OpenCore

Reference Manual (0.7~~.7~~.8)

[2022.01.21]

Failsafe: `false`
Description: Set to `true` to hide auxiliary entries from the picker menu.

An entry is considered auxiliary when at least one of the following applies:

- Entry is macOS recovery.
- Entry is macOS Time Machine.
- Entry is explicitly marked as `Auxiliary`.
- Entry is system (e.g. `Reset NVRAM`).

To display all entries, the picker menu can be reloaded into "Extended Mode" by pressing the `Spacebar` key. Hiding auxiliary entries may increase boot performance on multi-disk systems.

4. `LauncherOption`
   Type: `plist string`
   Failsafe: `Disabled`
   Description: Register the launcher option in the firmware preferences for persistence.

   Valid values:

   - `Disabled` — do nothing.
   - `Full` — create or update the top priority boot option in UEFI variable storage at bootloader startup.
     - For this option to work, `RequestBootVarRouting` is required to be enabled.
   - `Short` — create a short boot option instead of a complete one.
     - This variant is useful for some older types of firmware, typically from Insyde, that are unable to manage full device paths.
   - `System` — create no boot option but assume specified custom option is blessed.
     - This variant is useful when relying on `ForceBooterSignature` quirk and OpenCore launcher path management happens through `bless` utilities without involving OpenCore.

   This option allows integration with third-party operating system installation and upgrades (which may overwrite the `\EFI\BOOT\BOOTx64.efi` file). The BOOTx64.efi file is no longer used for bootstrapping OpenCore if a custom option is created. The custom path used for bootstrapping can be specified by using the `LauncherPath` option.

   Note 1: Some types of firmware may have NVRAM implementation flaws, no boot option support, or other incompatibilities. While unlikely, the use of this option may result in boot failures and should only be used exclusively on boards known to be compatible. Refer to acidanthera/bugtracker#1222 for some known issues affecting Haswell and other boards.

   Note 2: While NVRAM resets executed from OpenCore would not typically erase the boot option created in `Bootstrap`, executing NVRAM resets prior to loading OpenCore will erase the boot option. Therefore, for significant implementation updates, such as was the case with OpenCore 0.6.4, an NVRAM reset should be executed with `Bootstrap` disabled, after which it can be re-enabled.

   Note 3: Some versions of Intel Visual BIOS (e.g. on Intel NUC) have an unfortunate bug whereby if any boot option is added referring to a path on a USB drive, from then on that is the only boot option which will be shown when any USB drive is inserted. If OpenCore is started from a USB drive on this firmware with `LauncherOption` set to `Full` or `Short`, this applies and only the OpenCore boot entry will be seen afterwards, when any other USB is inserted (this highly non-standard BIOS behaviour affects other software as well). The best way to avoid this is to leave `LauncherOption` set to `Disabled` or `System` on any version of OpenCore which will be started from a USB drive on this firmware. If the problem has already occurred the quickest reliable fix is:

   - Enable the system UEFI Shell in Intel Visual BIOS
   - With power off, insert an OpenCore USB
   - Power up and select the system UEFI Shell
   - Since the system shell does not include `bcfg`, use the system shell to start OpenCore's OpenShell (e.g. by entering the command `FS2:\EFI\OC\Tools\OpenShell.efi` , but you will need to work out which drive is correct for OpenCore and modify the drive number `FS#:` accordingly)
   - Within OpenShell, use `bcfg boot dump` to display the NVRAM boot options and then use `bcfg boot rm #` (where `#` is the number of the OpenCore boot entry) to remove the OpenCore entry

It is alternatively possible to start OpenShell directly from the OpenCore boot menu, if you have a working configured OpenCore for the system. In that case, and if OpenCore has `RequestBootVarRouting` enabled, it will be necessary to run the command `\EFI\OC\Tools\OpenControl.efi disable` before using `bcfg`. (After `OpenControl disable`, it is necessary to either reboot or run `OpenControl restore`, before booting an operating system.) It is also possible to use `efibootmgr` within Linux to remove the offending entry, if you have a working version of Linux on the machine. Linux must be started either not via OpenCore, or via OpenCore with `RequestBootVarRouting` disabled for this to work.

