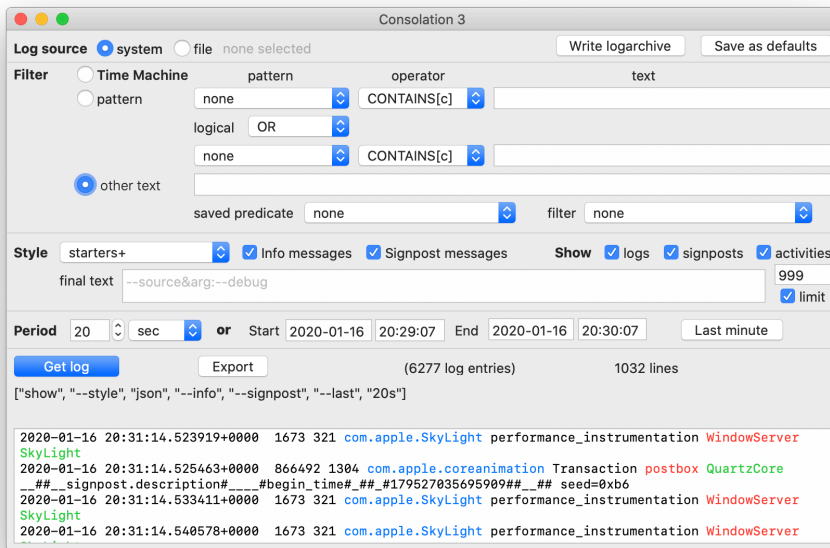


Start

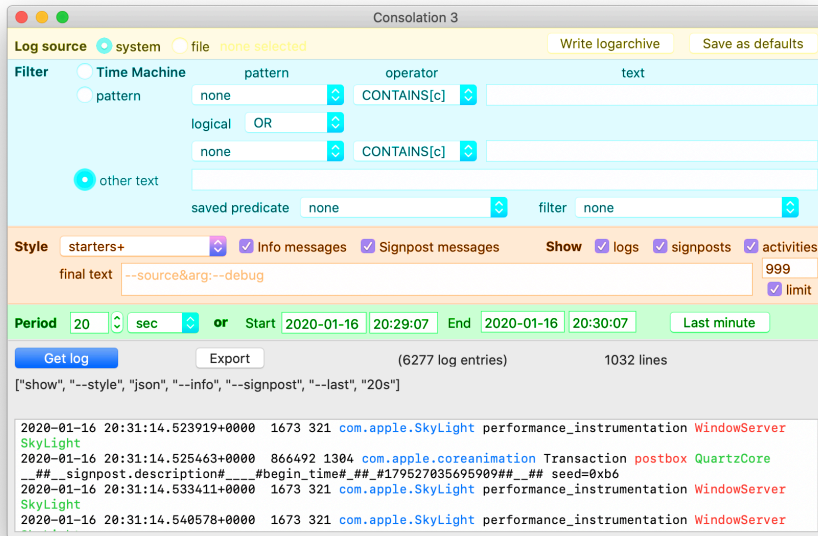
Consolation 3 gives unique access to the macOS unified log introduced in Sierra. Unlike Console, it analyses any entries which have already been written to the log. Unlike the `log show` command in Terminal, Consolation can work with full log extracts in JSON format, showing you exactly what you want to know, and exporting in styled Rich Text, or to CSV for use in other apps. Its graphical interface makes the log accessible to all users, and makes intensive log analysis quick and simple.

