OpenCore

Reference Manual (0.7.8.9)

[2022.03.02]

Here *ParseDarwinVersion* argument is assumed to be 3 integers obtained by splitting Darwin kernel version string from left to right by the `.` symbol. *FindDarwinVersion* function looks up Darwin kernel version by locating `"Darwin Kernel Version` $\kappa.\lambda.\mu$`"` string in the kernel image.

7. `MinKernel`
   Type: `plist string`
   Failsafe: Empty
   Description: Adds kernel extension on specified macOS version or newer.

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

8. `PlistPath`
   Type: `plist string`
   Failsafe: Empty
   Description: Kext `Info.plist` path relative to bundle (e.g. `Contents/Info.plist`).

7.4 Block Properties

1. `Arch`
   Type: `plist string`
   Failsafe: `Any` (Apply to any supported architecture)
   Description: Kext block architecture (`i386`, `x86_64`).

2. `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.

3. `Enabled`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Set to `true` to block this kernel extension.

4. `Identifier`
   Type: `plist string`
   Failsafe: Empty
   Description: Kext bundle identifier (e.g. `com.apple.driver.AppleTyMCEDriver`).

5. `MaxKernel`
   Type: `plist string`
   Failsafe: Empty
   Description: Blocks kernel extension on specified macOS version or older.

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

6. `MinKernel`
   Type: `plist string`
   Failsafe: Empty
   Description: Blocks kernel extension on specified macOS version or newer.

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

7. `Strategy`
   Type: `plist string`
   Failsafe: `Disable` (Forcibly make the kernel driver kmod startup code return failure)
   Description: Determines the behaviour of kernel driver blocking.

   Valid values:

   - `Disable` — Forcibly make the kernel driver kmod startup code return failure.
   - `Exclude` — Remove the kernel driver from the kernel cache by dropping plist entry and filling in zeroes.

   Note: It is risky to `Exclude` a kext that is a dependency of others.

Note 2: At this moment `Exclude` is only applied to `prelinkedkernel` and newer mechanisms.

Note 3: In most cases strategy `Exclude` requires the new kext to be injected as a replacement.

7.5 Emulate Properties

1. `Cpuid1Data`
   Type: `plist data`, 16 bytes
   Failsafe: All zero
   Description: Sequence of `EAX`, `EBX`, `ECX`, `EDX` values to replace `CPUID (1)` call in XNU kernel.

   This property primarily meets three requirements:

   - Enabling support for an unsupported CPU model (e.g. Intel Pentium).
   - Enabling support for a CPU model not yet supported by a specific version of macOS (typically old versions).
   - 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 (`0x0A0655`):
     `Cpuid1Data`: `55 06 0A 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`
   - Alder Lake (`0x090672`) to Comet Lake (`0x0A0655`):
     `Cpuid1Data`: `55 06 0A 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`.

Requirement: 10.13 (not required for older)
Description: Prevent kernel from printing kext dump in the panic log preventing from observing panic details. Affects 10.13 and above.

16. `PowerTimeoutKernelPanic`
Type: `plist boolean`
Failsafe: `false`
Requirement: 10.15 (not required for older)
Description: Disables kernel panic on setPowerState timeout.

    An additional security measure was added to macOS Catalina (10.15) causing kernel panic on power change timeout for Apple drivers. Sometimes it may cause issues on misconfigured hardware, notably digital audio, which 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 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 KVM and other hypervisors it provides precomputed MSR 35h values solving kernel panic with `-cpu host`.
    - 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+).

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.

    On macOS 12+, it is no longer possible to set trim timeout for APFS filesystems. However, trim can be disabled when the timeout value is set to `0`.

19. `ThirdPartyDrives`

Development and debug kernels produce more useful kernel panic logs. Consider downloading and installing the `KernelDebugKit` from developer.apple.com when debugging a problem. To activate a development kernel, the boot argument `kcsuffix=development` should be added. Use the `uname -a` command to ensure that the current loaded kernel is a development (or a debug) kernel.

In cases where the OpenCore kernel panic saving mechanism is not used, kernel panic logs may still be found in the `/Library/Logs/DiagnosticReports` directory.

