# OpenCore

Reference Manual (1.0.5.6)

[2025.08.11]

Figure 1. Directory Structure

When directory boot is used, the directory structure used should follow the descriptions in the Directory Structure figure. Available entries include:

- `BOOTx64.efi` or `BOOTIa32.efi`
  Initial bootstrap loaders, which load `OpenCore.efi`. `BOOTx64.efi` is loaded by the firmware by default consistent with the UEFI specification. However, it may also be renamed and put in a custom location to allow OpenCore coexist alongside operating systems, such as Windows, that use `BOOTx64.efi` files as their loaders. Refer to the `LauncherOption` property for details.
- `boot`
  Duet bootstrap loader, which initialises the UEFI environment on legacy BIOS firmware and loads `OpenCore.efi` similarly to other bootstrap loaders. A modern Duet bootstrap loader will default to `OpenCore.efi` on the same partition when present.
- `ACPI`
  Directory used for storing supplemental ACPI information for the `ACPI` section.
- `Drivers`
  Directory used for storing supplemental `UEFI` drivers for `UEFI` section.
- `Kexts`
  Directory used for storing supplemental kernel information for the `Kernel` section.
- `Resources`
  Directory used for storing media resources such as audio files for screen reader support. Refer to the `UEFI Audio Properties` section for details. This directory also contains image files for graphical user interface. Refer to the OpenCanopy section for details.
- `Tools`
  Directory used for storing supplemental tools.
- `OpenCore.efi`
  Main booter application responsible for operating system loading. The directory `OpenCore.efi` resides in is called the `root directory`, which is set to `EFI\OC` by default. When launching `OpenCore.efi` directly or through a custom launcher however, other directories containing `OpenCore.efi` files are also supported.
- `config.plist`
  `OC Config`.
- `vault.plist`
  Hashes for all files potentially loadable by `OC Config`.
- `vault.sig`
  Signature for `vault.plist`.
- `SysReport`
  Directory containing system reports generated by `SysReport` option.
- `nvram.plist`
  OpenCore variable import file.
- `nvram.fallback`
  OpenCore variable import fallback file.
- `nvram.used`
  Renamed previous OpenCore variable import file after switch to fallback file.
- `opencore-YYYY-MM-DD-HHMMSS.txt`
  OpenCore log file.
- `panic-YYYY-MM-DD-HHMMSS.txt`
  Kernel panic log file.

*Note*: It is not guaranteed that paths longer than `OC_STORAGE_SAFE_PATH_MAX` (~~128~~ 192 characters including the 0-terminator) will be accessible within OpenCore.

## 3.2 Installation and Upgrade

To install OpenCore, replicate the Configuration Structure described in the previous section in the EFI volume of a GPT partition. While corresponding sections of this document provide some information regarding external resources such as ACPI tables, UEFI drivers, or kernel extensions (kexts), completeness of the matter is out of the scope of this document. Information about kernel extensions may be found in a separate Kext List document available in the OpenCore repository. Vaulting information is provided in the Security Properties section of this document.

**Failsafe**: `false`
**Description**: Provide custom KASLR slide on low memory.

This option performs memory map analysis of the firmware and checks whether all slides (from `1` to `255`) can be used. As `boot.efi` generates this value randomly with `rdrand` or pseudo randomly `rdtsc`, there is a chance of boot failure when it chooses a conflicting slide. In cases where potential conflicts exist, this option forces macOS to select a pseudo random value from the available values. This also ensures that the `slide=` argument is never passed to the operating system (for security reasons).

*Note*: The need for this quirk is determined by the `OCABC: Only N/256 slide values are usable!` message in the debug log.

17. `ProvideMaxSlide`
**Type**: `plist integer`
**Failsafe**: `0`
**Description**: Provide maximum KASLR slide when higher ones are unavailable.

This option overrides the maximum slide of 255 by a user specified value between 1 and 254 (inclusive) when `ProvideCustomSlide` is enabled. It is assumed that modern firmware allocates pool memory from top to bottom, effectively resulting in free memory when slide scanning is used later as temporary memory during kernel loading. When such memory is not available, this option stops the evaluation of higher slides.

