



# **OpenCore**

Reference Manual (1.0.4)

[2025.02.23]

**Description:** Apple for macOS booter (typically `boot.efi`); or a name with a suffix, such as `bootmgfw.efi`, for a specific booter.

7. Limit

**Type:** plist integer

**Failsafe:** 0 (Search the entire booter)

**Description:** Maximum number of bytes to search for.

8. Mask

**Type:** plist data

**Failsafe:** Empty (Ignored)

**Description:** Data bitwise mask used during find comparison. Allows fuzzy search by ignoring not masked (set to zero) bits. Must be equal to `Find` in size if set.

9. Replace

**Type:** plist data

**Failsafe:** Empty

**Description:** Replacement data of one or more bytes.

10. ReplaceMask

**Type:** plist data

**Failsafe:** Empty (Ignored)

**Description:** Data bitwise mask used during replacement. Allows fuzzy replacement by updating masked (set to non-zero) bits. Must be equal to `Replace` in size if set.

11. Skip

**Type:** plist integer

**Failsafe:** 0 (Do not skip any occurrences)

**Description:** Number of found occurrences to skip before replacements are applied.

## 5.5 Quirks Properties

1. AllowRelocationBlock

**Type:** plist boolean

**Failsafe:** false

**Description:** Allows booting macOS through a relocation block.

The relocation block is a scratch buffer allocated in the lower 4 GB used for loading the kernel and related structures by `EfiBoot` on firmware where the lower memory region is otherwise occupied by (assumed) non-runtime data. Right before kernel startup, the relocation block is copied back to lower addresses. Similarly, all the other addresses pointing to the relocation block are also carefully adjusted. The relocation block can be used when:

- No better slide exists (all the memory is used)
- `slide=0` is forced (by an argument or safe mode)
- KASLR (slide) is unsupported (this is Mac OS X 10.7 or older)

This quirk typically requires `ProvideCustomSlide` and `AvoidRuntimeDefrag` to be enabled to function correctly. Hibernation is not supported when booting with a relocation block, which will only be used if required when the quirk is enabled.

*Note:* While this quirk is required to run older macOS versions on platforms with used lower memory, it is not compatible with some hardware and macOS 11. In such cases, consider using `EnableSafeModeSlide` instead.

2. AvoidRuntimeDefrag

**Type:** plist boolean

**Failsafe:** false

**Description:** Protect from `boot.efi` runtime memory defragmentation.

This option fixes UEFI runtime services (date, time, NVRAM, power control, etc.) support on firmware that uses SMM backing for certain services such as variable storage. SMM may try to access memory by physical addresses in non-SMM areas but this may sometimes have been moved by `boot.efi`. This option prevents `boot.efi` from moving such data.

*Note:* Most types of firmware, apart from Apple and VMware, need this quirk.

3. [ClearTaskSwitchBit](#)  
**Type:** `plist boolean`  
**Failsafe:** `false`

**Description:** Clear task switch bit during UEFI runtime services calls.

[This quirk resolves FPU-related crashes in UEFI runtime services when using either the 32-bit kernel on a 64-bit UEFI implementation, or the 64-bit kernel on a 32-bit UEFI implementation by removing the task switch \(TS\) bit from CR0 register during their execution. These crashes occur if there is any FPU/SSE instruction usage in UEFI runtime services as XNU is unable to handle exceptions from mixed-mode code. This quirk requires OC\\_FIRMWARE\\_RUNTIME protocol implemented in OpenRuntime.efi.](#)

*Note:* This quirk should only be required when running older macOS versions when the 32-bit kernel is used on a 64-bit UEFI implementation, such as Microsoft Hyper-V.

4. [DevirtualiseMmio](#)  
**Type:** `plist boolean`  
**Failsafe:** `false`

**Description:** Remove runtime attribute from certain MMIO regions.

This quirk reduces the stolen memory footprint in the memory map by removing the runtime bit for known memory regions. This quirk may result in an increase of KASLR slides available but without additional measures, it is not necessarily compatible with the target board. This quirk typically frees between 64 and 256 megabytes of memory, present in the debug log, and on some platforms, is the only way to boot macOS, which otherwise fails with allocation errors at the bootloader stage.