Starting with macOS Catalina, kernel panics are stored in JSON format and thus need to be preprocessed before passing to `kpdescribe.sh`:

```
cat Kernel.panic | grep macOSProcessedStackshotData |
  python -c 'import json,sys;print(json.load(sys.stdin)["macOSPanicString"])'
```

3. `DisableWatchDog`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Some types of firmware may not succeed in booting the operating system quickly, especially in debug mode. This results in the watchdog timer aborting the process. This option turns off the watchdog timer.

4. `DisplayDelay`
   Type: `plist integer`
   Failsafe: `0`
   Description: Delay in microseconds executed after every printed line visible onscreen (i.e. console).

5. `DisplayLevel`
   Type: `plist integer`, 64 bit
   Failsafe: `0`
   Description: EDK II debug level bitmask (sum) showed onscreen. Unless `Target` enables console (onscreen) printing, onscreen debug output will not be visible.

   The following levels are supported (discover more in DebugLib.h):

   - `0x00000002` (bit `1`) — `DEBUG_WARN` in `DEBUG`, `NOOPT`, `RELEASE`.
   - `0x00000040` (bit `6`) — `DEBUG_INFO` in `DEBUG`, `NOOPT`.
   - `0x00400000` (bit `22`) — `DEBUG_VERBOSE` in custom builds.
   - `0x80000000` (bit `31`) — `DEBUG_ERROR` in `DEBUG`, `NOOPT`, `RELEASE`.

6. `LogModules`
   Type: `plist string`
   Failsafe: `*`
   Description: Filter log entries by module.

   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. `SerialInit`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Perform serial port initialisation.

   This option will perform serial port initialisation within OpenCore prior to enabling (any) debug logging. Serial port configuration is defined via PCDs at compile time in `gEfiMdeModulePkgTokenSpaceGuid` GUID.

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.

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` — ~~OcDebugLogLib~~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
- `OCKM` — OcAppleKeyMapLib
- `OCL` — ~~OcDebugLogLib~~ OcLogAggregatorLib
- `OCM` — OcMiscLib
- `OCMCO` — OcMachoLib
- `OCME` — OcHeciLib
- `OCMM` — OcMemoryLib
- `OCPE` — OcPeCoffLib, OcPeCoffExtLib
- `OCPI` — OcFileLib, partition info
- `OCPNG` — OcPngLib
- `OCRAM` — OcAppleRamDiskLib
- `OCRTC` — OcRtcLib
- `OCSB` — OcAppleSecureBootLib

As an alternative, the first 8 bytes of `SystemUUID` can be used for `ApECID`, this is found in macOS 11 for Macs without the T2 chip.

With this value set and `SecureBootModel` valid (and not `Disabled`), it is possible to achieve `Full Security` of Apple Secure Boot.

To start using personalised Apple Secure Boot, the operating system must be reinstalled or personalised. ~~Unless~~ Until the operating system is personalised, only macOS DMG recovery ~~cannot~~ can be loaded. In cases where DMG recovery is missing, it can be downloaded by using the `macrecovery` utility and saved in `com.apple.recovery.boot` as explained in the Tips and Tricks section. Note that DMG loading needs to be set to `Signed` to use any DMG with Apple Secure Boot.

To personalise an existing operating system, use the `bless` command after loading to macOS DMG recovery. Mount the system volume partition, unless it has already been mounted, and execute the following command:

```
bless --folder "/Volumes/Macintosh HD/System/Library/CoreServices" \
  --bootefi --personalize
```

On macOS 11 and newer the dedicated `x86legacy` model always uses `ApECID`. When this configuration setting is left as `0` first 8 bytes of `system-id` variable are used instead.

On macOS versions before macOS 11, which introduced a dedicated `x86legacy` model for models without the T2 chip, personalised Apple Secure Boot may not work as expected. When reinstalling the operating system, the macOS Installer from macOS 10.15 and older will often run out of free memory on the `/var/tmp` partition when trying to install macOS with the personalised Apple Secure Boot. Soon after downloading the macOS installer image, an `Unable to verify macOS` error message will appear.