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

Contents

- [Main window layout](#)
- [Check Time Machine](#)
- [Inspect a crash, fault or event](#)
- [Set the source of the log](#)
- [Set the entries to be extracted](#)
 - [Predicates](#)
 - [Other text and final text](#)
- [Set the style of log content](#)
- [Set the period of log collection](#)
- [Get log and export](#)
- [Custom predicates](#)
- [Custom styles](#)
 - [Style definitions](#)
 - [Available fields](#)
- [Custom filters](#)
- [Preferences and settings](#)
- [Save log extracts](#)
- [Signposts](#)
- [Navigation](#)
- [Cause codes](#)
- [Updates](#)
- [Further information](#)
- [Change list](#)

Main window layout

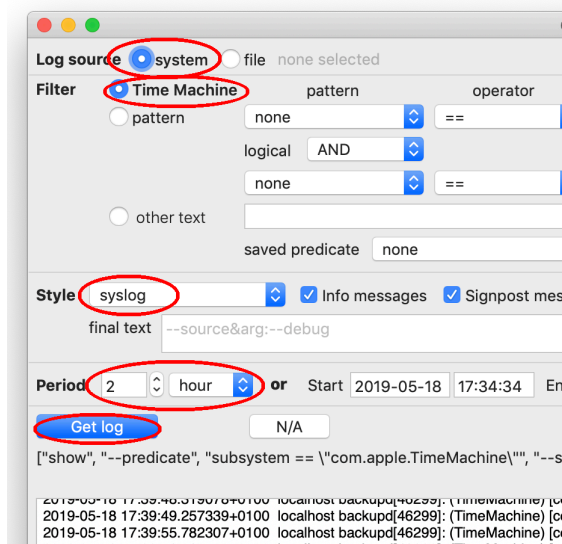


Consolation can open an almost unlimited number of windows, each containing different log extracts. Each window is structured into sections:

- at the top (yellow), → [select the source of the log](#), either the active system log or a logarchive
- below that (blue), → [set up the filter](#) for selecting which entries to browse using predicates
- in the middle (orange), → [set the style for display](#) of the log entries
- lower down (green), → [set the time period](#), or specify set times
- at the bottom (grey), → [get and view](#) that log extract.

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 Consolation next.

Check Time Machine



To browse Time Machine entries for the last 2 hours:

- set the Log source to **system**
- set the Filter to **Time Machine**
- set the Style to **syslog**
- set the Period to **2 hour**
- then click on the **Get log** button.

This is far simpler in my free utility **The Time Machine Mechanic** (T2M2), of course, which even interprets the log entries for you.

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

→ [Set the source of the log](#)

Inspect a crash, fault or event

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

- set the Log source to **system**
- set the Filter to **other text**, with the box to the right of it left empty
- set the Style to **syslog**, or use a custom style such as **starters+** supplied in the additional Property List
- to limit the number of log entries displayed when you're using a custom style, tick the **limit** checkbox, with the limit set above it at **999 - 2500**
- set the Period to **0**, then in the **Start** boxes enter a time about 5 seconds before the event, and in the **End** boxes enter a time about 20 seconds after the event. Be careful to use the correct time format of HH:MM:SS, and when you've finished editing them, press Enter or Tab to ensure those changes are registered properly
- then click on the **Get log** button.

Scroll through the entries, which include everything written to the log over that period of about 25 seconds, and you should be able to see that event occurring, whatever led up to it, and its consequences. If you're using a custom style, you can always increase the **limit** you set, to show more log entries. Type a new number in the box above the **limit** checkbox and press the Tab or Return key to refresh the log extract.

→ [Check Time Machine](#)

→ [Set the source of the log](#)

Set the source of the log

Log source ☒ system ☐ file none selected

Write logarchive

Save as defaults

To examine the active log on the current Mac, select **system**. To examine a previously-saved logarchive, click on **file** then select the logarchive, or an individual log file, with the extension `.tracev3`, within a logarchive bundle.

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

The unified log now collects and retains logs for long periods: in macOS Sierra 10.12.6 and later, this can be as long as 20 days. Trying to browse the 25 million log entries in a complete logarchive from a Mac is extremely slow, and ultimately futile. By selecting just a single log file within such a large logarchive, you can quickly home in on a period of interest, making browsing far more practicable.

Consolation also opens and browses logarchives written from iOS devices, Apple Watch, and Apple TV. It also works with logarchives obtained using the `log collect` command in Terminal.

