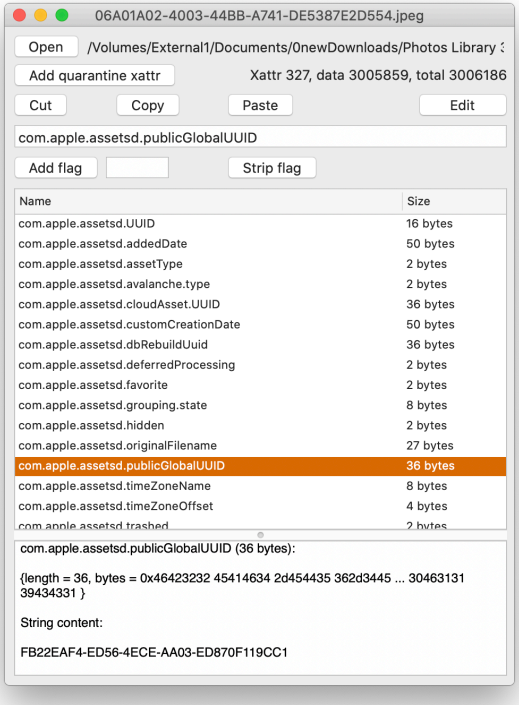
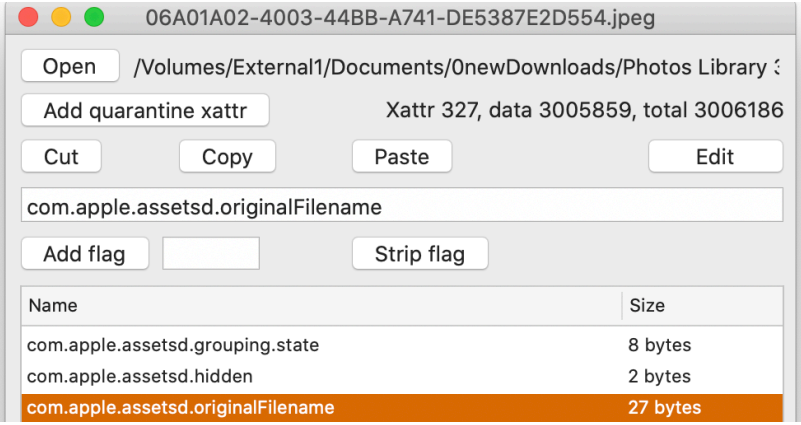
Start

xattred is the friendly and powerful editor of extended attributes (xattrs) for macOS El Capitan and above. To get help on its features and windows, click on a topic link below.



Basics



xattred's main window has nine functions to work with extended attributes:

- ➜ **Open** lists all the extended attributes for the selected file or folder, including the total size of the attributes. You can also use the **Open** and **Open Recent…** menu commands, or can **drag and drop** a file or folder onto the app icon.
- ➜ **Add quarantine xattr** makes an extended attribute which puts that item into Gatekeeper quarantine, adds it to the item, and enters the details into the quarantine database.
- ➜ **Cut** (⌘X) removes the selected extended attributes from the current item and stores them in the pasteboard.
- ➜ **Copy** (⌘C) makes copies of the selected extended attributes from the current item and stores them in the pasteboard.
- ➜ **Paste** (⌘V) adds copies of the extended attributes in the pasteboard to the current item.
- ➜ **Add flag** appends a protection flag to the selected extended attributes, using the flags set in the text box.
- ➜ **Strip flag** removes any protection flags from the selected extended attributes.
- You can **Save As…** one or all extended attributes to a text file.
- You can create **New** xattrs, and ➜ **Edit** the content of existing xattrs.

There are three additional windows: ➜ **Crawler** ➜ **Stripper** and ➜ **Compare**.

Open

To create a new window in which you can open the xattrs of a file or folder, use the **New** command in the **File** menu and click on the **Open** button in that window. Until you have opened a file or folder, all other buttons remain disabled. Alternatively, you can open an existing file using the **Open** command in the **File** menu, or you can open a file or folder by dragging and dropping it onto the app icon (in the Finder or the Dock).

When you click on the **Open** button, you will be prompted to select the file or folder which you wish to inspect, in a standard Open File dialog which drops down as a sheet over that window. The advantage of using the **Open** button is that this dialog shows hidden items too; when you use the **Open** menu command, you'll need to press ⌘⇧. to see hidden items.

