



OpenCore

Reference Manual (0.5.~~0~~.1)

[2019.09.25]

4.4 Block Properties

1. All
Type: plist boolean
Failsafe: false
Description: If set to **true**, all ACPI tables matching the condition will be dropped. Otherwise only first matched table.
2. Comment
Type: plist string
Failsafe: Empty string
Description: Arbitrary ASCII string used to provide human readable reference for the entry. It is implementation defined whether this value is used.
3. Enabled
Type: plist boolean
Failsafe: false
Description: This ACPI table will not be removed unless set to **true**.
4. OemTableId
Type: plist data, 8 bytes
Failsafe: All zero
Description: Match table OEM ID to be equal to this value unless all zero.
5. TableLength
Type: plist integer
Failsafe: 0
Description: Match table size to be equal to this value unless 0.
6. TableSignature
Type: plist data, 4 bytes
Failsafe: All zero
Description: Match table signature to be equal to this value unless all zero.

Note: Make sure not to specify table signature when the sequence needs to be replaced in multiple places. Especially when performing different kinds of renames.

4.5 Patch Properties

1. Comment
Type: plist string
Failsafe: Empty string
Description: Arbitrary ASCII string used to provide human readable reference for the entry. It is implementation defined whether this value is used.
2. Count
Type: plist integer
Failsafe: 0
Description: Number of patch occurrences to apply. 0 applies the patch to all occurrences found.
3. Enabled
Type: plist boolean
Failsafe: false
Description: This ACPI patch will not be used unless set to **true**.
4. Find
Type: plist data
Failsafe: Empty data
Description: Data to find. Must equal to **Replace** in size.
5. Limit
Type: plist integer

5 Booter

5.1 Introduction

This section allows to apply different kinds of UEFI modifications on Apple bootloader (`boot.efi`). The modifications currently provide various patches and environment alterations for different firmwares. Some of these features were originally implemented as a part of `AptioMemoryFix.efi`, which is no longer maintained. See [Tips and Tricks](#) section for migration steps.

If you are using this for the first time on a customised firmware, there is a list of checks to do first. Prior to starting please ensure that you have:

- Most up-to-date UEFI firmware (check your motherboard vendor website).
- `Fast Boot` and `Hardware Fast Boot` disabled in firmware settings if present.
- `Above 4G Decoding` or similar enabled in firmware settings if present. Note, that on some motherboards (notably ASUS WS-X299-PRO) this option causes adverse effects, and must be disabled. While no other motherboards with the same issue are known, consider this option to be first to check if you have erratic boot failures.
- `DisableIoMapper` quirk enabled, or `VT-d` disabled in firmware settings if present, or `ACPI DMAR` table dropped.
- No ‘slide’ boot argument present in NVRAM or anywhere else. It is not necessary unless you cannot boot at all or see `No slide values are usable! Use custom slide!` message in the log.
- `CFG Lock` (MSR 0xE2 write protection) disabled in firmware settings if present. Consider patching it if you have enough skills and no option is available. See [VerifyMsrE2](#) notes for more details.
- `CSM` (Compatibility Support Module) disabled in firmware settings if present. You may need to flash `GOP ROM` on NVIDIA 6xx/AMD 2xx or older. Use `GopUpdate` or `AMD UEFI GOP MAKER` in case you are not sure how.
- `EHCI/XHCI Hand-off` enabled in firmware settings **only** if boot stalls unless USB devices are disconnected.
- `VT-x`, `Hyper Threading`, `Execute Disable Bit` enabled in firmware settings if present.
- While it may not be required, sometimes you have to disable `Thunderbolt support`, `Intel SGX`, and `Intel Platform Trust` in firmware settings present.

When debugging sleep issues you may want to (temporarily) disable `Power Nap` and automatic power off, which appear to sometimes cause wake to black screen or boot loop issues on older platforms. The particular issues may vary, but in general you should check `ACPI` tables first. Here is an example of a bug found in some Z68 motherboards. To turn `Power Nap` and the others off run the following commands in Terminal:

```
sudo pmset autopoweroff 0
sudo pmset powernap 0
sudo pmset standby 0
```

Note: ~~these~~ [These](#) settings may reset at hardware change and in certain other circumstances. To view their current state use `pmset -g` command in Terminal.

5.2 Properties

1. Quirks

Type: `plist dict`

Description: Apply individual booter quirks described in [Quirks Properties](#) section below.

5.3 Quirks Properties

1. `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 many firmwares using `SMM` backing for select services like variable storage. `SMM` may try to access physical addresses, but they get moved by `boot.efi`.

Note: Most but Apple and VMware firmwares need this quirk.

Failsafe: Empty string

Description: Next executable path relative to bundle (e.g. Contents/MacOS/Lilu).

5. MatchKernelMaxKernel

Type: plist string

Failsafe: Empty string

Description: Adds kernel driver on ~~selected macOS version only. The selection happens based on prefix match with the kernel version, i.e. specified macOS version or older.~~

Kernel version interpretation is implemented as follows:

$$\begin{aligned} ParseDarwinVersion(\kappa, \lambda, \mu) = & (\lfloor \frac{\kappa}{10} \rfloor \cdot 10 + \kappa - \lfloor \frac{\kappa}{10} \rfloor \cdot 10) \cdot 10000 && \text{Where } \kappa \in (0, 99) \text{ is kernel version major} \\ & + (\lfloor \frac{\lambda}{10} \rfloor \cdot 10 + \lambda - \lfloor \frac{\lambda}{10} \rfloor \cdot 10) \cdot 100 && \text{Where } \lambda \in (0, 99) \text{ is kernel version minor} \\ & + (\lfloor \frac{\mu}{10} \rfloor \cdot 10 + \mu - \lfloor \frac{\mu}{10} \rfloor \cdot 10) && \text{Where } \mu \in (0, 99) \text{ is kernel version patch} \end{aligned}$$