To workaround this issue, allocate a dedicated RAM disk of 2 MBs for macOS personalisation by entering the following commands in the macOS recovery terminal before starting the installation:

```
disk=$(hdiutil attach -nomount ram://4096)
diskutil erasevolume HFS+ SecureBoot $disk
diskutil unmount $disk
mkdir /var/tmp/OSPersonalizationTemp
diskutil mount -mountpoint /var/tmp/OSPersonalizationTemp $disk
```

5. `AuthRestart`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enable `VirtualSMC`-compatible authenticated restart.

   Authenticated restart is a way to reboot FileVault 2 enabled macOS without entering the password. A dedicated terminal command can be used to perform authenticated restarts: `sudo fdesetup authrestart`. It is also used when installing operating system updates.

   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:

Most Linux distros require the `ext4_x64` driver, a few may require the `btrfs_x64` driver, and a few may require no additional file system driver: it depends on the filesystem of the boot partition of the installed distro, and on what filesystems are already supported by the system's firmware. LVM is not currently supported - this is because it is not believed that there is currently a stand-alone UEFI LVM filesystem driver.

Be aware of the `SyncRuntimePermissions` quirk, which may need to be set to avoid early boot failure (typically halting with a black screen) of the Linux kernel, due to a firmware bug of some firmware released after 2017. When present and not mitigated by this quirk, this affects booting via OpenCore with or without OpenLinuxBoot.

After installing OpenLinuxBoot, it is recommended to compare the options shown in the OpenCore debug log when booting (or attempting to boot) a given distro against the options seen using the shell command `cat /proc/cmdline` when the same distro has been booted via its native bootloader. In general (for safety and security of the running distro) these options should match, and if they do not it is recommended to use the driver arguments below (in particular `LINUX_BOOT_ADD_RO`, `LINUX_BOOT_ADD_RW`, ~~`partuuidopts`~~`autoopts:{PARTUUID}` and `autoopts`) to modify the options as required. Note however that the following differences are normal and do not need to be fixed:

- If the default bootloader is GRUB then the options generated by OpenLinuxBoot will not contain a `BOOT_IMAGE=...` value where the GRUB options do, and will contain an `initrd=...` value where the GRUB options do not.
- OpenLinuxBoot uses `PARTUUID` rather than filesystem `UUID` to identify the location of `initrd`, this is by design as UEFI filesystem drivers do not make Linux filesystem `UUID` values available.
- Less important graphics handover options (such as discussed in the Ubuntu example given in `autoopts` below) will not match exactly, this is not important as long as distro boots successfully.

If using OpenLinuxBoot with Secure Boot, users may wish to use the `shim-to-cert.tool` included in OpenCore utilities, which can be used to extract the public key needed to boot a distro's kernels directly, as done when using OpenCore with OpenLinuxBoot, rather than via GRUB shim. For non-GRUB distros, the required public key must be found by user research.

11.6.1 Configuration

The default parameter values should work well with no changes under most circumstances, but if required the following options for the driver may be specified in `UEFI/Drivers/Arguments`:

- `flags` - Default: all flags ~~except `LINUX_BOOT_ADD_DEBUG_INFO`~~ are set except the following:
  - `LINUX_BOOT_ADD_RW`,
  - `LINUX_BOOT_LOG_VERBOSE` and ~~`LINUX_BOOT_LOG_VERBOSE` are set .~~
  - `LINUX_BOOT_ADD_DEBUG_INFO`.

  Available flags are:

  - `0x00000001` (bit `0`) — `LINUX_BOOT_SCAN_ESP`, Allows scanning for entries on EFI System Partition.
  - `0x00000002` (bit `1`) — `LINUX_BOOT_SCAN_XBOOTLDR`, Allows scanning for entries on Extended Boot Loader Partition.
  - `0x00000004` (bit `2`) — `LINUX_BOOT_SCAN_LINUX_ROOT`, Allows scanning for entries on Linux Root filesystems.
  - `0x00000008` (bit `3`) — `LINUX_BOOT_SCAN_LINUX_DATA`, Allows scanning for entries on Linux Data filesystems.
  - `0x00000080` (bit `7`) — `LINUX_BOOT_SCAN_OTHER`, Allows scanning for entries on file systems not matched by any of the above.