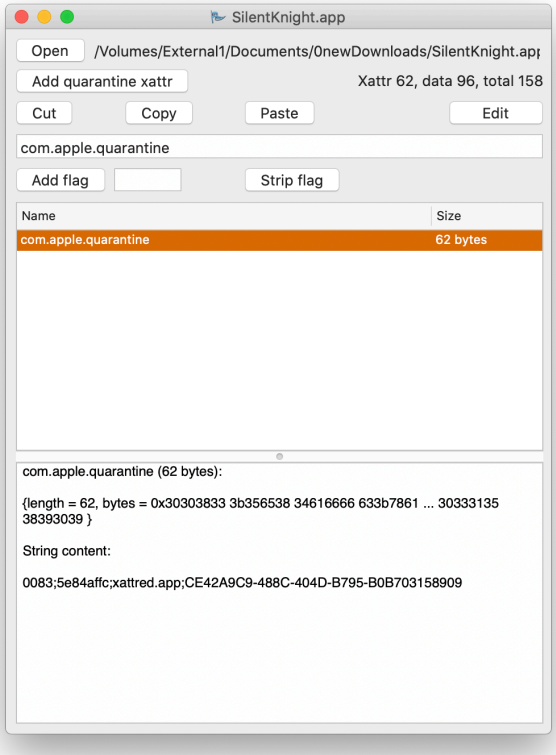
The extended attributes found are then listed in the scrolling table in the middle of the window, together with the size of each, the path to the file or folder will be displayed in the area next to the button, and the window will be named according to the file or folder which you opened. The area next to the **Add quarantine xattr** button will also show the total size of the attributes, the size of the data fork of the file, and the sum of those two values, as the total size of the file.

When you select one of the extended attributes in the table, it will be displayed in the large scrolling text box at the bottom. The first line in each group gives the name of that extended attribute and its size in bytes, following which its contents are shown. You can use the ⬇⬆ cursor keys to navigate the list of extended attributes.

Extended attributes are normally displayed in at least two different formats. The first shows them in raw hexadecimal within <> marks. The second format tries to decode that content as usefully as possible. For text content, this will be given as a string, normally in Unicode UTF-8 format. For those which are property lists, the XML source for the property list is given. For other binary content, extracted ASCII text is shown within «» marks.

➜ **Cut, Copy, Paste** ➜ **Edit**

Add quarantine xattr



To add a quarantine extended attribute to the current file or folder, which will force Gatekeeper to run a full check on the code signature within an app, click on the **Add quarantine xattr** button. xattred builds the content of the extended attribute, adds that to the current file or folder, and enters a new record in the user's quarantine database. This is valuable for developers who wish to check that an app is correctly signed for distribution.

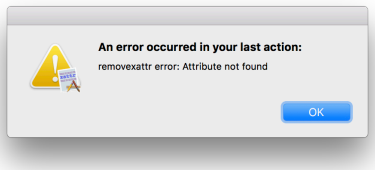
Cut, Copy, Paste

To cut or copy extended attributes from an item which is already open in xattred, select the xattrs in the table, then click on the **Cut** or **Copy** button as appropriate. The selected extended attributes will then be stored in the macOS General pasteboard. You can select multiple extended attributes, using the standard Shift and Command modifiers.

To paste extended attributes which have been cut or copied into the pasteboard, click on the xattred window of the item to which you want to add the xattrs, to bring it to the front, and click on the **Paste** button.

Extended attributes in the pasteboard use a custom data format which prevents them from being pasted into any documents outside of xattred, neither can you paste content from other apps into an extended attribute. You can, however, copy and paste freely between *text* representations of the attribute name and contents.

⚠️ Note that at present there is *no **Undo** command*. If you inadvertently cut the wrong xattrs, you will have to paste them back yourself; if you inadvertently paste the wrong xattrs, you will have to cut them out again.



