OpenCore

Reference Manual (0.7.~~6~~.7)

[2022.01.10]

cases, the use of open-source implementations with transparent binary generation (such as OCAT) is encouraged, given that other tools may contain malware. In addition, configurations created for a specific hardware setup should never be used on different hardware setups.

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.

When updating the LaTeX documentation (e.g. `Configuration.tex`) please do *not* rebuild the PDF files till merging to master happens. This avoids unnecessary merge conflicts:

- External contributors using the pull-request approach should request the maintainers to handle the PDF rebuild in the pull-request message.
- Internal contributors should rebuild the documentation at merge time in the same or in a separate commit. One can ask another maintainer to rebuild the documentation when lacking the necessary tools in the pull-request message.

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
. ./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 using Language Server Protocols. For example, Sublime Text with LSP for Sublime Text plugin. Add `compile_flags.txt` 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
```

- Enabling XCPM support for an unsupported CPU variant.

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

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

Note 3: For XCPM support it is recommended to use the following combinations. Be warned that one is required to set the correct frequency vectors matching the installed CPU.

- Haswell-E (`0x0306F2`) to Haswell (`0x0306C3`):
  `Cpuid1Data`: `C3 06 03 00 00 00 00 00 00 00 00 00 00 00 00 00`
  `Cpuid1Mask`: `FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
- Broadwell-E (`0x0406F1`) to Broadwell (`0x0306D4`):
  `Cpuid1Data`: `D4 06 03 00 00 00 00 00 00 00 00 00 00 00 00 00`
  `Cpuid1Mask`: `FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
- Comet Lake U62 (`0x0A0660`) to Comet Lake U42 (`0x0806EC`):
  `Cpuid1Data`: `EC 06 08 00 00 00 00 00 00 00 00 00 00 00 00 00`
  `Cpuid1Mask`: `FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
- Rocket Lake (`0x0A0670`) to Comet Lake (~~`0x0906EB`~~`0x0A0655`):
  `Cpuid1Data`: ~~`EB`~~ `55 06` ~~`09`~~ `0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00`
  `Cpuid1Mask`: `FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
- ~~Comet Lake U62 (~~Alder Lake (~~`0x0A0660`~~`0x090672`) to Comet Lake ~~U42 (~~(~~`0x0806EC`~~`0x0A0655`):
  `Cpuid1Data`: ~~`EC`~~ `55 06` ~~`08`~~ `0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00`
  `Cpuid1Mask`: `FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`

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

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

2. `Cpuid1Mask`
   Type: `plist data`, 16 bytes
   Failsafe: All zero
   Description: Bit mask of active bits in `Cpuid1Data`.

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

3. `DummyPowerManagement`
   Type: `plist boolean`
   Failsafe: `false`
   Requirement: 10.4
   Description: Disables `AppleIntelCpuPowerManagement`.

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

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

4. `MaxKernel`
   Type: `plist string`
   Failsafe: Empty
   Description: Emulates CPUID and applies `DummyPowerManagement` on specified macOS version or older.

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

sometimes fails to wake up. For debug kernels `setpowerstate_panic=0` boot argument should be used, which is otherwise equivalent to this quirk.

17. `ProvideCurrentCpuInfo`
    Type: `plist boolean`
    Failsafe: `false`
    Requirement: 10.8 (10.14)
    Description: Provides current CPU info to the kernel.

    This quirk ~~currently~~ works differently depending on the CPU:

    - For Microsoft Hyper-V it provides the correct TSC and FSB values to the kernel, as well as disables CPU topology validation ~~.~~ (10.8+).
    - For Intel CPUs it adds support for asymmetrical SMP systems (e.g. Intel Alder Lake) by patching core count to thread count along with the supplemental required changes (10.14+).

    ~~*Note*: These patches currently target Microsoft Hyper-V and may need to be extended for other purposes.~~

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

    The APFS filesystem is designed in a way that the space controlled via the spaceman structure is either used or free. This may be different in other filesystems where the areas can be marked as used, free, and *unmapped*. All free space is trimmed (unmapped/deallocated) at macOS startup. The trimming procedure for NVMe drives happens in LBA ranges due to the nature of the `DSM` command with up to 256 ranges per command. The more fragmented the memory on the drive is, the more commands are necessary to trim all the free space.