⚠ Consolation 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 Consolation running on Mojave or later. However, Consolation 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, further down the window, then click on the **Write logarchive** button. 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.

To save the current settings in this window (except Start and End times) as the defaults to be used for new windows opened in Consolation, click on the **Save as defaults** button, and Consolation will write the settings to its preferences file in `~/Library/Preferences`.

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

Set the entries to be extracted

Filter	Time Machine	pattern	operator	text
<input type="radio"/>	Time Machine	none	==	
<input type="radio"/>	pattern	logical AND		
<input type="radio"/>		none	==	
<input checked="" type="radio"/>	other text			
	saved predicate	com.apple.securityd		filter error

Consolation offers five ways to specify which entries in the log will be shown in a log extract:

- select **Time Machine** if you want to browse only entries relevant to Time Machine making its backups (shortcut Command-1)
- select **pattern**, then set the popups and boxes to the right of it to create a predicate (shortcut Command-2)
- select **other text** and enter (paste in) a specially-formatted option to the `log` command in the text box to the right of it
- select **other text** and set a **saved predicate** in the popup menu, from your custom list of predicates
- select **other text**, leave the text box empty, and the saved predicate set as **none**, which will return all log entries.

The **other text** option has a shortcut of Command-3.

→ [Predicates](#)

→ [Custom predicates](#)

→ [Other text and final text](#)

→ [Set the style of log content](#)

Predicates

The first popup menu in each of the two **pattern** lines determines what is examined in the filter. On offer are:

- **eventMessage** – specify a text pattern, or text, within the message, or an activity name.
- **processImagePath** – matches the text pattern in the name of the process which originated the event.
- **senderImagePath** – matches the text pattern in the name of the sender, which might be the name of a library, extension, or executable.
- **subsystem** – matches the subsystem specifier, e.g. `com.apple.TimeMachine`. Although potentially valuable, subsystems are not yet widely used, and discovering which is which is not easy. Use with caution.
- **eventType** – matches the type of event, such as `logEvent` (1024), `traceEvent` (768), `activityCreateEvent` (513), or `activityTransitionEvent` (514). Can be given as characters (case-sensitive) or digits as shown in parentheses. Use these only with the operators `==` or `!=`, as they are treated as numbers rather than text.
- **messageType** – matches the type of message for `logEvent` and `traceEvent`, and includes default (0), release (0), info (1), debug (2), error (16), and fault (17). Can be given as characters (case-sensitive) or digits as shown in parentheses. Use these only with the operators `==` or `!=`, as they are treated as numbers rather than text.
- **category** – matches the specified category, and should normally be used in conjunction with a subsystem filter; for the whole specifier `com.apple.TimeMachine.TMLogInfo`, the subsystem is `com.apple.TimeMachine` and the specifier is `TMLogInfo`.

The **operator** popup menu in each of the two pattern lines determines what the filter actually does. Operators available include:

- `==` is the equality operator, as in `=="com.apple.TimeMachine"`; this is also available without case sensitivity as `==[c]`, and without case or diacritic sensitivity as `==[cd]`
- `!=` is the inequality operator
- **BEGINSWITH** is for text which begins with the quoted text, and is case- and diacritic-sensitive
- **CONTAINS** is for text which contains the quoted text, and is case- and diacritic-sensitive
- **CONTAINS[c]** is for text which contains the quoted text, and is case-insensitive and diacritic-sensitive
- **CONTAINS[cd]** is for text which contains the quoted text, and is case-insensitive and diacritic-insensitive
- **ENDSWITH** is for text which ends with the quoted text, and is case- and diacritic-sensitive
- **LIKE** is for text comparisons using `?` and `*` wildcard characters; `?` matches exactly 1 character, and `*` matches none or more.
- **MATCHES** is for text comparisons using regex style, according to ICU v3.
- **IN** is an aggregate operation similar to SQL `IN`, where the left-hand side must appear in the collection specified by the right-hand side. For example, `name IN { 'Ben', 'Melissa', 'Nick' }`.

