OpenCore

Reference Manual (0.6.8.9)

[2021.04.27]

Copyright ©2018-2021 vit9696

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.
- 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 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.

The OC config file, as with any property list file, can be edited with any text editor, such as nano or vim. However, specialised software may provide a better experience. On macOS, the preferred GUI application is Xcode. For a lightweight cross-platform and open-source alternative, the ProperTree editor can be utilised.

It is strongly advised not to use any software that is aware of the internal configration structure as it constantly gets out of date and will cause incorrect configuration to be generated. If it is a must desprite the warning one should make sure to only use stable versions of OpenCore with explicit support for the particular version in the app. The choice of open-source implementations with transparent binary generation is encouraged (e.g. OCAT), since other tools may contain malware. Remember that a configuration made for a different hardware setup shall never be used on another hardware setup.

For BIOS booting, a third-party UEFI environment provider is required and OpenDuetPkg is one such UEFI environment provider for legacy systems. To run OpenCore on such a legacy system, OpenDuetPkg can be installed with a dedicated

tool — BootInstall (bundled with OpenCore). Third-party utilities can be used to perform this on systems other than macOS.

For upgrade purposes, refer to the `Differences.pdf` document which provides information about changes to the configuration (as compared to the previous release) as well as to the `Changelog.md` document (which contains a list of modifications across all published updates).

3.3 Contribution

OpenCore can be compiled as a standard EDK II package and requires the EDK II Stable package. The currently supported EDK II release is hosted in acidanthera/audk. Required patches for this package can be found in the `Patches` directory.

The only officially supported toolchain is `XCODE5`. Other toolchains might work but are neither supported nor recommended. Contributions of clean patches are welcome. Please do follow EDK II C Codestyle.

To compile with `XCODE5`, besides Xcode, users should also install NASM and MTOC. The latest Xcode version is recommended for use despite the toolchain name. An example command sequence is as follows:

```
git clone --depth=1 https://github.com/acidanthera/audk UDK
cd UDK
git submodule update --init --recommend-shallow
git clone --depth=1 https://github.com/acidanthera/OpenCorePkg
source edksetup.sh
make -C BaseTools
build -a X64 -b RELEASE -t XCODE5 -p OpenCorePkg/OpenCorePkg.dsc
```

Listing 1: Compilation Commands

For IDE usage Xcode projects are available in the root of the repositories. Another approach could be Sublime Text with EasyClangComplete plugin. Add `.clang_complete` file with similar content to the UDK root:

```
-I/UefiPackages/MdePkg
-I/UefiPackages/MdePkg/Include
-I/UefiPackages/MdePkg/Include/X64
-I/UefiPackages/MdeModulePkg
-I/UefiPackages/MdeModulePkg/Include
-I/UefiPackages/MdeModulePkg/Include/X64
-I/UefiPackages/OpenCorePkg/Include/AMI
-I/UefiPackages/OpenCorePkg/Include/Acidanthera
-I/UefiPackages/OpenCorePkg/Include/Apple
-I/UefiPackages/OpenCorePkg/Include/Apple/X64
-I/UefiPackages/OpenCorePkg/Include/Duet
-I/UefiPackages/OpenCorePkg/Include/Generic
-I/UefiPackages/OpenCorePkg/Include/Intel
-I/UefiPackages/OpenCorePkg/Include/Microsoft
-I/UefiPackages/OpenCorePkg/Include/Nvidia
-I/UefiPackages/OpenCorePkg/Include/VMware
-I/UefiPackages/OvmfPkg/Include
-I/UefiPackages/ShellPkg/Include
-I/UefiPackages/UefiCpuPkg/Include
-IInclude
-include
/UefiPackages/MdePkg/Include/Uefi.h
-fshort-wchar
-Wall
-Wextra
-Wno-unused-parameter
-Wno-missing-braces
-Wno-missing-field-initializers
```

```
-Wno-tautological-compare
-Wno-sign-compare
-Wno-varargs
-Wno-unused-const-variable
-DOC_TARGET_NOOPT=1
-DNO_MSABI_VA_FUNCS=1
```

Listing 2: ECC Configuration

Warning: Tool developers modifying `config.plist` or any other OpenCore files must ensure that their tools check the `opencore-version` NVRAM variable (see the Debug Properties section below) and warn users if the version listed is unsupported or prerelease. The OpenCore configuration may change across releases and such tools shall ensure that they carefully follow this document. Failure to do so may result in such tools being considered to be malware and blocked by any means.