    The following notes apply to all of the above options:

    Note 1: Apple filesystems APFS and HFS are never scanned.

    Note 2: Regardless of the above flags, a file system must first be allowed by `Misc/Security/ScanPolicy` before it can be seen by OpenLinuxBoot or any other `OC_BOOT_ENTRY_PROTOCOL` driver.

    Note 3: It is recommended to enable scanning `LINUX_ROOT` and `LINUX_DATA` in both OpenLinuxBoot flags and `Misc/Security/ScanPolicy` in order to be sure to detect all valid Linux installs, since Linux boot filesystems are very often marked as `LINUX_DATA`.

  - `0x00000100` (bit `8`) — `LINUX_BOOT_ALLOW_AUTODETECT`, If set allows autodetecting and linking `vmlinuz*` and `init*` ramdisk files when `loader/entries` files are not found.
  - `0x00000200` (bit `9`) — `LINUX_BOOT_USE_LATEST`, When a Linux entry generated by OpenLinuxBoot is selected as the default boot entry in OpenCore, automatically switch to the latest kernel when a new version is installed.

When this option is set, an internal menu entry id is shared between kernel versions from the same install of Linux. Linux boot options are always sorted highest kernel version first, so this means that the latest kernel version of the same install always shows as the default, with this option set.

Note: This option is recommended on all systems.

- `0x00000400` (bit `10`) — `LINUX_BOOT_ADD_RO`, This option applies to autodetected Linux only (i.e. not to BLSpec or Fedora-style distributions which have `/loader/entries/*.conf` files). Some distributions run a filesystem check on loading which requires the root filesystem to initially be mounted read-only via the `ro` kernel option, which requires this option to be added to the autodetected options. Set this bit to add this option on autodetected distros; should be harmless but very slightly slow down boot time (due to requried remount as read-write) on distros which do not require it. When there are multiple distros and it is required to specify this option for specific distros only, use ~~`partuuidopts`~~`autoopts:{`~~`partuuid`~~`PARTUUID}+=ro` to manually add the option where required, instead of using this flag.
- `0x00000800` (bit `11`) — `LINUX_BOOT_ADD_RW`, Like `LINUX_BOOT_ADD_RO`, this option applies to autodetected Linux only. It is not required for most distros (which usually require either `ro` or nothing to be added to detected boot options), but is required on some Arch-derived distros, e.g. EndeavourOS. When there are multiple distros and it is required to specify this option for specific distros only, use ~~`partuuidopts`~~`autoopts:{`~~`partuuid`~~`PARTUUID}+=rw` to manually add the option where required, instead of using this flag. If this option and `LINUX_BOOT_ADD_RO` are both specified, only this option is applied and `LINUX_BOOT_ADD_RO` is ignored.
- `0x00002000` (bit `13`) — `LINUX_BOOT_ALLOW_CONF_AUTO_ROOT`, In some instances of `BootLoaderSpecByDefault` in combination with `ostree`, the `/loader/entries/*.conf` files do not specify a required `root=...` kernel option – it is added by GRUB. If this bit is set and this situation is detected, then automatically add this option. (Required for example by Endless OS.)
- `0x00004000` (bit `14`) — `LINUX_BOOT_LOG_VERBOSE`, Add additional debug log info about files encountered and autodetect options added while scanning for Linux boot entries.
- `0x00008000` (bit `15`) — `LINUX_BOOT_ADD_DEBUG_INFO`, Adds a human readable file system type, followed by the first eight characters of the partition's unique partition uuid, to each generated entry name. Can help with debugging the origin of entries generated by the driver when there are multiple Linux installs on one system.

Flag values can be specified in hexadecimal beginning with `0x` or in decimal, e.g. `flags=0x80` or `flags=128`. It is also possible to specify flags to add or remove, using syntax such as `flags+=0xC000` to add all debugging options or `flags-=0x400` to remove the `LINUX_BOOT_ADD_RO` option.

- ~~`partuuidopts`~~`autoopts:{`~~`partuuid`~~`PARTUUID}[+]="{options}"` - Default: not set.

