OpenCore

Reference Manual (0.8.2.3)

[2022.07.24]

4 ACPI

4.1 Introduction

ACPI (Advanced Configuration and Power Interface) is an open standard to discover and configure computer hardware. The ACPI specification defines standard tables (e.g. `DSDT`, `SSDT`, `FACS`, `DMAR`) and various methods (e.g. `_DSM`, `_PRW`) for implementation. Modern hardware needs few changes to maintain ACPI compatibility and some options for such changes are provided as part of OpenCore.

To compile and disassemble ACPI tables, the iASL compiler developed by ACPICA can be used. A GUI front-end to iASL compiler can be downloaded from Acidanthera/MaciASL.

ACPI changes apply globally (to every operating system) with the following effective order:

- ~~Patch~~Delete is processed.
- ~~Delete~~Quirks ~~is~~ are processed.
- ~~Add~~Patch is processed.
- ~~Quirks~~Add ~~are~~ is processed.

Applying the changes globally resolves the problems of incorrect operating system detection (consistent with the ACPI specification, not possible before the operating system boots), operating system chainloading, and difficult ACPI debugging. Hence, more attention may be required when writing changes to `_OSI`.

Applying the patches early makes it possible to write so called "proxy" patches, where the original method is patched in the original table and is implemented in the patched table.

There are several sources of ACPI tables and workarounds. Commonly used ACPI tables are provided with OpenCore, VirtualSMC, VoodooPS2, and WhateverGreen releases. Besides those, several third-party instructions may be found on the AppleLife Laboratory and DSDT subforums (e.g. Battery register splitting guide). A slightly more user-friendly explanation of some tables included with OpenCore can also be found in Dortania's Getting started with ACPI guide. For more exotic cases, there are several alternatives such as daliansky's ACPI sample collection (English Translation by 5T33Z0 et al). Please note however, that suggested solutions from third parties may be outdated or may contain errors.

4.2 Properties

1. `Add`
   Type: `plist array`
   Failsafe: Empty
   Description: Load selected tables from the `OC/ACPI` directory.

   To be filled with `plist dict` values, describing each add entry. Refer to the Add Properties section below for details.

2. `Delete`
   Type: `plist array`
   Failsafe: Empty
   Description: Remove selected tables from the ACPI stack.

   To be filled with `plist dict` values, describing each delete entry. Refer to the Delete Properties section below for details.

3. `Patch`
   Type: `plist array`
   Failsafe: Empty
   Description: Perform binary patches in ACPI tables before table addition or removal.

   To be filled with `plist dictionary` values describing each patch entry. Refer to the Patch Properties section below for details.

4. `Quirks`
   Type: `plist dict`
   Description: Apply individual ACPI quirks described in the Quirks Properties section below.

5 Booter

5.1 Introduction

This section allows the application of different types of UEFI modifications to operating system bootloaders, primarily the Apple bootloader (`boot.efi`). The modifications currently provide various patches and environment alterations for different firmware types. Some of these features were originally implemented as part of AptioMemoryFix.efi, which is no longer maintained. Refer to the Tips and Tricks section for instructions on migration.

Booter changes apply with the following effective order:

- `Quirks` are processed.
- `Patch` is processed.

If this is used for the first time on customised firmware, the following requirements should be met before starting:

- Most up-to-date UEFI firmware (check the motherboard vendor website).
- `Fast Boot` and `Hardware Fast Boot` disabled in firmware settings if present.
- `Above 4G Decoding` or similar enabled in firmware settings if present. Note that on some motherboards, notably the ASUS WS-X299-PRO, this option results in adverse effects and must be disabled. While no other motherboards with the same issue are known, this option should be checked first whenever erratic boot failures are encountered.
- `DisableIoMapper` quirk enabled, or `VT-d` disabled in firmware settings if present, or ACPI DMAR table deleted.
- **No** 'slide' boot argument present in NVRAM or anywhere else. It is not necessary unless the system cannot be booted at all or `No slide values are usable! Use custom slide!` message can be seen in the log.
- `CFG Lock` (MSR `0xE2` write protection) disabled in firmware settings if present. Refer to the ControlMsrE2 notes for details.
- `CSM` (Compatibility Support Module) disabled in firmware settings if present. On NVIDIA 6xx/AMD 2xx or older, GOP ROM may have to be flashed first. Use GopUpdate (see the second post) or AMD UEFI GOP MAKER in case of any potential confusion.
- `EHCI/XHCI Hand-off` enabled in firmware settings `only` if boot stalls unless USB devices are disconnected.
- `VT-x`, `Hyper Threading`, `Execute Disable Bit` enabled in firmware settings if present.
- While it may not be required, sometimes `Thunderbolt support`, `Intel SGX`, and `Intel Platform Trust` may have to be disabled in firmware settings present.