    Depending on the SSD controller and the level of drive fragmenation, the trim procedure may take a considerable amount of time, causing noticeable boot slowdown. The APFS driver explicitly ignores previously unmapped areas and repeatedly trims them on boot. To mitigate against such boot slowdowns, the macOS driver introduced a timeout (`9.999999` seconds) that stops the trim operation when not finished in time.

    On several controllers, such as Samsung, where the deallocation process is relatively slow, this timeout can be reached very quickly. Essentially, it means that the level of fragmentation is high, thus macOS will attempt to trim the same lower blocks that have previously been deallocated, but never have enough time to deallocate higher blocks. The outcome is that trimming on such SSDs will be non-functional soon after installation, resulting in additional wear on the flash.

    One way to workaround the problem is to increase the timeout to an extremely high value, which at the cost of slow boot times (extra minutes) will ensure that all the blocks are trimmed. Set this option to a high value, such as `4294967295`, to ensure that all blocks are trimmed. Alternatively, use over-provisioning, if supported, or create a dedicated unmapped partition where the reserve blocks can be found by the controller. Conversely, the trim operation can be disabled by setting a very low timeout value. e.g. `999`. Refer to this article for details.

19. `ThirdPartyDrives`
    Type: `plist boolean`
    Failsafe: `false`
    Requirement: 10.6 (not required for older)
    Description: Apply vendor patches to IOAHCIBlockStorage.kext to enable native features for third-party drives, such as TRIM on SSDs or hibernation support on 10.15 and newer.

    Note: This option may be avoided on user preference. NVMe SSDs are compatible without the change. For AHCI SSDs on modern macOS version there is a dedicated built-in utility called `trimforce`. Starting from 10.15 this utility creates `EnableTRIM` variable in `APPLE_BOOT_VARIABLE_GUID` namespace with `01 00 00 00` value.

20. `XhciPortLimit`
    Type: `plist boolean`
    Failsafe: `false`
    Requirement: 10.11 (not required for older)

    * `1` — enables DeviceTree logging.
  - `kc-read-size=VALUE` — Chunk size used for buffered I/O from network or disk for prelinkedkernel reading and related. Set to 1MB (0x100000) by default, can be tuned for faster booting.
  - `log-level=VALUE` — log level bitmask.
    * `0x01` — enables trace logging (default).
  - `serial=VALUE` — enables serial logging.
    * `0` — disables serial logging (default).
    * `1` — enables serial logging for `EXITBS:END` onwards.
    * `2` — enables serial logging for `EXITBS:START` onwards.
    * `3` — enables serial logging when debug protocol is missing.
    * `4` — enables serial logging unconditionally.
  - `timestamps=VALUE` — enables timestamp logging.
    * `0` — disables timestamp logging.
    * `1` — enables timestamp logging (default).
  - `log=VALUE` — deprecated starting from 10.15.
    * `1` — AppleLoggingConOutOrErrSet/AppleLoggingConOutOrErrPrint (classical ConOut/StdErr)
    * `2` — AppleLoggingStdErrSet/AppleLoggingStdErrPrint (StdErr or serial?)
    * `4` — AppleLoggingFileSet/AppleLoggingFilePrint (BOOTER.LOG/BOOTER.OLD file on EFI partition)
  - `debug=VALUE` — deprecated starting from 10.15.
    * `1` — enables print something to BOOTER.LOG (stripped code implies there may be a crash)
    * `2` — enables perf logging to /efi/debug-log in the device three
    * `4` — enables timestamp printing for styled printf calls
  - `level=VALUE` — deprecated starting from 10.15. Verbosity level of DEBUG output. Everything but `0x80000000` is stripped from the binary, and this is the default value.

  Note: Enable the `AppleDebug` option to display verbose output from `boot.efi` on modern macOS versions. This will save the log to the general OpenCore log file. For versions before 10.15.4, set `bootercfg` to `log=1`. This will print verbose output onscreen.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:bootercfg-once`
  Booter arguments override removed after first launch. Otherwise equivalent to `bootercfg`.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:csr-data`
  Specify sources of kexts which will be approved regardless of SIP `CSR_ALLOW_UNAPPROVED_KEXTS` value.
