OpenCore

Reference Manual (1.0.2.3)

[2024.11.20]

8.2 Properties

1. `BlessOverride`
   Type: `plist array`
   Failsafe: Empty
   Description: Add custom scanning paths through the bless model.

   To be filled with `plist string` entries containing absolute UEFI paths to customised bootloaders such as `\EFI\debian\grubx64.efi` for the Debian bootloader. This allows non-standard boot paths to be automatically discovered by the OpenCore picker. Designwise, they are equivalent to predefined blessed paths, such as `\System\Library\CoreServices\boot.efi` or `\EFI\Microsoft\Boot\bootmgfw.efi`, but unlike predefined bless paths, they have the highest priority.

2. `Boot`
   Type: `plist dict`
   Description: Apply the boot configuration described in the Boot Properties section below.

3. `Debug`
   Type: `plist dict`
   Description: Apply debug configuration described in the Debug Properties section below.

4. `Entries`
   Type: `plist array`
   Failsafe: Empty
   Description: Add boot entries to OpenCore picker.

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

5. `Security`
   Type: `plist dict`
   Description: Apply the security configuration described in the Security Properties section below.

6. `Serial`
   Type: `plist dict`
   Description: Perform serial port initialisation and configure PCD values required by `BaseSerialPortLib16550` for serial ports to properly function. Values are listed and described in the Serial Properties and Serial Custom Properties section below.

   By enabling `Init`, this section ensures that the serial port is initialised when it is not done by firmware. In order for OpenCore to print logs to the serial port, bit `3` (i.e. serial logging) for `Target` under section `Misc->Debug` must be set.

   When debugging with serial ports, `BaseSerialPortLib16550` only recognises internal ones provided by the motherboard by default. If the option `Override` is enabled, this section will override the PCD values listed in BaseSerialPortLib16550.inf such that external serial ports (e.g. from a PCI card) will also function properly. Specifically, when troubleshooting macOS, in addition to overriding these PCD values, it is also necessary to turn the `CustomPciSerialDevice` kernel quirk on in order for the XNU to use such exterior serial ports.

   Refer to MdeModulePkg.dec for the explanations of each key.

7. `Tools`
   Type: `plist array`
   Failsafe: Empty
   Description: Add tool entries to the OpenCore picker.

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

   Note: Certain UEFI tools, such as UEFI Shell, can be very dangerous and **MUST NOT** appear in production configurations, ~~paticularly~~ particularly in vaulted configurations as well as those protected by secure boot, as such tools can be used to bypass the secure boot chain. Refer to the UEFI section for examples of UEFI tools.

disklabel utility or the bless --folder {FOLDER_PATH} --label {LABEL_TEXT} command. When pre-rendered labels are disabled or missing, use label text in .contentDetails (or .disk_label.contentDetails) file next to bootloader if present instead, otherwise the entry name itself will be rendered.

- 0x0004 — OC_ATTR_USE_GENERIC_LABEL_IMAGE, provides predefined label images for boot entries without custom entries. This may however give less detail for the actual boot entry.
- 0x0008 — OC_ATTR_HIDE_THEMED_ICONS, prefers builtin icons for certain icon categories to match the theme style. For example, this could force displaying the builtin Time Machine icon. Requires OC_ATTR_USE_VOLUME_ICON.
- 0x0010 — OC_ATTR_USE_POINTER_CONTROL, enables pointer control in the OpenCore picker when available. For example, this could make use of mouse or trackpad to control UI elements.
- 0x0020 — OC_ATTR_SHOW_DEBUG_DISPLAY, enable display of additional timing and debug information, in Builtin picker in DEBUG and NOOPT builds only.
- 0x0040 — OC_ATTR_USE_MINIMAL_UI, use minimal UI display, no Shutdown or Restart buttons, affects OpenCanopy and builtin picker.
- 0x0080 — OC_ATTR_USE_FLAVOUR_ICON, provides flexible boot entry content description, suitable for picking the best media across different content sets:
  When enabled, the entry icon in OpenCanopy and the audio assist entry sound in OpenCanopy and builtin boot picker are chosen by something called content flavour. To determine content flavour the following algorithm is used:
  - For a Tool the value is read from Flavour field.
  - For an automatically discovered entry, including for boot entry protocol entries such as those generated by the OpenLinuxBoot driver, it is read from the .contentFlavour file next to the bootloader, if present.
  - For a custom entry specified in the Entries section it is read from the .contentFlavour file next to the bootloader if Flavour is Auto, otherwise it is specified via the Flavour value itself.
  - If read flavour is Auto or there is no .contentFlavour, entry flavour is chosen based on the entry type (e.g. Windows automatically gets Windows flavour).