Logical operators which can be used to combine two filter patterns include:

- **AND** - simple, logical AND - both patterns are true
- **OR** - simple, logical OR - either pattern is true
- **AND NOT** - logical AND, but the second pattern is NOT true
- **OR NOT** - logical OR, but the second pattern is NOT true.

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

Other text and final text

The text box **other text** lets you to enter any other valid predicate which you wish, or to omit the predicate filter altogether and view all log entries. When the **other text** radio button is selected, the arguments in the **other text** box are submitted instead of any predicate which you might have entered in the popup menus above. To see the whole unfiltered log, leave the **other text** box completely empty.

In either text box, you must separate the text into arguments, as each is submitted separately when the command is run. Separate arguments using the text `&arg:` without any spaces. To pass the two arguments `--source` and `--debug`, enter the following into the **other text** or **final text** box:

```
--source&arg:--debug
```

To hand-code your own predicate, you might enter the following in the **other text** box:

```
--predicate&arg:subsystem == "com.apple.TimeMachine"
```


⚠ You *don't* need to surround the predicate itself with single quotes, as you would in the command line. This is because that whole chunk is treated as an argument, and is not submitted through a command shell, which would expect it those single quotes.

The contents of the **other text** box replace any settings for predicates made using the popup menus, which are ignored. Any text entered in the **final text** box is appended to the end of the `log show` command after any predicates set using popup menus, and follows the same convention using `&arg:` to separate its arguments.

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

→ [Set the style of log content](#)

Set the style of log content

Style starters+ 

☒ Info messages ☒ Signpost messages

Show ☒ logs ☒ signposts ☒ activities

final text

☒ limit

In the **Style** popup menu, Consolation offers three built-in styles for log extracts:

- **syslog**, which resembles a traditional Unix log, giving timestamp, process, process ID, and the message
- **default**, which gives timestamp, thread, type, activity, process ID, TTL, and the message
- **json**, which gives full entries formatted as JSON data, which is most useful for export to CSV format.

In addition to those, Consolation offers **custom styles**, in which you include any of the available fields in the order you want, with the addition of three colours, as well as default white/black. If you install the custom Property List supplied, this offers minimal styles like **basic**, and the more inclusive **starters+**.

When you get a log extract in JSON or a custom style (which actually gets entries in JSON format, then applies that style to them), switching between those items in this popup menu instantly changes format of the entries currently displayed. Switching between **syslog**, **default**, and any JSON-based style (including custom styles) requires you to get the log extract again, because those styles are applied by the `log show` command itself.

Checkboxes for **Info messages** and **Signpost messages** are set as **Get log** options, so you'll need to get a fresh log excerpt to apply them. The three checkboxes to **Show logs**, **signposts** and **activities** are applied instantly if the current style is JSON or a custom style.

To add any further options to the `log show` command, enter them into the **final text** box, which should otherwise be left empty.

When the **limit** checkbox is ticked, only the number of log entries set in the text box above will be displayed when using a custom style. This prevents you from ending up with tens or hundreds of thousands of entries by accident. It also works dynamically: change that number and press the Tab or Enter key and the listing will be updated immediately when the box is ticked. Uncheck the box to view all log entries.

→ [Set the period of collection](#)

→ [Custom styles](#)

→ [Other text and final text](#)

Set the period of collection

Period	<input type="text" value="0"/>		hour		or	Start	<input type="text" value="2019-05-18"/>	<input type="text" value="17:44:40"/>	End	<input type="text" value="2019-05-18"/>	<input type="text" value="17:55:40"/>	<input type="button" value="Last minute"/>
--------	--------------------------------	--	------	--	----	-------	---	---------------------------------------	-----	---	---------------------------------------	--

Consolation offers two methods for specifying the period of logs to be examined:

- if the number to the right of **Period** *isn't* **0** (zero), then the integer set there will be used with the time units in the popup menu, e.g. **2 hour** will examine the last 2 hours of the log
- if that period is **0** (zero), then the interval between the **Start** and **End** dates and times will be used instead.