  Example contents:
  `<dict><key>kext-allowed-teams</key><array><string>{DEVELOPER-TEAM-ID}</string></array></dict>%00`
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:efiboot-perf-record`
  Enable performance log saving in `boot.efi`. Performance log is saved to physical memory and is pointed to by the `efiboot-perf-record-data` and `efiboot-perf-record-size` variables. Starting from 10.15.4, it can also be saved to the OpenCore log by setting the `AppleDebug` option.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:fmm-computer-name`
  Current saved host name. ASCII string.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:nvda_drv`
  NVIDIA Web Driver control variable. Takes ASCII digit `1` or `0` to enable or disable installed driver.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:run-efi-updater`
  Override EFI firmware updating support in macOS (MultiUpdater, ThorUtil, and so on). Setting this to `No` or alternative boolean-castable value will prevent any firmware updates in macOS starting with 10.10 at least.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:StartupMute`
  Mute startup chime sound in firmware audio support. 8-bit integer. The value of `0x00` means unmuted. Missing variable or any other value means muted.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:SystemAudioVolume`
  System audio volume level for firmware audio support. 8-bit unsigned integer. The bit of `0x80` means muted. ~~Lower~~ The remaining bits are used to encode ~~volume range specific to installed audio codec. The~~ the raw amplifier gain setting to be applied to the audio amplifier in use. Exactly what this value means depends on the codec (and potentially on the specific amplifier within the codec). This value is capped by macOS to the `MaximumBootBeepVolume` AppleHDA layout value~~to avoid too loud~~, to avoid over-loud audio playback in the firmware.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:SystemAudioVolumeDB`
  Current system audio volume level in decibels (dB). 8-bit signed integer. The value represents the audio offset (gain if positive, attenuation if negative) in dB relative to the amplifier reference value of 0 dB. Exactly which

volume level is represented by 0 dB depends on the codec (and potentially on the specific amplifier within the codec) but it is normally at or near the maximum available amplifier volume. Typical values of this variable range from approximately -60 (the exact value depends on the audio hardware) up to exactly 0. On non-typical audio hardware, the value may go above zero.
Note: Unlike `SystemAudioVolume`, this value is not capped.

- `5EDDA193-A070-416A-85EB-2A1181F45B18:PEXConf`
  PCI expansion slot configuration for `MacPro7,1`. 8-byte sequence describing default PCI slot configuration. Each byte refers to a configuration for a dedicated PCI slot.
  - Slot 1 resides at `IOService:/AppleACPIPlatformExpert/PC01@0/AppleACPIPCI/BR1A@0` and its path is hardcoded. This slot is not behind a muxer.
  - Slot 3 resides at `IOService:/AppleACPIPlatformExpert/PC03@0/AppleACPIPCI/BR3A@0` and its path is hardcoded. This slot is not behind a muxer.
  - Slots 2, 4-8 are dynamic and are matched based on `AAPL,slot-name` property with `Slot-N` value, where `N` is the slot number. All these slots are behind the muxer.

  Refer to the support page for more details on how `MacPro7,1` slots are configured.
- `5EDDA193-A070-416A-85EB-2A1181F45B18:SlotUtilPEXConf`
  User PCI expansion slot configuration for `MacPro7,1`. 8-byte sequence describing user PCI slot configuration.

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~~ 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.

systemd-boot users (probably almost exclusively Arch Linux users) should be aware that OpenLinuxBoot does not support the systemd-boot–specific Boot Loader Interface; therefore `efibootmgr` rather than `bootctl` must be used for any low-level Linux command line interaction with the boot menu.

11.7 AudioDxe

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

11.7.1 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 on most Apple hardware, and possibly some others) may be specified in `UEFI/Drivers/Arguments`:

- `--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`.

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

11.8 Properties

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

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

~~Audio support~~ 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 ~~hardware and audio~~ 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.

3. `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.

4. `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.

5. `Input`
   Type: `plist dict`
   Failsafe: None
   Description: Apply individual settings designed for input (keyboard and mouse) in the Input Properties section below.

6. `Output`
   Type: `plist dict`
   Failsafe: None
   Description: Apply individual settings designed for output (text and graphics) in the Output Properties section below.

7. `ProtocolOverrides`
   Type: `plist dict`
   Failsafe: None
   Description: Force builtin versions of certain protocols described in the ProtocolOverrides Properties section below.

   Note: all protocol instances are installed prior to driver loading.

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. `PointerPollMin`
   Type: `plist integer`
   Failsafe: `0`
   Description: Configure minimal pointer polling period in ms.