  The Flavour value is a sequence of : separated names limited to 64 characters of printable 7-bit ASCII. This is designed to support up to approximately five names. Each name refers to a flavour, with the first name having the highest priority and the last name having the lowest priority. Such a structure allows describing an entry in a more specific way, with icons selected flexibly depending on support by the audio-visual pack. A missing audio or icon file means the next flavour should be tried, and if all are missing the choice happens based on the type of the entry. Example flavour values: BigSur:Apple, Windows10:Windows. OpenShell:UEFIShell:Shell.

  Using flavours means that you can switch between icon sets easily, with the flavour selecting the best available icons from each set. E.g. specifying icon flavour Debian:Linux will use the icon Debian.icns if provided, then will try Linux.icns, then will fall back to the default for an OS, which is HardDrive.icns.

  Things to keep in mind:
  - For security reasons Ext<Flavour>.icns and <Flavour>.icns are both supported, and only Ext<Flavour>.icns will be used if the entry is on an external drive (followed by default fallback ExtHardDrive.icns).
  - Where both apply .VolumeIcon.icns takes ~~precence~~ precedence over .contentFlavour.
  - In order to allow icons and audio assist to work correctly for tools (e.g. for UEFI Shell), system default boot entry icons (see Docs/Flavours.md) specified in the Flavour setting for Tools or Entries will continue to apply even when flavour is disabled. Non-system icons will be ignored in this case. In addition, the flavours UEFIShell and NVRAMReset are given special processing, identifying their respective tools to apply correct audio-assist, default builtin labels, etc.
  - A list of recommended flavours is provided in Docs/Flavours.md.
- 0x0100 — OC_ATTR_USE_REVERSED_UI, reverse position of Shutdown and Restart buttons, affects OpenCanopy and builtin picker. The reversed setting matches older macOS, and since it was the previous default in OpenCore it may better match some custom backgrounds. Only applicable when OC_ATTR_USE_MINIMAL_UI is not set.
- 0x0200 — OC_ATTR_REDUCE_MOTION, reduce password and menu animation in OpenCanopy, leaving only animations which communicate information not otherwise provided.
  Note: These same animations, plus additional animations whose information is provided by voice-over, are automatically disabled when PickerAudioAssist is enabled.

9. PickerAudioAssist
   Type: plist boolean

Failsafe: `false`
Description: Enable screen reader by default in the OpenCore picker.

For the macOS bootloader, screen reader preference is set in the `preferences.efires` archive in the `isVOEnabled.int32` file and is controlled by the operating system. For OpenCore screen reader support, this option is an independent equivalent. Toggling screen reader support in both the OpenCore picker and the macOS bootloader FileVault 2 login window can also be done by using the `Command` + `F5` key combination.

Note: The screen reader requires working audio support. Refer to the `UEFI Audio Properties` section for details.

10. `PickerMode`
Type: `plist string`
Failsafe: `Builtin`
Description: Choose picker used for boot management.

`PickerMode` describes the underlying boot management with an optional user interface responsible for handling boot options.

The following values are supported:

- `Builtin` — boot management is handled by OpenCore, a simple text-only user interface is used.
- `External` — an external boot management protocol is used if available. Otherwise, the `Builtin` mode is used.
- `Apple` — Apple boot management is used if available. Otherwise, the `Builtin` mode is used.

Upon success, the `External` mode may entirely disable all boot management in OpenCore except for policy enforcement. In the `Apple` mode, it may additionally bypass policy enforcement. Refer to the OpenCanopy plugin for an example of a custom user interface.

The OpenCore built-in picker contains a set of actions chosen during the boot process. The list of supported actions is similar to Apple BDS and typically can be accessed by holding `action hotkeys` during the boot process.

The following actions are currently considered:

- `Default` — this is the default option, and it lets the built-in OpenCore picker load the default boot option as specified in the Startup Disk preference pane.
- `ShowPicker` — this option forces the OpenCore picker to be displayed. This can typically be achieved by holding the `OPT` key during boot. Setting `ShowPicker` to `true` will make `ShowPicker` the default option.
- `BootApple` — this options performs booting to the first Apple operating system found unless the chosen default operating system is one from Apple. Hold the `X` key down to choose this option.
- `BootAppleRecovery` — this option performs booting into the Apple operating system recovery partition. This is either that related to the default chosen operating system, or first one found when the chosen default operating system is not from Apple or does not have a recovery partition. Hold the `CMD+R` hotkey combination down to choose this option.

