



OpenCore

Reference Manual (0.8.~~0~~.1)

[2022.05.24]

2. **AppleXcpmCfgLock**

Type: plist boolean

Failsafe: false

Requirement: 10.8 (not required for older)

Description: Disables PKG_CST_CONFIG_CONTROL (0xE2) MSR modification in XNU kernel, commonly causing early kernel panic, when it is locked from writing (XCPM power management).

Note: This option should be avoided whenever possible. Refer to the **AppleCpuPmCfgLock** description for details.

3. **AppleXcpmExtraMsrs**

Type: plist boolean

Failsafe: false

Requirement: 10.8 (not required for older)

Description: Disables multiple MSR access critical for certain CPUs, which have no native XCPM support.

This is typically used in conjunction with the **Emulate** section on Haswell-E, Broadwell-E, Skylake-SP, and similar CPUs. More details on the XCPM patches are outlined in [acidanthera/bugtracker#365](#).

Note: Additional not provided patches will be required for Ivy Bridge or Pentium CPUs. It is recommended to use **AppleIntelCpuPowerManagement.kext** for the former.

4. **AppleXcpmForceBoost**

Type: plist boolean

Failsafe: false

Requirement: 10.8 (not required for older)

Description: Forces maximum performance in XCPM mode.

This patch writes 0xFF00 to MSR_IA32_PERF_CONTROL (0x199), effectively setting maximum multiplier for all the time.

Note: While this may increase the performance, this patch is strongly discouraged on all systems but those explicitly dedicated to scientific or media calculations. Only certain Xeon models typically benefit from the patch.

5. **CustomPciSerialDevice**

Type: plist boolean

Failsafe: false

Requirement: 10.7

Description: Performs change of PMIO register base address on a customised PCI serial device.

The patch changes the PMIO register base address that the XNU kernel uses for serial input and output, from that of the default built-in COM1 serial port 0x3F8, to the base address stored in the first IO BAR of a specified PCI device or to a specific base address (e.g. 0x2F8 for COM2).

Note: By default, serial logging is disabled. **serial=3** boot argument, which enables serial input and output, should be used for XNU to print logs to the serial port.

Note 2: In addition to this patch, kext **Apple16X50PCI0** should be prevented from attaching to have **kprintf** method working properly. This can be achieved by setting (i.e. **Delete**, then **Add**) the **class-code** property of the PCI serial port device to FFFFFFFF in **DeviceProperties** section. As an alternative solution, a code-less kext **PCISerialDisable.kext** shown in the spoiler **PCISerialDisable.kext/Contents/Info.plist** at [acidanthera/bugtracker#1954](#), may also be used. [In addition, for certain Thunderbolt cards the IOKit personality IOPCITunnelCompatible also needs to be set to true, which can be done by the PCISerialThunderboltEnable.kext attached at acidanthera/bugtracker#2003.](#)

Note 3: For this patch to be correctly applied, **Override** must be enabled with all keys properly set in **Custom**, under section **Misc->Serial**.

Note 4: This patch is for PMIO support and is therefore not applied if **UseMmio** under section **Misc->Serial->Custom** is false. For MMIO, there are boot arguments **pcie_mmio_uart=ADDRESS** and **mmio_uart=ADDRESS** that allow the kernel to use MMIO for serial port access.

Note 5: The serial baud rate must be correctly set in both **BaudRate** under section **Misc->Serial->Custom** and via **serialbaud=VALUE** boot argument, both of which should match against each other. The default baud rate is 115200.

This option enables Aquantia AQtion based 10GbE network cards support, which used to work natively before macOS 10.15.4.

Note: In order for Aquantia cards to properly function, `DisableIoMapper` must be disabled, DMAR ACPI table must not be dropped, and VT-d must be enabled in BIOS.

Note 2: While this patch should enable ethernet support for all Aquantia AQtion series, it has only been tested on AQC-107s based 10GbE network cards.