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, shorter periods can be used, even below 10 seconds if you wish. The value in the Period box is limited to between 0 and 999 inclusive.

Start and **End** dates *must* be given in the format YYYY-MM-DD, and times as HH:MM:SS, and should match those shown by the system clock, i.e. they're expressed according to your Mac's local timezone as set. ⚠ Strange things happen when summer time starts and ends. You may need to experiment with settings if they span such a change in the system clock.

The **Period** set inevitably refers to different log entries as real time passes. If you want to take a quick look at events over the last minute, rather than using 1 minute, click on the **Last minute** button, shortcut Command-P. This will set the Period to 0, Start to one minute ago, and End to the current time.

The **Period** is also used when saving a logarchive. In that case, you can't use the Start and End times, but must set a number and a time unit at the left of this section.

If you switch from typing in digits into the **Period** box to using its stepper control, the first click on that may result in an unexpected number appearing for the **Period**. To avoid that, use the *Tab* or *Return* key to complete editing before using the stepper.

→ [Get log and export](#)

Get log and Export

When you've set the sections above, click on the **Get log** button (shortcut Return) to run the `log show` command, obtain and display the log extract. If that command returns an error, that should be displayed. To help you (and me) understand why an error occurred, the box immediately below the button shows the command arguments used as a Swift array of strings, which you can select, copy and paste elsewhere.

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.

Most of the time, the button to the right of **Get log** carries the characters **N/A** (not applicable), and it is inactive: clicking on it does nothing. When you have just obtained a log excerpt using the JSON format option or a custom style, the button changes to show **Export**. Clicking on it then will automatically convert the log excerpt from JSON format to CSV (comma-separated values), which you can then **Save** or **Save as** using menu commands, and can import into Microsoft Excel, Numbers, FileMaker Pro, or any other app which handles CSV format.

⚠ If you want to preserve that log extract in its current format, now is the time to **Save** it, as exporting will irrevocably change it – there is no Undo.

Save that text file using a unique file name (⚠ use **Save as...** if you have just saved it in JSON format, to avoid overwriting that). You can change its extension in the Finder to `.csv` if you wish, and that may make it easier to import into some apps. Apple's Numbers spreadsheet requires the file to have the extension `.csv` before it will import it, but Microsoft Excel doesn't.

→ [Get log and export](#) (completed)

Get log and Export (completed)

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.

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. If that is important to you, save the JSON format file and refer to that for content which has been lost due to truncation.

[Get log](#)[Export](#)

22637 log entries

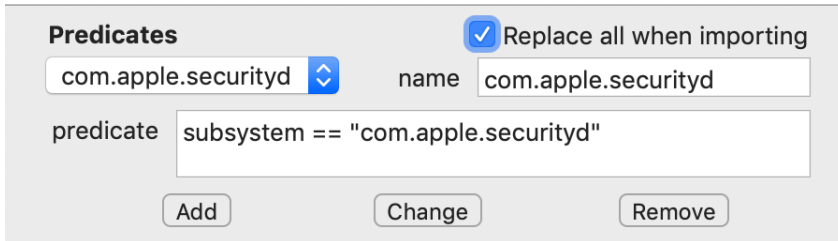
23240 lines

```
["show", "--style", "json", "--info", "--signpost", "--last", "1m"]
```

The text near the Get log button provides you with information about your log extract. To the right are displayed the total number of log entries contained in it, 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.

Below are the command elements which Consolation submitted to obtain the extract: append the log command and drop its string formatting, and you should be ready to use the same command in Terminal, if you prefer.

Custom predicates

The screenshot shows the 'Predicates' section of a preferences window. At the top, there is a checked checkbox labeled 'Replace all when importing'. Below this, there are two input fields: 'name' and 'predicate'. The 'name' field contains 'com.apple.securityd'. The 'predicate' field contains 'subsystem == "com.apple.securityd"'. At the bottom of the section, there are three buttons: 'Add', 'Change', and 'Remove'.