   This is the minimal period the OpenCore builtin AppleEvent driver polls pointer devices (e.g. mice, trackpads) for motion events. The current implementation defaults to 10 ms. Setting `0` leaves this default unchanged.

   Note: The OEM Apple implementation uses a polling rate of 2 ms.

7. `PointerPollMax`
   Type: `plist integer`
   Failsafe: `0`
   Description: Configure maximum pointer polling period in ms.

   This is the maximum period the OpenCore builtin AppleEvent driver polls pointer devices (e.g. mice, trackpads) for motion events. The period is increased up to this value as long as the devices do not respond in time. The current implementation defaults to 80 ms. Setting `0` leaves this default unchanged.

   Certain trackpad drivers often found in Dell laptops can be very slow to respond when no physical movement happens. This can affect OpenCanopy and FileVault 2 user interface responsiveness and loading times. Increasing the polling periods can reduce the impact.

   Note: The OEM Apple implementation uses a polling rate of 2 ms.

8. `PointerPollMask`
   Type: `plist integer, 32 bit`
   Failsafe: `-1`
   Description: Configure indices of polled pointers.

   Selects pointer devices to poll for AppleEvent motion events. `-1` implies all devices. A bit sum is used to determine particular devices. E.g. to enable devices 0, 2, 3 the value will be `1+4+8` (corresponding powers of two). A total of 32 configurable devices is supported.

   Certain pointer devices can be present in the firmware even when no corresponding physical devices are available. These devices usually are placeholders, aggregate devices, or proxies. Gathering information from these devices may result in inaccurate motion activity in the user interfaces and even cause performance issues. Disabling such pointer devices is recommended for laptop setups having issues of this kind.

   The amount of pointer devices available in the system can be found in the log. Refer to `Found N pointer devices` message for more details.

   Note: Has no effect when using the OEM Apple implementation (see `AppleEvent` setting).

9. `PointerSpeedDiv`
   Type: `plist integer`
   Failsafe: `1`
   Description: Configure pointer speed divisor in 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.

10. `PointerSpeedMul`
    Type: `plist integer`
    Failsafe: `1`

Description: Configure pointer speed multiplier in 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.

11.11 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,~~ If using `AudioDxe`, the available controller device paths are also output on lines formatted like this:

   ```
   HDA: Connecting controller - PciRoot(0x0)/Pci(0x1B,0x0)
   ```

   Finally, `gfxutil -f HDEF` command can be used in macOS ~~.~~ to obtain the device path.

   Specifying an empty device path ~~will result~~ results in the first available codec and audio controller being used. The value of `AudioCodec` is ignored in this case. This can be a convenient initial option to try to get UEFI audio working. Manual settings as above will be required when this default value does not work.

3. ~~`AudioOut`~~`AudioOutMask`
   Type: `plist integer`
   Failsafe: ~~`0`~~`-1`
   Description: ~~Index of the output port of the specified codec starting from 0.~~ Bit field indicating which output channels to use for UEFI sound.

   ~~This typically contains the index of the green out of the builtin analog audio controller (~~Audio mask is 1 « audio output (equivalently 2 ~~`HDEF`~~)~~.~~ ^ audio output). E.g. for audio output `0` the bitmask is `1`, for output `3` it is `8`, and for outputs `0` and `3` it is `9`.

   The number of available output nodes (`N`) for each HDA codec is shown in the debug log (marked in bold-italic), audio outputs `0` to `N - 1` may be selected:

   ```
   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)
   ```

~~The quickest way to find the right port is to bruteforce the values from~~ When `AudioDxe` is used then additional information about each output channel is logged during driver binding, including the bitmask for each output. The bitmask values for the desired outputs should be added together to obtain the `AudioOutMask` value:

```
HDA: | Port widget @ 0x9 is an output (pin defaults 0x2B4020) (bitmask 1)
HDA: | Port widget @ 0xA is an output (pin defaults 0x90100112) (bitmask 2)
HDA: | Port widget @ 0xB is an output (pin defaults 0x90100110) (bitmask 4)
HDA: | Port widget @ 0x10 is an output (pin defaults 0x4BE030) (bitmask 8)
```

Further information on the available output channels may be found from a Linux codec dump using the command:

```
cat /proc/asound/card{n}/codec#{m}
```

Using `AudioOutMask`, it is possible to play sound to more than one channel (e.g. main speaker plus bass speaker; headphones plus speakers) as long as all the chosen outputs support the sound file format in use; if any do not then no sound will play and a warning will be logged.

