used to switching quickly between them to help you browse a log extract. *The addition of colour makes a very substantial difference.*

When you **Save** a log extract, you have the option of saving it as plain text, or as Rich Text (RTF). If you opt for RTF, colours displayed in Ulbow's window will be preserved in that file.

→ [Style definitions](#)

Style definitions

A style, at its most basic, simply lists the different fields to be shown in the log excerpt, in the order in which they are to be shown. A popular minimal style might consist simply of three integers separated by single space characters:

```
0 2 10
```

This will result in the display, in sequence, of the timestamp (field 0), messageType (2), and eventMessage (10). You can display fields in any order that you wish.

The timestamp field can be displayed in 3 variants: as it comes, e.g.

```
2017-07-26 19:47:54.951146+0100
```

or you can add a formatting character of h or d for just the time part of the timestamp, or the timestamp without the UTC shift. So using 0h would produce 19:47:54.951146 and 0d would produce 2017-07-26 19:47:54.951146

Other fields have two groups of formatting options:

- content options determine how much of the field is shown:
 - the default is to show the whole field
 - t shows only the first 20 characters in the field
 - T shows the last 20 characters
 - f shows all characters after the last slash / in the field. This specifically intended to eliminate very long pathnames in the processImagePath and sendImagePath fields.
- colour options set the colour of the text to be used:
 - the default is standard black/white text
 - r displays red
 - g displays green
 - b displays blue.

These can be used in any order and combination, provided that no more than one of the content options is used, and no more than one of the colour options, for any given field. So

```
0h 2g 6 9 4b 3 11fr 13fg 10
```

will display the timestamp without the date in black, the messageType in green, threadID and processID in black, the subsystem in blue, the category in black, the processImagePath truncated to just the file name in red, the senderImagePath truncated to just the file name in green, and the eventMessage in black. You cannot use 11tfr or 10trg, for example.

→ [Available fields](#)

Available fields

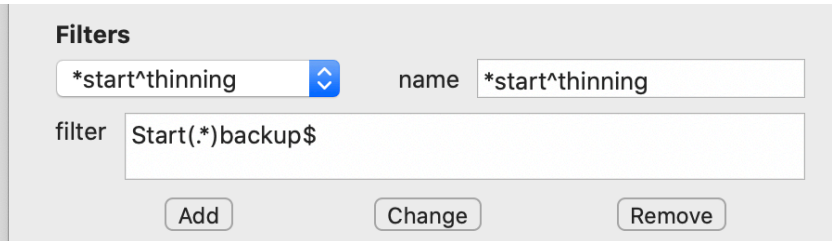
The full list of available fields and their numbers is:

- 0. timestamp, in full e.g. 2017-07-26 20:24:59.326229+0100, or with h or d format
- 1. machTimestamp, in system ticks, e.g. 608403543041193
- 2. messageType, e.g. Default
- 3. category, e.g. security_exception
- 4. subsystem, e.g. com.apple.securityd
- 5. processUniqueID, e.g. 156
- 6. threadID, e.g. 868
- 7. traceID, e.g. 833721519476834308
- 8. senderProgramCounter, e.g. 193733726
- 9. processID, e.g. 156
- 10. eventMessage, e.g. MacOS error: -67062, can usefully be truncated with t/T format
- 11. processImagePath, e.g. /usr/libexec/taskgated, can be truncated with t/T or f format
- 12. processImageUUID, e.g. 4F6F0B24-7A18-3AF9-853F-8F72F6C7D7C7
- 13. senderImagePath, e.g. /System/Library/Frameworks/Security.framework/Versions/A/Security, can be truncated with t/T or f format
- 14. senderImageUUID, e.g. 005E8C96-40B6-35E3-B58B-888A5F5957C2
- 15. timeZoneName, may be blank.
- 16. eventType, one of signpostEvent, activityCreateEvent, or logEvent (not Sierra)
- 17. activityIdentifier, e.g. 32688 (not Sierra)
- 18. parentActivityIdentifier, e.g. 0 (not Sierra)
- 19. creatorActivityID, e.g. 0 (not Sierra)
- 20. signpostID, e.g. 123 (not Sierra)
- 21. signpostName, e.g. LoopTest (not Sierra or High Sierra)
- 22. signpostType, one of begin, end, or event (not Sierra or High Sierra)
- 23. signpostScope, e.g. process (not Sierra or High Sierra)
- 24. formatString, the format string used to convert variable content into the output string, e.g. %{public}@ (not Sierra or High Sierra)
- 25. source, e.g. null. Currently, the only value is null, so Ulbow always generates the text null for this key (not Sierra, and in High Sierra this is field 21)
- 26. backtrace. This is a complex structured field in Mojave/Catalina only, and is currently ignored by Ulbow.