3.4 Coding conventions

As with any other project, we have conventions that we follow during development. All third-party contributors are advised to adhere to the conventions listed below before submitting patches. To minimise abortive work and the potential rejection of submissions, third-party contributors should initially raise issues to the Acidanthera Bugtracker for feedback before submitting patches.

Organisation. The codebase is contained in the `OpenCorePkg` repository, which is the primary EDK II package.

- Whenever changes are required in multiple repositories, separate pull requests should be sent to each.
- Committing the changes should happen firstly to dependent repositories, secondly to primary repositories to avoid automatic build errors.
- Each unique commit should compile with `XCODE5` and preferably with other toolchains. In the majority of the cases it can be checked by accessing the CI interface. Ensuring that static analysis finds no warnings is preferred.
- External pull requests and tagged commits must be validated. That said, commits in master may build but may not necessarily work.
- Internal branches should be named as follows: `author-name-date`, e.g. `vit9696-ballooning-20191026`.
- Commit messages should be prefixed with the primary module (e.g. library or code module) the changes were made in. For example, `OcGuardLib: Add OC_ALIGNED macro`. For non-library changes `Docs` or `Build` prefixes are used.

Design. The codebase is written in a subset of freestanding C11 (C17) supported by most modern toolchains used by EDK II. Applying common software development practices or requesting clarification is recommended if any particular case is not discussed below.

- Never rely on undefined behaviour and try to avoid implementation defined behaviour unless explicitly covered below (feel free to create an issue when a relevant case is not present).
- Use `OcGuardLib` to ensure safe integral arithmetics avoiding overflows. Unsigned wraparound should be relied on with care and reduced to the necessary amount.
- Check pointers for correct alignment with `OcGuardLib` and do not rely on the architecture being able to dereference unaligned pointers.
- Use flexible array members instead of zero-length or one-length arrays where necessary.
- Use static assertions (`STATIC_ASSERT`) for type and value assumptions, and runtime assertions (`ASSERT`) for precondition and invariant sanity checking. Do not use runtime assertions to check for errors as they should never alter control flow and potentially be excluded.
- Assume `UINT32`/`INT32` to be `int`-sized and use `%u`, `%d`, and `%x` to print them.
- Assume `UINTN`/`INTN` to be of unspecified size, and cast them to `UINT64`/`INT64` for printing with `%Lu`, `%Ld` and so on as normal.
- Do not rely on integer promotions for numeric literals. Use explicit casts when the type is implementation-dependent or suffixes when type size is known. Assume `U` for `UINT32` and `ULL` for `UINT64`.
- Do ensure unsigned arithmetics especially in bitwise maths, shifts in particular.
- `sizeof` operator should take variables instead of types where possible to be error prone. Use `ARRAY_SIZE` to obtain array size in elements. Use `L_STR_LEN` and `L_STR_SIZE` macros from `OcStringLib` to obtain string literal sizes to ensure compiler optimisation.

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 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 titles for boot entries:
  – `.disk_label` (`.disk_label_2x`) file near 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, (`.contentDetails` or `.disk_label.contentDetails`) are to be rendered instead.
- `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.

