OpenCore

Reference Manual (0.7.~~5~~.6)

[2021.11.28]

14. `ProvideCustomSlide`
Type: `plist boolean`
Failsafe: `false`
Description: Provide custom KASLR slide on low memory.

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

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

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

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

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

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

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

- The Memory map size must not exceed 4096 bytes as the Apple kernel maps it as a single 4K page. As some types of firmware can have very large memory maps, potentially over 100 entries, the Apple kernel will crash on boot.
- The Memory attributes table is ignored. `EfiRuntimeServicesCode` memory statically gets `RX` permissions while all other memory types get `RW` permissions. As some firmware drivers may write to global variables at runtime, the Apple kernel will crash at calling UEFI runtime services unless the driver `.data` section has a `EfiRuntimeServicesData` type.

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

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

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

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

This quirk reduces GPU PCI BAR sizes for Apple macOS up to the specified value or lower if it is unsupported. The specified value follows PCI Resizable BAR spec. ~~Use 0 for 1 MB, 1 for 2 MB, 2 for 4 MB, and so on up to 19 for 512 GB.~~ While Apple macOS supports a theoretical 1 GB maximum, ~~which is 10.~~in practice all non-default

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

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

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

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

Note~~1~~: See `ResizeGpuBars` quirk for general GPU PCI BAR size configuration and more details about the technology.

~~*Note 2*: Certain GPU drivers do not support non-standard BAR sizes, causing sleep wake issues, for this reason for macOS it is recommended to use minimal supported BAR sizes, i.e. specify `0` (1 MB).~~

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

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

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

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

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

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

Some types of firmware fail to properly handle runtime permissions:

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

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

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

- 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`).

   Note 1: 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.

   Note 2: ~~OC~~ OpenCore 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).

VirtualSMC performs authenticated restarts by splitting and saving disk encryption keys between NVRAM and RTC, which despite being removed as soon as OpenCore starts, may be considered a security risk and thus is optional.

6. `BlacklistAppleUpdate`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Ignore boot options trying to update Apple peripheral firmware (e.g. `MultiUpdater.efi`).

   Note: Certain operating systems, such as macOS Big Sur, are incapable of disabling firmware updates by using the `run-efi-updater` NVRAM variable.

7. `DmgLoading`
   Type: `plist string`
   Failsafe: `Signed`
   Description: Define Disk Image (DMG) loading policy used for macOS Recovery.

   Valid values:

   - `Disabled` — loading DMG images will fail. The `Disabled` policy will still let the macOS Recovery load in most cases as typically, there are `boot.efi` files compatible with Apple Secure Boot. Manually downloaded DMG images stored in `com.apple.recovery.boot` directories will not load, however.
   - `Signed` — only Apple-signed DMG images will load. Due to the design of Apple Secure Boot, the `Signed` policy will let any Apple-signed macOS Recovery load regardless of the Apple Secure Boot state, which may not always be desired. While using signed DMG images is more desirable, verifying the image signature may slightly slow the boot time down (by up to 1 second).
   - `Any` — any DMG images will mount as normal filesystems. The `Any` policy is strongly discouraged and will result in boot failures when Apple Secure Boot is active.

8. `EnablePassword`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enable password protection to facilitate sensitive operations.

   Password protection ensures that sensitive operations such as booting a non-default operating system (e.g. macOS recovery or a tool), resetting NVRAM storage, trying to boot into a non-default mode (e.g. verbose mode or safe mode) are not allowed without explicit user authentication by a custom password. Currently, password and salt are hashed with 5000000 iterations of SHA-512.

   Note: This functionality is still under development and is not ready for production environments.

9. `ExposeSensitiveData`
   Type: `plist integer`
   Failsafe: `0x6`
   Description: Sensitive data exposure bitmask (sum) to operating system.

