



OpenCore

Reference Manual (~~0.5.9~~0.6.0)

[2020.06.16]

1 Introduction

This document provides information on OpenCore user configuration file format used to setup the correct functioning of macOS operating system. It is to be read as the official clarification of expected OpenCore behaviour. All deviations, if found in published OpenCore releases, shall be considered documentation or implementation bugs, and are requested to be reported through Acidanthera Bugtracker. [Errata sheet is available in](#) OpenCorePkg repository.

This document is structured as a specification, and is not meant to provide a step by step algorithm for configuring end-user board support package (BSP). The intended audience of the document are programmers and engineers with basic understanding of macOS internals and UEFI functioning. For these reasons this document is available exclusively in English, and all other sources or translations of this document are unofficial and may contain errors.

Third-party articles, utilities, books, and alike may be more useful for a wider audience as they could provide guide-like material. However, they are prone to their authors' preferences, tastes, this document misinterpretation, and essential obsolescence. In case you use these sources, for example, Dortania's OpenCore Desktop Guide and related material, please ensure to follow this document for every made decision and judge its consequences.

Be warned that regardless of the sources used you are required to fully understand every dedicated OpenCore configuration option and concept prior to reporting any issues in Acidanthera Bugtracker.

1.1 Generic Terms

- **plist** — Subset of ASCII Property List format written in XML, also known as XML plist format version 1. Uniform Type Identifier (UTI): `com.apple.property-list`. Plists consist of **plist** objects, which are combined to form a hierarchical structure. Due to plist format not being well-defined, all the definitions of this document may only be applied after plist is considered valid by running `plutil -lint`. External references: <https://www.apple.com/DTDs/PropertyList-1.0.dtd>, `man plutil`.
- **plist type** — plist collections (**plist array**, **plist dictionary**, **plist key**) and primitives (**plist string**, **plist data**, **plist date**, **plist boolean**, **plist integer**, **plist real**).
- **plist object** — definite realisation of **plist type**, which may be interpreted as value.
- **plist array** — array-like collection, conforms to `array`. Consists of zero or more **plist** objects.
- **plist dictionary** — map-like (associative array) collection, conforms to `dict`. Consists of zero or more **plist** keys.
- **plist key** — contains one **plist** object going by the name of **plist key**, conforms to `key`. Consists of printable 7-bit ASCII characters.
- **plist string** — printable 7-bit ASCII string, conforms to `string`.
- **plist data** — base64-encoded blob, conforms to `data`.
- **plist date** — ISO-8601 date, conforms to `date`, unsupported.
- **plist boolean** — logical state object, which is either true (1) or false (0), conforms to `true` and `false`.
- **plist integer** — possibly signed integer number in base 10, conforms to `integer`. Fits in 64-bit unsigned integer in two's complement representation, unless a smaller signed or unsigned integral type is explicitly mentioned in specific **plist** object description.
- **plist real** — floating point number, conforms to `real`, unsupported.
- **plist metadata** — value cast to data by the implementation. Permits passing **plist string**, in which case the result is represented by a null-terminated sequence of bytes (aka C string), **plist integer**, in which case the result is represented by 32-bit little endian sequence of bytes in two's complement representation, **plist boolean**, in which case the value is one byte: 01 for `true` and 00 for `false`, and **plist data** itself. All other types or larger integers invoke undefined behaviour.

Type	Value
<code>plist integer</code>	0 (<integer>0</integer>)
<code>plist boolean</code>	False (<false/>)
<code>plist tristate</code>	False (<false/>)

2.3 Configuration Structure

OC `config` is separated into following sections, which are described in separate sections of this document. By default it is tried to not enable anything and optionally provide kill switches with `Enable` property for `plist dict` entries. In general the configuration is written idiomatically to group similar actions in subsections:

- `Add` provides support for data addition. Existing data will not be overridden, and needs to be handled separately with `Delete` if necessary.
- `Delete` provides support for data removal.
- `Patch` provides support for data modification.
- `Quirks` provides support for specific hacks.

Root configuration entries consist of the following:

- `ACPI`
- `Booter`
- `DeviceProperties`
- `Kernel`
- `Misc`
- `NVRAM`
- `PlatformInfo`
- `UEFI`

It is possible to perform basic validation of the configuration by using `ConfigValidityocvalidate` utility. Please note, that `ConfigValidityocvalidate` must match the used OpenCore release and may not be able to detect all configuration flaws present in the file.

Note: Currently most properties try to have defined values even if not specified in the configuration for safety reasons. This behaviour should not be relied upon, and all fields must be properly specified in the configuration.