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 `isVOEnabled.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 workaround 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

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 *):

`BootKicker*`	Enter Apple BootPicker menu (exclusive for Macs with compatible GPUs).
`ChipTune*`	Test BeepGen protocol and generate audio signals of different style and length.
`CleanNvram*`	Reset NVRAM alternative bundled as a standalone tool.
`GopStop*`	Test GraphicsOutput protocol with a simple scenario.
`KeyTester*`	Test keyboard input in `SimpleText` mode.
`MemTest86`	Memory testing utility.
`OpenControl*`	Unlock and lock back NVRAM protection for other tools to be able to get full NVRAM access when launching from OpenCore.
`OpenShell*`	OpenCore-configured `UEFI Shell` for compatibility with a broad range of firmware.
`PavpProvision`	Perform EPID provisioning (requires certificate data configuration).
`ResetSystem*`	Utility to perform system reset. Takes reset type as an argument: ~~ColdReset~~coldreset, ~~Firmware~~firmware, ~~Shutdown~~shutdown, ~~WarmReset~~warmreset. Defaults to ~~ColdReset~~coldreset.
`RtcRw*`	Utility to read and write RTC (CMOS) memory.
`ControlMsrE2*`	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 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 saved in 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).

Failsafe: 0
Description: Minimal allowed APFS driver version.

The APFS driver version connects the APFS driver with the macOS release. 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 version of APFS in OpenCore. The default version will increase with time and thus this setting is recommended. Currently set to the latest point release from High Sierra from App Store (748077008000000).
- -1 — permit any version to load (strongly discouraged).
- Other — use custom minimal APFS version, e.g. 1412101001000000 from macOS Catalina 10.15.4. APFS versions can be found in OpenCore boot log and OcApfsLib.

11.8 AppleInput Properties

1. AppleEvent
   Type: plist string
   Failsafe: Auto
   Description: Determine whether OC builtin or OEM Apple Event protocol is used.

   This option determines whether Apple's OEM Apple Event protocol is used (where available), or whether OpenCore's reversed engineered and updated re-implementation is used. In general OpenCore's re-implementation should be preferred, since it contains updates such as noticeably improved fine mouse cursor movement and configurable key repeat delays.

   - Auto — ~~Performs automatic choice of implementation . Because of optimisations used to achieve fast boot times, this actually means that the OpenCore re-implementation will be found and used except in the case that OpenCore was been explicitly selected and started (not just auto-booted) from Apple 's boot picker (where present). ~~Use OEM Apple Event implementation if available, connected and recent enough to be used, otherwise use OC reimplementation. On non-Apple hardware this will use the OpenCore builtin implementation. On some Macs (e.g. classic Mac Pro) this will find the Apple implementation. On both older and newer Macs than this, this option will always or often use the OC implementation. On older Macs this is because the implementation available is too old to be used, on newer Macs it is because of optimisations added by Apple which do not connect the Apple Event protocol except when needed – e.g. except when the Apple boot picker is explicitly started. Due to its somewhat unpredicatable results, this option is not normally recommended.
   - Builtin — ~~Use~~ Always use OpenCore's updated re-implementation of the Apple Event protocol. ~~Recommended~~Use of this setting is recommended even on Apple hardware, due to improvements (better fine mouse control, configurable key delays) made in the OC re-implementation of the protocol.
   - OEM — Assume Apple's protocol will be available at driver connection. ~~This results in Apple 's implementation being reliably used on Apple systems. It results~~ On all Apple hardware where a recent enough Apple OEM version of the protocol is available – whether or not connected automatically by Apple's firmware – this option will reliably access the Apple implementation. On all other systems, this option will result in no keyboard or mouse support~~otherwise. ~~. For the reasons stated, Builtin is recommended in preference to this option in most cases.

2. CustomDelays
   Type: plist ~~string~~boolean
   Failsafe: ~~Auto~~false
   Description: Enable custom key repeat delays ~~.~~ when using the OpenCore implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see AppleEvent setting).

   - ~~Auto — Treated as Enabled when KeySupport is~~ true ~~and Disabled otherwise.~~
   - ~~Enabled~~ — The values of KeyInitialDelay and KeySubsequentDelay are used.
   - ~~Disabled~~false — Apple default values of 500ms (50) and 50ms (5) are used.

3. KeyInitialDelay
   Type: plist integer
   Failsafe: ~~0~~50 (~~no initial delay, immediate~~ 500ms before first key repeat)

Description: ~~Configure initial keyboard repeat delay~~ Configures the initial delay before keyboard key repeats in OpenCore implementation of Apple Event protocol, in units of 10ms.