   - `0x01` — Expose the printable booter path as ~~an~~ a UEFI variable.
   - `0x02` — Expose the OpenCore version as ~~an~~ a UEFI variable.
   - `0x04` — Expose the OpenCore version in the OpenCore picker menu title.
   - `0x08` — Expose OEM information as a set of UEFI variables.

   The exposed booter path points to OpenCore.efi or its booter depending on the load order. To obtain the booter path, use the following command in macOS:

```
nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-path
```

   To use a booter path to mount a booter volume, use the following command in macOS:

```
u=$(nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-path | sed 's/.*GPT,\([^,]*\),.*/\1/'); \
  if [ "$u" != "" ]; then sudo diskutil mount $u ; fi
```

   To obtain the current OpenCore version, use the following command in macOS:

```
nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:opencore-version
```

11 UEFI

11.1 Introduction

UEFI (Unified Extensible Firmware Interface) is a specification that defines a software interface between an operating system and platform firmware. This section allows loading additional UEFI modules as well as applying tweaks to the onboard firmware. To inspect firmware contents, apply modifications and perform upgrades UEFITool and supplementary utilities can be used.

11.2 Drivers

Depending on the firmware, a different set of drivers may be required. Loading an incompatible driver may lead the system to unbootable state or even cause permanent firmware damage. Some of the known drivers are listed below:

`AudioDxe*`	HDA audio support driver in UEFI firmware for most Intel and some other analog audio controllers. Staging driver, refer to acidanthera/bugtracker#740 for known issues in AudioDxe.
`btrfs_x64`	Open source BTRFS file system driver, required for booting with OpenLinuxBoot from a file system which is now quite commonly used with Linux.
`BiosVideo*`	CSM video driver implementing graphics output protocol based on VESA and legacy BIOS interfaces. Used for UEFI firmware with fragile GOP support (e.g. low resolution). Requires `ReconnectGraphicsOnConnect`. Included in OpenDuet out of the box.
`CrScreenshotDxe*`	Screenshot making driver saving images to the root of OpenCore partition (ESP) or any available writeable filesystem upon pressing `F10`. This is a modified version of `CrScreenshotDxe` driver by Nikolaj Schlej.
`ExFatDxe`	Proprietary ExFAT file system driver for Bootcamp support commonly found in Apple firmware. For Sandy Bridge and earlier CPUs, the `ExFatDxeLegacy` driver should be used due to the lack of `RDRAND` instruction support.
`ext4_x64`	Open source EXT4 file system driver, required for booting with OpenLinuxBoot from the file system most commonly used with Linux.
`HfsPlus`	Recommended. Proprietary HFS file system driver with bless support commonly found in Apple firmware. For Sandy Bridge and earlier CPUs, the `HfsPlusLegacy` driver should be used due to the lack of `RDRAND` instruction support.
`HiiDatabase*`	HII services support driver from `MdeModulePkg`. This driver is included in most types of firmware starting with the Ivy Bridge generation. Some applications with GUI, such as UEFI Shell, may need this driver to work properly.
`EnhancedFatDxe`	FAT filesystem driver from `FatPkg`. This driver is embedded in all UEFI firmware and cannot be used from OpenCore. Several types of firmware have defective FAT support implementation that may lead to corrupted filesystems on write attempts. Embedding this driver within the firmware may be required in case writing to the EFI partition is needed during the boot process.
`NvmExpressDxe*`	NVMe support driver from `MdeModulePkg`. This driver is included in most firmware starting with the Broadwell generation. For Haswell and earlier, embedding it within the firmware may be more favourable in case a NVMe SSD drive is installed.
`OpenCanopy*`	OpenCore plugin implementing graphical interface.
`OpenRuntime*`	OpenCore plugin implementing `OC_FIRMWARE_RUNTIME` protocol.
`OpenLinuxBoot*`	OpenCore plugin implementing `OC_BOOT_ENTRY_PROTOCOL` to allow direct detection and booting of Linux distributiuons from OpenCore, without chainloading via GRUB.
`OpenUsbKbDxe*`	USB keyboard driver adding support for `AppleKeyMapAggregator` protocols on top of a custom USB keyboard driver implementation. This is an alternative to builtin `KeySupport`, which may work better or worse depending on the firmware.
`OpenPartitionDxe*`	Partition management driver with Apple Partitioning Scheme support. This driver can be used to support loading older DMG recoveries such as macOS 10.9 using Apple Partitioning Scheme. OpenDuet already includes this driver.
`Ps2KeyboardDxe*`	PS/2 keyboard driver from `MdeModulePkg`. `OpenDuetPkg` and some types of firmware may not include this driver, but it is necessary for PS/2 keyboard to work. Note, unlike `OpenUsbKbDxe` this driver has no `AppleKeyMapAggregator` support and thus requires `KeySupport` to be enabled.