Consolation can store custom predicates for you, which are ideal for those which you use often. To add and edit custom predicates, bring a Consolation window to the front, and use the **Preferences...** command in the main app menu to drop down the editor.

The first item in that menu will always be **none**, and that cannot be removed or edited. 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 Consolation'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, click on **Close** and those new items will be saved to Consolation's preferences, and immediately available in all windows.

→ [Custom styles](#)

Custom styles



The screenshot shows a 'Styles' dialog box. At the top, there's a title 'Styles'. Below it, there's a dropdown menu with 'starters+' selected, a 'name' field with 'starters+', and a 'style' field with '0 2g 6 9 4b 3 11fr 13fg 10'. At the bottom, there are three buttons: 'Add', 'Change', and 'Remove'.

Consolation can store custom styles for you, which are ideal for those which you use often. To add and edit custom styles, bring a Consolation window to the front, and use the **Preferences...** command in the main app menu to drop down the editor.

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, click on **Close** and those new items will be saved to Consolation's preferences, and immediately available in all windows.

User-defined styles depend on the log extract being obtained in JSON format. If you want to switch from either the **syslog** or **default** style to a custom style, you have to click on the **Get log** button again. However, changing from the JSON or any user-defined style to another user-defined style is performed simply by changing the selected style in the popup menu: the style will be applied instantly. You can also export in CSV format from a user-defined style without having to click **Get log** again. 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 (including CSV after exporting), or as Rich Text (RTF). If you opt for RTF, colours displayed in Consolation'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

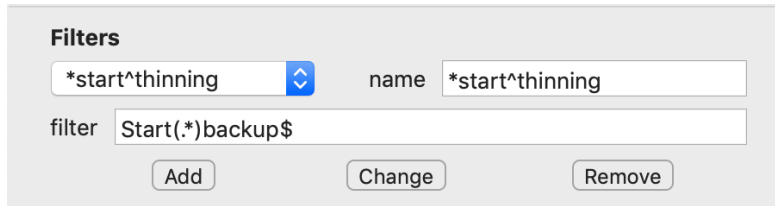
The full list of available fields and their numbers is:

- 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 Consolation 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 only, and is currently ignored by Consolation.

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 and later, fields 0-25 are supported. To check which are available for the version of macOS on which Consolation is running, use the **Fields help...** menu command in the **Help** menu.

→ [Custom filters](#)

Custom filters

The screenshot shows a 'Filters' dialog box with a title bar. Inside, there are two text input fields: 'name' and 'filter'. The 'name' field contains the text '*start^thinning' and has a small dropdown arrow on its right. The 'filter' field contains the text 'Start(*)backup\$'. Below these fields are three buttons: 'Add', 'Change', and 'Remove'.

In Consolation, you can 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.

Consolation 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, bring a Consolation window to the front, and use the **Preferences...** command in the main app menu to drop down the editor.

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.

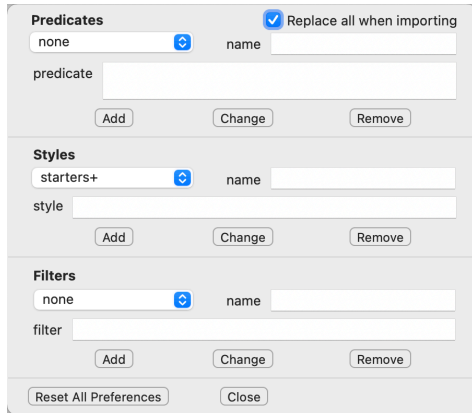
When your new entry is ready, click on the **Add** button to add it to the popup menu, and continue to add the filters which you want to store. To change a filter already in the menu, select that menu item and its **name** and **filter** will be loaded ready. Use the Tab or Enter key to finish editing, 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, click on **Close** and those new items will be saved to Consolation's preferences, and immediately available in all windows.

Filters *only* work on log extracts which have been obtained in JSON format, such as those with a custom style, with or without a custom predicate.