~~Configures the initial delay before key repeat.~~ The Apple OEM default value is `50` (500ms).

Note 1: ~~When~~ On systems not using `KeySupport`, this setting may be freely used to configure key repeat behaviour.

Note 2: On systems using `KeySupport`, but which do not show the 'two long delays' behavior (see Note 3) and/or which always show a solid 'set default' indicator (see `KeyForgetThreshold`) then this setting may also be freely used to configure key repeat initial delay behaviour, except that it should never be set to less than `KeyForgetThreshold` to avoid uncontrolled key repeats.

Note 3: On some systems using `KeySupport`, you may find that you ~~get an~~ see one additional slow key repeat before normal speed key repeat starts, when holding a key down. If so, ~~your initial key repeat delay is being driven by your BIOS firmware and cannot be overriden by OC (due to technical limitations of how `KeySupport` works, to derive raw key data from the non-raw key data which is all that UEFI makes available). To avoid this minor but undesired effect of two long repeats, you can simply cancel the second, Apple Event, repeat by setting `KeyInitialDelay` to~~ you may wish to configure `0``KeyInitialDelay` ~~. When doing this you should also set~~ and `KeySubsequentDelay` ~~to at least the value of your `KeyForgetThreshold` setting (see more information in the~~ according to the instructions at Note 3 of `KeySubsequentDelay`~~setting). The instructions in this note only apply on systems using `KeySupport`~~.

4. `KeySubsequentDelay`
   Type: `plist integer`
   Failsafe: ~~1~~`5` (50ms between subsequent key repeats)
   Description: ~~Configure subsequent keyboard repeat delay~~ Configures the gap between keyboard key repeats in OpenCore implementation of Apple Event protocol, in units of 10ms.

   ~~Configures the gap between key repeats.~~ The Apple OEM default value is `5` (50ms). `0` is an invalid value for this option (will issue a debug log warning and use `1` instead).

   Note 1: ~~When~~ On systems not using `KeySupport`, this setting may be freely used to configure key repeat behaviour.

   Note 2: On systems using `KeySupport`, but which do not show the 'two long delays' behaviour (see Note 3) and/or which always show a solid 'set default' indicator (see `KeyForgetThreshold`) (which should apply to many/most systems using `AMI` `KeySupport` mode) then this setting may be freely used to configure key repeat subsequent delay behaviour, except that it should never be set to less than `KeyForgetThreshold` to avoid uncontrolled key repeats.

   Note 3: On some systems using `KeySupport`, particularly `KeySupport` in non-`AMI` mode, you may find that after configuring `KeyForgetThreshold` you get one additional slow key repeat before normal speed key repeat starts~~. If so, set~~, when holding a key down. On systems where this is the case, it is an unavoidable artefect of using `KeySupport` to emulate raw keyboard data, which is not made available by UEFI. While this 'two long delays' issue has minimal effect on overall usability, nevertheless you may wish to resolve it, and it is possible to do so as follows:

   - Set `CustomDelays` to `true`
   - Set `KeyInitialDelay` to `0` ~~and set~~
   - Set `KeySubsequentDelay` to at least the value of your `KeyForgetThreshold` setting ~~. The reason for this is that the key-smoothing~~

   The above procedure works as follows:

   - Setting `KeyInitialDelay` to `0` cancels the Apple Event initial repeat delay (when using the OC builtin Apple Event implementation with `CustomDelays` enabled), therefore the only long delay you will see is the the non-configurable and non-avoidable initial long delay introduced by the BIOS key support on these machines.
   - Key-smoothing parameter `KeyForgetThreshold` effectively acts as the shortest time for which a key can appear to be held, therefore a key repeat delay of less than this will guarantee at least one extra repeat for every key press, however quickly the key is physically tapped. ~~(~~

- In the unlikely event that you still get frequent, or occasional, double key responses after setting `KeySubsequentDelay` equal to your system's value of `KeyForgetThreshold`, then increase `KeySubsequentDelay` by one or two more until this effect goes away~~ - for greatest keyboard responsiveness, use the lowest value which avoids multiple keypresses.) The instructions in this note only apply on systems using `KeySupport`~~.

