



OpenCore

Reference Manual (~~0.6.9~~0.7.0)

[2021.06.04]

- Enabling XCPM support for an unsupported CPU variant.

Note 1: It may also be the case that the CPU model is supported but there is no power management supported (e.g. virtual machines). In this case, `MinKernel` and `MaxKernel` can be set to restrict CPU virtualisation and dummy power management patches to the particular macOS kernel version.

Note 2: Only the value of `EAX`, which represents the full CPUID, typically needs to be accounted for and remaining bytes should be left as zeroes. The byte order is Little Endian. For example, `C3 06 03 00` stands for CPUID `0x0306C3` (Haswell).

Note 3: For XCPM support it is recommended to use the following combinations.

- Haswell-E (0x0306F2) to Haswell (0x0306C3):
`Cpuid1Data: C3 06 03 00 00 00 00 00 00 00 00 00 00 00 00 00`
`Cpuid1Mask: FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
- Broadwell-E (0x0406F1) to Broadwell (0x0306D4):
`Cpuid1Data: D4 06 03 00 00 00 00 00 00 00 00 00 00 00 00 00`
`Cpuid1Mask: FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
- Rocket Lake (0x0A0670) to Comet Lake (0x0906EB):
`Cpuid1Data: EB 06 09 00 00 00 00 00 00 00 00 00 00 00 00 00`
`Cpuid1Mask: FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
- [Comet Lake U62 \(0x0A0660\) to Comet Lake U42 \(0x0806EC\)](#):
`Cpuid1Data: EC 06 08 00 00 00 00 00 00 00 00 00 00 00 00 00`
`Cpuid1Mask: FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`

Note 4: Be aware that the following configurations are unsupported by XCPM (at least out of the box):

- Consumer Ivy Bridge (0x0306A9) as Apple disabled XCPM for Ivy Bridge and recommends legacy power management for these CPUs. `_xcpm_bootstrap` should manually be patched to enforce XCPM on these CPUs instead of this option.
- Low-end CPUs (e.g. Haswell+ Pentium) as they are not supported properly by macOS. Legacy workarounds for older models can be found in the `Special NOTES` section of [acidanthera/bugtracker#365](#).

2. `Cpuid1Mask`

Type: plist data, 16 bytes

Failsafe: All zero

Description: Bit mask of active bits in `Cpuid1Data`.

When each `Cpuid1Mask` bit is set to 0, the original CPU bit is used, otherwise set bits take the value of `Cpuid1Data`.

3. `DummyPowerManagement`

Type: plist boolean

Failsafe: false

Requirement: 10.4

Description: Disables `AppleIntelCpuPowerManagement`.

Note 1: This option is a preferred alternative to `NullCpuPowerManagement.kext` for CPUs without native power management driver in macOS.

Note 2: While this option is typically needed to disable `AppleIntelCpuPowerManagement` on unsupported platforms, it can also be used to disable this kext in other situations (e.g. with `Cpuid1Data` left blank).

4. `MaxKernel`

Type: plist string

Failsafe: Empty

Description: Emulates CPUID and applies `DummyPowerManagement` on specified macOS version or older.

Note: Refer to the `Add MaxKernel` description for matching logic.

5. `MinKernel`

Type: plist string

Failsafe: Empty

Description: Emulates CPUID and applies `DummyPowerManagement` on specified macOS version or newer.

Note: Refer to the `Add MaxKernel` description for matching logic.

Failsafe: false
Requirement: 11
Description: Force x86 scheme for IMG4 verification.

Note: This option is required on virtual machines when using `SecureBootModel` different from `x86legacy`.

12. `IncreasePciBarSize`

Type: plist boolean
Failsafe: false
Requirement: 10.10
Description: Increases 32-bit PCI bar size in `IOPCIFamily` from 1 to 4 GBs.

Note: This option should be avoided whenever possible. A need for this option indicates misconfigured or defective firmware.

13. `LapicKernelPanic`

Type: plist boolean
Failsafe: false
Requirement: 10.6 (64-bit)
Description: Disables kernel panic on LAPIC interrupts.

14. `LegacyCommpage`

Type: plist boolean
Failsafe: false
Requirement: 10.4 - 10.6
Description: Replaces the default 64-bit `commpage` bcopy implementation with one that does not require SSSE3, useful for legacy platforms. This prevents a `commpage no match for last` panic due to no available 64-bit bcopy functions that do not require SSSE3.

15. `PanicNoKextDump`

Type: plist boolean
Failsafe: false
Requirement: 10.13 (not required for older)
Description: Prevent kernel from printing kext dump in the panic log preventing from observing panic details. Affects 10.13 and above.

16. `PowerTimeoutKernelPanic`

Type: plist boolean
Failsafe: false
Requirement: 10.15 (not required for older)
Description: Disables kernel panic on `setPowerState` timeout.

An additional security measure was added to macOS Catalina (10.15) causing kernel panic on power change timeout for Apple drivers. Sometimes it may cause issues on misconfigured hardware, notably digital audio, which sometimes fails to wake up. For debug kernels `setpowerstate_panic=0` boot argument should be used, which is otherwise equivalent to this quirk.

17. [ProvideCurrentCpuInfo](#)

[Type:](#) plist boolean
[Failsafe:](#) false
[Requirement:](#) 10.8
[Description:](#) Provides current CPU info to the kernel.

[This quirk currently provides the correct TSC and FSB values to the kernel, as well as disables CPU topology validation.](#)

[Note:](#) These patches currently target Microsoft Hyper-V and may need to be extended for other purposes.

18. `SetApfsTrimTimeout`

Type: plist integer
Failsafe: -1
Requirement: 10.14 (not required for older)
Description: Set trim timeout in microseconds for APFS filesystems on SSDs.

An entry is considered auxiliary when at least one of the following applies:

- Entry is macOS recovery.
- Entry is macOS Time Machine.
- Entry is explicitly marked as **Auxiliary**.
- Entry is system (e.g. **Reset NVRAM**).

To display all entries, the picker menu can be reloaded into “Extended Mode” by pressing the **Spacebar** key. Hiding auxiliary entries may increase boot performance on multi-disk systems.

4. LauncherOption

Type: plist string

Failsafe: Disabled

Description: Register the launcher option in the firmware preferences for persistence.

Valid values:

- **Disabled** — do nothing.
- **Full** — create or update the top priority boot option in UEFI variable storage at bootloader startup.
 - For this option to work, **RequestBootVarRouting** is required to be enabled.
- **Short** — create a short boot option instead of a complete one.
 - This variant is useful for some older types of firmware, typically from Insyde, that are unable to manage full device paths.
- **System** — create no boot option but assume specified custom option is blessed.
 - This variant is useful when relying on **ForceBooterSignature** quirk and OpenCore launcher path management happens through **bless** utilities without involving OpenCore.

This option allows integration with third-party operating system installation and upgrades (which may overwrite the `\EFI\BOOT\BOOTx64.efi` file). The `BOOTx64.efi` file is no longer used for bootstrapping OpenCore if a custom option is created. The custom path used for bootstrapping can be specified by using the **LauncherPath** option.