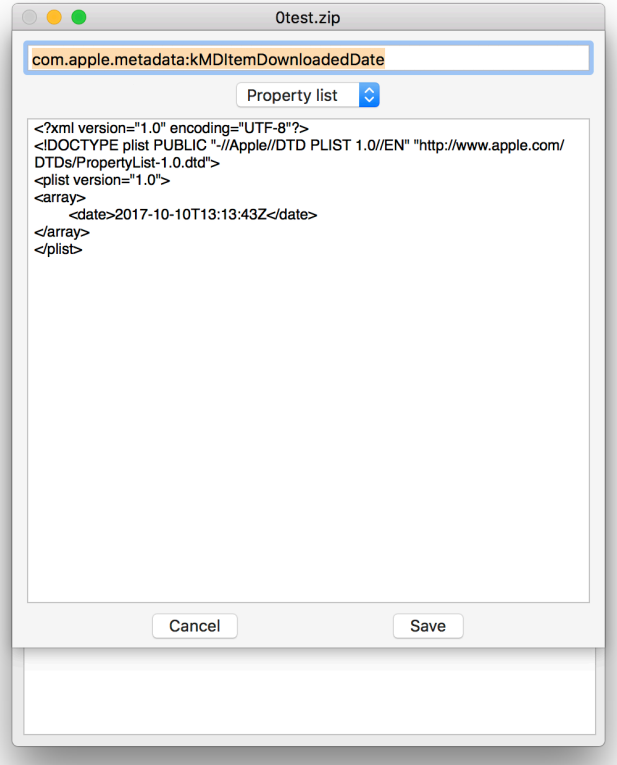
If an error occurs, a meaningful message should be displayed in an alert to explain what went wrong.

At any time, you can use the **Save as…** command in the **File** menu to save the contents of the bottom scrolling text area (a listing of the current extended attribute) as a text file. Default names are based on the most recently-selected file or folder name. If no extended attribute is selected, then this command will save a text version of all the extended attributes of the open file.

➜ **Edit**

Edit

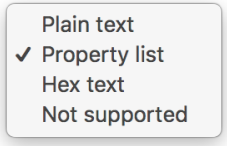
When an existing extended attribute is selected in the table view, the button at the right reads **Edit**; when no extended attribute is selected, the button reads **New**. Click on **Edit** to edit the currently-selected extended attribute, or on **New** to edit a new one. Either action drops down the xattr editor sheet. If you cannot find an easy spot to click on in order to deselect a row in the table, Command-click on the selected row.



➜ continue

Edit (continued)

When creating a new extended attribute, both text boxes will be empty; when editing an existing one, they will already be populated with the current contents of that extended attribute. The top text box contains the name (type) of the extended attribute, including any subtype and serialisation, e.g. com.apple.rootless or com.apple.metadata:kMDItemDownloadedDate



Below the attribute name is a popup menu, listing the formats in which the contents can be edited. Currently, there are three different formats:

- **Plain text** is treated as a simple Unicode UTF-8 string without any surrounding quotation marks, e.g.
  0083;5a4d439b;xattred.app;0358D6CB-A801-420A-A5EE-97220E8C0D04
- **Property list** is treated as a standard XML property list. For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<array>
        <date>2017-10-10T13:13:43Z</date>
</array>
</plist>
```

- **Hex text** is a string of hexadecimal with embedded spaces (optional), in identical format to the hexadecimal data shown for an extended attribute (the top section, between <> marks, in the normal window). This is converted to real hex bytes when stored. You can put the string inside <> marks if you wish, or you can omit them; if they are found when saving the content, they are automatically stripped from front and back before saving. For example,