5. `PointerSpeedDiv`
   Type: `plist integer`
   Failsafe: `1`~~)~~
   Description: Configure pointer speed divisor in OpenCore implementation of Apple Event protocol. Has no effect when using the OEM Apple implementation (see `AppleEvent` setting).

   Configures the divisor for pointer movements. The Apple OEM default value is `1`. `0` is an invalid value for this option.

   Note: The recommended value for this option is `1`. This value may optionally be modified in combination with `PointerSpeedMul`, according to user preference, to achieve customised mouse movement scaling.

6. `PointerSpeedMul`
   Type: `plist integer`
   Failsafe: ~~`0`~~`1`~~)~~
   Description: Configure pointer speed multiplier in OpenCore implementation of Apple Event protocol. Has no effect when using the OEM Apple implementation (see `AppleEvent` setting).

   Configures the multiplier for pointer movements. The Apple OEM default value is `1`.

   Note: The recommended value for this option is `1`. This value may optionally be modified in combination with `PointerSpeedDiv`, according to user preference, to achieve customised mouse movement scaling.

11.9 Audio Properties

1. `AudioCodec`
   Type: `plist integer`
   Failsafe: `0`
   Description: Codec address on the specified audio controller for audio support.

   This typically contains the first audio codec address on the builtin analog audio controller (`HDEF`). Audio codec addresses, e.g. `2`, can be found in the debug log (marked in bold-italic):

   ```
   OCAU: 1/3 PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x1)/VenMsg(<redacted>,00000000) (4 outputs)
   OCAU: 2/3 PciRoot(0x0)/Pci(0x3,0x0)/VenMsg(<redacted>,00000000) (1 outputs)
   OCAU: 3/3 PciRoot(0x0)/Pci(0x1B,0x0)/VenMsg(<redacted>,02000000) (7 outputs)
   ```

   As an alternative, this value can be obtained from `IOHDACodecDevice` class in I/O Registry containing it in `IOHDACodecAddress` field.

2. `AudioDevice`
   Type: `plist string`
   Failsafe: Empty
   Description: Device path of the specified audio controller for audio support.

   This typically contains builtin analog audio controller (`HDEF`) device path, e.g. `PciRoot(0x0)/Pci(0x1b,0x0)`. The list of recognised audio controllers can be found in the debug log (marked in bold-italic):

   ```
   OCAU: 1/3 PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x1)/VenMsg(<redacted>,00000000) (4 outputs)
   OCAU: 2/3 PciRoot(0x0)/Pci(0x3,0x0)/VenMsg(<redacted>,00000000) (1 outputs)
   OCAU: 3/3 PciRoot(0x0)/Pci(0x1B,0x0)/VenMsg(<redacted>,02000000) (7 outputs)
   ```

   As an alternative, `gfxutil -f HDEF` command can be used in macOS. Specifying an empty device path will result in the first available audio controller being used.

3. `AudioOut`
   Type: `plist integer`
   Failsafe: `0`
   Description: Index of the output port of the specified codec starting from 0.

Volume level range read from `SystemAudioVolume` varies depending on the codec. To transform read value in `[0, 127]` range into raw volume range `[0, 100]` the read value is scaled to `VolumeAmplifier` percents:

$$RawVolume = MIN(\frac{SystemAudioVolume * VolumeAmplifier}{100}, 100)$$

Note: the transformation used in macOS is not linear, but it is very close and this nuance is thus ignored.

11.10 Input Properties

1. `KeyFiltering`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enable keyboard input sanity checking.

   Apparently some boards such as the GA Z77P-D3 may return uninitialised data in `EFI_INPUT_KEY` with all input protocols. This option discards keys that are neither ASCII, nor are defined in the UEFI specification (see tables 107 and 108 in version 2.8).

2. `KeyForgetThreshold`
   Type: `plist integer`
   Failsafe: `0`
   Description: Treat duplicate key presses as held keys if they arrive during this timeout, in 10 ms units. Only applies to systems using `KeySupport`.