Note 1: On non-Apple firmware `KeySupport`, `OpenUsbKbDxe`, or similar drivers are required for key handling. However, not all of the key handling functions can be implemented on several types of firmware.

Note 2: In addition to `OPT`, OpenCore supports using both the `Escape` and `Zero` keys to enter the OpenCore picker when `ShowPicker` is disabled. `Escape` exists to support co-existence with the Apple picker (including OpenCore `Apple` picker mode) and to support firmware that fails to report held `OPT` key, as on some PS/2 keyboards. In addition, `Zero` is provided to support systems on which `Escape` is already assigned to some other pre-boot firmware feature. In systems which do not require `KeySupport`, pressing and holding one of these keys from after power on until the picker appears should always be successful. The same should apply when using `KeySupport` mode if it is correctly configured for the system, i.e. with a long enough `KeyForgetThreshold`. If pressing and holding the key is not successful to reliably enter the picker, multiple repeated keypresses may be tried instead.

Note 3: On Macs with problematic GOP, it may be difficult to re-bless OpenCore if its bless status is lost. The `BootKicker` utility can be used to work around this problem, if set up as a Tool in OpenCore with `FullNvramAccess` enabled. It will launch the Apple picker, which allows selection of an item to boot next (with `Enter`), or next and from then on until the next change (with `CTRL+Enter`). Note that after the selection is made, the system *will reboot* before the chosen entry is booted. While this behaviour might seem surprising, it can be

Note that enabling Apple Secure Boot is demanding on invalid configurations, faulty macOS installations, and on unsupported setups.

Things to consider:

(a) As with T2 Macs, all unsigned kernel extensions as well as several signed kernel extensions, including NVIDIA Web Drivers, cannot be installed.
(b) The list of cached kernel extensions may be different, resulting in a need to change the list of `Added` or `Forced` kernel extensions. For example, `IO80211Family` cannot be injected in this case.
(c) System volume alterations on operating systems with sealing, such as macOS 11, may result in the operating system being unbootable. Do not try to disable system volume encryption unless Apple Secure Boot is disabled.
(d) Boot failures might occur when the platform requires certain settings, but they have not been enabled because the associated issues were not discovered earlier. Be extra careful with `IgnoreInvalidFlexRatio` or `HashServices`.
(e) Operating systems released before Apple Secure Boot was released (e.g. macOS 10.12 or earlier), will still boot until UEFI Secure Boot is enabled. This is so because Apple Secure Boot treats these as incompatible and they are then handled by the firmware (as Microsoft Windows is).
(f) On older CPUs (e.g. before Sandy Bridge), enabling Apple Secure Boot might cause slightly slower loading (by up to 1 second).
(g) As the `Default` value will increase with time to support the latest major released operating system, it is not recommended to use the `ApECID` and the `Default` settings together.
(h) Installing macOS with Apple Secure Boot enabled is not possible while using HFS+ target volumes. This may include HFS+ formatted drives when no spare APFS drive is available.

The installed operating system may have sometimes outdated Apple Secure Boot manifests on the `Preboot` partition, resulting in boot failures. This is likely to be the case when an "OCB: Apple Secure Boot prohibits this boot entry, enforcing!" message is logged.

When this happens, either reinstall the operating system or copy the manifests (files with `.im4m` extension, such as `boot.efi.j137.im4m`) from `/usr/standalone/i386` to `/Volumes/Preboot/<UUID>/System/Library/CoreServices`. Here, `<UUID>` is the system volume identifier. On HFS+ installations, the manifests should be copied to `/System/Library/CoreServices` on the system volume.

For more details on how to configure Apple Secure Boot with UEFI Secure Boot, refer to the UEFI Secure Boot section.

13. `Vault`
Type: `plist string`
Failsafe: `Secure`
Description: Enables the OpenCore vaulting mechanism.

Valid values:

- `Optional` — require nothing, no vault is enforced, insecure.
- `Basic` — require `vault.plist` file present in `OC` directory. This provides basic filesystem integrity verification and may protect from unintentional filesystem corruption.
- `Secure` — require `vault.sig` signature file for `vault.plist` in `OC` directory. This includes `Basic` integrity checking but also attempts to build a trusted bootchain.