5. `LauncherPath`
   Type: `plist string`
   Failsafe: `Default`
   Description: Launch path for the `LauncherOption` property.

   `Default` points to `OpenCore.efi`. User specified paths, e.g. `\EFI\SomeLauncher.efi`, can be used to provide custom loaders, which are supposed to load `OpenCore.efi` themselves.

6. `PickerAttributes`
   Type: `plist integer`
   Failsafe: `0`
   Description: Sets specific attributes for the OpenCore picker.

   Different OpenCore pickers may be configured through the attribute mask containing OpenCore-reserved (`BIT0`~`BIT15`) and OEM-specific (`BIT16`~`BIT31`) values.

   Current OpenCore values include:

   - `0x0001` — `OC_ATTR_USE_VOLUME_ICON`, provides custom icons for boot entries:
     OpenCore will attempt loading a volume icon by searching as follows, and will fallback to the default icon on failure:
     - `.VolumeIcon.icns` file at `Preboot` volume in per-volume directory (`/System/Volumes/Preboot/{GUID}/` when mounted at the default location within macOS) for APFS (if present).
     - `.VolumeIcon.icns` file at the `Preboot` volume root (`/System/Volumes/Preboot/`, when mounted at the default location within macOS) for APFS (otherwise).
     - `.VolumeIcon.icns` file at the volume root for other filesystems.

     Note 1: The Apple picker partially supports placing a volume icon file at the operating system's `Data` volume root, `/System/Volumes/Data/`, when mounted at the default location within macOS. This approach is flawed: the file is neither accessible to OpenCanopy nor to the Apple picker when FileVault 2, which is meant to be the default choice, is enabled. Therefore, OpenCanopy does not attempt supporting Apple's approach. A volume icon file may be placed at the root of the `Preboot` volume for compatibility with both OpenCanopy and the Apple picker, or use the `Preboot` per-volume location as above with OpenCanopy as a preferred alternative to Apple's approach.

     Note 2: Be aware that using a volume icon on any drive overrides the normal OpenCore picker behaviour for that drive of selecting the appropriate icon depending on whether the drive is internal or external.

   - `0x0002` — `OC_ATTR_USE_DISK_LABEL_FILE`, ~~provides~~ use custom prerendered titles for boot entries from `.disk_label` (`.disk_label_2x`) file next to the bootloader for all filesystems. ~~Prerendered~~ These labels can be generated via the `disklabel` utility or the `bless --folder {FOLDER_PATH} --label {LABEL_TEXT}` command. When prerendered labels are disabled or missing, use label text in ~~(~~`.contentDetails` (or `.disk_label.contentDetails`) ~~will be rendered~~ 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 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`.

7. `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.

8. `PollAppleHotKeys`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enable `modifier hotkey` handling in the OpenCore picker.

   In addition to `action hotkeys`, which are partially described in the `PickerMode` section and are typically handled by Apple BDS, modifier keys handled by the operating system bootloader (`boot.efi`) also exist. These keys allow changing the behaviour of the operating system by providing different boot modes.

   On certain firmware, using modifier keys may be problematic due to driver incompatibilities. To workaround this problem, this option allows registering certain hotkeys in a more permissive manner from within the OpenCore picker. Such extensions include support for tapping on key combinations before selecting the boot item, and for

NVRAM reset. Refer to acidanthera/bugtracker#995 for details.

Note 2: Resetting NVRAM will also erase any boot options not backed up using the bless command. For example, Linux installations to custom locations not specified in BlessOverride

2. `AllowSetDefault`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Allow `CTRL+Enter` and `CTRL+Index` handling to set the default boot option in the OpenCore picker.