When all available output channels on the codec support the available sound file format then a value of ~~0 to~~ `-1` will play sound to all channels simultaneously. If this does not work it will usually be quickest to try each available output channel one by one, by setting `AudioOutMask` to 1, 2, 4, etc., up to 2 ^ `N - 1`, in order to work out which channel(s) produce sound.

4. `AudioSupport`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Activate audio support by connecting to a backend driver.

   Enabling this setting routes audio playback from builtin protocols to ~~a dedicated audio port (~~specified (~~`AudioOut`~~`AudioOutMask` dedicated audio ports of the specified codec (`AudioCodec`), located on the specified audio controller (`AudioDevice`).

5. ~~`MinimumVolume`~~`DisconnectHda`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Disconnect HDA controller before loading drivers.

   May be required on some systems (e.g. Apple hardware, VMware Fusion guest) to allow a UEFI sound driver (such as `AudioDxe`) to take control of the audio hardware.

   Note: In addition to this option, most Apple hardware also requires the `--gpio-setup` driver argument which is dealt with in the AudioDxe section.

6. `MaximumGain`
   Type: `plist integer`
   Failsafe: ~~`0`~~`-15`
   Description: ~~Minimal heard volume level from~~ Maximum gain to use for UEFI audio, specified in decibels (dB) with respect to amplifier reference level of 0 ~~to~~ dB (see note 1).

   All UEFI audio will use this gain setting when the system amplifier gain read from the ~~100~~`SystemAudioVolumeDB` NVRAM variable is higher than this. This is to avoid over-loud UEFI audio when the system volume is set very high, or the `SystemAudioVolumeDB` NVRAM value has been misconfigured.

   Note 1: Decibels (dB) specify gain (postive values; increase in volume) or attenuation (negative values; decrease in volume) compared to some reference level. When you hear the sound level of a jet plane expressed as 120 decibels, say, the reference level is the sound level just audible to an average human. However generally in acoustic science and computer audio any reference level can be specified. Intel HDA and macOS natively use decibels to specify volume level. On most Intel HDA hardware the reference level of 0 dB is the *loudest* volume of the hardware, and all lower volumes are therefore negative numbers. The quietest volume on typical sound hardware is around -55 dB to -60 dB.

   Note 2: Matching how macOS handles decibel values, this value is converted to a signed byte; therefore values outside −128 dB to +127 dB (which are well beyond physically plausible volume levels) are not allowed.

   Note 3: Digital audio output – which does not have a volume slider in-OS – ignores this and all other gain settings, only mute settings are relevant.

7. `MinimumAssistGain`
   Type: `plist integer`
   Failsafe: `-30`
   Description: Minimum gain in decibels (dB) to use for picker audio assist.

   The screen reader will use this ~~volume level when the calculated volume level~~ amplifier gain if the system amplifier gain read from the `SystemAudioVolumeDB` NVRAM variable is lower than ~~`MinimumVolume` and the~~ this.

   Note 1: In addition to this setting, because audio assist must be audible to serve its function, audio assist is not muted even if the OS sound is muted or the `StartupMute` NVRAM variable is set.

   Note 2: See `MaximumGain` for an explanation of decibel volume levels.

8. `MinimumAudibleGain`
   Type: `plist integer`
   Failsafe: `-128`
   Description: Minimum gain in decibels (dB) at which to attempt to play any sound.

   The boot chime will not play if the ~~calculated volume level~~ system amplifier gain level in the `SystemAudioVolumeDB` NVRAM variable is lower than ~~`MinimumVolume`~~this.

   Note 1: This setting is designed to save unecessary pauses due to audio setup at inaudible volume levels, when no sound will be heard anyway. Whether there are inaudible volume levels depends on the hardware. On some hardware (including Apple) the audio values are well enough matched to the hardware that the lowest volume levels available are very quiet but audible, whereas on some other hardware combinations, the lowest part of the volume range may not be audible at all.

   Note 2: See `MaximumGain` for an explanation of decibel volume levels.

9. `PlayChime`
   Type: `plist string`
   Failsafe: `Auto`
   Description: Play chime sound at startup.