  Allows manually specifying kernel options to use in autodetect mode for a given partition only. Replace the text `{PARTUUID}` with the specific partition UUID on which the kernels are stored (in normal layout, the partition which contains `/boot`), e.g. `autoopts:11223344-5566-7788-99aa-bbccddeeff00+="vt.handoff=7"`. If specified with `+=` then these options are used in addition to any autodetected options, if specified with `=` they are used instead. Used for autodetected Linux only ~~. Values~~ – values specified here are never used for entries created from `/loader/entries/*.conf` files.

  Note: The ~~`partuuid`~~`PARTUUID` value to be specified here is typically the same as the `PARTUUID` seen in `root=PARTUUID=...` in the Linux kernel boot options (view using `cat /proc/cmdline`)~~for autodetected Debian-style distros, but is not the same for Fedora-style distros booted from `/loader/entries/*.conf` files. Typically this option should not be needed in the latter case, but in case it is, to find out the unique partition uuid to use look for `LNX:` entries in the OpenCore debug log file.~~. Alternatively, and for more advanced scenarios, it is possible to examine how the distro's partitions are mounted using the Linux `mount` command, and then find out the partuuid of relevant mounted partitions by examining the output of `ls -l /dev/disk/by-partuuid`.

- `autoopts[+]="{options}"` - Default: None specified. ~~The~~

  Allows manually specifying kernel options to use ~~for autodetected Linux only. The value here is never used for entries created from `/loader/entries/*.conf` files. `partuuidopts` may be~~ in autodetect mode. The alternative format `autoopts:{PARTUUID}` is more suitable where there are multiple distros, but `autoopts` with no PARTUUID required ~~is~~ may be more convenient for just one distro. If specified with `+=` then these are used in addition to autodetected options, if specified with `=` they are used instead. Used for autodetected Linux only – values specified here are never used for entries created from `/loader/entries/*.conf` files.

As example usage, it is possible to use `+=` format to add a `vt.handoff` options, such as `autopts+="vt.handoff=7"` or `autopts+="vt.handoff=3"` (check `cat /proc/cmdline` when booted via the distro's default bootloader) on Ubuntu and related distros, in order to add the `vt.handoff` option to the auto-detected GRUB defaults, and avoid a flash of text showing before the distro splash screen.

11.6.2 Additional information

OpenLinuxBoot can detect the `loader/entries/*.conf` files created according to the Boot Loader Specification or the closely related systemd BootLoaderSpecByDefault. The former is specific to systemd-boot and is used by Arch Linux, the latter applies to most Fedora-related distros including Fedora itself, RHEL and variants.

Where the above files are not present, OpenLinuxBoot can autodetect and boot `{boot}/vmlinuz*` kernel files directly. It links these automatically – based on the kernel version in the filename – to their associated `{boot}/init*` ramdisk files. This applies to most Debian-related distros, including Debian itself, Ubuntu and variants.

When autodetecting in `/boot` as part of the root filesystem, OpenLinuxBoot looks in `/etc/default/grub` for kernel boot options and `/etc/os-release` for the distro name. When autodetecting in a standalone boot partition (i.e. when `/boot` has its own mount point), OpenLinuxBoot cannot autodetect kernel arguments and all kernel arguments except `initrd=...` must be fully specified by hand using `autoopts=...` or `autoopts:{partuuid}=...` (`+=` variants of these options will not work, as these only add additional arguments).

BootLoaderSpecByDefault (but not pure Boot Loader Specification) can expand GRUB variables in the `*.conf` files – and this is used in practice in certain distros such as CentOS. In order to handle this correctly, when this situation is detected OpenLinuxBoot extracts all variables from `{boot}/grub2/grubenv` and also any unconditionally set variables from `{boot}/grub2/grub.cfg`, and then expands these where required in `*.conf` file entries.

The only currently supported method of starting Linux kernels relies on their being compiled with EFISTUB. This applies to almost all modern distros, particularly those which use systemd. Note that most modern distros use systemd as their system manager, even though most do not use systemd-boot as their bootloader.

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.

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

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.~~
- --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. 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.

11.8 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.

   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.