This option is useful on all types of firmware, except for some very old ones such as Sandy Bridge. On certain firmware, a list of addresses that need virtual addresses for proper NVRAM and hibernation functionality may be required. Use the `MmioWhitelist` section for this.

5. [DisableSingleUser](#)  
**Type:** `plist boolean`  
**Failsafe:** `false`

**Description:** Disable single user mode.

This is a security option that restricts the activation of single user mode by ignoring the `CMD+S` hotkey and the `-s` boot argument. The behaviour with this quirk enabled is supposed to match T2-based model behaviour. Refer to this archived article to understand how to use single user mode with this quirk enabled.

*Note:* When Apple Secure Boot is enabled single user mode is always disabled.

6. [DisableVariableWrite](#)  
**Type:** `plist boolean`  
**Failsafe:** `false`

**Description:** Protect from macOS NVRAM write access.

This is a security option that restricts NVRAM access in macOS. This quirk requires `OC_FIRMWARE_RUNTIME` protocol implemented in `OpenRuntime.efi`.

*Note:* This quirk can also be used as an ad hoc workaround for defective UEFI runtime services implementations that are unable to write variables to NVRAM and results in operating system failures.

7. [DiscardHibernateMap](#)  
**Type:** `plist boolean`  
**Failsafe:** `false`

**Description:** Reuse original hibernate memory map.

This option forces the XNU kernel to ignore a newly supplied memory map and assume that it did not change after waking from hibernation. This behaviour is required by Windows to work. Windows mandates preserving runtime memory size and location after S4 wake.

*Note:* This may be used to workaround defective memory map implementations on older, rare legacy hardware. Examples of such hardware are Ivy Bridge laptops with Insyde firmware such as the Acer V3-571G. Do not use this option without a full understanding of the implications.

## 7 Kernel

### 7.1 Introduction

This section allows the application of different kinds of kernelspace modifications on Apple Kernel (XNU). The modifications currently provide driver (kext) injection, kernel and driver patching, and driver blocking.

Kernel and kext changes apply with the following effective order:

- **Block** is processed.
- **Add** and **Force** are processed.
- **Emulate** and **Quirks** are processed.
- **Patch** is processed.

### 7.2 Properties

#### 1. Add

**Type:** plist array

**Failsafe:** Empty

**Description:** Load selected kernel extensions (kexts) from the `OC/Kexts` directory.

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

*Note 1:* The load order is based on the order in which the kexts appear in the array. Hence, dependencies must appear before kexts that depend on them.

*Note 2:* To track the dependency order, inspect the `OSBundleLibraries` key in the `Info.plist` file of the kext being added. Any kext included under the key is a dependency that must appear before the kext being added.

*Note 3:* Kexts may have inner kexts (**Plugins**) included in the bundle. Such **Plugins** must be added separately and follow the same global ordering rules as other kexts.

#### 2. Block

**Type:** plist array

**Failsafe:** Empty

**Description:** Remove selected kernel extensions (kexts) from the prelinked kernel.

To be filled with `plist dictionary` values, describing each blocked kext. Refer to the Block Properties section below for details.

#### 3. Emulate

**Type:** plist dict

**Description:** Emulate certain hardware in kernelspace via parameters described in the Emulate Properties section below.

#### 4. Force

**Type:** plist array

**Failsafe:** Empty

**Description:** Load kernel extensions (kexts) from the system volume if they are not cached.

To be filled with `plist dict` values, describing each kext. Refer to the Force Properties section below for details. This section resolves the problem of injecting kexts that depend on other kexts, which are not otherwise cached. The issue typically affects older operating systems, where various dependency kexts, such as `IOAudioFamily` or `IONetworkingFamily` may not be present in the kernel cache by default.

*Note 1:* The load order is based on the order in which the kexts appear in the array. Hence, dependencies must appear before kexts that depend on them.

*Note 2:* **Force** happens before **Add**.

*Note 3:* The signature of the “forced” kext is not checked in any way. This makes using this feature extremely dangerous and undesirable for secure boot.

*Note 4:* This feature may not work on encrypted partitions in newer operating systems.

**WARNING:** Choose variables carefully, as the nvram.plist file is not vaulted. For instance, do not include `boot-args` or `csr-active-config`, as these can be used to bypass SIP.