Note 1: Some types of firmware may have NVRAM implementation flaws, no boot option support, or other incompatibilities. While unlikely, the use of this option may result in boot failures and should only be used exclusively on boards known to be compatible. Refer to [acidanthera/bugtracker#1222](#) for some known issues affecting Haswell and other boards.

Note 2: While NVRAM resets executed from OpenCore would not typically erase the boot option created in **Bootstrap**, executing NVRAM resets prior to loading OpenCore will erase the boot option. Therefore, for significant implementation updates, such as was the case with OpenCore 0.6.4, an NVRAM reset should be executed with **Bootstrap** disabled, after which it can be re-enabled.

5. LauncherPath

Type: plist string

Failsafe: Default

Description: Launch path for the **LauncherOption** property.

Default points to `OpenCore.efi`. User specified paths, e.g. `\EFI\SomeLauncher.efi`, can be used to provide custom loaders, which are supposed to load `OpenCore.efi` themselves.

6. PickerAttributes

Type: plist integer

Failsafe: 0

Description: Sets specific attributes for the OpenCore picker.

Different OpenCore pickers may be configured through the attribute mask containing OpenCore-reserved (BIT0~BIT15) and OEM-specific (BIT16~BIT31) values.

Current OpenCore values include:

- 0x0001 — `OC_ATTR_USE_VOLUME_ICON`, provides custom icons for boot entries:
~~For Tools, OpenCore will attempt loading a custom icon and fallback to a default icon on failure:–~~
 - ~~`ResetNVRAM` — `Resources\Image\ResetNVRAM.icns` — `ResetNVRAM.icns` from icons directory.–~~
 - ~~`Tools\<TOOL_RELATIVE_PATH>.icns` — icon near the tool file with appended `.icns` extension.–~~

~~For custom boot Entries, OpenCore will attempt loading a custom icon and fallback to the volume icon or the default icon on failure:-~~

- ~~- `<ENTRY_PATH>.icns` — icon near the entry file with appended `.icns` extension.~~

~~For all other entries, OpenCore will attempt loading a volume icon [volume icon](#) by searching as follows, and will fallback to the default icon on failure:~~

- ~~- `.VolumeIcon.icns` file at Preboot volume in per-volume directory (`/System/Volumes/Preboot/{GUID}/` when mounted at the default location within macOS) for APFS (if present).~~
- ~~- `.VolumeIcon.icns` file at the Preboot volume root (`/System/Volumes/Preboot/`, when mounted at the default location within macOS) for APFS (otherwise).~~
- ~~- `.VolumeIcon.icns` file at the volume root for other filesystems.~~

Note 1: The Apple picker partially supports placing a volume icon file at the operating system's Data volume root, `/System/Volumes/Data/`, when mounted at the default location within macOS. This approach is flawed: the file is neither accessible to OpenCanopy nor to the Apple picker when FileVault 2, which is meant to be the default choice, is enabled. Therefore, OpenCanopy does not attempt supporting Apple's approach. A volume icon file may be placed at the root of the Preboot volume for compatibility with both OpenCanopy and the Apple picker, or use the Preboot per-volume location as above with OpenCanopy as a preferred alternative to Apple's approach.

Note 2: Be aware that using a volume icon on any drive overrides the normal OpenCore picker behaviour for that drive of selecting the appropriate icon depending on whether the drive is internal or external.

- `0x0002` — `OC_ATTR_USE_DISK_LABEL_FILE`, provides custom ~~rendered~~ [prerendered](#) titles for boot entries ~~:-~~
- [from](#) `.disk_label` (`.disk_label_2x`) file ~~near~~ [next to the](#) bootloader for all filesystems.
- ~~`<TOOL_NAME>.lbl` (`<TOOL_NAME>.l2x`) file near tool for Tools.~~ Prerendered labels can be generated via the `disklabel` utility or the `bless` command. When disabled or missing ~~text labels,~~ [label text in](#) (`.contentDetails` or `.disk_label.contentDetails`) ~~are to be rendered instead~~ [will be rendered if present instead, otherwise the entry name itself will be rendered.](#)
- `0x0004` — `OC_ATTR_USE_GENERIC_LABEL_IMAGE`, provides predefined label images for boot entries without custom entries. This may however give less detail for the actual boot entry.
- `0x0008` — `OC_ATTR_HIDE_THEMED_ICONS`, prefers builtin icons for certain icon categories to match the theme style. For example, this could force displaying the builtin Time Machine icon. Requires `OC_ATTR_USE_VOLUME_ICON`.
- `0x0010` — `OC_ATTR_USE_POINTER_CONTROL`, enables pointer control in the OpenCore picker when available. For example, this could make use of mouse or trackpad to control UI elements.
- `0x0020` — `OC_ATTR_SHOW_DEBUG_DISPLAY`, enable display of additional timing and debug information, in Builtin picker in `DEBUG` and `NOOPT` builds only.
- `0x0040` — `OC_ATTR_USE_MINIMAL_UI`, use minimal UI display, no Shutdown or Restart buttons, affects OpenCanopy and builtin picker.
- ~~`0x0080` — `OC_ATTR_USE_FLAVOUR_ICON`, provides flexible boot entry content description, suitable for picking the best media across different content sets:~~ [When enabled, the entry icon in OpenCanopy and the audio assist entry sound in OpenCanopy and builtin boot picker are chosen by something called content flavour. To determine content flavour the following algorithm is used:](#)
 - ~~- For a Tool the value is read from `Flavour` field.~~
 - ~~- For an automatically discovered entry it is read from the `.contentFlavour` file next to the bootloader, if present.~~
 - ~~- For a custom entry it is read from the `.contentFlavour` file next to the bootloader if `Flavour` is `Auto`, otherwise specified via the `Flavour` value itself.~~
 - ~~- If read flavour is `Auto` or there is no `.contentFlavour`, entry flavour is chosen based on the entry type (e.g. Windows automatically gets Windows flavour).~~

[The Flavour value is a sequence of : separated names limited to 64 characters of printable 7-bit ASCII. This is designed to support up to approximately five names. Each name refers to a flavour, with the first name having the highest priority and the last name having the lowest priority. Such a structure allows describing an entry in a more specific way, with icons selected flexibly depending on support by the audio-visual pack. A missing audio or icon file means the next flavour should be tried, and if all are missing the choice happens based on the type of the entry. Example flavour values: `BigSur:Apple`, `Windows10:Windows`, `OpenShell:UEFIShell:Shell`.](#)

[Using flavours means that you can switch between icon sets easily, with the flavour selecting the best available icons from each set. E.g. specifying icon flavour `Debian:Linux` will use the icon `Debian.icns` if](#)

provided, then will try `Linux.icns`, then will fall back to the default for an OS, which is `HardDrive.icns`.