Note 3: To address AppleVTD incompatibilities after applying this quirk, the Reserved Memory Region section of the corresponding device in the DMAR ACPI table might be removed. This table should be disassembled and edited, then recompiled to AML with tool iASL. For the patched DMAR table to be added, the original one should be deleted. More details can be found at comment on commit 2441455.

13. `ForceSecureBootScheme`

Type: plist boolean

Failsafe: false

Requirement: 11

Description: Force x86 scheme for IMG4 verification.

Note: This option is required on virtual machines when using `SecureBootModel` different from `x86legacy`.

14. `IncreasePciBarSize`

Type: plist boolean

Failsafe: false

Requirement: 10.10

Description: Allows IOPCIFamily to boot with 2 GB PCI BARs.

Normally macOS restricts PCI BARs to 1 GB. Enabling this option (still) does not let macOS actually use PCI devices with larger BARs.

Note: This option should be avoided whenever possible. A need for this option indicates misconfigured or defective firmware.

15. `LapicKernelPanic`

Type: plist boolean

Failsafe: false

Requirement: 10.6 (64-bit)

Description: Disables kernel panic on LAPIC interrupts.

16. `LegacyCommpage`

Type: plist boolean

Failsafe: false

Requirement: 10.4 - 10.6

Description: Replaces the default 64-bit commpage bcopy implementation with one that does not require SSSE3, useful for legacy platforms. This prevents a `commpage no match for last panic` due to no available 64-bit bcopy functions that do not require SSSE3.

17. `PanicNoKextDump`

Type: plist boolean

Failsafe: false

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.

18. `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

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 reliable detection of the **Shift** key when selecting the boot item, in order to work around the fact that hotkeys which are continuously held during boot cannot be reliably detected on many PS/2 keyboards.

This list of known **modifier hotkeys** includes:

- **CMD+C+MINUS** — disable board compatibility checking.
- **CMD+K** — boot release kernel, similar to **kcsuffix=release**.
- **CMD+S** — single user mode.
- **CMD+S+MINUS** — disable KASLR slide, requires disabled SIP.
- **CMD+V** — verbose mode.
- **Shift+Enter**, **Shift+Index** — safe mode, may be used in combination with **CTRL+Enter**, **CTRL+Index**.

9. ShowPicker

Type: plist boolean

Failsafe: false

Description: Show a simple picker to allow boot entry selection.

10. TakeoffDelay

Type: plist integer, 32 bit

Failsafe: 0

Description: Delay in microseconds executed before handling the OpenCore picker startup and **action hotkeys**.

Introducing a delay may give extra time to hold the right **action hotkey** sequence to, for instance, boot into recovery mode. On most systems, the appearance of the initial boot logo is a good indication of the time from which hotkeys can be held down. Earlier than this, the key press may not be registered. On some platforms, setting this option to a minimum of 5000–10000 microseconds ~~may be~~ is also required to access **action hotkeys** due to the nature of the keyboard driver.

If the boot chime is configured (see audio configuration options) then at the expense of slower startup, an even longer delay of half to one second (500000–1000000) may be used to create behaviour similar to a real Mac, where the chime itself can be used as a signal for when hotkeys can be pressed. The boot chime is inevitably later in the boot sequence in OpenCore than on Apple hardware, due to the fact that non-native drivers have to be loaded and connected first. Configuring the boot chime and adding this longer additional delay can also be useful in systems where fast boot time and/or slow monitor signal synchronisation may cause the boot logo not to be shown at all on some boots or reboots.

11. Timeout

Type: plist integer, 32 bit

Failsafe: 0

Description: Timeout in seconds in the OpenCore picker before automatic booting of the default boot entry. Set to 0 to disable.

12. 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.
- ~~**ResetNvram** — this option erases certain UEFI variables and is normally executed by holding down the CMD+OPT+P+R key combination during boot. Another way to erase UEFI variables is to choose **Reset NVRAM** in the OpenCore picker. This option requires **AllowNvramReset** to be set to **true**.~~
- **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 ~~key-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 access the Apple picker. The **BootKicker** utility can be blessed to workaround this problem even without loading OpenCore. On some Macs however, the **BootKicker** utility cannot be run from OpenCore.