   Enabling this setting plays the boot chime using the builtin audio support. The volume level is determined by the ~~`MinimumVolume`~~`SystemAudioVolumeDB` ~~and `VolumeAmplifier` settings as well as the `SystemAudioVolume`~~ NVRAM variable. ~~Possible values include~~Supported values are:

   - `Auto` — Enables chime when `StartupMute` NVRAM variable is not present or set to `00`.
   - `Enabled` — Enables chime unconditionally.
   - `Disabled` — Disables chime unconditionally.

   Note 1: `Enabled` can be used ~~in separate from~~ separately from the `StartupMute` NVRAM variable to avoid conflicts when the firmware is able to play the boot chime.

   Note 2: Regardless of this setting, the boot chime will not play if system audio is muted, i.e. if the `SystemAudioVolume` NVRAM variable has bit `0x80` set.

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` property instead.

11. `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.

12. ~~`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.~~

11.12 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.

11.13 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

This option implements standard UEFI pointer protocol (`EFI_SIMPLE_POINTER_PROTOCOL`) through certain OEM protocols. The option may be useful on Z87 ASUS boards, where `EFI_SIMPLE_POINTER_PROTOCOL` is defective.

7. `PointerSupportMode`
   Type: `plist string`
   Failsafe: Empty
   Description: Set OEM protocol used for internal pointer driver.

   Currently the only supported variant is `ASUS`, using specialised protocol available on certain Z87 and Z97 ASUS boards. More details can be found in `LongSoft/UefiTool#116`. The value of this property cannot be empty if `PointerSupport` is enabled.

8. `TimerResolution`
   Type: `plist integer`
   Failsafe: `0`
   Description: Set architecture timer resolution.

   This option allows updating the firmware architecture timer period with the specified value in `100` nanosecond units. Setting a lower value typically improves performance and responsiveness of the interface and input handling.

   The recommended value is `50000` (`5` milliseconds) or slightly higher. Select ASUS Z87 boards use `60000` for the interface. Apple boards use `100000`. In case of issues, this option can be left as `0` to not change the timer resolution.

11.14 Output Properties

1. `TextRenderer`
   Type: `plist string`
   Failsafe: `BuiltinGraphics`
   Description: Chooses renderer for text going through standard console output.

   Currently two renderers are supported: `Builtin` and `System`. `System` renderer uses firmware services for text rendering. `Builtin` bypassing firmware services and performs text rendering on its own. Different renderers support a different set of options. It is recommended to use `Builtin` renderer, as it supports HiDPI mode and uses full screen resolution.

   UEFI firmware typically supports `ConsoleControl` with two rendering modes: `Graphics` and `Text`. Some types of firmware do not support `ConsoleControl` and rendering modes. OpenCore and macOS expect text to only be shown in `Graphics` mode and graphics to be drawn in any mode. Since this is not required by UEFI specification, exact behaviour varies.

   Valid values are combinations of text renderer and rendering mode:

   - `BuiltinGraphics` — Switch to `Graphics` mode and use `Builtin` renderer with custom `ConsoleControl`.
   - `BuiltinText` — Switch to `Text` mode and use `Builtin` renderer with custom `ConsoleControl`.
   - `SystemGraphics` — Switch to `Graphics` mode and use `System` renderer with custom `ConsoleControl`.
   - `SystemText` — Switch to `Text` mode and use `System` renderer with custom `ConsoleControl`.
   - `SystemGeneric` — Use `System` renderer with system `ConsoleControl` assuming it behaves correctly.

   The use of `BuiltinGraphics` is straightforward. For most platforms, it is necessary to enable `ProvideConsoleGop` and set `Resolution` to `Max`. The `BuiltinText` variant is an alternative `BuiltinGraphics` for some very old and defective laptop firmware, which can only draw in `Text` mode.

   The use of `System` protocols is more complicated. Typically, the preferred setting is `SystemGraphics` or `SystemText`. Enabling `ProvideConsoleGop`, setting `Resolution` to `Max`, enabling `ReplaceTabWithSpace` is useful on almost all platforms. `SanitiseClearScreen`, `IgnoreTextInGraphics`, and `ClearScreenOnModeSwitch` are more specific, and their use depends on the firmware.

   Note: Some Macs, such as the `MacPro5,1`, may have incompatible console output when using modern GPUs, and thus only `BuiltinGraphics` may work for them in such cases. NVIDIA GPUs may require additional firmware upgrades.

2. `ConsoleMode`
   Type: `plist string`