Things to keep in mind:

- For security reasons `Ext<Flavour>.icns` and `<Flavour>.icns` are both supported, and only `Ext<Flavour>.icns` will be used if the entry is on an external drive (followed by default fallback `ExtHardDrive.icns`).
- Where both apply `.VolumeIcon.icns` takes precedence over `.contentFlavour`.
- In order to allow icons and audio assist to work correctly for tools (e.g. for UEFI Shell), system default boot entry icons (see `Docs/Flavours.md`) specified in the `Flavour` setting for `Tools` or `Entries` will continue to apply even when flavour is disabled. Non-system icons will be ignored in this case. In addition, the flavours `UEFIShell` and `NVRAMReset` are given special processing, identifying their respective tools to apply correct audio-assist, default builtin labels, etc.
- A list of recommended flavours is provided in `Docs/Flavours.md`.

7. PickerAudioAssist

Type: plist boolean

Failsafe: false

Description: Enable screen reader by default in the OpenCore picker.

For the macOS bootloader, screen reader preference is set in the `preferences.efires` archive in the `isV0Enabled.int32` file and is controlled by the operating system. For OpenCore screen reader support, this option is an independent equivalent. Toggling screen reader support in both the OpenCore picker and the macOS bootloader FileVault 2 login window can also be done by using the `Command + F5` key combination.

Note: The screen reader requires working audio support. Refer to the `UEFI Audio Properties` section for details.

8. PollAppleHotKeys

Type: plist boolean

Failsafe: false

Description: Enable modifier hotkey handling in the OpenCore picker.

In addition to `action hotkeys`, which are partially described in the `PickerMode` section and are typically handled by Apple BDS, modifier keys handled by the operating system bootloader (`boot.efi`) also exist. These keys allow changing the behaviour of the operating system by providing different boot modes.

On certain firmware, using modifier keys may be problematic due to driver incompatibilities. To workaroud this problem, this option allows registering certain hotkeys in a more permissive manner from within the OpenCore picker. Such extensions include support for tapping on key combinations before selecting the boot item, and for reliable detection of the `Shift` key when selecting the boot item, in order to work around the fact that hotkeys which are continuously held during boot cannot be reliably detected on many PS/2 keyboards.

This list of known `modifier hotkeys` includes:

- `CMD+C+MINUS` — disable board compatibility checking.
- `CMD+K` — boot release kernel, similar to `kcsuffix=release`.
- `CMD+S` — single user mode.
- `CMD+S+MINUS` — disable KASLR slide, requires disabled SIP.
- `CMD+V` — verbose mode.
- `Shift+Enter`, `Shift+Index` — safe mode, may be used in combination with `CTRL+Enter`, `CTRL+Index`.

9. ShowPicker

Type: plist boolean

Failsafe: false

Description: Show a simple picker to allow boot entry selection.

10. TakeoffDelay

Type: plist integer, 32 bit

Failsafe: 0

Description: Delay in microseconds executed before handling the OpenCore picker startup and `action hotkeys`.

Introducing a delay may give extra time to hold the right `action hotkey` sequence to, for instance, boot into recovery mode. On some platforms, setting this option to a minimum of 5000–10000 microseconds may be required to access `action hotkeys` due to the nature of the keyboard driver.

Type: plist string

Failsafe: Auto

Description: Choose specific icon set to be used for boot management.

~~The following values are supported~~ An icon set is a directory path relative to `Resources/Image`, where the icons and an optional manifest are located. It is recommended for the artists to use provide their sets in the `Vendor/Set` format, e.g. `Acidanthera\GoldenGate`.

Sample resources provided as a part of OcBinaryData repository provide the following icon set:

- ~~Auto~~ `Acidanthera\GoldenGate` — ~~Automatically select one set of icons based on the DefaultBackground colour~~ macOS 11 styled icon set.
- ~~Default~~ `Acidanthera\Syrah` — ~~Normal icon set (without prefix)~~ macOS 10.10 styled icon set.
- ~~Old~~ `Acidanthera\Chardonnay` — ~~Vintage icon set (Old filename prefix)~~ macOS 10.4 styled icon set.

For convenience purposes there also are predefined aliases:

- ~~Modern~~ `Auto` — ~~Nouveau icon set (Automatically select one set of icons based on the ModernDefaultBackground filename prefix)~~ colour: `Acidanthera\GoldenGate` for Syrah Black and `Acidanthera\Chardonnay` for Light Gray.
- ~~Other value~~ `Default` — ~~Custom icon set if supported by installed resources~~ `Acidanthera\GoldenGate`.

8.4 Debug Properties

1. AppleDebug

Type: plist boolean

Failsafe: false

Description: Enable writing the `boot.efi` debug log to the OpenCore log.

Note: This option only applies to 10.15.4 and newer.

2. ApplePanic

Type: plist boolean

Failsafe: false

Description: Save macOS kernel panic output to the OpenCore root partition.

The file is saved as `panic-YYYY-MM-DD-HHMMSS.txt`. It is strongly recommended to set the `keepsyms=1` boot argument to see debug symbols in the panic log. In cases where it is not present, the `kpdescribe.sh` utility (bundled with OpenCore) may be used to partially recover the stacktrace.

Development and debug kernels produce more useful kernel panic logs. Consider downloading and installing the `KernelDebugKit` from developer.apple.com when debugging a problem. To activate a development kernel, the boot argument `kcsuffix=development` should be added. Use the `uname -a` command to ensure that the current loaded kernel is a development (or a debug) kernel.

In cases where the OpenCore kernel panic saving mechanism is not used, kernel panic logs may still be found in the `/Library/Logs/DiagnosticReports` directory.

Starting with macOS Catalina, kernel panics are stored in JSON format and thus need to be preprocessed before passing to `kpdescribe.sh`:

```
cat Kernel.panic | grep macOSProcessedStackshotData |  
python -c 'import json,sys;print(json.load(sys.stdin)["macOSPanicString"])'
```

3. DisableWatchDog

Type: plist boolean

Failsafe: false

Description: Some types of firmware may not succeed in booting the operating system quickly, especially in debug mode. This results in the watchdog timer aborting the process. This option turns off the watchdog timer.

4. DisplayDelay

Type: plist integer

Failsafe: 0

Description: Delay in microseconds executed after every printed line visible onscreen (i.e. console).

UEFI variable log does not include some messages and has no performance data. To maintain system integrity, the log size is limited to 32 kilobytes. Some types of firmware may truncate it much earlier or drop completely if they have no memory. Using the `non-volatile` flag will cause the log to be written to NVRAM flash after every printed line.

To obtain UEFI variable logs, use the following command in macOS:

```
nvrnm 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-log |  
awk '{gsub(/%0d%0a%00/, ""); gsub(/%0d%0a/, "\n")}1'
```

Warning: Certain firmware appear to have defective NVRAM garbage collection. As a result, they may not be able to always free space after variable deletion. Do not enable `non-volatile` NVRAM logging on such devices unless specifically required.

While the OpenCore boot log already contains basic version information including build type and date, this information may also be found in the `opencore-version` NVRAM variable even when boot logging is disabled.