   `AppleKeyMapAggregator` protocol is supposed to contain a fixed length buffer of currently pressed keys. However, the majority of the drivers which require `KeySupport` report key presses as interrupts, with automatically generated key repeat behaviour with some defined initial and subsequent delay. As a result, to emulate the raw key behaviour required by several Apple boot systems, we use a timeout to merge multiple repeated keys which are submitted within a small timeout window.

   This option allows setting this timeout based on the platform. The recommended value for the majority of platforms is from `5` (`50` milliseconds) to `7` (`70` milliseconds), although values up to `9` (`90` milliseconds) have been observed to be required on some PS/2 systems. For reference, holding a key on VMware will repeat roughly every `20` milliseconds and the equivalent value for APTIO V is `30-40` milliseconds. `KeyForgetThreshold` should be configured to be longer than this. Thus, it is possible to configure a lower `KeyForgetThreshold` value on platforms with a faster native driver key repeat rate, for more responsive input, and it is required to set a higher value on slower platforms.

   Pressing keys one after the other results in delays of at least `60` and `100` milliseconds for the same platforms. Ideally, `KeyForgetThreshold` should remain lower than this value, to avoid merging real key presses.

   ~~*Note*: If you wish to fine tune this value , a good heuristic is to~~ Tuning the value of `KeyForgetThreshold` is necessary for accurate and responsive keyboard input on systems on which `KeySupport` is enabled, and it is recommended to follow the instructions below to tune it correctly for your system.

   Note 1: To tune `KeyForgetThreshold`, you may use the 'set default' indicator within either OpenCanopy or the builtin picker. ~~When~~ If `KeyForgetThreshold` is ~~configured correctly, this indicator should~~ too low then the 'set default' indicator will continue to flicker while `CTRL` or `=/+` is held down. You should configure the lowest value which avoids this flicker. On some systems (e.g. Aptio IV and potentially other systems using `AMI` `KeySupport` mode) you will be able to find a minimum `KeyForgetThreshold` value at which the 'set default' indicator goes on and stays on with no flicker at all - if so, use this value. On most other systems using `KeySupport`, you will find that the 'set default' indicator will flicker once, when first pressing and holding the `CTRL` or `=/+` key, and then after a further very brief interval ~~should~~ will go on and stay on. ~~(The initial flicker~~ On such systems, you should chose the lowest value of `KeyForgetThreshold` at which you see only one initial flicker and then no subsequent flickering. (Where this happens, it is an unavoidable ~~artefact~~ artefect on those systems of using `KeySupport` to emulate raw keyboard data~~.) If~~ , which is not made available by UEFI.)

   Note 2: `KeyForgetThreshold` ~~is configured too low for the system, then~~ should never need to be more than about `9` or `10` at most. If it is set to a value much higher than this, it will result in noticeably unresponsive keyboard input. Therefore, for overall key responsiveness, it is strongly recommended to configure a relatively lower value, at which the 'set default' indicator ~~will continue to flickerwhile `CTRL` or `=/+` is held. Configure the lowest value which causes this indicator to go on and stay on after the first initial flicker~~ flickers once and then

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. ~~`AppleEvent`**Type**: `plist boolean`**Failsafe**: `false`**Description**: Replaces the Apple Event protocol with a builtin version. This may be used to ensure FileVault 2 compatibility on VMs and legacy Macs.~~

Description: Replaces the Device Property protocol with a builtin version. This may be used to ensure full compatibility on VMs and legacy Macs.

Note: This will discard all previous entries if the protocol was already installed, so all properties required for safe operation of the system must be specified in the configuration file.

15. `FirmwareVolume`
Type: `plist boolean`
Failsafe: `false`
Description: Wraps Firmware Volume protocols, or installs a new version, to support custom cursor images for FileVault 2. Set to `true` to ensure FileVault 2 compatibility on anything other than on VMs and legacy Macs.

Note: Several virtual machines, including VMware, may have corrupted cursor images in HiDPI mode and thus, may also require enabling this setting.