```
d6a77d92 6889fccf 1e5e0dc2 f9ccad16 bbd2eb70
```

- **Not supported** is currently handled as hex text.

➜ continue

Edit (concluded)

xattred tries to recognise the correct format of the content of each extended attribute. This is most reliable for property lists and for plain text. Binary content can sometimes appear indistinguishable from plain text, and when you open the attribute to edit its content, binary content may be displayed in plain text format. If that happens, switch the popup menu to the correct format, in this case **Hex text**. The contents will then be reloaded in hex text format for you to edit.

When you change the format set in the popup menu, xattred does *not* try to convert the current contents of the editor view into the new format set, but re-loads the unedited data from the attribute. This ensures that you can at any time undo all changes made to the content by selecting the same format in the menu, and so that you can select the correct format in the event of ambiguity.

When the format is set to **Property list**, xattr attempts to parse that property list as soon as you click the **Save** button. If it cannot parse it correctly, it will not save your changes, and you should either correct the property list in the editor so that it does parse correctly and can be saved, or click on **Cancel** to leave the attribute unchanged.

When you have finished editing an extended attribute, check that the popup menu is set to the correct format for that xattr, then click on the **Save** button to save your changes to that attribute, or **Cancel** to abandon them, and return to the main window. ⚠️ *There is no **Undo** at present.*

⚠️ Do not attempt to edit the extended attributes of the sole or original copy of any important file. Always use a copy, and check that its attributes are all correct before using that. Defective or corrupt extended attributes can cause other problems, such as prevent the mdworker process to correctly index files for Spotlight, and they can cause apps to crash.

➜ **Cut, Copy, Paste**

Add and strip flags

Extended attributes may be stripped when items are copied or moved, including to different volumes and iCloud. macOS has an almost undocumented system which uses flags suffixed to the name of extended attributes to determine when they are stripped or preserved in different situations. These flags are specified after the hash **#** character, but many apps don't recognise them, and treat the flag as if it's part of the name.

To add one or more flags to selected extended attributes, enter the flag characters, such as **PS**, in the text box next to the **Add flag** button, and click that button. To remove all flags from selected extended attributes, click on the **Strip flag** button. Flags can't be removed from extended attributes which are specially protected, including com.apple.macl.

Flags can be upper or lower case letters C, N, P and S. Upper case *sets* (enables) that property, whilst lower case *clears* (disables) that property. The properties are currently:

- **C**: XATTR_FLAG_CONTENT_DEPENDENT, which ties the flag and the file contents, so the xattr is rewritten when the file data changes. This is normally used for checksums and hashes, text encoding, and position information. The xattr is preserved for copy and share, but not in a safe save.
- **P**: XATTR_FLAG_NO_EXPORT, which doesn't export the xattr, but normally preserves it during copying.
- **N**: XATTR_FLAG_NEVER_PRESERVE, which ensures the xattr is never copied, even when copying the file.
- **S**: XATTR_FLAG_SYNCABLE, which ensures the xattr is preserved even during syncing. Default behaviour is for xattrs to be stripped during syncing, to minimise the amount of data to be transferred, but this will override that default.

The most useful of these is **S**, which used alone after the # normally makes the extended attribute as well-preserved as possible.

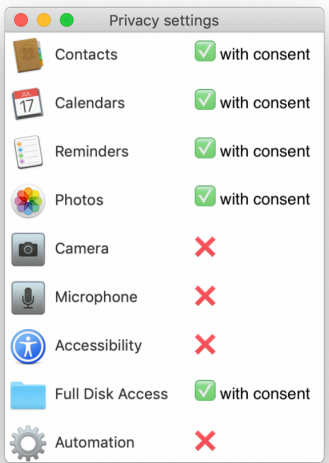
These must operate within another general restriction of xattrs: their name cannot exceed a maximum of 127 UTF-8 characters.

Privacy protection (Mojave and later)