#### 5. WriteFlash

**Type:** plist boolean

**Failsafe:** false

**Description:** Enables writing to flash memory for all added variables.

*Note:* This value should be enabled on most types of firmware but is left configurable to account for firmware that may have issues with NVRAM variable storage garbage collection or similar.

The `nvram` command can be used to read NVRAM variable values from macOS by concatenating the GUID and name variables separated by a `:` symbol. For example, `nvram 7C436110-AB2A-4BBB-A880-FE41995C9F82:boot-args`.

A continuously updated variable list can be found in a corresponding document: [NVRAM Variables](#).

## 9.3 Mandatory Variables

**Warning:** These variables may be added by the PlatformNVRAM or Generic subsections of the PlatformInfo section. Using `PlatformInfo` is the recommended way of setting these variables.

The following variables are mandatory for macOS functioning:

- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeatures`  
32-bit `FirmwareFeatures`. Present on all Macs to avoid extra parsing of SMBIOS tables.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeaturesMask`  
32-bit `FirmwareFeaturesMask`. Present on all Macs to avoid extra parsing of SMBIOS tables.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:MLB`  
`BoardSerialNumber`. Present on newer Macs (2013+ at least) to avoid extra parsing of SMBIOS tables, especially in `boot.efi`.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ROM`  
Primary network adapter MAC address or replacement value. Present on newer Macs (2013+ at least) to avoid accessing special memory region, especially in `boot.efi`.

## 9.4 Recommended Variables

The following variables are recommended for faster startup or other improvements:

- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:BridgeOSHardwareModel`  
Bridge OS hardware model variable used to propagate to IODT bridge-model by `EfiBoot`. Read by `hw.target sysctl`, used by `SoftwareUpdateCoreSupport`.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:csr-active-config`  
32-bit System Integrity Protection bitmask. Declared in XNU source code in `csr.h`.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeatures`  
Combined `FirmwareFeatures` and `ExtendedFirmwareFeatures`. Present on newer Macs to avoid extra parsing of SMBIOS tables.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeaturesMask`  
Combined `FirmwareFeaturesMask` and `ExtendedFirmwareFeaturesMask`. Present on newer Macs to avoid extra parsing of SMBIOS tables.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_BID`  
Hardware `BoardProduct` (e.g. `Mac-35C1E88140C3E6CF`). Not present on real Macs, but used to avoid extra parsing of SMBIOS tables, especially in `boot.efi`.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_MLB`  
Hardware `BoardSerialNumber`. Override for `MLB`. Present on newer Macs (2013+ at least).
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_ROM`  
Hardware `ROM`. Override for `ROM`. Present on newer Macs (2013+ at least).
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:SSN`  
Serial number. Present on newer Macs (2013+ at least).
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:prev-lang:kbd`  
ASCII string defining default keyboard layout. Format is `lang-COUNTRY:keyboard`, e.g. `ru-RU:252` for Russian locale and ABC keyboard. Also accepts short forms: `ru:252` or `ru:0` (U.S. keyboard, compatible with 10.9). Full

decoded keyboard list from `AppleKeyboardLayouts-L.dat` can be found here. Using non-latin keyboard on 10.14 will not enable ABC keyboard, unlike previous and subsequent macOS versions, and is thus not recommended in case 10.14 is needed.

- `7C436110-AB2A-4BBB-A880-FE41995C9F82:security-mode`  
ASCII string defining FireWire security mode. Legacy, can be found in `IOFireWireFamily` source code in `IOFireWireController.cpp`. It is recommended not to set this variable, which may speedup system startup. Setting to `full` is equivalent to not setting the variable and `none` disables FireWire security.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:UIScale`  
One-byte data defining `boot.efi` user interface scaling. Should be `01` for normal screens and `02` for HiDPI screens.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:ForceDisplayRotationInEFI`  
32-bit integer defining display rotation. Can be `0` for no rotation or any of 90, 180, 270 for matching rotation in degrees.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:DefaultBackgroundColor`  
Four-byte BGRA data defining `boot.efi` user interface background colour. Standard colours include `BF BF BF 00` (Light Gray) and `00 00 00 00` (Syrah Black). Other colours may be set at user's preference.