16. `HashServices`
Type: `plist boolean`
Failsafe: `false`
Description: Replaces Hash Services protocols with builtin versions. Set to `true` to ensure FileVault 2 compatibility on platforms with defective SHA-1 hash implementations. This can be determined by an invalid cursor size when `UIScale` is set to `02`. Platforms earlier than APTIO V (Haswell and older) are typically affected.

17. `OSInfo`
Type: `plist boolean`
Failsafe: `false`
Description: Replaces the OS Info protocol with a builtin version. This protocol is typically used by the firmware and other applications to receive notifications from the macOS bootloader.

18. `UnicodeCollation`
Type: `plist boolean`
Failsafe: `false`
Description: Replaces unicode collation services with builtin versions. Set to `true` to ensure UEFI Shell compatibility on platforms with defective unicode collation implementations. Legacy Insyde and APTIO platforms on Ivy Bridge, and earlier, are typically affected.

11.13 Quirks Properties

1. `ActivateHpetSupport`
Type: `plist boolean`
Failsafe: `false`
Description: Activates HPET support.

Older boards like ICH6 may not always have HPET setting in the firmware preferences, this option tries to force enable it.

2. `EnableVectorAcceleration`
Type: `plist boolean`
Failsafe: `false`
Description: Enable AVX vector acceleration of SHA-512 and SHA-384 hashing algorithms.

3. `DisableSecurityPolicy`
Type: `plist boolean`
Failsafe: `false`
Description: Disable platform security policy.

Note: This setting disables various security features of the firmware, defeating the purpose of any kind of Secure Boot. Do NOT enable if using UEFI Secure Boot.

4. `ExitBootServicesDelay`
Type: `plist integer`
Failsafe: `0`
Description: Adds delay in microseconds after `EXIT_BOOT_SERVICES` event.

This is a very rough workaround to circumvent the `Still waiting for root device` message on some APTIO IV firmware (ASUS Z87-Pro) particularly when using FileVault 2. It appears that for some reason, they execute code in parallel to `EXIT_BOOT_SERVICES`, which results in the SATA controller being inaccessible from macOS. A better approach is required and Acidanthera is open to suggestions. Expect 3 to 5 seconds to be adequate when this quirk is needed.

5. `ForgeUefiSupport`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Implement partial UEFI 2.x support on EFI 1.x firmware.

   This setting allows running some software written for UEFI 2.x firmware like NVIDIA GOP Option ROMs on hardware with older EFI 1.x firmware like `MacPro5,1`.

6. `IgnoreInvalidFlexRatio`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Some types of firmware (such as APTIO IV) may contain invalid values in the `MSR_FLEX_RATIO` (`0x194`) MSR register. These values may cause macOS boot failures on Intel platforms.

   Note: While the option is not expected to harm unaffected firmware, its use is recommended only when specifically required.

7. `ReleaseUsbOwnership`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Attempt to detach USB controller ownership from the firmware driver. While most types of firmware manage to do this properly, or at least have an option for this, some do not. As a result, the operating system may freeze upon boot. Not recommended unless specifically required.

8. `ReloadOptionRoms`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Query PCI devices and reload their Option ROMs if available.

   For example, this option allows reloading NVIDIA GOP Option ROM on older Macs after the firmware version is upgraded via `ForgeUefiSupport`.

9. `RequestBootVarRouting`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Request redirect of all `Boot` prefixed variables from `EFI_GLOBAL_VARIABLE_GUID` to `OC_VENDOR_VARIABLE_GUID`.

   This quirk requires `OC_FIRMWARE_RUNTIME` protocol implemented in `OpenRuntime.efi`. The quirk lets default boot entry preservation at times when the firmware deletes incompatible boot entries. In summary, this quirk is required to reliably use the Startup Disk preference pane in firmware that is not compatible with macOS boot entries by design.

   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 corrupted in any way.

10. `TscSyncTimeout`
    Type: `plist integer`
    Failsafe: `0`
    Description: Attempts to perform TSC synchronisation with a specified timeout.

    The primary purpose of this quirk is to enable early bootstrap TSC synchronisation on some server and laptop models when running a debug XNU kernel. For the debug kernel the TSC needs to be kept in sync across the cores