xattred is designed so that it will edit the extended attributes of any item for which you have permissions. However, Mojave adds restrictions to preserve the privacy of your personal data. If you want to use xattred to edit *any* file or folder, including those in what Mojave considers to be protected data, then you must add the app to the list of apps with **Full Disk Access** in the **Privacy** tab of your **Security & Privacy** pane.

If you don't do that, but try to access protected data, then you should be prompted to give consent to xattred to do so; if you agree, it will then be added to the appropriate list of apps in the **Privacy** settings.

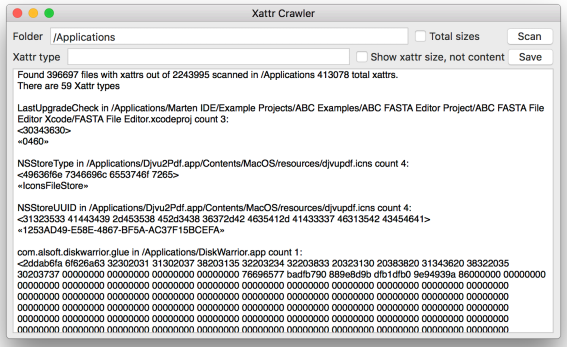
If xattred crashes when you try to access any protected items, this is most probably because you need to add it to the **Full Disk Access** list. If you're unsure what access it can have, its **Privacy settings** command in the **Help** menu displays a window which explains succinctly:



➜ Open

Crawler

To open a **Crawler** window to search for extended attributes, use the **Open Crawler…** command in the **Window** menu.



The Crawler offers two main searches for xattrs in any given folder:

- **all types**, with the Xattr type text box left empty
- **single type**, with the Xattr type text box containing the full identifier of a xattr type, e.g. com.apple.FinderInfo.

When searching for **all types**, there are two options set by the **Total sizes** checkbox:

- with that box empty (unchecked), clicking on the **Scan** button will return information about the total number of xattrs, the number of different types, and will provide one example of each type encountered
- with that box ticked, clicking on the **Scan** button will return the total number of files encountered with xattrs, the total, and average sizes of those xattrs.

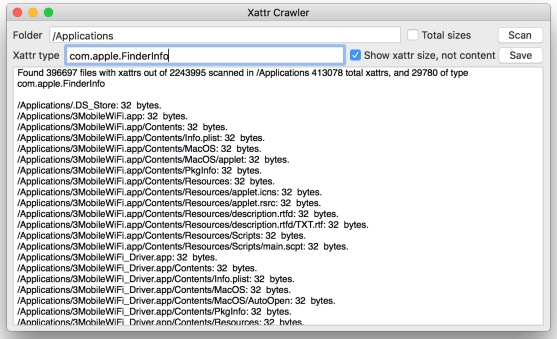
When searching for a **single type**, ensure that the **Total sizes** checkbox is empty (unchecked). You then have two options set by the **Show xattr size, not content** checkbox:

- with that box empty (unchecked), clicking on the **Scan** button will return the total number of those xattrs found, and will list them all
- with that box ticked, clicking on the **Scan** button will still return a list of all the files with that xattr, but rather than listing every xattr in full, only its size will be given.

➜ continue

➜ Start

Crawler (continued)



You can specify which folder to scan by typing its path into the **Folder** text box, where you can use the ~ symbol as shorthand for the current user's Home folder, e.g. as in ~/Documents. If that text box is left empty, then clicking on the **Scan** button produces an **Open Folder** dialog, in which you should select the folder which you wish to scan, then click on the **Scan** button.

Scans performed use deep traversal to examine every file and folder within the selected folder. These are time- and memory-intensive. Scanning a large /Applications folder with over two million files can take several minutes and consume 10 GB of memory. However, this memory is managed by macOS and such scans can be run on Macs with less physical memory without difficulty. Scans are also cached whenever possible, so repeating a slightly different scan on the same folder can be very much quicker after an initial scan of the same folder.

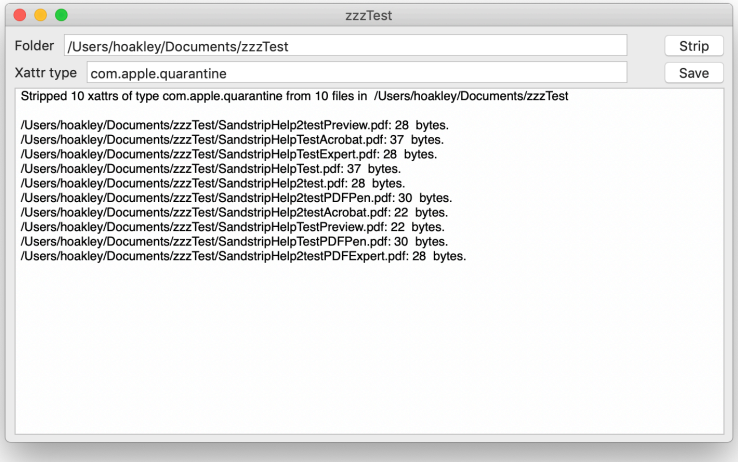
Scans are performed in a background process, with the 'busy spinner' active, to enable other activities while they complete. However, with very large results, brief periods of the 'spinning beachball' may be experienced at the end of a scan while the output is generated. As that has to be performed as part of the app's main process, that is unavoidable.

Click on the **Save** button to save the latest scan results to a text file. Close the Scanner window by clicking on its red close button, in the normal way.

➜ back

Stripper

Use the main window to cut individual attributes from single items. If you want to remove the same extended attribute type from many files, use the **Open Stripper…** command in the **Window** menu.