File logging will create a file named `opencore-YYYY-MM-DD-HHMMSS.txt` (in UTC) under the EFI volume root with log contents (the upper case letter sequence is replaced with date and time from the firmware). Please be warned that some file system drivers present in firmware are not reliable and may corrupt data when writing files through UEFI. Log writing is attempted in the safest manner and thus, is very slow. Ensure that `DisableWatchDog` is set to `true` when a slow drive is used. Try to avoid frequent use of this option when dealing with flash drives as large I/O amounts may speed up memory wear and render the flash drive unusable quicker.

When interpreting the log, note that the lines are prefixed with a tag describing the relevant location (module) of the log line allowing better attribution of the line to the functionality.

The list of currently used tags is as follows.

Drivers and tools:

- BMF — OpenCanopy, bitmap font
- BS — Bootstrap
- GSTT — GoptStop
- HDA — AudioDxe
- KKT — KeyTester
- MMDD — MmapDump
- OCPAVP — PavpProvision
- OCRST — ResetSystem
- OCUI — OpenCanopy
- OC — OpenCore main, also OcMainLib
- VMOPT — VerifyMemOpt

Libraries:

- AAPL — OcDebugLogLib, Apple EfiBoot logging
- OCABC — OcAfterBootCompatLib
- OCAE — OcAppleEventLib
- OCAK — OcAppleKernelLib
- OCAU — OcAudioLib
- OCA — OcAcpiLib
- OCBP — OcAppleBootPolicyLib
- OCB — OcBootManagementLib
- [OCBLT — OcBlitLib](#)
- OCCL — OcAppleChunkListLib
- OCCPU — OcCpuLib
- OCC — OcConsoleLib
- OCDC — OcDriverConnectionLib
- OCDH — OcDataHubLib
- OCDI — OcAppleDiskImageLib
- OCDM — OcDeviceMiscLib
- OCFS — OcFileLib

- OCFV — OcFirmwareVolumeLib
- OCHS — OcHashServicesLib
- OCIA — OcAppleImg4Lib
- OCIC — OcImageConversionLib
- OCII — OcInputLib
- OCJS — OcApfsLib
- OCKM — OcAppleKeyMapLib
- OCL — OcDebugLogLib
- OCM — OcMiscLib
- OCMCO — OcMachoLib
- OCME — OcHeciLib
- OCMM — OcMemoryLib
- OCPE — OcPeCoffLib, OcPeCoffExtLib
- OCPI — OcFileLib, partition info
- OCPNG — OcPngLib
- OCRAM — OcAppleRamDiskLib
- OCRTC — OcRtcLib
- OCSB — OcAppleSecureBootLib
- OCSMB — OcSmbiosLib
- OCSMC — OcSmcLib
- OCST — OcStorageLib
- OCS — OcSerializedLib
- OCTPL — OcTemplateLib
- OCUC — OcUnicodeCollationLib
- OCUT — OcAppleUserInterfaceThemeLib
- OCXML — OcXmlLib

8.5 Security Properties

1. AllowNvramReset

Type: plist boolean

Failsafe: false

Description: Allow CMD+OPT+P+R handling and enable showing NVRAM Reset entry in OpenCore picker.

Note 1: It is known that some Lenovo laptops have a firmware bug, which makes them unbootable after performing NVRAM reset. Refer to [acidanthera/bugtracker#995](#) for details.

Note 2: Resetting NVRAM will also erase any boot options not backed up using the bless command. For example, Linux installations to custom locations not specified in BlessOverride

2. AllowSetDefault

Type: plist boolean

Failsafe: false

Description: Allow CTRL+Enter and CTRL+Index handling to set the default boot option in the OpenCore picker.

Note 1: May be used in combination with Shift+Enter or Shift+Index when PollAppleHotKeys is enabled.

Note 2: In order to support systems with unresponsive modifiers during preboot (which includes V1 and V2 KeySupport mode on some firmware) OpenCore also allows holding the =/+ key in order to trigger 'set default' mode.

3. [AllowToggleSip](#)

Type: [plist boolean](#)

Failsafe: [false](#)

Description: [Enable entry for disabling and enabling System Integrity Protection in OpenCore picker.](#)

[This will toggle Apple NVRAM variable csr-active-config between 0 for SIP Enabled and a practical default value for SIP Disabled \(currently 0x26F\).](#)

[Note1:](#) [It is strongly recommended not to make a habit of running macOS with SIP disabled. Use of this boot option may make it easier to quickly disable SIP protection when genuinely needed - it should be re-enabled](#)

again afterwards.

Note2: OC uses 0x26F even though csrutil disable on Big Sur sets 0x7F. To explain the choice:

- csrutil disable --no-internal actually sets 0x6F, and this is preferable because CSR_ALLOW_APPLE_INTERNAL (0x10) prevents updates (unless you are running an internal build of macOS).
- CSR_ALLOW_UNAPPROVED_KEXTS (0x200) is generally useful, in the case where you do need to have SIP disabled, as it allows installing unsigned kexts without manual approval in System Preferences.
- CSR_ALLOW_UNAUTHENTICATED_ROOT (0x800) is not practical as it prevents incremental (non-full) OTA updates.

Note3: For any other value which you may need to use, it is possible to configure CsrUtil.efi as a TextMode Tools entry to configure a different value, e.g. use toggle 0x6F in Arguments to toggle the SIP disabled value set by default by csrutil disable --no-internal in Big Sur.

4. ApECID

Type: plist integer, 64 bit

Failsafe: 0

Description: Apple Enclave Identifier.

Setting this value to any non-zero 64-bit integer will allow using personalised Apple Secure Boot identifiers. To use this setting, generate a random 64-bit number with a cryptographically secure random number generator. As an alternative, the first 8 bytes of SystemUUID can be used for ApECID, this is found in macOS 11 for Macs without the T2 chip.

With this value set and SecureBootModel valid (and not Disabled), it is possible to achieve Full Security of Apple Secure Boot.

To start using personalised Apple Secure Boot, the operating system must be reinstalled or personalised. Unless the operating system is personalised, macOS DMG recovery cannot be loaded. In cases where DMG recovery is missing, it can be downloaded by using the macrecovery utility and saved in com.apple.recovery.boot as explained in the Tips and Tricks section. Note that DMG loading needs to be set to Signed to use any DMG with Apple Secure Boot.

To personalise an existing operating system, use the bless command after loading to macOS DMG recovery. Mount the system volume partition, unless it has already been mounted, and execute the following command:

```
bless bless --folder "/Volumes/Macintosh HD/System/Library/CoreServices" \  
--bootefi --personalize
```

On macOS versions before macOS 11, which introduced a dedicated x86legacy model for models without the T2 chip, personalised Apple Secure Boot may not work as expected. When reinstalling the operating system, the macOS Installer from macOS 10.15 and older will often run out of free memory on the /var/tmp partition when trying to install macOS with the personalised Apple Secure Boot. Soon after downloading the macOS installer image, an Unable to verify macOS error message will appear.