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 2021/01/01.
- `-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 `OcApfsLib`.

6. `MinVersion`
Type: `plist integer`
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 allow macOS Big Sur and newer (`1600000000000000`).
- `-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`.

11.9 AppleInput Properties

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

This option determines whether ~~Apple's~~ the 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` — Use the OEM Apple Event implementation if available, connected and recent enough to be used, otherwise use ~~OC reimplementation~~the OpenCore re-implementation. On non-Apple hardware, this will use the OpenCore builtin implementation. On some Macs ~~(e.g. classic Mac Pro) this will find~~ such as Classic Mac Pros, this will prefer the Apple implementation ~~. On~~ but on both older and newer ~~Macs than this~~Mac models than these, this option will ~~always or often use the OC implementation~~typically use the OpenCore re-implementation instead. On older Macs, this is because the implementation available is too old to be used ~~,~~ while 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~~ typically recommended.
- `Builtin` — Always use OpenCore's updated re-implementation of the Apple Event protocol. Use of this setting is recommended even on Apple hardware, due to improvements (better fine mouse control, configurable key delays) made in the ~~OC~~ OpenCore re-implementation of the protocol.
- `OEM` — Assume Apple's protocol will be available at driver connection. 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. For the reasons stated, `Builtin` is recommended in preference to this option in most cases.

2. `CustomDelays`
Type: `plist boolean`

Failsafe: `false`
Description: Enable custom key repeat delays when using the OpenCore ~~implementation~~ re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see `AppleEvent` setting).

- `true` — The values of `KeyInitialDelay` and `KeySubsequentDelay` are used.
- `false` — Apple default values of 500ms (`50`) and 50ms (`5`) are used.

3. `KeyInitialDelay`
Type: `plist integer`
Failsafe: `50` (500ms before first key repeat)
Description: Configures the initial delay before keyboard key repeats in ~~OpenCore implementation of~~ the OpenCore re-implementation of the Apple Event protocol, in units of 10ms.

The Apple OEM default value is `50` (500ms).

Note 1: 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 see one additional slow key repeat before normal speed key repeat starts, when holding a key down. If so, you may wish to configure `KeyInitialDelay` and `KeySubsequentDelay` according to the instructions at Note 3 of `KeySubsequentDelay`.

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

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: 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, 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`
- Set `KeySubsequentDelay` to at least the value of your `KeyForgetThreshold` setting

The above procedure works as follows:

- Setting `KeyInitialDelay` to `0` cancels the Apple Event initial repeat delay (when using the ~~OC~~ OpenCore 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.

5. `GraphicsInputMirroring`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Apple's own implementation of AppleEvent prevents keyboard input during graphics applications from appearing on the basic console input stream.

   With the default setting of `false`, ~~OC~~OpenCore's builtin implementation of AppleEvent replicates this behaviour.

   On non-Apple hardware this can stop keyboard input working in graphics-based applications such as Windows BitLocker which use non-Apple key input methods.

   The recommended setting on all hardware is `true`.