   Note 1: May be used in combination with `Shift+Enter` or `Shift+Index` when `PollAppleHotKeys` is enabled.

   Note 2: In order to support systems with unresponsive modifiers during preboot (which includes `V1` and `V2` `KeySupport` mode on some firmware) OpenCore also allows holding the `=/+` key in order to trigger 'set default' mode.

3. `AllowToggleSip`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enable entry for disabling and enabling System Integrity Protection in OpenCore picker.

   This will toggle Apple NVRAM variable `csr-active-config` between `0` for SIP Enabled and a practical default value for SIP Disabled~~(currently `0x26F`)~~.

   Note 1: It is strongly recommended not to make a habit of running macOS with SIP disabled. Use of this boot option may make it easier to quickly disable SIP protection when genuinely needed - it should be re-enabled again afterwards.

   Note 2: OpenCore uses ~~`0x26F`~~`0x27F` ~~even though~~ while `csrutil disable` on ~~Big Sur~~ macOS Big Sur and Monterey sets `0x7F`. ~~To explain the choice:~~

   - ~~`csrutil disable --no-internal` actually sets `0x6F`, and this is preferable because `CSR_ALLOW_APPLE_INTERNAL` (`0x10`) prevents updates (unless you are running an internal build of macOS).~~
   - `CSR_ALLOW_UNAPPROVED_KEXTS` (`0x200`) is generally useful, in the case where you do need to have SIP disabled anyway, as it allows installing unsigned kexts without manual approval in System Preferences.
   - `CSR_ALLOW_UNAUTHENTICATED_ROOT` (`0x800`) is not ~~practical as it prevents incremental (non-full)~~ included, as it is very easy when using it to inadvertently break OS seal and prevent incremental OTA updates.

   Note3: For any other value which you may need to use, it is possible to configure `CsrUtil.efi` as a `TextMode` `Tools` entry to configure a different value, e.g. use `toggle` ~~`0x6F`~~`0x77` in `Arguments` to toggle the SIP disabled value set by default ~~by `csrutil disable --no-internal` in Big Sur~~in macOS Catalina.

4. `ApECID`
   Type: `plist integer`, 64 bit
   Failsafe: `0`
   Description: Apple Enclave Identifier.

   Setting this value to any non-zero 64-bit integer will allow using personalised Apple Secure Boot identifiers. To use this setting, generate a random 64-bit number with a cryptographically secure random number generator. 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 the operating system is personalised, macOS DMG recovery cannot 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:

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 ~~Linux boot options (shown with `cat /proc/cmdline`) seen when booting via OpenLinuxBoot and via the distro's original bootloader.~~ 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` 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 ~~, expect~~ then the options generated by OpenLinuxBoot ~~not to~~ will not contain a `BOOT_IMAGE=...` value where the GRUB options do, and ~~to~~ will contain an `initrd=...` value ~~while~~ where the GRUB options do not. ~~All remaining options should match (option order does not matter) — perhaps excluding less~~
- 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) ~~. If they do not , it is recommended to manually add the missing options, e.g. with `partuuidopts:{partuuid}+={opts}` to target a specific distro (or just with `autoopts+={opts}`, which applies to all installed distros, if only one distro is in use).~~ 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 ~~required public key to validate~~ 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` and `LINUX_BOOT_LOG_VERBOSE` are set.

  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. ~~to Debian-style distrubutions,~~ not to BLSpec ~~and~~ or Fedora-style distributions ~~with~~ which have `/loader/entries/*.conf` files). Some ~~distrubtions~~ 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. ~~To~~ When there are multiple distros and it is required to specify this option for specific distros only, use `partuuidopts:{partuuid}+=ro` ~~instead of~~ 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:{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:{partuuid}[+]="{options}"` - Default: not set.

  Allows specifying kernel options for a given partition only. 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.

  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`) 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 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 more suitable where there are multiple distros, but `autoopts` with no PARTUUID required is 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. 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.