When running on Sierra, only fields 0-15 are accessible. This is a limitation imposed by the log command in that version of macOS. When running on High Sierra, fields 0-21 are accessible, although field 21 (source) always returns null. When running on Mojave or later, fields 0-25 are supported. To check which are available for the version of macOS on which Ulbow is running, use the **Fields help...** menu command in the **Help** menu.

→ [Custom filters](#)

Custom filters



The screenshot shows a 'Filters' window with two input fields. The 'name' field contains the text '*start^thinning' and has a small blue dropdown arrow to its right. The 'filter' field contains the text 'Start(.*?)backup\$'. Below the fields are three buttons: 'Add', 'Change', and 'Remove'.

In Ulbow, you select the log entries to be included in a log extract using a predicate which includes text search of the `eventMessage` field. However, to apply predicates you have to obtain a fresh log extract using **Get log**, which takes time, and predicates are limited in the types of search which they can implement.

Ulbow lets you specify a search string to be applied as a filter to the existing log extract, which is usually much quicker, often almost instantaneous, and supports two different types of search: *simple*, and using *regular expressions* (regex). To add and edit custom filters, use the **Preferences...** command in the main app menu to open the Preferences window.

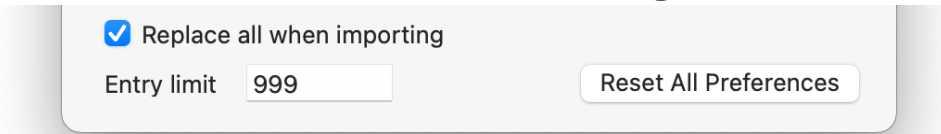
To add a new filter, open the popup menu in Preferences and select the **New...** item at the end. To add a simple filter, in the **name** box, give that filter a suitable name (which *mustn't* start with an asterisk `*`) to appear in the popup menu. In the **filter** box, enter the search string to be found using localised case-insensitive search. To add a regular expression for the filter, which is case-sensitive, ensure the first character of its name is an asterisk `*`, and type the regular expression into the filter box, as shown above.

When your new entry is ready, click on the **Add** button to add it to the popup menu, and continue to add the filters which you want to store. To change a filter already in the menu, select that menu item and its **name** and **filter** will be loaded ready. Use the Tab or Enter key to finish editing, then click on the **Change** button. If you don't click on the button, your changes will be ignored. To remove a filter from the menu, select it in the menu, then click on the **Remove** button.

Once you have finished editing your filter library, simply close the Preferences window and those new items will be saved to Ulbow's preferences, and immediately available in all windows.

→ [Preferences and settings](#)

Preferences and settings



Changes which you make in the Preferences window aren't just made available in the respective popup menus in the currently open window, but they're also saved automatically to the app's preferences file: you don't have to use the **Save as defaults** command in the Ulbow app menu to save changes made to custom predicates, filters, or styles.

You can return Ulbow's settings to the defaults at any time by clicking on the **Reset All Preferences** button. This doesn't affect font size or auto-update settings, though. This will overwrite any custom settings for predicates, filters, or styles; if you want to preserve those, export them first, as described below.

To help you move your custom predicates, filters, and styles between Macs, and to enable you to build libraries of those custom items for different purposes, Ulbow can export them to, and import them from, property list files. These features are accessed through two menu commands in the main app menu: **Import custom settings...** and **Export custom settings...** These work *only* with custom predicates, filters, and styles: if you want to move other settings which are saved as defaults to the app's preferences file, then you'll have to use that preferences file to do so. These commands are available when any document window is open in the app.

To export your current library of custom predicates, filters, and styles, use the **Export custom settings...** menu command. You'll then see a standard file save dialog, in which you should provide a name for the property list file, and locate it.

→ [Preferences and settings](#) (continued)

Preferences and settings (continued)

There are two options for importing custom settings from property list files, which are set by the **Replace all when importing** checkbox at the foot of the Preferences window. If that box is ticked (checked), then the custom settings which you import will replace *all* the existing custom predicates, filters, and styles. If that box is not ticked (unchecked), then your current custom settings will be merged with the imported ones. Imported items which have the same name as existing custom items (separately for each section) will not be imported; the only items which will be imported are those with different names from those currently in the list of custom predicates/styles/filters.