   Note: AppleEvent's default behaviour is intended to prevent unwanted queued keystrokes from appearing after exiting graphics-based UEFI applications; this issue is already handled separately within OpenCore.

   - `true` — Allow keyboard input to reach graphics mode apps which are not using Apple input protocols.
   - `false` — Prevent key input mirroring to non-Apple protocols when in graphics mode.

6. `PointerSpeedDiv`
   Type: `plist integer`
   Failsafe: `1`
   Description: Configure pointer speed divisor in ~~OpenCore implementation of~~ the OpenCore re-implementation of the 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.

7. `PointerSpeedMul`
   Type: `plist integer`
   Failsafe: `1`
   Description: Configure pointer speed multiplier in ~~OpenCore implementation of~~ the OpenCore re-implementation of the 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.

11.10 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.

8. `SetupDelay`
   Type: `plist integer`
   Failsafe: `0`
   Description: Audio codec reconfiguration delay in microseconds.

   Some codecs require a vendor-specific delay after the reconfiguration (e.g. volume setting). This option makes it configurable. A typical delay can be up to 0.5 seconds.

9. `VolumeAmplifier`
   Type: `plist integer`
   Failsafe: `0`
   Description: Multiplication coefficient for system volume to raw volume linear translation from `0` to `1000`.

   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.

11.11 Drivers Properties

1. `Comment`
   Type: `plist string`
   Failsafe: Empty
   Description: Arbitrary ASCII string used to provide human readable reference for the entry. Whether this value is used is implementation defined.

2. `Path`
   Type: `plist string`
   Failsafe: Empty
   Description: Path of file to be loaded as a UEFI driver from `OC/Drivers` directory.

3. `Enabled`
   Type: `plist boolean`
   Failsafe: `false`
   Description: If `false` this driver entry will be ignored.

4. `Arguments`
   Type: `plist string`
   Failsafe: Empty
   Description: Some ~~OC~~ OpenCore plugins accept optional additional arguments which may be specified as a string here.

11.12 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

Failsafe: `Disabled`
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. `ReconnectGraphicsOnConnect`
Type: `plist boolean`
Failsafe: `false`
Description: Reconnect all graphics drivers during driver connection.

    On certain firmware, it may be desireable to use an alternative graphics driver, for example BiosVideo.efi, providing better screen resolution options on legacy machines, or a driver supporting `ForceResolution`. This option attempts to disconnect all currently connected graphics drivers before connecting newly loaded drivers.

    Note: This option requires `ConnectDrivers` to be enabled.

12. `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.

13. `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.

14. `UIScale`
Type: `plist integer`, 8 bit
Failsafe: `-1`
Description: User interface scaling factor.

Corresponds to `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:UIScale` variable.

- `1` — 1x scaling, corresponds to normal displays.
- `2` — 2x scaling, corresponds to HiDPI displays.
- `-1` — leaves the current variable unchanged.
- `0` — automatically chooses scaling based on the current resolution.

Note 1: Automatic scale factor detection works on the basis of total pixel area and may fail on small HiDPI displays, in which case the value may be manually managed using the NVRAM section.

Note 2: When switching from manually specified NVRAM variable to this preference an NVRAM reset may be needed.

15. `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.

11.14 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`

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. `ForceOcWriteFlash`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enables writing to flash memory for all ~~OpenCore~~ OpenCore-managed NVRAM system variables.

   Note: This value should be disabled on most types of firmware but is left configurable to account for firmware that may have issues with volatile variable storage overflows or similar. Boot issues across multiple OSes can be observed on e.g. Lenovo Thinkpad T430 and T530 without this quirk. Apple variables related to Secure Boot and hibernation are exempt from this for security reasons. Furthermore, some OpenCore variables are exempt for different reasons, such as the boot log due to an available user option, and the TSC frequency due to timing issues. When toggling this option, a NVRAM reset may be required to ensure full functionality.

6. `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`.

7. `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.

8. `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.

9. `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`.