In this window, enter the xattr type you want to strip in the **Xattr type** textbox. Choose the folders or files you wish to cover either by clicking on the **Strip** button and selecting them in the Open File dialog, or type in the full path to the location in the **Folder** textbox then click on the **Strip** button.

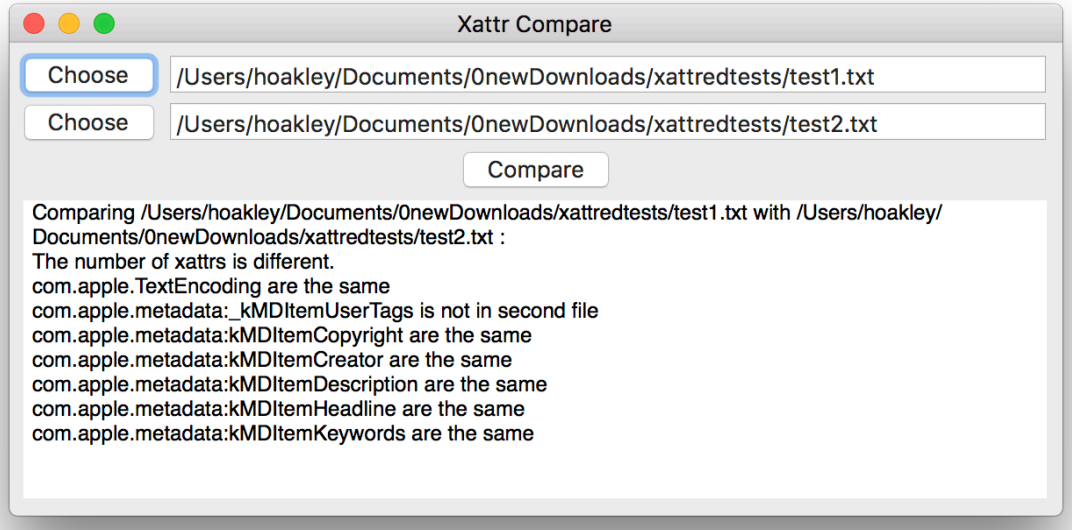
xattred will then scan the item(s) you selected, and remove all the xattrs of that type. It reports details of all items from which it was able to strip the specified xattr. The **Save** button writes the contents of the window to a text file.

⚠ **There is *no Undo:* once stripped, those xattrs are gone for ever.**

⚠ Be cautious of side-effects. For instance, stripping all the quarantine xattrs could allow malware to avoid detection by Gatekeeper/XProtect, thus to affect your Mac.

Compare

To compare all the extended attributes between two files in xattred, open a **Compare** window using the **Open Compare…** command in the **Window** menu.



Click on each of the two **Choose** buttons to choose the two files, displaying their full paths in the boxes at the side; you can also edit those boxes directly. Once you are happy that those paths are correct, click on the **Compare** button, and their similarities and differences will be listed below. xattred doesn't compare data forks, though.

Common xattrs

Among more than 120 types in current use are:

com.apple.cscachefs	iCloud Drive app tag
com.apple.diskimages.fsck	record of disk image integrity check
com.apple.diskimages.recentcksum	disk image checksum
com.apple.FinderInfo	information for the Finder
com.apple.icloud.itemName	iCloud Drive placeholder filename
com.apple.LaunchServices.OpenWith	sets a custom app to open a file
com.apple.logd.metadata	log metadata
com.apple.metadata:com_apple_backup_excludeItem	exclude from backups
com.apple.metadata:kMDItemCopyright	records copyright info
com.apple.metadata:kMDItemCreator	records the app which created a file
com.apple.metadata:kMDItemDescription	arbitrary information about a file
com.apple.metadata:kMDItemDownloadedDate	the download datestamp
com.apple.metadata:kMDItemHeadline	an arbitrary text headline
com.apple.metadata:kMDItemWhereFroms	origin of downloaded file
com.apple.metadata:_kMDItemUserTags	Finder tags
com.apple.metadata:_kTimeMachineNewestSnapshot	old backup details
com.apple.metadata:_kTimeMachineOldestSnapshot	old backup details
com.apple.quarantine	the quarantine flag
com.apple.ResourceFork	a classic Mac resource fork
com.apple.rootless	protects with SIP
com.apple.TextEncoding	reveals text file encoding
com.apple.uuiddb.boot-uuid	log metadata
lock	a log file lock
org.openmetainfo: and org.openmetainfo.time:	families of third-party metadata

Updates

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

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

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

which works properly through the preferences server `cfprefsd`.

Support & further information

Further information about and support for xattred is available from its product page, which is easily accessed through the **xattred Support** command in the **Help** menu, which opens that page in your default browser.

Extended attributes associate metadata with individual files and folders. In Classic MacOS, many files have **resource forks** to contain structured metadata: a classic app, for example, stores definitions of windows, menus, dialogs, etc., in its resource fork. In macOS, files and folders can also have forks, which are implemented as extended attributes: a resource fork becomes resource metadata in a xattr. Extended attributes are quite widely used in other file systems, in Linux and BSD, for example.

macOS and a few applications use xattrs for various purposes. The most prominent is implementing the **quarantine flag** which indicates that an app has been downloaded from the internet and requires full Gatekeeper checks. Other xattrs attach details of the website from which a file was downloaded, and they are used by macOS Server. They are not normally used to store metadata such as EXIF, and those associated with other media files, which are normally part of the file data.