To import an existing library of custom settings from a property list file, use the **Import custom settings...** menu command. You will then see a standard file open dialog, in which you should select the property list to be imported.

Ulbow's preferences file is managed by macOS's defaults server `cfprefsd`, which means that you must be careful if you try to edit or delete it, as you may find that server defeats your purpose.

The property lists exported and imported for custom settings aren't managed as preferences files, and you are encouraged to edit them as you wish. For example, Ulbow doesn't provide any means of changing the order of items in the popup menus. Using any good text, XML or PList editor, you can alter an exported property list to set the menus as you wish, then import that file replacing all existing settings.

→ [Preferences and settings](#) (concluded)

Preferences and settings (concluded)

The property list should consist of the following:

- a dictionary containing one to three items
- those (up to) three items should be arrays, with the key values `specFiltList` (for filters), `specPredList` (for predicates), or `specStyleList` (for styles)
- each of those consists of an array of arrays, the innermost consisting of two strings
- the string pairs consist of the name (first element) and the value (second element)
- the first element for `specFiltList` and `specPredList` should consist of the menu item none:

```
<array>
  <string>none</string>
  <string></string>
</array>
```

Note that each property list file does not have to provide all three of `specFiltList`, `specPredList` and `specStyleList`. Indeed, when the **Replace all when importing** checkbox is ticked/checked, you can use this to your advantage: if the property list doesn't contain an item with the key value `specPredList`, for example, then existing custom predicates will not be replaced.

Not only can you assemble libraries of different suites of custom settings, but by careful use of their contents, you can have property list files which only contain custom predicates, for instance.

The other item in the Preferences window lets you set the maximum **Entry limit** which is used when Limit entries shown is enabled in the View menu.

Finally, the **Save as Defaults** command in the Ulbow app menu saves all your current settings *apart from the reference time* to Ulbow's references file.

Save log extracts

Ulbow offers two formats for saved log extracts: plain text (Unicode UTF-8), and Rich Text (RTF). You can also select and copy any text you wish from the displayed log extract.

If you want to perform further analysis of a log excerpt, then you should normally save it in plain text format.

If you want to browse the excerpt outside Ulbow, but do not wish to perform further analysis, then consider using RTF, which preserves the colours applied by the current custom style. The extension .rtf will be automatically supplied to the file name when you go to save it.

The RTF file written is compatible with display in Dark Mode. Use a Rich Text editor such as my free DelightEd, and it looks really great in Dark Mode.

To save the log excerpt in the frontmost window, use the **Save...** command in the **File** menu. This displays a standard file save dialog with the choice of formats, and the contents of the bottom panel will be saved to the text file that you specify there. Added to the end of the file is a timestamp for the time that the excerpt was made, and the command, e.g.:

```
At 2019-12-28 14:11:27 +0000 for command log show --predicate subsystem ==  
"com.apple.TimeMachine" --style syslog --info --last 3h
```

You can also convert your log extract and → [save it in CSV format](#) for importing into a spreadsheet or another app.

Save as CSV for spreadsheets, etc.

Ulbow can readily convert your current log extract into CSV format, which you can then Save as a text file and import into any spreadsheet and many other apps for further analysis. This works on the whole extract, ignoring any limit set on what is displayed. The only options which determine what is included are the **Show log entries**, **Show Signposts** and **Show Activities** commands in the View menu, and the Predicate used to select log entries for inclusion. Simply use the **Convert to CSV** command in the **File** menu to convert the entries, then **Save...** or **Save As...** a *text* file. You may need to change its extension to .csv to make import easier. To revert back to the normal styled display, reset the **Style** or use another control which reparses and displays the entries afresh.

CSV export text is written in compliance with IETF RFC 4180, using a comma as the only delimiter, and setting all fields other than numeric values inside double quotation marks "...". In particular, this means that any double quotation marks " within a field are converted to double-doubles "" and only appear in fields which are themselves set in double quotation marks. Because of its content, the `formatString` field is excluded from all CSV conversions.

Some apps which import CSV, notably Microsoft Excel, do not cope with newline characters in such fields, although they are permitted under the RFC. Accordingly, newlines are replaced with a single space character to make the CSV as compatible as possible with apps likely to be used for import.

⚠ Logs may contain some entries with very long, multiline `eventMessage` fields, which many apps will not import correctly. To work around this, following replacement of newline characters in the `eventMessage` field with single space characters, values of that field are truncated to a maximum of 255 Unicode characters, which drops the end of some `eventMessage` data.