When debugging sleep issues, Power Nap and automatic power off (which appear to sometimes cause wake to black screen or boot loop issues on older platforms) may be temporarily disabled. The specific issues may vary, but ACPI tables should typically be looked at first.

Here is an example of a defect found on some Z68 motherboards. To turn Power Nap and the others off, run the following commands in Terminal:

```
sudo pmset autopoweroff 0
sudo pmset powernap 0
sudo pmset standby 0
```

Note: These settings may be reset by hardware changes and in certain other circumstances. To view their current state, use the `pmset -g` command in Terminal.

5.2 Properties

1. `MmioWhitelist`
   Type: `plist array`
   Failsafe: Empty
   Description: To be filled with `plist dict` values, describing addresses critical for particular firmware functioning when `DevirtualiseMmio` quirk is in use. Refer to the MmioWhitelist Properties section below for details.

2. `Patch`
   Type: `plist array`
   Failsafe: Empty
   Description: Perform binary patches in booter.

7 Kernel

7.1 Introduction

This section allows the application of different kinds of kernelspace modifications on Apple Kernel (XNU). The modifications currently provide driver (kext) injection, kernel and driver patching, and driver blocking.

Kernel and kext changes apply with the following effective order:

- `Block` is processed.
- `Emulate` and `Quirks` are processed.
- `Patch` is processed.
- `Add` and `Force` are processed.

7.2 Properties

1. `Add`
   Type: `plist array`
   Failsafe: Empty
   Description: Load selected kernel extensions (kexts) from the `OC/Kexts` directory.

   To be filled with `plist dict` values, describing each kext. Refer to the Add Properties section below for details.

   Note 1: The load order is based on the order in which the kexts appear in the array. Hence, dependencies must appear before kexts that depend on them.

   Note 2: To track the dependency order, inspect the `OSBundleLibraries` key in the `Info.plist` file of the kext being added. Any kext included under the key is a dependency that must appear before the kext being added.

   Note 3: Kexts may have inner kexts (`Plugins`) included in the bundle. Such `Plugins` must be added separately and follow the same global ordering rules as other kexts.

2. `Block`
   Type: `plist array`
   Failsafe: Empty
   Description: Remove selected kernel extensions (kexts) from the prelinked kernel.

   To be filled with `plist dictionary` values, describing each blocked kext. Refer to the Block Properties section below for details.

3. `Emulate`
   Type: `plist dict`
   Description: Emulate certain hardware in kernelspace via parameters described in the Emulate Properties section below.

4. `Force`
   Type: `plist array`
   Failsafe: Empty
   Description: Load kernel extensions (kexts) from the system volume if they are not cached.

   To be filled with `plist dict` values, describing each kext. Refer to the Force Properties section below for details. This section resolves the problem of injecting kexts that depend on other kexts, which are not otherwise cached. The issue typically affects older operating systems, where various dependency kexts, such as `IOAudioFamily` or `IONetworkingFamily` may not be present in the kernel cache by default.

   Note 1: The load order is based on the order in which the kexts appear in the array. Hence, dependencies must appear before kexts that depend on them.

   Note 2: `Force` happens before `Add`.

   Note 3: The signature of the "forced" kext is not checked in any way. This makes using this feature extremely dangerous and undesirable for secure boot.

   Note 4: This feature may not work on encrypted partitions in newer operating systems.

(f) Download and run Modified GRUB Shell compiled by brainsucker or use a newer version by datasone.
(g) Enter `setup_var 0x123 0x00` command, where `0x123` should be replaced by the actual offset, and reboot.

Warning: Variable offsets are unique not only to each motherboard but even to its firmware version. Never ever try to use an offset without checking.

On selected platforms, the `ControlMsrE2` tool can also change such hidden options. Pass desired argument: `lock`, `unlock` for `CFG Lock`. Or pass `interactive` to find and modify other hidden options.