To workaround this issue, allocate a dedicated RAM disk of 2 MBs for macOS personalisation by entering the following commands in the macOS recovery terminal before starting the installation:

```
disk=$(hdiutil attach -nomount ram://4096)  
diskutil erasevolume HFS+ SecureBoot $disk  
diskutil unmount $disk  
mkdir /var/tmp/OSPersonalizationTemp  
diskutil mount -mountpoint /var/tmp/OSPersonalizationTemp $disk
```

5. AuthRestart

Type: plist boolean

Failsafe: false

Description: Enable VirtualSMC-compatible authenticated restart.

Authenticated restart is a way to reboot FileVault 2 enabled macOS without entering the password. A dedicated terminal command can be used to perform authenticated restarts: `sudo fdesetup authrestart`. It is also used when installing operating system updates.

- (h) Installing macOS with Apple Secure Boot enabled is not possible while using HFS+ target volumes. This may include HFS+ formatted drives when no spare APFS drive is available.

The installed operating system may have sometimes outdated Apple Secure Boot manifests on the `Preboot` partition, resulting in boot failures. This is likely to be the case when an “OCB: Apple Secure Boot prohibits this boot entry, enforcing!” message is logged.

When this happens, either reinstall the operating system or copy the manifests (files with `.im4m` extension, such as `boot.efi.j137.im4m`) from `/usr/standalone/i386` to `/Volumes/Preboot/<UUID>/System/Library/CoreServices`. Here, `<UUID>` is the system volume identifier. On HFS+ installations, the manifests should be copied to `/System/Library/CoreServices` on the system volume.

For more details on how to configure Apple Secure Boot with UEFI Secure Boot, refer to the UEFI Secure Boot section.

8.6 Entry Properties

1. Arguments
Type: plist string
Failsafe: Empty
Description: Arbitrary ASCII string used as boot arguments (load options) of the specified entry.
2. Auxiliary
Type: plist boolean
Failsafe: false
Description: Set to `true` to hide this entry when `HideAuxiliary` is also set to `true`. Press the `Spacebar` key to enter “Extended Mode” and display the entry when hidden.
3. Comment
Type: plist string
Failsafe: Empty
Description: Arbitrary ASCII string used to provide a human readable reference for the entry. Whether this value is used is implementation defined.
4. Enabled
Type: plist boolean
Failsafe: false
Description: Set to `true` activate this entry.
5. [Flavour](#)
Type: [plist string](#)
Failsafe: [Auto](#)
Description: [Specify the content flavour for this entry. See OC_ATTR_USE_FLAVOUR_ICON flag for documentation.](#)
6. Name
Type: plist string
Failsafe: Empty
Description: Human readable entry name displayed in the OpenCore picker.
7. Path
Type: plist string
Failsafe: Empty
Description: Entry location depending on entry type.
 - **Entries** specify external boot options, and therefore take device paths in the `Path` key. Care should be exercised as these values are not checked. Example: `PciRoot(0x0)/Pci(0x1,0x1)/.../EFI/COOL.EFI`
 - **Tools** specify internal boot options, which are part of the bootloader vault, and therefore take file paths relative to the `OC/Tools` directory. Example: `OpenShell.efi`.
8. RealPath
Type: plist boolean

- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeatures`
Combined `FirmwareFeatures` and `ExtendedFirmwareFeatures`. Present on newer Macs to avoid extra parsing of SMBIOS tables.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeaturesMask`
Combined `FirmwareFeaturesMask` and `ExtendedFirmwareFeaturesMask`. Present on newer Macs to avoid extra parsing of SMBIOS tables.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_BID`
Hardware `BoardProduct` (e.g. `Mac-35C1E88140C3E6CF`). Not present on real Macs, but used to avoid extra parsing of SMBIOS tables, especially in `boot.efi`.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_MLB`
Hardware `BoardSerialNumber`. Override for `MLB`. Present on newer Macs (2013+ at least).
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_ROM`
Hardware `ROM`. Override for `ROM`. Present on newer Macs (2013+ at least).
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:SSN`
Serial number. Present on newer Macs (2013+ at least).
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:prev-lang:kbd`
ASCII string defining default keyboard layout. Format is `lang-COUNTRY:keyboard`, e.g. `ru-RU:252` for Russian locale and ABC keyboard. Also accepts short forms: `ru:252` or `ru:0` (U.S. keyboard, compatible with 10.9). Full decoded keyboard list from `AppleKeyboardLayouts-L.dat` can be found [here](#). Using non-latin keyboard on 10.14 will not enable ABC keyboard, unlike previous and subsequent macOS versions, and is thus not recommended in case 10.14 is needed.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:security-mode`
ASCII string defining FireWire security mode. Legacy, can be found in `IOFireWireFamily` source code in `IOFireWireController.cpp`. It is recommended not to set this variable, which may speedup system startup. Setting to `full` is equivalent to not setting the variable and `none` disables FireWire security.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:UIScale`
One-byte data defining `boot.efi` user interface scaling. Should be `01` for normal screens and `02` for HiDPI screens.
- [`7C436110-AB2A-4BBB-A880-FE41995C9F82:ForceDisplayRotationInEFI` 32-bit integer defining display rotation. Can be 0 for no rotation or any of 90, 180, 270 for matching rotation in degrees.](#)
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:DefaultBackgroundColor`
Four-byte BGRA data defining `boot.efi` user interface background colour. Standard colours include `BF BF BF 00` (Light Gray) and `00 00 00 00` (Syrah Black). Other colours may be set at user's preference.

9.5 Other Variables

The following variables may be useful for certain configurations or troubleshooting:

- `7C436110-AB2A-4BBB-A880-FE41995C9F82:boot-args`
Kernel arguments, used to pass configuration to Apple kernel and drivers. There are many arguments, which may be found by looking for the use of `PE_parse_boot_argn` function in the kernel or driver code. Some of the known boot arguments include:
 - `acpi_layer=0xFFFFFFFF`
 - `acpi_level=0xFFFF5F` (implies `ACPI_ALL_COMPONENTS`)
 - `arch=i386` (force kernel architecture to `i386`, see `KernelArch`)
 - `batman=VALUE` (`AppleSmartBatteryManager` debug mask)
 - `batman-nosmc=1` (disable `AppleSmartBatteryManager` SMC interface)
 - `cpus=VALUE` (maximum number of CPUs used)
 - `debug=VALUE` (debug mask)
 - `io=VALUE` (`IOKit` debug mask)
 - `ioaccel_debug=VALUE` (`IOAccelerator` debug mask)
 - `keepsyms=1` (show panic log debug symbols)
 - `kextlog=VALUE` (kernel extension loading debug mask)
 - `nvram-log=1` (enables `AppleEFIVRAM` logs)
 - `nv_disable=1` (disables NVIDIA GPU acceleration)
 - `nvda_drv=1` (legacy way to enable NVIDIA web driver, removed in 10.12)
 - `npcci=0x2000` (legacy, disables `kIOPCIConfiguratorPFM64`)

and related. Set to 1MB (0x100000) by default, can be tuned for faster booting.