Signposts

macOS High Sierra (at least by version 10.13.5), Mojave and later support an additional type of log entry: Signposts. Ulbow supports the additional keys available in their logs, and provides tools to help you use them by showing only certain types of log entry. *When running on Sierra, these tools aren't available, and the additional keys have no effect, as its log files and log command tool don't recognise them.*

To include Signposts when using Ulbow on High Sierra or later, ensure the **Get Signposts** command in the **View** menu is ticked before clicking on **Get log**. The log extract which is then obtained includes three types of log entry, determined by the `eventType` value: regular log messages, Signposts, and Activities. Sierra logs (and when running Ulbow on Sierra) only support regular log messages and Activities, and have limited discrimination between them.

Once you've fetched your log extract, use the command options in the **View** menu to change which entries are displayed. If you want to see only Signposts, for example, uncheck the **Show log entries** and **Show Activities** boxes, leaving the **Show Signposts** box ticked ✓. Those don't alter the *content* of the log extract, but change only the entries which are displayed.

To get the most out of Signposts, you will also want a custom style which shows their special key values, such as

```
0h 2 3g 4 16r 20b 21 22g 23
```

which is also valuable for viewing Activities. This is included in a new custom settings Property List which you can import into Ulbow to get you started.

To get the most from Signposts:

- capture and analyse your logs on High Sierra 10.13.5 or later
- ensure the **Get Signpost** command is ticked
- select a suitable custom style in the **Style** menu
- uncheck the **Show log entries** and **Show Activities** commands
- set up other controls to capture the Signposts you want to analyse.

Navigation

Start of the boot process

before about macOS Sierra 10.12.4 or .5: `BOOT_TIME` in the `eventMessage` field

after macOS Sierra 10.12.5: `=== system boot:` followed by the UUID of the startup event.

Shutdown

before about macOS Sierra 10.12.4 or .5: `SHUTDOWN_TIME` in the `eventMessage` field

after macOS Sierra 10.12.5 the last event logged has an `eventMessage` field containing

`=== system wallclock time adjusted`

Login process starts with an `eventMessage` from `loginwindow` of

`Login Window Application Started`

following which there is a series of entries from `SecurityAgent` which step through the login process. Once that is successful, `accountsd` will unlock and open keychains, although the most useful information is censored with `<private>`.

System wake

the kernel writes `PMRD: System Wake` marking the start of waking up. Another mark of the system becoming fully awake is when CTS logs `User Active` twice in succession..

System sleep

the kernel writes `gIOLastWakeAbsTime:` with the system ticks immediately before sleep

→ [Cause codes](#)

Cause codes

When a Mac starts up (including following a restart) or wakes from sleep, the reason for the previous shutdown (or initiation of a restart) or sleep is reported in the log. Unfortunately, Apple considers these reasons to be of no concern to the user or system administrator, so they are given as numbers, and Apple doesn't release information on what the numbers – cause codes – mean.

To view recent cause codes for your Mac, use a filter predicate like:

```
eventMessage CONTAINS[c] shutdown cause
```

```
eventMessage CONTAINS[c] sleep cause
```

You can OR them into a single predicate to view both. For more sophisticated filtering, their messageType is **Release**, senderImage is **AppleSMC**, and processImage is **kernel**.

You should then see entries similar to the shortened:

```
2017-02-26 22:13:34.210351+0000 kernel: (AppleSMC) Previous shutdown cause: 5
```

```
2017-02-26 11:52:19.006041+0000 kernel: (AppleSMC) Previous sleep cause: 5
```

Cause codes which are negative refer to hardware causes originating mainly from the SMC, and those which are positive refer to software. A special code of 0 indicates an intermediate, which can occur when there is sudden loss of power on some systems, or (for system sleep) when a laptop goes into SafeSleep to preserve its remaining reserve battery power.