As a last resort, consider patching the BIOS (for advanced users only).

2. `AppleXcpmCfgLock`
Type: `plist boolean`
Failsafe: `false`
Requirement: 10.8 (not required for older)
Description: Disables `PKG_CST_CONFIG_CONTROL` (`0xE2`) MSR modification in XNU kernel, commonly causing early kernel panic, when it is locked from writing (XCPM power management).

Note: This option should be avoided whenever possible. Refer to the `AppleCpuPmCfgLock` description for details.

3. `AppleXcpmExtraMsrs`
Type: `plist boolean`
Failsafe: `false`
Requirement: 10.8 (not required for older)
Description: Disables multiple MSR access critical for certain CPUs, which have no native XCPM support.

This is typically used in conjunction with the `Emulate` section on Haswell-E, Broadwell-E, Skylake-SP, and similar CPUs. More details on the XCPM patches are outlined in acidanthera/bugtracker#365.

Note: Additional not provided patches will be required for Ivy Bridge or Pentium CPUs. It is recommended to use `AppleIntelCpuPowerManagement.kext` for the former.

4. `AppleXcpmForceBoost`
Type: `plist boolean`
Failsafe: `false`
Requirement: 10.8 (not required for older)
Description: Forces maximum performance in XCPM mode.

This patch writes `0xFF00` to `MSR_IA32_PERF_CONTROL` (`0x199`), effectively setting maximum multiplier for all the time.

Note: While this may increase the performance, this patch is strongly discouraged on all systems but those explicitly dedicated to scientific or media calculations. Only certain Xeon models typically benefit from the patch.

5. `CustomPciSerialDevice`
Type: `plist boolean`
Failsafe: `false`
Requirement: 10.7
Description: Performs change of PMIO register base address on a customised PCI serial device.

The patch changes the PMIO register base address that the XNU kernel uses for serial input and output, from that of the default built-in COM1 serial port `0x3F8`, to the base address stored in the first IO BAR of a specified PCI device or to a specific base address (e.g. `0x2F8` for COM2).

Note: By default, serial logging is disabled. `serial=3` boot argument, which enables serial input and output, should be used for XNU to print logs to the serial port.

Note 2: In addition to this patch, kext `Apple16X50PCI0` should be prevented from attaching to have `kprintf` method working properly. This can be achieved by ~~setting (i. e. `Delete`, then `Add`) the `class-code` property of the PCI serial port device to `FFFFFFFF` in `DeviceProperties` section. As an alternative solution, a codeless kext `PCIeSerialDisable.kext` shown in the spoiler `PCIeSerialDisable.kext/Contents/Info.plist` at acidanthera/bugtracker may also be used.~~ using PCIeSerialDisable. In addition, for certain Thunderbolt cards the IOKit personality `IOPCITunnelCompatible` also needs to be set to `true`, which can be done by the `PCIeSerialThunderboltEnable.kext` attached at acidanthera/bugtracker#2003.

This option filters logging generated by specific modules, both in the log and onscreen. Two modes are supported:

- `+` — Positive filtering: Only present selected modules.
- `-` — Negative filtering: Exclude selected modules.

When multiple ones are selected, comma (`,`) should be used as the splitter. For instance, `+OCCPU,OCA,OCB` means *only* `OCCPU`, `OCA`, `OCB` being printed, while `-OCCPU,OCA,OCB` indicates these modules being filtered out (i.e. *not* logged). When no symbol is specified, positive filtering (`+`) will be used. `*` indicates all modules being logged.

Note 1: Acronyms of libraries can be found in the `Libraries` section below.

Note 2: Messages printed before the configuration of log protocol cannot be filtered.

7. `SysReport`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Produce system report on ESP folder.

   This option will create a `SysReport` directory in the ESP partition unless already present. The directory will contain ACPI, SMBIOS, and audio codec dumps. Audio codec dumps require an audio backend driver to be loaded.

   Note: To maintain system integrity, the `SysReport` option is **not** available in `RELEASE` builds. Use a `DEBUG` build if this option is required.

8. `Target`
   Type: `plist integer`
   Failsafe: `0`
   Description: A bitmask (sum) of enabled logging targets. Logging output is hidden by default and this option must be set when such output is required, such as when debugging.