13. **PickerVariant**

Type: plist string

Failsafe: Auto

Description: Choose specific icon set to be used for boot management.

An icon set is a directory path relative to **Resources\Image**, where the icons and an optional manifest are located. It is recommended for the artists to use provide their sets in the **Vendor\Set** format, e.g. **Acidanthera\GoldenGate**.

Sample resources provided as a part of OcBinaryData repository provide the following icon set:

- **Acidanthera\GoldenGate** — macOS 11 styled icon set.
- **Acidanthera\Syrah** — macOS 10.10 styled icon set.

- OCM — OcMiscLib
- OCMCO — OcMachoLib
- OCME — OcHeciLib
- OCMM — OcMemoryLib
- OCPE — OcPeCoffLib, OcPeCoffExtLib
- OCPI — OcFileLib, partition info
- OCPNG — OcPngLib
- OCRAM — OcAppleRamDiskLib
- OCRTC — OcRtcLib
- OCSB — OcAppleSecureBootLib
- OCSMB — OcSmbiosLib
- OCSMC — OcSmcLib
- OCST — OcStorageLib
- OCS — OcSerializedLib
- OCTPL — OcTemplateLib
- OCUC — OcUnicodeCollationLib
- OCUT — OcAppleUserInterfaceThemeLib
- OCXML — OcXmlLib

8.5 Security Properties

1. ~~AllowNvramReset~~**Type:** ~~plist boolean~~**Failsafe:** ~~false~~**Description:** ~~Allow CMD+OPT+P+R handling and enable showing NVRAM Reset entry in OpenCore picker.~~

~~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: 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.~~

~~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 0x27F while csrutil-disable on macOS Big Sur and Monterey sets 0x7F.~~

- 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 included, as it is very easy when using it to inadvertently break OS seal and prevent incremental OTA updates.

~~Note 3: 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 0x77 in Arguments to toggle the SIP disabled value set by default in macOS Catalina.~~

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.

8.6 Serial Properties

1. Custom

Type: plist dict

Description: Update serial port properties in `BaseSerialPortLib16550`.

This section lists the PCD values that are used by the `BaseSerialPortLib16550`. When option `Override` is set to `false`, this dictionary is optional.

2. Init

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.

Refer to the `Debugging` section for details.

3. Override

Type: plist boolean

Failsafe: false

Description: Override serial port properties. When this option is set to `false`, no keys from `Custom` will be overridden.

This option will override serial port properties listed in the `Serial Custom Properties` section below.

8.6.1 Serial Custom Properties

1. BaudRate

Type: plist integer

Failsafe: 115200

Description: Set the baud rate for serial port.

This option will override the value of `gEfiMdeModulePkgTokenSpaceGuid.PcdSerialBaudRate` defined in `Mde-ModulePkg.dec`.

2. ClockRate

Type: plist integer

Failsafe: 1843200

Description: Set the clock rate for serial port.

This option will override the value of `gEfiMdeModulePkgTokenSpaceGuid.PcdSerialClockRate` defined in `Mde-ModulePkg.dec`.

3. DetectCable
Type: plist boolean
Failsafe: false
Description: Enable serial port cable detection.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialDetectCable defined in MdeModulePkg.dec.
4. ExtendedTxFifoSize
Type: plist integer
Failsafe: 64
Description: Set the extended transmit FIFO size for serial port.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialExtendedTxFifoSize defined in MdeModulePkg.dec.
5. FifoControl
Type: plist integer
Failsafe: 0x07
Description: Configure serial port FIFO Control settings.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialFifoControl defined in MdeModulePkg.dec.
6. LineControl
Type: plist integer
Failsafe: 0x07
Description: Configure serial port Line Control settings.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialLineControl defined in MdeModulePkg.dec.
7. PciDeviceInfo
Type: plist data
Failsafe: 0xFF
Description: Set PCI serial device information.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialPciDeviceInfo defined in MdeModulePkg.dec.

Note: The maximum allowed size of this option is 41 bytes. Refer to acidanthera/bugtracker#1954 for more details.