Extended attributes are not stored in the main data for files, but in the **Attributes** area of the volume metadata. As such they are out of reach of normal file tools, and can only be accessed using tools specifically intended to work with xattrs. Each file and folder can have an effectively unlimited number of xattrs, each of which can be more than 100 KB.

Because they are part of the volume metadata, some versions of macOS may not include the space occupied by them when calculating free and used disk space. It is possible to fill a volume with extended attributes and run it out of free space, and that space may also be ignored by services which manage storage use, for example space allocations for clients in a server system. These should be pathological problems, but could also result from malicious activity.

Most file systems to which macOS can write either handle xattrs natively (HFS+, APFS), or macOS uses a scheme to preserve them. NFS is an important exception, and files copied to NFS will have all their xattrs stripped. iCloud also strips many xattrs, but not all.

➜ continue

➜ Start

Further info (continued)

Metadata stored in xattrs is usually hidden from the user, although Finder's **Get Info** may show the content of some xattrs such as com.apple.metadata:kMDItemDownloadedDate and kMDItemWhereFroms. Some third-party apps may expose relevant metadata. For example, better text editors show and allow the setting of the information stored in com.apple.TextEncoding, which records the encoding scheme used by many text files.

The standard tool for working with xattrs is the command **xattr**. xattrs are listed with
xattr -l itempath
where **itempath** is the path to the file or folder to be examined.

Copying xattrs using this command is complex, and requires writing the xattr which is printed in hex form, e.g.
xattr -wx com.apple.FinderInfo "`xattr -px com.apple.FinderInfo thisitem`" thatitem
copies the com.apple.FinderInfo xattr from **thisitem** to **thatitem**.

Some old commands, including cpio, zip, and pax, may omit xattrs unless they are specifically included using an option: if xattrs are to be preserved when using such commands, check with the man page and preferably perform a small-scale test before proceeding.

Occasionally, bugs in specific apps can attach incorrect or very large xattrs to files quite inappropriately. These can in turn cause some services to fail, most commonly the mdworker daemon responsible for indexing metadata for Spotlight. These usually present with bizarre symptoms, and until they are recognised as being caused by xattrs, they remain baffling.

In certain circumstances, trying to open a document in macOS 10.8 and later can result in a security error and refusal.
The three conditions required for this security error to occur are:

1. The document must have a **quarantine flag** attached to it.
2. The document must have an **OpenWith** extended attribute attached to it, setting a non-default app to open that document.
3. The document must be opened using the Finder, e.g. by double-clicking it, or dropping it onto the app icon, or another mechanism which is handled by LaunchServices.

All three conditions must be satisfied.

➜ continue

Further info (concluded)

The **quarantine flag**, an extended attribute of type com.apple.quarantine, can be set either when it was downloaded from the Internet using a method which sets this flag, such as Safari, or by opening or saving the document in an app which runs in a sandbox, such as Apple's apps bundled with macOS, all those supplied by the App Store, and many others supplied by other means. It is not normally attached by apps which have merely been hardened and/or notarized.

The **OpenWith** extended attribute of type com.apple.LaunchServices.OpenWith is attached to documents when the user changes the app set to open that document in the Finder's Get Info dialog. If the user sets that back to the default app used to open that document type, although the extended attribute remains in place, the security mechanism recognises that the default app is being used and this mechanism isn't then triggered.

Opening the document from within an app doesn't act via LaunchServices, and is unaffected by this mechanism.