- **log-level=VALUE** — log level bitmask.
 - * 0x01 — enables trace logging (default).
- **serial=VALUE** — enables serial logging.
 - * 0 — disables serial logging (default).
 - * 1 — enables serial logging for EXITBS:END onwards.
 - * 2 — enables serial logging for EXITBS:START onwards.
 - * 3 — enables serial logging when debug protocol is missing.
 - * 4 — enables serial logging unconditionally.
- **timestamps=VALUE** — enables timestamp logging.
 - * 0 — disables timestamp logging.
 - * 1 — enables timestamp logging (default).
- **log=VALUE** — deprecated starting from 10.15.
 - * 1 — AppleLoggingConOutOrErrSet/AppleLoggingConOutOrErrPrint (classical ConOut/StdErr)
 - * 2 — AppleLoggingStdErrSet/AppleLoggingStdErrPrint (StdErr or serial?)
 - * 4 — AppleLoggingFileSet/AppleLoggingFilePrint (BOOTER.LOG/BOOTER.OLD file on EFI partition)
- **debug=VALUE** — deprecated starting from 10.15.
 - * 1 — enables print something to BOOTER.LOG (stripped code implies there may be a crash)
 - * 2 — enables perf logging to /efi/debug-log in the device three
 - * 4 — enables timestamp printing for styled printf calls
- **level=VALUE** — deprecated starting from 10.15. Verbosity level of DEBUG output. Everything but 0x80000000 is stripped from the binary, and this is the default value.

Note: Enable the **AppleDebug** option to display verbose output from **boot.efi** on modern macOS versions. This will save the log to the general OpenCore log file. For versions before 10.15.4, set **booterconfig** to **log=1**. This will print verbose output onscreen.

- **7C436110-AB2A-4BBB-A880-FE41995C9F82:booterconfig-once**
Booter arguments override removed after first launch. Otherwise equivalent to **booterconfig**.
- **7C436110-AB2A-4BBB-A880-FE41995C9F82:csr-data**
[Specify sources of kexts which will be approved regardless of SIP CSR_ALLOW_UNAPPROVED_KEXTS value.](#)
[Example contents:](#)

```
<dict><key>kext-allowed-teams</key><array><string>{DEVELOPER-TEAM-ID}</string></array></dict>%00
```
- **7C436110-AB2A-4BBB-A880-FE41995C9F82:efiboot-perf-record**
Enable performance log saving in **boot.efi**. Performance log is saved to physical memory and is pointed to by the **efiboot-perf-record-data** and **efiboot-perf-record-size** variables. Starting from 10.15.4, it can also be saved to the OpenCore log by setting the **AppleDebug** option.
- **7C436110-AB2A-4BBB-A880-FE41995C9F82:fmm-computer-name**
Current saved host name. ASCII string.
- **7C436110-AB2A-4BBB-A880-FE41995C9F82:nvda_drv**
NVIDIA Web Driver control variable. Takes ASCII digit 1 or 0 to enable or disable installed driver.
- **7C436110-AB2A-4BBB-A880-FE41995C9F82:run-efi-updater**
Override EFI firmware updating support in macOS (MultiUpdater, ThorUtil, and so on). Setting this to No or alternative boolean-castable value will prevent any firmware updates in macOS starting with 10.10 at least.
- **7C436110-AB2A-4BBB-A880-FE41995C9F82:StartupMute**
Mute startup chime sound in firmware audio support. 8-bit integer. The value of 0x00 means unmuted. Missing variable or any other value means muted.
- **7C436110-AB2A-4BBB-A880-FE41995C9F82:SystemAudioVolume**
System audio volume level for firmware audio support. 8-bit integer. The bit of 0x80 means muted. Lower bits are used to encode volume range specific to installed audio codec. The value is capped by **MaximumBootBeepVolume** AppleHDA layout value to avoid too loud audio playback in the firmware.
- **5EDDA193-A070-416A-85EB-2A1181F45B18:PEXConf**
PCI expansion slot configuration for MacPro7,1. 8-byte sequence describing default PCI slot configuration. Each byte refers to a configuration for a dedicated PCI slot.
 - Slot 1 resides at **IOService:/AppleACPIPlatformExpert/PC01@0/AppleACPIPCI/BR1A@0** and its path is hardcoded. This slot is not behind a muxer.
 - Slot 3 resides at **IOService:/AppleACPIPlatformExpert/PC03@0/AppleACPIPCI/BR3A@0** and its path is hardcoded. This slot is not behind a muxer.
 - Slots 2, 4-8 are dynamic and are matched based on **AAPL,slot-name** property with **Slot-N** value, where N

Description: Update Data Hub fields in non-Automatic mode.

Note: This section is ignored and may be removed when `Automatic` is `true`.

10. Memory

Type: plist dictionary

Description: Define custom memory configuration.

Note: This section is ignored and may be removed when `CustomMemory` is `false`.

11. PlatformNVRAM

Type: plist dictionary

Description: Update platform NVRAM fields in non-Automatic mode.

Note: This section is ignored and may be removed when `Automatic` is `true`.

12. SMBIOS

Type: plist dictionary

Description: Update SMBIOS fields in non-Automatic mode.

Note: This section is ignored and may be removed when `Automatic` is `true`.

10.2 Generic Properties

1. SpoofVendor

Type: plist boolean

Failsafe: false

Description: Sets SMBIOS vendor fields to `Acidanthera`.

It can be dangerous to use “Apple” in SMBIOS vendor fields for reasons outlined in the `SystemManufacturer` description. However, certain firmware may not provide valid values otherwise, which could obstruct the operation of some software.

2. ~~AdviseWindows~~AdviseFeatures

Type: plist boolean

Failsafe: false

Description: ~~Forces Windows support in~~ Updates `FirmwareFeatures` with supported bits.

Added bits to `FirmwareFeatures`:

- `FW_FEATURE_SUPPORTS_CSM_LEGACY_MODE` (0x1) - Without this bit, it is not possible to reboot to Windows installed on a drive with an EFI partition that is not the first partition on the disk.
- `FW_FEATURE_SUPPORTS_UEFI_WINDOWS_BOOT` (0x20000000) - Without this bit, it is not possible to reboot to Windows installed on a drive with an EFI partition that is the first partition on the disk.
- `FW_FEATURE_SUPPORTS_APFS` (0x00080000) - Without this bit, it is not possible to install macOS on an APFS disk.

Note: On most newer firmwares these bits are already set, the option may be necessary when "upgrading" the firmware with new features.

3. MaxBIOSVersion

Type: plist boolean

Failsafe: false

Description: Sets `BIOSVersion` to 9999.999.999.999.999, recommended for legacy Macs when using `Automatic PlatformInfo`, to avoid BIOS updates in unofficially supported macOS versions.

4. SystemMemoryStatus

Type: plist string

Failsafe: Auto

Description: Indicates whether system memory is upgradable in `PlatformFeature`. This controls the visibility of the Memory tab in “About This Mac”.

Valid values:

- Auto — use the original `PlatformFeature` value.

11.3 Tools and Applications

Standalone tools may help to debug firmware and hardware. Some of the known tools are listed below. While some tools can be launched from within OpenCore (Refer to the Tools subsection for more details), most should be run separately either directly or from `Shell`.