*Note*: The need for this quirk is determined by random boot failures when `ProvideCustomSlide` is enabled and the randomized slide falls into the unavailable range. When `AppleDebug` is enabled, the debug log typically contains messages such as `AAPL: [EB|'LD:LKC] } Err(0x9)`. To find the optimal value, append `slide=X`, where `X` is the slide value, to the `boot-args` and select the largest one that does not result in boot failures.

18. `RebuildAppleMemoryMap`
**Type**: `plist boolean`
**Failsafe**: `false`
**Description**: Generate macOS compatible Memory Map.

The Apple kernel has several limitations on parsing the UEFI memory map:

- The Memory map size must not exceed 4096 bytes as the Apple kernel maps it as a single 4K page. As some types of firmware can have very large memory maps, potentially over 100 entries, the Apple kernel will crash on boot.
- The Memory attributes table is ignored. `EfiRuntimeServicesCode` memory statically gets `RX` permissions while all other memory types get `RW` permissions. As some firmware drivers may write to global variables at runtime, the Apple kernel will crash at calling UEFI runtime services unless the driver `.data` section has a `EfiRuntimeServicesData` type.
- Apple kernel Memory map entry consolidation may work incorrectly for low memory descriptors, which are initially marked as preallocated by Apple kernel, but then may get consolidated and lose their preallocation status due to a bug. Since Apple kernel later frees low memory, this may result in use-after-free errors and various kinds of kernel panics at boot time. The issue was fixed in Mac OS X 10.7 kernel.

To workaround these limitations, this quirk applies memory attribute table permissions to the memory map passed to the Apple kernel and optionally attempts to unify contiguous slots of similar types if the resulting memory map exceeds 4 KB.

*Note 1*: Since several types of firmware come with incorrect memory protection tables, this quirk often comes paired with `SyncRuntimePermissions`.

*Note 2*: The need for this quirk is determined by early boot failures. This quirk replaces `EnableWriteUnprotector` on firmware supporting Memory Attribute Tables (MAT). ~~This quirk is typically unnecessary when using `OpenDuetPkg` but may be required to boot Mac OS X 10.6, and earlier, for reasons that are as yet unclear.~~

19. `ResizeAppleGpuBars`
**Type**: `plist integer`
**Failsafe**: `-1`
**Description**: Reduce GPU PCI BAR sizes for compatibility with macOS.

This quirk reduces GPU PCI BAR sizes for Apple macOS up to the specified value or lower if it is unsupported. The specified value follows PCI Resizable BAR spec. While Apple macOS supports a theoretical 1 GB maximum,

in practice all non-default values may not work correctly. For this reason the only supported value for this quirk is the minimal supported BAR size, i.e. `0`. Use `-1` to disable this quirk.

For development purposes one may take risks and try other values. Consider a GPU with 2 BARs:

- `BAR0` supports sizes from 256 MB to 8 GB. Its value is 4 GB.
- `BAR1` supports sizes from 2 MB to 256 MB. Its value is 256 MB.

*Example 1*: Setting `ResizeAppleGpuBars` to 1 GB will change `BAR0` to 1 GB and leave `BAR1` unchanged.
*Example 2*: Setting `ResizeAppleGpuBars` to 1 MB will change `BAR0` to 256 MB and `BAR0` to 2 MB.
*Example 3*: Setting `ResizeAppleGpuBars` to 16 GB will make no changes.

*Note*: See `ResizeGpuBars` quirk for general GPU PCI BAR size configuration and more details about the technology.

20. `SetupVirtualMap`
**Type**: `plist boolean`
**Failsafe**: `false`
**Description**: Setup virtual memory at `SetVirtualAddresses`.

Some types of firmware access memory by virtual addresses after a `SetVirtualAddresses` call, resulting in early boot crashes. This quirk workarounds the problem by performing early boot identity mapping of assigned virtual addresses to physical memory.

*Note 1*: The need for this quirk is determined by early boot failures.

*Note 2*: This quirk is not compatible with 32-bit kernels.

21. `SignalAppleOS`
**Type**: `plist boolean`
**Failsafe**: `false`
**Description**: Report macOS being loaded through OS Info for any OS.

This quirk is useful on Mac firmware, which loads different operating systems with different hardware configurations. For example, it is supposed to enable Intel GPU in Windows and Linux in some dual-GPU MacBook models.

22. `SyncRuntimePermissions`
**Type**: `plist boolean`
**Failsafe**: `false`
**Description**: Update memory permissions for the runtime environment.