Note 2: This option can be set by running the FindSerialPort tool.
8. RegisterAccessWidth
Type: plist integer
Failsafe: 8
Description: Set serial port register access width.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialRegisterAccessWidth defined in MdeModulePkg.dec.
9. RegisterBase
Type: plist integer
Failsafe: 0x03F8
Description: Set the base address of serial port registers.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialRegisterBase defined in MdeModulePkg.dec.
10. RegisterStride
Type: plist integer
Failsafe: 1
Description: Set the serial port register stride in bytes.

11 UEFI

11.1 Introduction

UEFI (Unified Extensible Firmware Interface) is a specification that defines a software interface between an operating system and platform firmware. This section allows loading additional UEFI modules as well as applying tweaks to the onboard firmware. To inspect firmware contents, apply modifications and perform upgrades UEFITool and supplementary utilities can be used.

11.2 Drivers

Depending on the firmware, a different set of drivers may be required. Loading an incompatible driver may lead the system to unbootable state or even cause permanent firmware damage. Some of the known drivers are listed below:

AudioDxe*	HDA audio support driver in UEFI firmware for most Intel and some other analog audio controllers. Staging driver, refer to acidanthera/bugtracker#740 for known issues in AudioDxe.
btrfs_x64	Open source BTRFS file system driver, required for booting with OpenLinuxBoot from a file system which is now quite commonly used with Linux.
BiosVideo*	CSM video driver implementing graphics output protocol based on VESA and legacy BIOS interfaces. Used for UEFI firmware with fragile GOP support (e.g. low resolution). Requires ReconnectGraphicsOnConnect . Included in OpenDuet out of the box.
CrScreenshotDxe*	Screenshot making driver saving images to the root of OpenCore partition (ESP) or any available writeable filesystem upon pressing F10. This is a modified version of CrScreenshotDxe driver by Nikolaj Schlej.
ExFatDxe	Proprietary ExFAT file system driver for Bootcamp support commonly found in Apple firmware. For Sandy Bridge and earlier CPUs, the ExFatDxeLegacy driver should be used due to the lack of RDRAND instruction support.
ext4_x64	Open source EXT4 file system driver, required for booting with OpenLinuxBoot from the file system most commonly used with Linux.
HfsPlus	Recommended. Proprietary HFS file system driver with bless support commonly found in Apple firmware. For Sandy Bridge and earlier CPUs, the HfsPlusLegacy driver should be used due to the lack of RDRAND instruction support.
HiiDatabase*	HII services support driver from MdeModulePkg. This driver is included in most types of firmware starting with the Ivy Bridge generation. Some applications with GUI, such as UEFI Shell, may need this driver to work properly.
EnhancedFatDxe	FAT filesystem driver from FatPkg. This driver is embedded in all UEFI firmware and cannot be used from OpenCore. Several types of firmware have defective FAT support implementation that may lead to corrupted filesystems on write attempts. Embedding this driver within the firmware may be required in case writing to the EFI partition is needed during the boot process.
NvmExpressDxe*	NVMe support driver from MdeModulePkg. This driver is included in most firmware starting with the Broadwell generation. For Haswell and earlier, embedding it within the firmware may be more favourable in case a NVMe SSD drive is installed.
OpenCanopy*	OpenCore plugin implementing graphical interface.
OpenRuntime*	OpenCore plugin implementing OC_FIRMWARE_RUNTIME protocol.
OpenLinuxBoot*	OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL to allow direct detection and booting of Linux distributions from OpenCore, without chainloading via GRUB.
<u>OpenNtfsDxe*</u>	<u>New Technologies File System (NTFS) read-only driver. NTFS is the primary file system for Microsoft Windows versions that are based on Windows NT.</u>
OpenUsbKbDxe*	USB keyboard driver adding support for AppleKeyMapAggregator protocols on top of a custom USB keyboard driver implementation. This is an alternative to builtin KeySupport, which may work better or worse depending on the firmware.
OpenPartitionDxe*	Partition management driver with Apple Partitioning Scheme support. This driver can be used to support loading older DMG recoveries such as macOS 10.9 using Apple Partitioning Scheme. OpenDuet already includes this driver.