To boot into OpenShell or any other tool directly save `OpenShell.efi` under the name of `EFI\BOOT\BOOTX64.EFI` on a FAT32 partition. It is typically unimportant whether the partition scheme is GPT or MBR.

While the previous approach works both on Macs and other computers, an alternative Mac-only approach to bless the tool on an HFS+ or APFS volume:

```
sudo bless --verbose --file /Volumes/VOLNAME/DIR/OpenShell.efi \
--folder /Volumes/VOLNAME/DIR/ --setBoot
```

Listing 3: Blessing tool

Note 1: `/System/Library/CoreServices/BridgeVersion.bin` should be copied to `/Volumes/VOLNAME/DIR`.

Note 2: To be able to use the `bless` command, disabling System Integrity Protection is necessary.

Note 3: To be able to boot Secure Boot might be disabled if present.

Some of the known tools are listed below (builtin tools are marked with *):

<code>BootKicker*</code>	Enter Apple BootPicker menu (exclusive for Macs with compatible GPUs).
<code>ChipTune*</code>	Test BeepGen protocol and generate audio signals of different style and length.
<code>CleanNvram*</code>	Reset NVRAM alternative bundled as a standalone tool.
<code>CsrUtil*</code>	Simple implementation of SIP-related features of Apple csrutil.
<code>GopStop*</code>	Test GraphicsOutput protocol with a simple scenario.
<code>KeyTester*</code>	Test keyboard input in SimpleText mode.
<code>MemTest86</code>	Memory testing utility.
<code>OpenControl*</code>	Unlock and lock back NVRAM protection for other tools to be able to get full NVRAM access when launching from OpenCore.
<code>OpenShell*</code>	OpenCore-configured UEFI Shell for compatibility with a broad range of firmware.
<code>PavpProvision</code>	Perform EPID provisioning (requires certificate data configuration).
<code>ResetSystem*</code>	Utility to perform system reset. Takes reset type as an argument: <code>coldreset</code> , <code>firmware</code> , <code>shutdown</code> , <code>warmreset</code> . Defaults to <code>coldreset</code> .
<code>RtcRw*</code>	Utility to read and write RTC (CMOS) memory.
<code>ControlMsrE2*</code>	Check CFG Lock (MSR 0xE2 write protection) consistency across all cores and change such hidden options on selected platforms.

11.4 OpenCanopy

OpenCanopy is a graphical OpenCore user interface that runs in `External PickerMode` and relies on `OpenCorePkg OcBootManagementLib` similar to the builtin text interface.

OpenCanopy requires graphical resources located in `Resources` directory to run. Sample resources (fonts and images) can be found in `OcBinaryData` repository. Customised icons can be found over the internet (e.g. [here](#) or [there](#)).

OpenCanopy provides full support for `PickerAttributes` and offers a configurable builtin icon set. The ~~default~~ chosen icon set ~~depends~~ [may depend](#) on the `DefaultBackgroundColor` variable value. ~~For Light Gray Refer to Oldicon-set will be used, for other colours -- the one without a prefix~~ [PickerVariant for more details](#).

Predefined icons are saved in the [PickerVariant-derived subdirectory of the](#) `\EFI\OC\Resources\Image` directory. A full list of supported icons (in `.icns` format) is provided below. When optional icons are missing, the closest available icon will be used. External entries will use `Ext`-prefixed icon if available (e.g. `OldExtHardDrive.icns`).

Note: In the following all dimensions are normative for the 1x scaling level and shall be scaled accordingly for other levels.

- `Cursor` — Mouse cursor (mandatory, up to 144x144).
- `Selected` — Selected item (mandatory, 144x144).
- `Selector` — Selecting item (mandatory, up to 144x40).
- `Left` — Scrolling left (mandatory, 40x40).

- **Right** — Scrolling right (mandatory, 40x40).
- **HardDrive** — Generic OS (mandatory, 128x128).
- **Background** — Centred background image.
- **Apple** — Apple OS (128x128).
- **AppleRecv** — Apple Recovery OS (128x128).
- **AppleTM** — Apple Time Machine (128x128).
- **Windows** — Windows (128x128).
- **Other** — Custom entry (see **Entries**, 128x128).
- **ResetNVRAM** — Reset NVRAM system action or tool (128x128).
- **Shell** — Entry with UEFI Shell name for e.g. **OpenShell** (128x128).
- **Tool** — Any other tool (128x128).

Predefined labels are saved in the `\EFI\OC\Resources\Label` directory. Each label has `.1b1` or `.12x` suffix to represent the scaling level. Full list of labels is provided below. All labels are mandatory.

- **EFIBoot** — Generic OS.
- **Apple** — Apple OS.
- **AppleRecv** — Apple Recovery OS.
- **AppleTM** — Apple Time Machine.
- **Windows** — Windows.
- **Other** — Custom entry (see **Entries**).
- **ResetNVRAM** — Reset NVRAM system action or tool.
- [SIPDisabled — Toogle SIP tool with SIP disabled.](#)
- [SIPEnabled — Toogle SIP tool with SIP enabled.](#)
- **Shell** — Entry with UEFI Shell name (e.g. **OpenShell**).
- **Tool** — Any other tool.

Note: All labels must have a height of exactly 12 px. There is no limit for their width.

Label and icon generation can be performed with bundled utilities: **disklabel** and **icnspack**. Font is Helvetica 12 pt times scale factor.

Font format corresponds to AngelCode binary BMF. While there are many utilities to generate font files, currently it is recommended to use **dpFontBaker** to generate bitmap font (using **CoreText** produces best results) and **fonverter** to export it to binary format.

11.5 OpenRuntime

OpenRuntime is an OpenCore plugin implementing `OC_FIRMWARE_RUNTIME` protocol. This protocol implements multiple features required for OpenCore that are otherwise not possible to implement in OpenCore itself as they are needed to work in runtime, i.e. during operating system functioning. Feature highlights:

- NVRAM namespaces, allowing to isolate operating systems from accessing select variables (e.g. **RequestBootVarRouting** or **ProtectSecureBoot**).
- Read-only and write-only NVRAM variables, enhancing the security of OpenCore, Lilu, and Lilu plugins, such as **VirtualSMC**, which implements **AuthRestart** support.
- NVRAM isolation, allowing to protect all variables from being written from an untrusted operating system (e.g. **DisableVariableWrite**).
- UEFI Runtime Services memory protection management to workaround read-only mapping (e.g. **EnableWriteUnprotector**).

11.6 Properties

1. APFS

Type: plist dict

Failsafe: None

Description: Provide APFS support as configured in the APFS Properties section below.

2. Audio

Type: plist dict

8. Quirks

Type: plist dict

Failsafe: None

Description: Apply individual firmware quirks described in the Quirks Properties section below.

9. ReservedMemory

Type: plist array

Description: To be filled with `plist dict` values, describing memory areas exclusive to specific firmware and hardware functioning, which should not be used by the operating system. Examples of such memory regions could be the second 256 MB corrupted by the Intel HD 3000 or an area with faulty RAM. Refer to the ReservedMemory Properties section below for details.