## 9.5 Other Variables

The following variables may be useful for certain configurations or troubleshooting:

- `7C436110-AB2A-4BBB-A880-FE41995C9F82:boot-args`  
Kernel arguments, used to pass configuration to Apple kernel and drivers. There are many arguments, which may be found by looking for the use of `PE_parse_boot_argn` function in the kernel or driver code. Some of the known boot arguments include:
  - `acpi_layer=0xFFFFFFFF`
  - `acpi_level=0xFFFFF5F` (implies `ACPI_ALL_COMPONENTS`)
  - `arch=i386` (force kernel architecture to `i386`, see `KernelArch`)
  - `batman=VALUE` (`AppleSmartBatteryManager` debug mask)
  - `batman-nosmc=1` (disable `AppleSmartBatteryManager` SMC interface)
  - `cpus=VALUE` (maximum number of CPUs used)
  - `debug=VALUE` (debug mask)
  - `io=VALUE` (IOKit debug mask)
  - `ioaccel_debug=VALUE` (`IOAccelerator` debug mask)
  - `keepsyms=1` (show panic log debug symbols)
  - `kextlog=VALUE` (kernel extension loading debug mask)
  - `nvram-log=1` (enables `AppleEFINVRAM` logs)
  - `nv_disable=1` (disables NVIDIA GPU acceleration)
  - `nvda_drv=1` (legacy way to enable NVIDIA web driver, removed in 10.12)
  - `npci=0x2000` (legacy, disables `kIOPCIConfiguratorPFM64`)
  - `lapic_dont_panic=1` (disable lapic spurious interrupt panic on AP cores)
  - `panic_on_display_hang=1` (trigger panic on display hang)
  - `panic_on_gpu_hang=1` (trigger panic on GPU hang)
  - `serial=VALUE` (configure serial logging mode) — The following bits are used by XNU:
    - \* `0x01` (`SERIALMODE_OUTPUT`, bit 0) — Enable serial output.
    - \* `0x02` (`SERIALMODE_INPUT`, bit 1) — Enable serial input.
    - \* `0x04` (`SERIALMODE_SYNCRAIN`, bit 2) — Enable serial drain synchronisation.
    - \* `0x08` (`SERIALMODE_BASE_TTY`, bit 3) — Load Base/Recovery/FVUnlock TTY.
    - \* `0x10` (`SERIALMODE_NO_IOLOG`, bit 4) — Prevent IOLogs writing to serial.
  - `slide=VALUE` (manually set KASLR slide)
  - `smcdebug=VALUE` (`AppleSMC` debug mask)
  - `spin_wait_for_gpu=1` (reduces GPU timeout on high load)
  - `-amd_no_dgpu_accel` (alternative to WhateverGreen's `-radvesa` for new GPUs)
  - `-nehalem_error_disable` (disables the `AppleTyMCEDriver`)
  - `-no_compat_check` (disable model checking on 10.7+)
  - `-s` (single mode)
  - `-v` (verbose mode)
  - `-x` (safe mode)

- **http** - Boolean flag, enabled if present.

If specified enable HTTP(S) Boot. Disable PXE Boot unless the **pxe** flag is also present. If neither flag is present, both are enabled by default.

- **https** - Boolean flag, enabled if present.

If enabled, allow only **https://** URIs for HTTP(S) Boot. Additionally has the same behaviour as the **http** flag.

- **ipv4** - Boolean flag, enabled if present.

If specified enable IPv4 for PXE and HTTP(S) Boot. Disable IPV6 unless the **ipv6** flag is also present. If neither flag is present, both are enabled by default.

- **ipv6** - Boolean flag, enabled if present.

If specified enable IPv6 for PXE and HTTP(S) Boot. Disable IPV4 unless the **ipv4** flag is also present. If neither flag is present, both are enabled by default.

- **pxe** - Boolean flag, enabled if present.

If specified enable PXE Boot, and disable HTTP(S) Boot unless the **http** or **https** flags are present. If none of these flags are present, both PXE and HTTP(S) Boot are enabled by default.

- **static4:{MAC\_ADDR}[\VLAN\_ID] [= "{IP},{MASK},{GATEWAY}[, {DNS}] "** - String value.