If those three conditions are met, a check of the document by XProtect returns an error -67062, which results in display of the error alert, and that attempt to open the document is terminated. The alert gives no option to open the document by any other means.

macOS provides various ways of working around this security control, of which the simplest is to use the Finder's contextual menu to **Open** the document. A security dialog then gives the user the option to proceed to open that document using the non-default app. If the user agrees, the quarantine flag is changed to indicate that the document can be opened without future blocking, and it is opened in the chosen app. The quarantine flag is not removed from the document.

Other simple ways to work around this include opening the document using the **Open** command within the app, and changing its **OpenWith** setting to the default app, which of course then leads to the document being opened by that app.

If you want to prevent this from happening proactively, macOS doesn't provide any direct way to inspect or edit the quarantine flag, but my free utilities **Sandstrip** and **Pratique** do. The former strips only quarantine flags which have been set by the sandbox; the latter changes quarantine flags on selected documents to indicate that they have cleared quarantine.

➜ back
➜ Change list

Change list

1.3 release:

- Universal App for both Intel and Apple Silicon Macs.

1.2 release:

- altered views to improve behaviour on resizing
- added Add and Strip flag
- enabled multiple selection and cut/copy/paste
- added keystroke shortcuts.

1.1 release:

- added automatic check and download of updates.

1.0 release:

- added Stripper
- added PDF Help.

1.0b10:

- added support for cursor keys to navigate list of xattrs
- ported to Swift 4.2.1 and Xcode 10.1.

1.0b9:

- added privacy usage strings to Info.plist
- added Privacy settings window to Help
- notarized for Mojave.

1.0b8:

- added Compare feature and window.

1.0b7:

- added Browse updates menu command
- fixed data access conflict
- added full support for Dark Mode
- built with Xcode 10ß.

1.0b6:

- enabled Open menu command
- enabled drag and drop to open
- changed Save to Save as…

1.0b5:

- added functionality from unreleased app in Xattr Crawler window
- put deep directory traversal in crawler into background
- added busy spinner.

1.0b4:

- replaced quarantine xattr text output with fork size information.

1.0b3:

- tidied File menu to remove Open command
- changed Save behaviour to save all xattrs when none is selected
- tweaked text outputs
- improved document naming.

1.0b2:

- brought quarantine xattr code to compatibility with El Capitan
- built to run on El Capitan and later.

1.0b1:

- reworked xattr class typing
- fixed handling of plists for xattrs using property lists, so extending full plist editing
- reload data when switched format in the editor
- made editor sheet resizable.

0.7a1:

- added segue sheet content editor
- enabled text and hex editing of xattrs, and basic binary plist editing (limited to NSString)
- fixed crashing bug when pasting from pasteboard which had no xattr data
- added type info to xattr class
- restructured xattr class to generate all string output
- enable/disable/retitle action buttons on context.

0.6a2:

- altered settings for the file open dialog to reveal hidden items and see inside bundles
- tidied view refresh following a double-click to reveal binary plist data.

0.6a1:

- added NSCoder support to resource class
- added cut/copy/paste support with clipboard
- couldn't get Edit menu support working, so added buttons in lieu.

0.5a1 (unreleased):

- rewritten to use resource classes
- TableView much improved
- added decoding of binary plists
- changed text view content to current xattr only.

0.4a3:

- fixed window behaviour on resizing.

0.4a2:

- ported first to Swift 3.2 then to 4.0, and built in Xcode 9 to run on Sierra and High Sierra. Fingers crossed!

0.3a1:

- rewrote button actions to operate on currently Opened file or folder, rather than having to open an item for every action
- architectural changes to retain xattr lists and data once Opened, and update xattrs following button actions
- added TableView listing xattrs and their sizes

0.2a3:

- first proper alpha release.

4 August 2020.