Some types of firmware fail to properly handle runtime permissions:

- They incorrectly mark `OpenRuntime` as not executable in the memory map.
- They incorrectly mark `OpenRuntime` as not executable in the memory attributes table.
- They lose entries from the memory attributes table after `OpenRuntime` is loaded.
- They mark items in the memory attributes table as read-write-execute.

This quirk attempts to update the memory map and memory attributes table to correct this.

*Note*: The need for this quirk is indicated by early boot failures (note: includes halt at black screen as well as more obvious crash). Particularly likely to affect early boot of Windows or Linux (but not always both) on affected systems. Only firmware released after 2017 is typically affected.

When `--gpio-setup` is enabled (i.e. non-zero), then `0` is a special value for `--gpio-pins`, meaning that the pin mask will be auto-generated based on the reported number of GPIO pins on the specified codec (see `AudioCodec`), e.g. if the codec's audio out function group reports 4 GPIO pins, a mask of `0xF` will be used. The value in use can be seen in the debug log in a line such as:

```
HDA: GPIO setup on pins 0x0F - Success
```

Values for driver parameters can be specified in hexadecimal beginning with `0x` or in decimal, e.g. `--gpio-pins=0x12` or `--gpio-pins=18`.

- `--restore-nosnoop` - Boolean flag, enabled if present.

  AudioDxe clears the Intel HDA No Snoop Enable (NSNPEN) bit. On some systems, this change must be reversed on exit in order to avoid breaking sound in Windows or Linux. If so, this flag should be added to AudioDxe driver arguments. Not enabled by default, since restoring the flag can prevent sound from working in macOS on some other systems.

- `--use-conn-none` - Boolean flag, enabled if present.

  On some sound cards enabling this option will enable additional usable audio channels (e.g. the bass or treble speaker of a pair, where only one is found without it).

  *Note*: Enabling this option may increase the available channels, in which case any custom setting of `AudioOutMask` may need to be changed to match the new channel list.

## 11.11 OpenVariableRuntimeDxe

Provides in-memory emulated NVRAM implementation. This can be useful on systems with fragile (e.g. MacPro5,1, see discussion linked from this forum post) or incompatible NVRAM implementations. This driver is included by default in OpenDuet.

In addition to installing emulated NVRAM, this driver additionally installs an OpenCore compatible protocol enabling the following:

- NVRAM values are loaded from `NVRAM/nvram.plist` (or from `NVRAM/nvram.fallback` if it is present and `NVRAM/nvram.plist` is missing) on boot
- The Reset NVRAM option installed by the `ResetNvramEntry` driver removes the above files instead of affecting underlying NVRAM
- `CTRL+Enter` in the OpenCore bootpicker updates or creates `NVRAM/nvram.plist`

Recommended configuration settings for this driver:

- `OpenVariableRuntimeDxe.efi` loaded using `LoadEarly=true`. OpenDuet users should not load this driver, as ~~it~~ a firmware driver serving the same purpose is included in OpenDuet.
- `OpenRuntime.efi` specified after `OpenVariableRuntimeDxe.efi` (when applicable), also loaded using `LoadEarly=true` for correct operation of `RequestBootVarRouting`.
  - `RequestBootVarRouting` is never strictly needed while using emulated NVRAM, but it can be convenient to leave it set on a system which needs to switch between real and emulated NVRAM.
  - `RequestBootVarRouting` is never required on an OpenDuet system, since there are no BIOS-managed boot entries to protect, therefore on OpenDuet recommended settings are `LoadEarly=false` for `OpenRuntime.efi` and `RequestBootVarRouting=false`.
- `LegacySchema` populated.
  - For simpler testing (allows arbitrary test variables), and future-proofing against changes in the variables required by macOS updates, use `<string>*</string>` settings, as described in notes below.
  - For increased security, populate sections with known required keys only, as shown in OpenCore's sample `.plist` files.
- `ExposeSensitiveData` with at least bit `0x1` set to make `boot-path` variable containing the OpenCore EFI partition UUID available to the `Launchd.command` script.

Variable loading happens prior to the NVRAM `Delete` (and `Add`) phases. Unless `LegacyOverwrite` is enabled, it will not overwrite any existing variable. Variables allowed for loading and for saving with `CTRL+Enter` must be specified in `LegacySchema`.