Text renderer supports colour arguments as a sum of foreground and background colors according to UEFI specification. The value of black background and black foreground (0) is reserved. List of colour names:

- 0x00 — EFI_BLACK
- 0x01 — EFI_BLUE
- 0x02 — EFI_GREEN
- 0x03 — EFI_CYAN
- 0x04 — EFI_RED
- 0x05 — EFI_MAGENTA
- 0x06 — EFI_BROWN
- 0x07 — EFI_LIGHTGRAY
- 0x08 — EFI_DARKGRAY
- 0x09 — EFI_LIGHTBLUE
- 0x0A — EFI_LIGHTGREEN
- 0x0B — EFI_LIGHTCYAN
- 0x0C — EFI_LIGHTRED
- 0x0D — EFI_LIGHTMAGENTA
- 0x0E — EFI_YELLOW
- 0x0F — EFI_WHITE
- 0x10 — EFI_BACKGROUND_BLACK
- 0x11 — EFI_BACKGROUND_BLUE
- 0x12 — EFI_BACKGROUND_GREEN
- 0x13 — EFI_BACKGROUND_CYAN
- 0x14 — EFI_BACKGROUND_RED
- 0x15 — EFI_BACKGROUND_MAGENTA
- 0x16 — EFI_BACKGROUND_BROWN
- 0x17 — EFI_BACKGROUND_LIGHTGRAY

Note: This option may not work well with **System** text renderer. Setting a background different from black could help testing proper GOP functioning.

2. HibernateMode

Type: plist string

Failsafe: None

Description: Hibernation detection mode. The following modes are supported:

- None — Avoid hibernation for your own good.
- Auto — Use RTC and NVRAM detection.
- RTC — Use RTC detection.
- NVRAM — Use NVRAM detection.

3. HideAuxiliary

Type: plist boolean

Failsafe: false

Description: Hides auxiliary entries from picker menu by default.

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. ~~Clean~~Reset NVRAM).

To see all entries picker menu needs to be reloaded in extended mode by pressing **Spacebar** key. Hiding auxiliary entries may increase boot performance for multidisk systems.

4. PickerAttributes

Type: plist integer

Failsafe: 0

Description: Sets specific attributes for picker.

Different pickers may be configured through the attribute mask containing OpenCore-reserved (BIT0~BIT15) and

- OCCL — OcAppleChunkListLib
- OCCPU — OcCpuLib
- OCC — OcConsoleLib
- OCDH — OcDataHubLib
- OCDI — OcAppleDiskImageLib
- OCFSQ — OcFileLib, UnblockFs quirk
- OCFS — OcFileLib
- OCFV — OcFirmwareVolumeLib
- OCHS — OcHashServicesLib
- OCIC — OcImageConversionLib
- OCII — OcInputLib
- OCJS — OcApfsLib
- OCKM — OcAppleKeyMapLib
- OCL — OcDebugLogLib
- OCMCO — OcMachoLib
- OCME — OcHeciLib
- OCMM — OcMemoryLib
- 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 boot picker.

[Note: Resetting NVRAM will also erase all the boot options otherwise not backed up with bless \(e.g. Linux\).](#)

2. AllowSetDefault

Type: plist boolean

Failsafe: false

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

3. 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. To perform authenticated restart one can use a dedicated terminal command: `sudo fdsetup authrestart`. It is also used when installing operating system updates.

VirtualSMC performs authenticated restart by saving disk encryption key split in NVRAM and RTC, which despite being removed as soon as OpenCore starts, may be considered a security risk and thus is optional.

4. BlacklistAppleUpdate

Type: plist boolean

Failsafe: false

Description: Ignore boot options trying to update Apple peripheral firmware (e.g. `MultiUpdater.efi`).

11.3 Tools

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 many 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. In general it is 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: You may have to copy `/System/Library/CoreServices/BridgeVersion.bin` to `/Volumes/VOLNAME/DIR`.

Note 2: To be able to use `bless` you may have to disable System Integrity Protection.

Note 3: To be able to boot you may have to disable Secure Boot 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>GopStop*</code>	Test GraphicsOutput protocol with a simple scenario.
<code>HdaCodecDump*</code>	Parse and dump High Definition Audio codec information (requires <code>AudioDxe</code>).
<code>KeyTester*</code>	Test keyboard input in <code>SimpleText</code> 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 <code>Shell</code> for compatibility with a broad range of firmwares.
<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>WarmResetFirmware</code> , <code>Shutdown</code> , <code>WarmReset</code> . Defaults to <code>ColdReset</code> .
<code>RtcRw*</code>	Utility to read and write RTC (CMOS) memory.
<code>VerifyMsrE2*</code>	Check CFG Lock (MSR 0xE2 write protection) consistency across all cores.