→ [Cause codes](#) (concluded)

Cause codes (concluded)

Notable hardware cause codes:

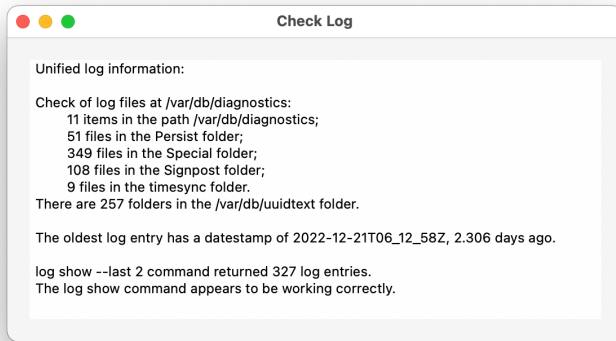
- -3 multiple temperature sensors too high
- -60 bad master directory block, serious disk error
- -61, -62 unresponsive app resulting in forced shutdown
- -64 kernel panic, probably due to firmware issue
- -71 memory too hot
- -74 battery too hot
- -75 MagSafe power adaptor communication problem
- -78 incorrect input current from power adaptor
- -79 incorrect current from battery
- -86, -95 proximity temperature (heatsink etc.) too high
- -100 power supply too hot
- -101 display too hot
- -103 battery voltage too low
- -104 unknown battery fault
- -127 PMU/SMC forced shutdown for another cause
- -128 unknown, possibly battery is at the end of its life, but can also occur when the SMC initiates an automatic restart following a kernel panic.

Common software cause codes:

- 3 is a 'dirty' shutdown resulting from a forced restart or shutdown
- 5 is a 'clean' shutdown or sleep initiated by the user.

Thanks to Graham Perrin and others who revealed this information on StackExchange and elsewhere.

Check log



This window shows useful information about the active log on the current Mac which can be helpful in diagnosing log problems, and tells you how far back those logs go in time. Use the **Check Log** command in the app menu to open this window. You should then see:

- there are at least 4 items in the path /var/db/diagnostics
- there is at least 1 file in each of the the Persist, Special and timesync folders
- there is at least 1 file in the Signpost folder when running Mojave or later; that folder doesn't exist in earlier versions of macOS, though
- the `log show --last 2` command returns at least 2 log entries.

If any of those are incorrect, then there's something wrong with that Mac's log system.

Information for the oldest log entry refers to log files in the Persist folder. Although there will still be some log entries prior to that time, they're generally incomplete and provided by remaining files in the Special folder. If that period isn't long enough, it means too many log entries are being added to the log.

→ [controlling log size](#)

Controlling log size

Unlike conventional logs, those in the Unified log are maintained by `logd` so that the total size of the `Persist` folder is about 525 MB in size. As there appears no way to increase that, if the rate of new log entries being added is high, this means that older entries are removed sooner than you might want. Currently the only way to control this is to exclude entries from particular subsystems or processes from being retained in the log files when they're written to disk. When you've identified which subsystems or processes you want to exclude, in this example those from the subsystem `com.apple.bluetooth`, use either of two methods to exclude them.

In Terminal, first check current setting using the command

```
sudo log config --status --subsystem com.apple.bluetooth
```

which should return

```
Mode for 'com.apple.bluetooth' INFO PERSIST_INFO
```

saving Info and above entries to disk. Disable that using the command

```
sudo log config --mode "persist:off" --subsystem com.apple.bluetooth
```

and no further entries from that subsystem should be saved to disk.

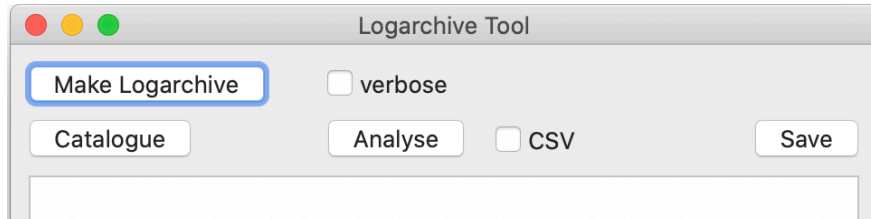
This command creates a property list in `/Library/Preferences/Logging/Subsystems` named

`com.apple.bluetooth.plist` Instead of using that command, you can create your own property lists in that folder. Use the examples provided there and in the read-only equivalent at `/System/Library/Preferences/Logging/Subsystems` which provides system defaults.

Remember to disable these restrictions well before using a feature like `sysdiagnose`, which rely on log extracts.

→ [check log](#)

Create logarchives



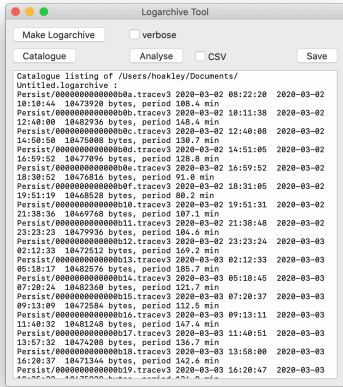
Ulbrow offers two ways of creating logarchives, portable bundles containing a set of log files which can be accessed through the **Open Logarchive...** menu command. The **→ Write Logarchive...** command in the **File** menu uses the standard `log collect` tool to create these, but can only use the active system log on that Mac as its source. If you want to create a logarchive from another Mac, or a salvaged collection of log files, open the **Logarchive Tool** in the **Window** menu.

