OpenCore

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

[2022.01.16]

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 alteratively 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 custom prerendered titles for boot entries from `.disk_label` (`.disk_label_2x`) file next to the bootloader for all filesystems. Prerendered labels can be generated via the `disklabel` utility or the `bless` command. When disabled or missing, label text in (`.contentDetails` or `.disk_label.contentDetails`) will be rendered 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.

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:

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