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.

OpenCanopy provides full support for `PickerAttributes` and offers a configurable builtin icon set. The default chosen icon set depends on the `DefaultBackgroundColor` variable value. For `Light Gray Old` icon set will be used, for other colours — the one without a prefix.

Predefined icons are put to `\EFI\OC\Resources\Image` directory. Full list of supported icons (in `.icns` format) is provided below. Missing optional icons will use the closest available icon. External entries will use `Ext`-prefixed icon if available (e.g. `OldExtHardDrive.icns`).

- `Cursor` — Mouse cursor (mandatory).
- `Selected` — Selected item (mandatory).
- `Selector` — Selecting item (mandatory).
- `HardDrive` — Generic OS (mandatory).
- `Apple` — Apple OS.
- `AppleRecv` — Apple Recovery OS.
- `AppleTM` — Apple Time Machine.
- `Windows` — Windows.
- `Other` — Custom entry (see `Entries`).

Description: Forcibly wraps Firmware Volume protocols or installs new to support custom cursor images for File Vault 2. Should be set to **true** to ensure File Vault 2 compatibility on everything but VMs and legacy Macs.

Note: Several virtual machines including VMware may have corrupted cursor image in HiDPI mode and thus may also require this setting to be enabled.

13. HashServices

Type: plist boolean

Failsafe: false

Description: Forcibly reinstalls Hash Services protocols with builtin versions. Should be set to **true** to ensure File Vault 2 compatibility on platforms providing broken SHA-1 hashing. Can be diagnosed by invalid cursor size with **UIScale** set to 02, in general platforms prior to APTIO V (Haswell and older) are affected.

14. OSInfo

Type: plist boolean

Failsafe: false

Description: Forcibly reinstalls OS Info protocol with builtin versions. This protocol is generally used to receive notifications from macOS bootloader, by the firmware or by other applications.

15. UnicodeCollation

Type: plist boolean

Failsafe: false

Description: Forcibly reinstalls unicode collation services with builtin version. Should be set to **true** to ensure UEFI Shell compatibility on platforms providing broken unicode collation. In general legacy Insyde and APTIO platforms on Ivy Bridge and earlier are affected.

11.12 Quirks Properties

1. DeduplicateBootOrder

Type: plist boolean

Failsafe: false

Description: Remove duplicate entries in **BootOrder** variable in **EFI_GLOBAL_VARIABLE_GUID**.

This quirk requires **RequestBootVarRouting** to be enabled and therefore **OC_FIRMWARE_RUNTIME** protocol implemented in **OpenRuntime.efi**.

By redirecting **Boot** prefixed variables to a separate GUID namespace with the help of **RequestBootVarRouting** quirk we achieve multiple goals:

- Operating systems are jailed and only controlled by OpenCore boot environment to enhance security.
- Operating systems do not mess with OpenCore boot priority, and guarantee fluent updates and hibernation wakes for cases that require reboots with OpenCore in the middle.
- Potentially incompatible boot entries, such as macOS entries, are not deleted or anyhow corrupted.

However, some firmwares do their own boot option scanning upon startup by checking file presence on the available disks. Quite often this scanning includes non-standard locations, such as Windows Bootloader paths. Normally it is not an issue, but some firmwares, ASUS firmwares on APTIO V in particular, have bugs. For them scanning is implemented improperly, and firmware preferences may get accidentally corrupted due to **BootOrder** entry duplication (each option will be added twice) making it impossible to boot without [cleaning-resetting](#) NVRAM.

To trigger the bug one should have some valid boot options (e.g. OpenCore) and then install Windows with **RequestBootVarRouting** enabled. As Windows bootloader option will not be created by Windows installer, the firmware will attempt to create it itself, and then corrupt its boot option list.

This quirk removes all duplicates in **BootOrder** variable attempting to resolve the consequences of the bugs upon OpenCore loading. It is recommended to use this key along with **BootProtect** option.

2. ExitBootServicesDelay

Type: plist integer

Failsafe: 0

Description: Adds delay in microseconds after **EXIT_BOOT_SERVICES** event.

This is a very ugly quirk to circumvent "Still waiting for root device" message on select APTIO IV firmwares, namely ASUS Z87-Pro, when using FileVault 2 in particular. It seems that for some reason they execute code