→ [Preferences and settings](#)

Preferences and settings



Changes which you make in the Preferences sheet are not just made available in the respective popup menus in the currently open window, but they are also saved automatically to the app's preferences file: you don't have to click on the **Save as defaults** button in the window to save changes made to custom predicates, filters, or styles.

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, Consolation 3 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 will have to use that preferences file to do so. As with the **Preferences...** command, these commands are available when any 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 will 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 top right of the Preferences sheet. 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.

The **Reset All Preferences** button replaces all your current settings with the defaults for Consolation.

Consolation'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 are not managed as preferences files, and you are encouraged to edit them as you wish. For example, Consolation 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` must consist of the menu item `none`:

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

- `specStyleList` does not include the standard styles of `syslog`, `default`, or `json`, which are supplied by `Consolation`.

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. If you supply an item with that key value and just the `none` menu item, then all existing custom predicates will be removed.

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.

Save log extracts

Consolation 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. This is essential if you have converted the excerpt to CSV; in that case, set the format as plain text, and give the file the extension of .csv.

If you want to browse the excerpt outside Consolation, 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 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 2017-02-08 14:11:27 +0000 for command log show --predicate subsystem ==  
"com.apple.TimeMachine" --style syslog --info --last 3h
```

⚠ If you want to use the JSON format to save your log excerpt, and then wish to process or import that text file, you will need to remove that final line from the file, as most code handling JSON data will throw an error if it encounters it.

Signposts

macOS High Sierra (at least by version 10.13.5) and later now support an additional type of log entry: Signposts. Consolation supports the additional keys available in High Sierra's logs, and provides tools to help you use them by showing only certain types of log entry. For compatibility with Sierra, these tools and the additional keys have little or no effect when using Consolation in Sierra, as its log files and log command tool don't recognise them.

To include Signposts when using Consolation on High Sierra or later, ensure the **Signpost messages** checkbox is ticked before clicking on **Get log**. You should also select a custom style (or JSON) in the **Style** popup menu. 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 Consolation on Sierra) only support regular log messages and Activities, and have limited discrimination between them.

Once you've fetched your log extract, use the checkboxes next to **Show** to change which entries are displayed. If you want to see only Signposts, for example, uncheck the **logs** and **activities** boxes, leaving the **signposts** box ticked ✓. Those controls 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 Consolation to get you started.

Your **Show** settings also affect CSV Export conversion: although the **filter** and **style** settings do not alter CSV output, the **Show** settings do. If you want to export only Signpost entries to your CSV file, tick the **signposts** checkbox and uncheck **logs** and **activities**. In Sierra, because of the much more limited support for the `eventType`, this has little effect, except that you can turn export of Activities off by unchecking the **activities** checkbox before exporting.

To get the most from Signposts:

- capture and analyse your logs on High Sierra 10.13.5 or Mojave
- ensure the **Signpost** messages box is ticked
- select a suitable custom style in the Style popup menu
- uncheck the **logs** and **activities** boxes in **Show**
- set up other controls to capture the Signposts you want to analyse
- click on **Export**, save as text, and change the extension to `.csv` for easy import into a spreadsheet.

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 does not 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.

Updates

Whenever you open Consolation, 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 Consolation 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, Consolation 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, Consolation 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 Consolation'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, Consolation will reset it automatically to that default.

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