   The following logging targets are supported:

   - `0x01` (bit `0`) — Enable logging, otherwise all log is discarded.
   - `0x02` (bit `1`) — Enable basic console (onscreen) logging.
   - `0x04` (bit `2`) — Enable logging to Data Hub.
   - `0x08` (bit `3`) — Enable serial port logging.
   - `0x10` (bit `4`) — Enable UEFI variable logging.
   - `0x20` (bit `5`) — Enable `non-volatile` UEFI variable logging.
   - `0x40` (bit `6`) — Enable logging to file.
   - `0x80` (bit `7`) — In combination with `0x40`, enable faster but unsafe (see Warning 2 below) file logging.

   Console logging prints less than the other variants. Depending on the build type (`RELEASE`, `DEBUG`, or `NOOPT`) different amount of logging may be read (from least to most).

   To obtain Data Hub logs, use the following command in macOS (Note that Data Hub logs do not log kernel and kext patches):

   ```
   ioreg -lw0 -p IODeviceTree | grep boot-log | sort | sed 's/.*<\(.*\)>.*/\1/' | xxd -r -p
   ```

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

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

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

   Warning 1: Certain firmware appear to have defective NVRAM garbage collection. As a result, they may not be able to always free space after variable deletion. Do not enable `non-volatile` NVRAM logging on such devices unless specifically required.

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

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

Warning 2: It is possible to enable fast file logging, which requires a fully compliant firmware FAT32 driver. On drivers with incorrect FAT32 write support (e.g. APTIO IV, but maybe others) this setting can result in corruption up to and including an unusable ESP filesystem, therefore be prepared to recreate the ESP partition and all of its contents if testing this option. This option can increase logging speed significantly on some suitable firmware, but may make little speed difference on some others.

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

The list of currently used tags is as follows.

Drivers and tools:

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

Libraries:

- `AAPL` — OcLogAggregatorLib, Apple EfiBoot logging
- `OCABC` — OcAfterBootCompatLib
- `OCAE` — OcAppleEventLib
- `OCAK` — OcAppleKernelLib
- `OCAU` — OcAudioLib
- `OCA` —- OcAcpiLib
- `OCBP` — OcAppleBootPolicyLib
- `OCB` — OcBootManagementLib
- `OCLBT` — OcBlitLib
- `OCCL` — OcAppleChunkListLib
- `OCCPU` — OcCpuLib
- `OCC` — OcConsoleLib
- `OCDC` — OcDriverConnectionLib
- `OCDH` — OcDataHubLib
- `OCDI` — OcAppleDiskImageLib
- `OCDM` — OcDeviceMiscLib
- `OCFS` — OcFileLib
- `OCFV` — OcFirmwareVolumeLib
- `OCHS` — OcHashServicesLib
- `OCI4` — OcAppleImg4Lib
- `OCIC` — OcImageConversionLib
- `OCII` — OcInputLib
- `OCJS` — OcApfsLib

Note 1: Due to using system NVRAM reset, this option is not compatible with the `--preserve-boot` option and will override it, therefore all BIOS boot entries will be removed.

Note 2: Due to using system NVRAM reset, the OpenCore boot option cannot be preserved and OpenCore will have to either be reselected in the native boot picker or re-blessed.

Note 3: On non-Apple hardware, this option will still set this variable but the variable will not be recognised by the firmware and no NVRAM reset will happen.

11.7.2 ToggleSipEntry

Provides a boot entry for enabling and disabling System Integrity Protection (SIP) in OpenCore picker.

While macOS is running, SIP involves multiple configured software protection systems, however all the information about which of these protections to enable is stored in the single Apple NVRAM variable `csr-active-config`. As long as this variable is set before macOS startup, SIP will be fully configured, so setting the variable using this boot option (or in any other way, before macOS starts) has exactly the same end result as configuring SIP using the `csrutil` command in macOS Recovery.

`csr-active-config` will be toggled between `0` for enabled, and a user-specified or default value for disabled. ~~The default value is `0x27F` (see below). Any other required value can~~

Options for the driver should be specified as ~~a single number~~ plain text values separated by whitespace in the `Arguments` ~~for this driver. This~~ section of `Driver` entry. Available options are:

- `--show-csr` - Boolean flag, enabled if present.

  If enabled, show the current hexadecimal value of `csr-active-config` in the boot entry name. This option will not work in OpenCanopy when used in combination with `OC_ATTR_USE_GENERIC_LABEL_IMAGE` in `PickerAttributes`.