The `vault.plist` file should contain SHA-256 hashes for all files used by OpenCore. The presence of this file is highly recommended to ensure that unintentional file modifications (including filesystem corruption) do not go unnoticed. To create this file automatically, use the `create_vault.sh` script. Notwithstanding the underlying file system, the path names and cases between `config.plist` and `vault.plist` must match.

The `vault.sig` file should contain a raw 256 byte RSA-2048 signature from a SHA-256 hash of `vault.plist`. The signature is verified against the public key embedded into `OpenCore.efi`.

To embed the public key, either one of the following should be performed:

- Provide public key during the `OpenCore.efi` compilation in `OpenCoreVault.c` file.
- Binary patch `OpenCore.efi` replacing zeroes with the public key between `=BEGIN OC VAULT=` and `==END OC VAULT==` ASCII markers.

11.6.1 Configuration

No additional configuration should work well in most circumstances, but if required the following options for the driver may be specified in `UEFI/Drivers/Arguments`:

- `--hide-devices` - String value, no default.

  When this option is present and has one or more values separated by semicolons (e.g. --hide-devices=PciRoot(0x0)/Pci(0x1F,0x2)/Sata(0x0,0xFFFF,0x0)/HD(2,GPT,...)), it disables scanning the specified disks for legacy operating system boot sectors.

11.7 OpenLinuxBoot

OpenLinuxBoot is an OpenCore plugin implementing `OC_BOOT_ENTRY_PROTOCOL`. It aims to automatically detect and boot most Linux distros without additional configuration.

Usage is as follows:

- Add `OpenLinuxBoot.efi` and also typically (see below) `ext4_x64.efi` to the `config.plist` `Drivers` section.
- Make sure `RequestBootVarRouting` and `LauncherOption` are enabled in `config.plist`; it is also recommended to enable `HideAuxiliary` in order to hide older Linux kernels except when required (they are added as auxiliary entries and so may then be shown by pressing the `Spacebar` key in the OpenCore boot menu).
- Install Linux as normal if this has not been done earlier – OpenLinuxBoot is not involved in this stage.
- Reboot into OpenCore: the installed Linux distribution should just appear and boot directly from OpenCore when selected, which it does without chainloading via GRUB.

If OpenCore has already been manually set up to boot Linux, e.g. via `BlessOverride` or via `Entries` then then these settings may be removed so that the Linux distribution is not displayed twice in the boot menu.

It is recommended to install Linux with its default bootloader, even though this will not be actively used when booting via OpenLinuxBoot. This is because OpenLinuxBoot has to detect the correct kernel options to use, and does so by looking in files left by the default bootloader. If no bootloader was installed (or these options cannot be found) booting is still possible, but the correct boot options must be manually specified before OpenLinuxBoot will attempt to start the distro.

OpenLinuxBoot typically requires filesystem drivers that are not available in firmware, such as EXT4 and BTRFS drivers. These drivers can be obtained from external sources. Drivers tested in basic scenarios can be downloaded from OcBinaryData. Be aware that these drivers are not tested for reliability in all ~~scenarious~~scenarios, nor did they undergo tamper-resistance testing, therefore they may carry potential security or data-loss risks.

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`, `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 install a user built, user signed Shim bootloader giving SBAT and MOK integration, as explained in OpenCore ShimUtils.

11.7.1 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 are set except the following:
  - `LINUX_BOOT_ADD_RW`,
  - `LINUX_BOOT_LOG_VERBOSE`,
  - `LINUX_BOOT_LOG_GRUB_VARS` and
  - `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~~ required 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 `autoopts:{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 `autoopts:{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.
- `0x00010000` (bit `16`) — `LINUX_BOOT_LOG_GRUB_VARS`, When a `BootLoaderSpecByDefault` setup is detected, log available GRUB variables found in `grub2/grubenv` and `grub2/grub.cfg`.
- `0x00020000` (bit `17`) — `LINUX_BOOT_FIX_TUNED`, In some circumstances, such as after upgrades which add TuneD to existing systems, the TuneD system tuning plugin may add its GRUB variables to `loader/entries/*.conf` files but not initialise them in `grub2/grub.cfg`. In order to avoid incorrect boots, OpenLinuxBoot treats used, non-initialised GRUB variables as an error. When this flag is set, empty values are added for the TuneD variables `tuned_params` and `tuned_initrd` if they are not present. This is required for OpenLinuxBoot on TuneD systems with this problem, and harmless otherwise.