To work with the **Make Logarchive** button, log files should be put inside a folder containing just two folders named `diagnostics` and `uuidtext`. The first contains the main files, structured into folders such as `Persist` and `Special`, which contain the major `.tracev3` files, and other key files such as `version.plist`. `uuidtext` contains many folders, starting with `00` and ending with `FF`. Click on the **Make Logarchive** button, select the folder containing those two folders, then in the Save File dialog choose the name and location of the logarchive to create from them. Tick the **verbose** checkbox beforehand if you want a full account of what was copied where.

This method should create a logarchive which can be accessed by Ulbrow, although it may not work with Console or some `log` commands. It's surprisingly resilient to damaged or missing contents, although it's quite possible to create logarchives which look good but only return errors when you try to access them. Sometimes manually tuning the `Info.plist` file can coax a fragmentary log into yielding at least some of its contents.

[→ Analyse logarchives](#)

Analyse logarchives



However you create them, logarchives can now be analysed by Ulbow in two ways: it can list all the `.tracev3` files in their main folders, constructing an inventory of them complete with periods over which each was collected and their sizes, and it can show basic statistics about each of those files in terms of frequency of log entries from different named processes (only in macOS 10.13 and later).

To view a catalogue for any logarchive, click on the **Catalogue** button and select the logarchive you want to analyse. The results will be displayed in the window, and you can save them as a text file using the **Save** button.

To view statistical analysis for the log files in a logarchive, first choose whether you want them in regular formatted text, or in CSV format (to import into a spreadsheet or database) when you tick the **CSV** checkbox. Then click on the **Analyse** button and select the logarchive to be analysed. Those from Sierra may not contain suitable analyses to perform this on. Use the **Save** button to save the window contents as text.

→ [Create logarchives](#)

Updates

Whenever you open Ulbow, 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 Ulbow 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, Ulbow 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, Ulbow 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 Ulbow'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.Ulbow 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 later is `monospacedSystemFont`, which is unfortunately not available in earlier versions of macOS. Both are used in their regular weight.

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

Change list

- 1.8:
 - added log period estimate to Check Log window
 - removed button to Check Log, reformatted results
 - fixed a bug in generating CSV by omitting string formatting field
 - extended Help book to cover controlling persistence
 - removed code integrity check.
- 1.7:
 - improved launch checks on time formatting
 - addressed changes in time formatting in macOS 11.2.
- 1.6:
 - added log checks when app has launched, to protect from macOS bugs.
- 1.5:
 - added Reset All Preferences button to Preferences
 - now creates a default set of preferences on its own, when needed.
- 1.4:
 - fixed minor version issues for Universal App release.
- 1.3:
 - removed Signposts controls when running on Sierra
 - built with Xcode 11.5.

1.2:

- built with Xcode 11.4 for release.

1.2b2:

- fixes a bug in handling multiple whitespaces in style strings.

1.2b1:

- added Logarchive Tool, with features to create logarchives and analyse them
- logarchive creation made more robust in the face of missing contents.

1.1:

- fixed obscure bug which was clipping initial log entries from Chart view
- removed time zone from Chart view tooltips
- set initial datestamp in Chart view to Chart start time
- re-checked charting calculations and adjusted.

1.1b3:

- added Save to PNG for Chart view
- added spot time feature to Chart view
- enabled direct editing of period, with remaining small bug when switching from text editing to stepper control
- added copy and paste of date-time settings
- added Get Debug option.

1.1b2:

- added Check Log feature.

1.1b1:

- added Chart view
- changed document and window titling.

1.0:

- first full release.

1.0b3:

- enlarged main window to accommodate AM/PM indicators and allow non-24-hour clock usage
- fixes a potential error in checking admin user privileges if the user is a member of few groups
- added CSV conversion.

1.0b2:

- changed font to monospace
- added auto-update feature
- added check for admin user

→ [Start](#)

→ [Contents](#)

- added logarchive creation
- added use of logarchive as source
- revised Help book further.

1.0b1:

- first beta release.

23 December 2022.