```
defaults write co.eclecticlight.Consolation3 updateCheckInt '10'
```

which works properly through the preferences server `cfprefsd`.

Further Information

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 **Consolation Support** command in the **Help** menu too.

Change list

- 3.11:
 - added Reset All Preferences button to Preferences
 - now sets its own defaults when started the first time.
- 3.10:
 - fixes a couple of minor version issues, and is a Universal App.
- 3.9:
 - first version ported to Big Sur.
- 3.8:
 - fixes a bug in handling multiple whitespace characters in custom styles.
- 3.7:
 - removes requirement to press Tab/Return to change Period, but a small glitch when switching to using the Stepper still
 - changed font to monospace, using 10.15 version where available
 - added ability to limit number of log entries displayed.
- 3.6:
 - fixes a problem detecting whether an admin user for those users who are members of very few groups.
- 3.5:
 - now detects whether being run as an admin user, and throws an alert and quits if it is not
 - further optimisations and tweaks to reparsing and redisplaying log extracts
 - added log entry and line counts.
- 3.4:
 - fixed three bugs affecting formatting in Catalina.
- 3.3:
 - fixed bug in Fields Help for Catalina
 - built with Xcode 11.
- 3.2:
 - added feature to change font size in log extract view
 - improved window size and position saving
 - added support for automatic checking of updates and downloading them.
- 3.1:
 - enabled copy to use Rich Text, so preserving colours.
- 3.0:
 - added code signature check on launch
 - added product support command

- new PDF help book
- changed from traditional Help book to PDF.

3.0b17:

- fixed bug which set last line of RTF text in black which didn't change with Dark Mode
- ported to Swift 4.2.1 and Xcode 10.1
- notarized for Mojave.

3.0b16:

- fixed Dark Mode support
- adjusted depth of Fields Help window according to version of macOS.

3.0b15:

- full support for Mojave, including Dark Mode for custom styles, as documented when running on Mojave
- changed order of fields for better handling of Signposts
- added Fields Help window
- added OS-dependent behaviours with respect to fields/keys
- compiled in Xcode 10beta.

3.0b14:

- added support for fetching Mojave's Signposts
- added Show logs/signposts/activities checkboxes
- added Browse updates to Help menu
- improved code in Export and JSON to text conversions
- remaining filter and style popups issue to fix before release
- two versions: s compiled in Sierra, h compiled in High Sierra with Xcode 10beta.

3.0b13:

- fixed an obscure bug setting start and end times incorrectly when switching from logarchive to system log
- fixed a bug preventing import and export of custom settings
- fixed a covert bug opening Preferences
- added Last minute button
- renamed Run command button to Get log
- added keystroke shortcuts to controls
- further interface tweaks.

3.0b12:

- added option to save log extracts as RTF, which preserves colour attributes supported by custom styles
- put log show and logarchiving commands into background processes to minimise spinning beachballs
- added 'busy spinners' for Run command and Write logarchive buttons
- reduced window minimum size for compatibility with smaller displays.

3.0b11:

- fixes a crash which could occur when using a custom style including traceID (field 7)
- fixes a crash which could occur when exporting to CSV.

3.0b10:

- ported to Swift 4.0 in Xcode 9.1
- synchronised preference changes with all windows
- rearranged controls to fit new broad window.

3.0b9:

- added ability to open .tracev3 log file within a logarchive
- fixed minor bugs in button connections

3.0b8:

- ported to Swift 3.2 and rebuilt using Xcode 9.0 GM.
- 3.0b7:
- improves behaviours in the preferences sheet.
- 3.0b6:
- added support for exporting and importing property list files containing custom predicates, filters, and styles
 - fixed updating of current window's custom popup menus when closing preferences sheet
 - added more Tooltips, including the preferences sheet.
- 3.0b5:
- added filter support, for both plain and regex searches
 - restored access to saved logarchives
 - fixed potential crashes trying to process empty JSON data
 - fixed rare false 0 return from parsing JSON data (for integer values).
- 3.0b4:
- restructured code handling JSON extracts, and re-worked logic
 - added colour options to styles.
- 3.0b3:
- fixed a crashing bug in Styles, if Consolation tried to handle any numeric field
 - improved range-checking of styles
 - added T format in styles
 - pre-parse list of fields in styles, to improve performance.
- 3.0b2:
- completed Preferences sheet, with styles and filters editors, although filters currently do nothing in the app itself
 - implemented first cut of styles, with support for h and d format options for datestamp field, and t and f options for others
 - Predicates/predicate text box grows to the right when the sheet is enlarged.
- 3.0b1:
- added new popup menus
 - added Preferences sheet, with predicate popup editor.

1 February 2021.