- **Windows** — Windows.
- **Other** — Custom entry (see **Entries**).
- **ResetNVRAM** — Reset NVRAM system action or tool.
- **SIPDisabled** — ~~Toogle~~ Toggle SIP tool with SIP disabled.
- **SIPEnabled** — ~~Toogle~~ Toggle SIP tool with SIP enabled.
- **Shell** — Entry with UEFI Shell name (e.g. **OpenShell**).
- **Tool** — Any other tool.

Note: All labels must have a height of exactly 12 px. There is no limit for their width.

Label and icon generation can be performed with bundled utilities: **disklabel** and **icnspack**. Font is Helvetica 12 pt times scale factor.

Font format corresponds to AngelCode binary BMF. While there are many utilities to generate font files, currently it is recommended to use dpFontBaker to generate bitmap font (using CoreText produces best results) and fonverter to export it to binary format.

11.5 OpenRuntime

OpenRuntime is an OpenCore plugin implementing **OC_FIRMWARE_RUNTIME** protocol. This protocol implements multiple features required for OpenCore that are otherwise not possible to implement in OpenCore itself as they are needed to work in runtime, i.e. during operating system functioning. Feature highlights:

- NVRAM namespaces, allowing to isolate operating systems from accessing select variables (e.g. **RequestBootVarRouting** or **ProtectSecureBoot**).
- Read-only and write-only NVRAM variables, enhancing the security of OpenCore, Lilu, and Lilu plugins, such as VirtualSMC, which implements **AuthRestart** support.
- NVRAM isolation, allowing to protect all variables from being written from an untrusted operating system (e.g. **DisableVariableWrite**).
- UEFI Runtime Services memory protection management to workaround read-only mapping (e.g. **EnableWriteUnprotector**).

11.6 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 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 scenarios, nor did they undergo tamper-resistance testing, therefore they may carry potential security or data-loss risks.

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 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.

11.7.1 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.

11.7.2 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. The default value is 0x27F (see below). Any other required value can be specified as a single number in the **Arguments** for this driver. This can be specified as hexadecimal, beginning with 0x, or as decimal.

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 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.

11.8.1 Configuration

Most UEFI audio configuration is handled via the **UEFI Audio Properties** section, but if required the following additional 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 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 of 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`.

- `--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.

11.9 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**.

11. **ReconnectGraphicsOnConnect**

Type: plist boolean

Failsafe: false

Description: Reconnect all graphics drivers during driver connection.

On certain firmware, it may be 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. **SanitiseClearScreen**

Type: plist boolean

Failsafe: false

Description: Some types of firmware reset screen resolutions to a failsafe value (such as 1024x768) on the attempts to clear screen contents when large display (e.g. 2K or 4K) is used. This option attempts to apply a workaround.

Note: This option only applies to the **System** renderer. On all known affected systems, **ConsoleMode** must be set to an empty string for this option to work.

14. **UIScale**

Type: plist integer, 8 bit

Failsafe: -1

Description: User interface scaling factor.

Corresponds to 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:UIScale variable.

- 1 — 1x scaling, corresponds to normal displays.
- 2 — 2x scaling, corresponds to HiDPI displays.
- -1 — leaves the current variable unchanged.
- 0 — automatically chooses scaling based on the current resolution.

Note 1: Automatic scale factor detection works on the basis of total pixel area and may fail on small HiDPI displays, in which case the value may be manually managed using the NVRAM section.

Note 2: When switching from manually specified NVRAM variable to this preference an NVRAM reset may be needed.

15. **UgaPassThrough**

Type: plist boolean

Failsafe: false

Description: Provide UGA protocol instances on top of GOP protocol instances.

Some types of firmware do not implement the legacy UGA protocol but this may be required for screen output by older EFI applications such as EfiBoot from 10.4.

11.16 ProtocolOverrides Properties

1. **AppleAudio**

Type: plist boolean

Failsafe: false

Description: Replaces Apple audio protocols with builtin versions.