- Numerical value - Default value `0x27F`.

  Specify the `csr-active-config` value to use to disabled SIP. This can be specified as hexadecimal, beginning with `0x`, or as decimal. For more info see Note 2 below.

Note 1: It is recommended not to run 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: The default value for disabling SIP with this boot entry is `0x27F`. For comparison, `csrutil disable` with no other arguments on macOS Big Sur and Monterey sets `0x7F`, and on Catalina it sets `0x77`. The OpenCore default value of `0x27F` is a variant of the Big Sur and Monterey value, chosen as follows:

- `CSR_ALLOW_UNAPPROVED_KEXTS` (`0x200`) is included in the default value, since it is generally useful, in the case where you need to have SIP disabled anyway, to be able to install unsigned kexts without manual approval in System Preferences.
- `CSR_ALLOW_UNAUTHENTICATED_ROOT` (`0x800`) is not included in the default value, as it is very easy when using it to inadvertently break OS seal and prevent incremental OTA updates.
- If unsupported bits from a later OS are specified in `csr-active-config` (e.g. specifying `0x7F` on Catalina) then `csrutil status` will report that SIP has a non-standard value, however protection will be functionally the same.

11.8 AudioDxe

High Definition Audio (HDA) support driver in UEFI firmware for most Intel and some other analog audio controllers.

Note: AudioDxe is a staging driver, refer to acidanthera/bugtracker#740 for known issues.

11.8.1 Configuration

Most UEFI audio configuration is handled via the `UEFI Audio Properties` section, but ~~if required the following additonal configuration options (which are needed to produce sound onmost Apple hardware, and possibly some others)may be specified in UEFI/Drivers/Arguments:~~ in addition some of the following configuration options may be required in order to allow AudioDxe to correctly drive certain devices. All options are specified as text strings, separated

by space if more than one option is required, in the `Arguments` property for the driver within the `UEFI/Drivers` section:

- `--codec-setup-delay` - Integer value, default `0`.

  Amount of time in milliseconds to wait for all widgets to come fully on, applied per codec during driver connection phase. In most systems this should not be needed and a faster boot will be achieved by using `Audio` section `SetupDelay` if any audio setup delay is required. Where required, values of up to one second may be needed.

- `--force-device` - String value, no default.

  When this option is present and has a value (e.g. `--force-device=PciRoot(0x0)/Pci(0x1f,0x3)`), it forces AudioDxe to connect to the specified PCI device, even if the device does not report itself as an HDA audio controller.
  During driver connection, AudioDxe automatically provides audio services on all supported codecs of all available HDA controllers. However, if the relevant controller is misreporting its identity (typically, it will be reporting itself as a legacy audio device instead of an HDA controller) then this argument may be required.
  Applies if the audio device can be made to work in macOS, but shows no sign of being detected by AudioDxe (e.g. when including `DEBUG_INFO` in `DisplayLevel` and using a DEBUG build of AudioDxe, no controller and codec layout information is displayed during the `Connecting drivers...` phase of OpenCore log).

- `--gpio-setup` - Default value is `0` (GPIO setup disabled) if argument is not provided, or `7` (all GPIO setup stages stages enabled) if the argument is provided with no value.

  Available values, which may be combined by adding, are:

  - `0x00000001` (bit `0`) — `GPIO_SETUP_STAGE_DATA`, set GPIO pin data high on specified pins. Required e.g. on `MacBookPro10,2` and `MacPro5,1`.
  - `0x00000002` (bit `1`) — `GPIO_SETUP_STAGE_DIRECTION`, set GPIO data direction to output on specified pins. Required e.g. on `MacPro5,1`.
  - `0x00000004` (bit `2`) — `GPIO_SETUP_STAGE_ENABLE`, enable specified GPIO pins. Required e.g. on `MacPro5,1`.

  If audio appears to be 'playing' on the correct codec, e.g. based on the debug log, but no sound is heard on any channel, it is suggested to use `--gpio-setup` (with no value) in the AudioDxe driver arguments. If specified with no value, all stages will be enabled (equivalent of specifying `7`). If this produces sound, it is then possible to try fewer bits, e.g. `--gpio-setup=1`, `--gpio-setup=3`, to find out which stages are actually required.