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.

- `autoopts:{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 specified here are never used for entries created from `/loader/entries/*.conf` files.

  Note: The `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`). 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.

  Allows manually specifying kernel options to use in autodetect mode. The alternative format `autoopts:{PARTUUID}` is more suitable where there are multiple distros, but `autoopts` with no PARTUUID required 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 `autoopts+="vt.handoff=7"` or `autoopts+="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.

11.7.2 Additional information

OpenLinuxBoot can detect the `loader/entries/*.conf` files created according to the Boot Loader Specification or the closely related ~~systemd~~ Fedora 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~~ Fedora `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 from OpenLinuxBoot 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.

11.8 Other Boot Entry Protocol drivers

In addition to the OpenLinuxBoot plugin, the following `OC_BOOT_ENTRY_PROTOCOL` plugins are made available to add optional, configurable boot entries to the OpenCore boot picker.

11.8.1 ResetNvramEntry

Adds a menu entry which resets NVRAM and immediately restarts. Additionally adds support for hotkey `CMD+OPT+P+R` to perform the same action. Note that on some combinations of firmware and drivers, the `TakeoffDelay` option must be configured in order for this and other builtin hotkeys to be reliably detected.

Note 1: It is known that some Lenovo laptops have a firmware bug, which makes them unbootable after performing NVRAM reset. Refer to acidanthera/bugtracker#995 for details.

Note 2: If `LauncherOption` is set to `Full` or `Short` then the OpenCore boot entry is protected. Resetting NVRAM will normally erase any other boot options not specified via `BlessOverride`, for example Linux installations to custom locations and not using the `OpenLinuxBoot` driver, or user-specified UEFI boot menu entries. To obtain reset NVRAM functionality which does not remove other boot options, it is possible to use the `--preserve-boot` option (though see the warning specified).

The following configuration options may be specified in the `Arguments` section for this driver:

- `--preserve-boot` - Boolean flag, enabled if present.

  If enabled, BIOS boot entries are not cleared during NVRAM reset. This option should be used with caution, as some boot problems can be fixed by clearing these entries.

- `--apple` - Boolean flag, enabled if present.

  On Apple firmware only, this performs a system NVRAM reset. This can result in additional, desirable operations such as NVRAM garbage collection. This is achieved by setting the `ResetNVRam` NVRAM variable. Where available, this has the same effect as pressing `CMD+OPT+P+R` during native boot, although note that if accessed from the menu entry only one boot chime will be heard.

  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.

11.8.2 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.

Options for the driver should be specified as plain text values separated by whitespace in the `Arguments` 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.

11.8.3 FirmwareSettings

Adds a menu entry which will reboot into UEFI firmware settings, when supported. No menu entry added and logs a warning when not supported.

11.9 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.

11.9.1 Configuration

Most UEFI audio configuration is handled via the `UEFI Audio Properties` section, but 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.

4. `JumpstartHotPlug`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Load APFS drivers for newly connected devices.

   Permits APFS USB hot plug which enables loading APFS drivers, both at OpenCore startup and during OpenCore picker display. Disable if not required.

5. `MinDate`
   Type: `plist integer`
   Failsafe: `0`
   Description: Minimal allowed APFS driver date.

   The APFS driver date connects the APFS driver with the calendar release date. Apple ultimately drops support for older macOS releases and APFS drivers from such releases may contain vulnerabilities that can be used to compromise a computer if such drivers are used after support ends. This option permits restricting APFS drivers to current macOS versions.

   - `0` — require the default supported release date of APFS in OpenCore. The default release date will increase with time and thus this setting is recommended. Currently set to 2021/01/01.
   - `-1` — permit any release date to load (strongly discouraged).
   - Other — use custom minimal APFS release date, e.g. `20200401` for 2020/04/01. APFS release dates can be found in OpenCore boot log and `OcApfsLib`.

6. `MinVersion`
   Type: `plist integer`
   Failsafe: `0`
   Description: Minimal allowed APFS driver version.

   The APFS driver version connects the APFS driver with the macOS release. Apple ultimately drops support for older macOS releases and APFS drivers from such releases may contain vulnerabilities that can be used to compromise a computer if such drivers are used after support ends. This option permits restricting APFS drivers to current macOS versions.