11.7 APFS Properties

1. EnableJumpstart

Type: plist boolean

Failsafe: false

Description: Load embedded APFS drivers from APFS containers.

An APFS EFI driver is bundled in all bootable APFS containers. This option performs the loading of signed APFS drivers (consistent with the `ScanPolicy`). Refer to the “EFI Jumpstart” section of the Apple File System Reference for details.

2. GlobalConnect

Type: plist boolean

Failsafe: false

Description: Perform full device connection during APFS loading.

Every handle is connected recursively instead of the partition handle connection typically used for APFS driver loading. This may result in additional time being taken but can sometimes be the only way to access APFS partitions on certain firmware, such as those on older HP laptops.

3. HideVerbose

Type: plist boolean

Failsafe: false

Description: Hide verbose output from APFS driver.

APFS verbose output can be useful for debugging.

4. JumpstartHotPlug

Type: plist boolean

Failsafe: false

Description: Load APFS drivers for newly connected devices.

Permits APFS USB hot plug which enables loading APFS drivers, both at OpenCore startup and during OpenCore picker ~~display~~display. Disable if not required.

5. MinDate

Type: plist integer

Failsafe: 0

Description: Minimal allowed APFS driver date.

The APFS driver date connects the APFS driver with the calendar release date. Apple ultimately drops support for older macOS releases and APFS drivers from such releases may contain vulnerabilities that can be used to compromise a computer if such drivers are used after support ends. This option permits restricting APFS drivers to current macOS versions.

- 0 — require the default supported release date of APFS in OpenCore. The default release date will increase with time and thus this setting is recommended. Currently set to 2018/06/21.
- -1 — permit any release date to load (strongly discouraged).
- Other — use custom minimal APFS release date, e.g. 20200401 for 2020/04/01. APFS release dates can be found in OpenCore boot log and `OcAptsLib`.

On certain firmware, such as on the MacPro5,1, this may provide better performance or fix rendering issues. However, this option is not recommended unless there is an obvious benefit as it may result in issues such as slower scrolling.

This renderer fully supports AppleEg2Info protocol and will provide screen rotation for all EFI applications. In order to provide seamless rotation compatibility with EfiBoot, builtin AppleFramebufferInfo should also be used, i.e. it may need to be overridden on Mac EFI.

7. GopPassThrough

Type: plist `booleanstring`

Failsafe: `falseDisabled`

Description: Provide GOP protocol instances on top of UGA protocol instances.

This option provides the GOP protocol via a UGA-based proxy for firmware that do not implement the protocol.

The supported values for the option are as follows:

- Enabled — provide GOP for all UGA protocols.
- Apple — provide GOP for AppleFramebufferInfo-enabled protocols.
- Disabled — do not provide GOP.

Note: This option requires ProvideConsoleGop to be enabled.

8. IgnoreTextInGraphics

Type: plist `boolean`

Failsafe: `false`

Description: Some types of firmware output text onscreen in both graphics and text mode. This is typically unexpected as random text may appear over graphical images and cause UI corruption. Setting this option to `true` will discard all text output when console control is in a different mode from `Text`.

Note: This option only applies to the `System` renderer.

9. ReplaceTabWithSpace

Type: plist `boolean`

Failsafe: `false`

Description: Some types of firmware do not print tab characters or everything that follows them, causing difficulties in using the UEFI Shell's builtin text editor to edit property lists and other documents. This option makes the console output spaces instead of tabs.

Note: This option only applies to `System` renderer.

10. ProvideConsoleGop

Type: plist `boolean`

Failsafe: `false`

Description: Ensure GOP (Graphics Output Protocol) on console handle.

macOS bootloader requires GOP or UGA (for 10.4 EfiBoot) to be present on console handle, yet the exact location of the graphics protocol is not covered by the UEFI specification. This option will ensure GOP and UGA, if present, are available on the console handle.

Note: This option will also replace incompatible implementations of GOP on the console handle, as may be the case on the MacPro5,1 when using modern GPUs.

11. ReconnectOnResChange

Type: plist `boolean`

Failsafe: `false`

Description: Reconnect console controllers after changing screen resolution.

On certain firmware, the controllers that produce the console protocols (simple text out) must be reconnected when the screen resolution is changed via GOP. Otherwise, they will not produce text based on the new resolution.

Note: On several boards this logic may result in black screen when launching OpenCore from Shell and thus it is optional. In versions prior to 0.5.2 this option was mandatory and not configurable. Please do not use this unless required.

12. SanitiseClearScreen

Type: plist boolean

Failsafe: false

Description: Some types of firmware reset screen resolutions to a failsafe value (such as 1024x768) on the attempts to clear screen contents when large display (e.g. 2K or 4K) is used. This option attempts to apply a workaround.

Note: This option only applies to the **System** renderer. On all known affected systems, **ConsoleMode** must be set to an empty string for this option to work.

13. UgaPassThrough

Type: plist boolean

Failsafe: false

Description: Provide UGA protocol instances on top of GOP protocol instances.

Some types of firmware do not implement the legacy UGA protocol but this may be required for screen output by older EFI applications such as EfiBoot from 10.4.

11.12 ProtocolOverrides Properties

1. AppleAudio

Type: plist boolean

Failsafe: false

Description: Replaces Apple audio protocols with builtin versions.

Apple audio protocols allow OpenCore and the macOS bootloader to play sounds and signals for screen reading or audible error reporting. Supported protocols are beep generation and VoiceOver. The VoiceOver protocol is specific to Gibraltar machines (T2) and is not supported before macOS High Sierra (10.13). Older macOS versions use the AppleHDA protocol (which is not currently implemented) instead.

Only one set of audio protocols can be available at a time, so this setting should be enabled in order to enable audio playback in the OpenCore user interface on Mac systems implementing some of these protocols.

Note: The backend audio driver needs to be configured in **UEFI Audio** section for these protocols to be able to stream audio.

2. AppleBootPolicy

Type: plist boolean

Failsafe: false

Description: Replaces the Apple Boot Policy protocol with a builtin version. This may be used to ensure APFS compatibility on VMs and legacy Macs.

Note: This option is advisable on certain Macs, such as the **MacPro5,1**, that are APFS compatible but on which the Apple Boot Policy protocol has recovery detection issues.

3. AppleDebugLog

Type: plist boolean

Failsafe: false

Description: Replaces the Apple Debug Log protocol with a builtin version.

4. [AppleEg2Info](#)

[**Type:** plist boolean](#)

[**Failsafe:** false](#)

[**Description:** Replaces the Apple EFI Graphics 2 protocol with a builtin version.](#)

[*Note:* This protocol allows newer EfiBoot versions \(at least 10.15\) to expose screen rotation to macOS. Refer to ForceDisplayRotationInEFI variable description on how to set screen rotation angle.](#)

5. AppleFramebufferInfo

Type: plist boolean

Failsafe: false

Description: Replaces the Apple Framebuffer Info protocol with a builtin version. This may be used to override framebuffer information on VMs and legacy Macs to improve compatibility with legacy EfiBoot such as the one in macOS 10.4.