  Note: Value `7` (all flags enabled) of this option – as required for the `MacPro5,1` – is compatible with most systems, but is known to cause problems with sound (previous sounds are not allowed to finish before new sounds start) on a small number of other systems, hence this option is not enabled by default.

- `--gpio-pins` - Default: `0`, auto-detect.

  Specifies which GPIO pins should be operated on by `--gpio-setup`. This is a bit mask, with possible values from `0x0` to `0xFF`. The usable maximum depends on the number if available pins on the audio out function group of the codec in use, e.g. it is `0x3` (lowest two bits) if two GPIO pins are present, `0x7` if three pins are present, etc.

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

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

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

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

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

11.9 Properties

1. `APFS`
   Type: `plist dict`
   Failsafe: None
   Description: Provide APFS support as configured in the APFS Properties section below.

2. `AppleInput`
   Type: `plist dict`
   Failsafe: None
   Description: Configure the re-implementation of the Apple Event protocol described in the AppleInput Properties section below.

3. `Audio`
   Type: `plist dict`
   Failsafe: None
   Description: Configure audio backend support described in the `Audio Properties` section below.

   Unless documented otherwise (e.g. `ResetTrafficClass`) settings in this section are for UEFI audio support only (e.g. OpenCore generated boot chime and audio assist) and are unrelated to any configuration needed for OS audio support (e.g. `AppleALC`).

   UEFI audio support provides a way for upstream protocols to interact with the selected audio hardware and resources. All audio resources should reside in `\EFI\OC\Resources\Audio` directory. Currently the supported audio file formats are MP3 and WAVE PCM. While it is driver-dependent which audio stream format is supported, most common audio cards support 16-bit signed stereo audio at 44100 or 48000 Hz.

   Audio file path is determined by audio type, audio localisation, and audio path. Each filename looks as follows: `[audio type]_[audio localisation]_[audio path].[audio ext]`. For unlocalised files filename does not include the language code and looks as follows: `[audio type]_[audio path].[audio ext]`. Audio extension can either be `mp3` or `wav`.

   - Audio type can be `OCEFIAudio` for OpenCore audio files or `AXEFIAudio` for macOS bootloader audio files.
   - Audio localisation is a two letter language code (e.g. `en`) with an exception for Chinese, Spanish, and Portuguese. Refer to `APPLE_VOICE_OVER_LANGUAGE_CODE` definition for the list of all supported localisations.
   - Audio path is the base filename corresponding to a file identifier. For macOS bootloader audio paths refer to `APPLE_VOICE_OVER_AUDIO_FILE` definition. For OpenCore audio paths refer to `OC_VOICE_OVER_AUDIO_FILE` definition. The only exception is OpenCore boot chime file, which is `OCEFIAudio_VoiceOver_Boot.mp3`.

   Audio localisation is determined separately for macOS bootloader and OpenCore. For macOS bootloader it is set in `preferences.efires` archive in `systemLanguage.utf8` file and is controlled by the operating system. For OpenCore the value of `prev-lang:kbd` variable is used. When native audio localisation of a particular file is missing, English language (`en`) localisation is used. Sample audio files can be found in OcBinaryData repository.

4. `ConnectDrivers`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Perform UEFI controller connection after driver loading.

   This option is useful for loading drivers following UEFI driver model as they may not start by themselves. Examples of such drivers are filesystem or audio drivers. While effective, this option may not be necessary for drivers performing automatic connection, and may slightly slowdown the boot.

   Note: Some types of firmware, particularly those made by Apple, only connect the boot drive to speed up the boot process. Enable this option to be able to see all the boot options when running multiple drives.

5. `Drivers`
   Type: `plist array`
   Failsafe: Empty
   Description: Load selected drivers from `OC/Drivers` directory.

   To be filled with `plist dict` values, describing each driver. Refer to the Drivers Properties section below.

10. `ResetTrafficClass`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Set HDA Traffic Class Select Register to `TC0`.

    AppleHDA kext will function correctly only if `TCSEL` register is configured to use `TC0` traffic class. Refer to Intel I/O Controller Hub 9 (ICH9) Family Datasheet (or any other ICH datasheet) for more details about this register.

    Note: This option is independent from `AudioSupport`. If AppleALC is used it is preferred to use AppleALC ~~alctsel~~alctcsel property instead.

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

    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.

11.13 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 OpenCore plugins accept optional additional arguments which may be specified as a string here.

11.14 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