   - `0` — require the default supported version of APFS in OpenCore. The default version will increase with time and thus this setting is recommended. Currently set to allow macOS Big Sur and newer (`1600000000000000`).
   - `-1` — permit any version to load (strongly discouraged).
   - Other — use custom minimal APFS version, e.g. `1412101001000000` from macOS Catalina 10.15.4. APFS versions can be found in OpenCore boot log and `OcApfsLib`.

11.13 AppleInput Properties

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

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

   - `Auto` — Use the OEM Apple Event implementation if available, connected and recent enough to be used, otherwise use the OpenCore re-implementation. On non-Apple hardware, this will use the OpenCore builtin implementation. On some Macs such as Classic Mac Pros, this will prefer the Apple implementation but on both older and newer Mac models than these, this option will typically use the OpenCore re-implementation instead. On older Macs, this is because the implementation available is too old to be used while on newer Macs, it is because of optimisations added by Apple which do not connect the Apple Event protocol except when needed – e.g. except when the Apple boot picker is explicitly started. Due to its somewhat ~~unpredicatable~~ unpredictable results, this option is not typically recommended.
   - `Builtin` — Always use OpenCore's updated re-implementation of the Apple Event protocol. Use of this setting is recommended even on Apple hardware, due to improvements (better fine mouse control, configurable

When all available output channels on the codec support the available sound file format then a value of `-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 specified (`AudioOutMask`) dedicated audio ports of the specified codec (`AudioCodec`), located on the specified audio controller (`AudioDevice`).

5. `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: `-15`
   Description: Maximum gain to use for UEFI audio, specified in decibels (dB) with respect to amplifier reference level of 0 dB (see note 1).

   All UEFI audio will use this gain setting when the system amplifier gain read from the `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~~ positive 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 amplifier gain if the system amplifier gain read from the `SystemAudioVolumeDB` NVRAM variable is lower than 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 system amplifier gain level in the `SystemAudioVolumeDB` NVRAM variable is lower than this.

Note 1: This setting is designed to save ~~unecessary~~ unnecessary 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 `SystemAudioVolumeDB` NVRAM variable. 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 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 `alctcsel` property instead.

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

11.15 Drivers Properties

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

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`

10. `ProvideConsoleGop`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Ensure GOP (Graphics Output Protocol) on console handle.

    macOS bootloader requires GOP or UGA (for 10.4 EfiBoot) to be present on console handle, yet the exact location of the graphics protocol is not covered by the UEFI specification. This option will ensure GOP and UGA, if present, are available on the console handle.

    Note: This option will also replace incompatible implementations of GOP on the console handle, as may be the case on the `MacPro5,1` when using modern GPUs.

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

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

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

12. `ReconnectOnResChange`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Reconnect console controllers after changing screen resolution.

    On certain firmware, the controllers that produce the console protocols (simple text out) must be reconnected when the screen resolution is changed via GOP. Otherwise, they will not produce text based on the new resolution.

    Note: On several boards this logic may result in black screen when launching OpenCore from Shell and thus it is optional. In versions prior to 0.5.2 this option was mandatory and not configurable. Please do not use this unless required.

13. `ReplaceTabWithSpace`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Some types of firmware do not print tab characters or everything that follows them, causing difficulties in using the UEFI Shell's builtin text editor to edit property lists and other documents. This option makes the console output spaces instead of tabs.

    Note: This option only applies to `System` renderer.

14. `Resolution`
    Type: `plist string`
    Failsafe: Empty (Maintain current screen resolution)
    Description: Sets console output screen resolution.

    - Set to `WxH@Bpp` (e.g. `1920x1080@32`) or `WxH` (e.g. `1920x1080`) formatted string to request custom resolution from GOP if available.
    - Set to `Max` to attempt using the largest available screen resolution.

    On HiDPI screens `APPLE_VENDOR_VARIABLE_GUID` `UIScale` NVRAM variable may need to be set to `02` to enable HiDPI scaling in `Builtin` text renderer, FileVault 2 UEFI password interface, and boot screen logo. Refer to the Recommended Variables section for details.

    Note: This will fail when console handle has no GOP protocol. When the firmware does not provide it, it can be added with `ProvideConsoleGop` set to `true`.

15. `SanitiseClearScreen`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Some types of firmware reset screen resolutions to a failsafe value (such as `1024x768`) on the