Specify static IPv4 address for the network interface with the MAC address given by **MAC\_ADDR**. **MAC\_ADDR** must be specified as 12 consecutive hex digits, with no spaces, colons or hyphens separating digit pairs. In some advanced use-cases such as iSCSI, the MAC address length may be some other even number length of hex digits. The required MAC address can be found in the names of the boot options produced by this driver. Note that hyphens separating digit pairs must be removed, as compared to the format displayed in boot option names. It is also possible to specify a VLAN ID to use on the interface, by adding a backslash followed by a 4 digit hex representation of the VLAN ID following the MAC address. The VLAN ID will also be shown in the boot entry name, but note that it must be converted from decimal in the boot entry name to a 4 digit hex number in this option.

Required elements in value are IP address in **IP**, network mask in **MASK** and gateway in **GATEWAY**. Optional is an additional space separated list of one or more DNS servers in **DNS**. **DNS** will be needed if the boot file URI includes a domain name rather than an IP address.

**MAC\_ADDR** is not optional.

If value is omitted, then any static IP for this MAC address (and VLAN ID when present) will be deleted.

- Example 1: **static4:112233445566="192.168.1.20,255.255.255.0,192.168.1.1,8.8.8.8 4.4.4.4"**.
- Example 2: **static4:112233445566\0001="10.0.0.2,255.255.255.0,10.0.0.1"**.

*Note 1:* This option is written to NVRAM and will remain present even if the option is removed from the driver **Arguments**, unless ~~NVRAM is cleared or~~ an alternative value is written or the value deleted, using this option.

*Note 2:* This setting will normally cause a static IP to be assigned during pre-boot, even in vendor-provided network stacks. However, due to a quirk of the design of PXE and HTTP boot, any such static assignment will then be ignored and DHCP used instead, during network boot. The OpenCore network stack (specifically **HttpBootDxe.efi**) is unusual in that it will allow HTTP boot from a static IP address, as long as an HTTP boot URI has also been specified, using the **uri** option for this driver (or e.g. in the OVMF admin screens if using OVMF, or similar options where present in other firmwares). If HTTP boot from static IP is required, then any pre-existing vendor-specific version of **HttpBootDxe.efi** will need to be unloaded (see **UEFI Unload** option) and the OpenCore version used instead.

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. [When set to Max all available resolutions will be listed in lines starting OCC: Mode in the debug log.](#)

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 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.

16. **TextRenderer**

**Type:** plist string

**Failsafe:** **BuiltinGraphics**

**Description:** Chooses renderer for text going through standard console output.

Currently two renderers are supported: **Builtin** and **System**. The **System** renderer uses firmware services for text rendering, however with additional options provided to sanitize the output. The **Builtin** renderer bypasses firmware services and performs text rendering on its own. Each renderer supports a different set of options. It is recommended to use the **Builtin** renderer, as it supports HiDPI mode and uses full screen resolution.

Each renderer provides its own **ConsoleControl** protocol (in the case of **SystemGeneric** only, this passes some operations through to the system **ConsoleControl** protocol, if one exists).

Valid values of this option are combinations of the renderer to use and the **ConsoleControl** mode to set on the underlying system **ConsoleControl** protocol before starting. To control the initial mode of the provided **ConsoleControl** protocol once started, use the **InitialMode** option.

- **BuiltinGraphics** — Switch to **Graphics** mode then use **Builtin** renderer with custom **ConsoleControl**.
- **BuiltinText** — Switch to **Text** mode then use **Builtin** renderer with custom **ConsoleControl**.
- **SystemGraphics** — Switch to **Graphics** mode then use **System** renderer with custom **ConsoleControl**.
- **SystemText** — Switch to **Text** mode then use **System** renderer with custom **ConsoleControl**.
- **SystemGeneric** — Use **System** renderer with custom a **ConsoleControl** protocol which passes its mode set and get operations through to system **ConsoleControl** when it exists.

The use of **BuiltinGraphics** is straightforward. For most platforms, it is necessary to enable **ProvideConsoleGop** and set **Resolution** to **Max**. The **BuiltinText** variant is an alternative to **BuiltinGraphics** for some very old