Kernel version comparison is implemented as follows:

$$\begin{aligned} \alpha = & \begin{cases} ParseDarwinVersion(\text{MinKernel}), & \text{If MinKernel is valid} \\ 0 & \text{Otherwise} \end{cases} \\ \beta = & \begin{cases} ParseDarwinVersion(\text{MaxKernel}), & \text{If MaxKernel is valid} \\ \infty & \text{Otherwise} \end{cases} \\ \gamma = & \begin{cases} ParseDarwinVersion(\text{FindDarwinVersion}()), & \text{If valid "Darwin Kernel Version" is found} \\ \infty & \text{Otherwise} \end{cases} \\ f(\alpha, \beta, \gamma) = & \alpha \leq \gamma \leq \beta \end{aligned}$$

Here *ParseDarwinVersion* argument is assumed to be 3 integers obtained by splitting Darwin kernel version string from left to right by the . symbol. *FindDarwinVersion* function looks up Darwin kernel version by locating "Darwin Kernel Version $\kappa.\lambda.\mu$ " string in the kernel image.

6. ~~16.7.0~~MinKernel~~will match macOS 10.12.6 and 16. will match any macOS 10.12.x version~~

Type: plist string

Failsafe: Empty string

Description: Adds kernel driver on specified macOS version or newer.

Note: Refer to Add MaxKernel description for matching logic.

7. PlistPath

Type: plist string

Failsafe: Empty string

Description: Next Info.plist path relative to bundle (e.g. Contents/Info.plist).

7.4 Block Properties

1. Comment

Type: plist string

Failsafe: Empty string

Description: Arbitrary ASCII string used to provide human readable reference for the entry. It is implementation defined whether this value is used.

2. Enabled

Type: plist boolean

Failsafe: false

Description: This kernel driver will not be blocked unless set to true.

3. Identifier
Type: plist string
Failsafe: Empty string
Description: Kext bundle identifier (e.g. `com.apple.driver.AppleTyMCEDriver`).
4. ~~MatchKernel~~MaxKernel
Type: plist string
Failsafe: Empty string
Description: Blocks kernel driver on ~~selected macOS version only. The selection happens based on prefix match with the kernel version, i. e. specified macOS version or older.~~
Note: Refer to Add MaxKernel description for matching logic.
5. ~~16.7.0MinKernel~~will match macOS 10.12.6 and 16. will match any macOS 10.12.x version
Type: plist string
Failsafe: Empty string
Description: Blocks kernel driver on specified macOS version or newer.
Note: Refer to Add MaxKernel description for matching logic.

7.5 Emulate Properties

1. Cpuid1Data
Type: plist data, 16 bytes
Failsafe: All zero
Description: Sequence of EAX, EBX, ECX, EDX values in Little Endian order to replace CPUID (1) call in XNU kernel. Normally it is only the value of EAX that needs to be taken care of, which represents the exact CPUID. And the remainders are to be left as zeroes. For instance, A9 06 03 00 stands for CPUID 0x0306A9 (Ivy Bridge). A good example can be found at [acidanthera/bugtracker#365](#). (See Special NOTES for Haswell+ low-end)
2. Cpuid1Mask
Type: plist data, 16 bytes
Failsafe: All zero
Description: Bit mask of active bits in Cpuid1Data. When each Cpuid1Mask bit is set to 0, the original CPU bit is used, otherwise set bits take the value of Cpuid1Data.

7.6 Patch Properties

1. Base
Type: plist string
Failsafe: Empty string
Description: Selects symbol-matched base for patch lookup (or immediate replacement) by obtaining the address of provided symbol name. Can be set to empty string to be ignored.
2. Comment
Type: plist string
Failsafe: Empty string
Description: Arbitrary ASCII string used to provide human readable reference for the entry. It is implementation defined whether this value is used.
3. Count
Type: plist integer
Failsafe: 0
Description: Number of patch occurrences to apply. 0 applies the patch to all occurrences found.
4. Enabled
Type: plist boolean
Failsafe: false
Description: This kernel patch will not be used unless set to `true`.
5. Find
Type: plist data

Failsafe: Empty data

Description: Data to find. Can be set to empty for immediate replacement at Base. Must equal to **Replace** in size otherwise.

6. Identifier

Type: plist string

Failsafe: Empty string

Description: Kext bundle identifier (e.g. `com.apple.driver.AppleHDA`) or `kernel` for kernel patch.

7. Limit

Type: plist integer

Failsafe: 0

Description: Maximum number of bytes to search for. Can be set to 0 to look through the whole kext or kernel.

8. Mask

Type: plist data

Failsafe: Empty data

Description: Data bitwise mask used during find comparison. Allows fuzzy search by ignoring not masked (set to zero) bits. Can be set to empty data to be ignored. Must equal to **Replace** in size otherwise.

9. ~~MatchKernel~~MaxKernel

Type: plist string

Failsafe: Empty string

Description: ~~Adds kernel driver to selected macOS version only. The selection happens based on prefix match with the kernel version, i. e.~~ Patches data on specified macOS version or older.

Note: Refer to Add MaxKernel description for matching logic.

10. ~~16.7.0MinKernel~~will match macOS 10.12.6 and 16. will match any macOS 10.12.x version

Type: plist string

Failsafe: Empty string

Description: Patches data on specified macOS version or newer.

Note: Refer to Add MaxKernel description for matching logic.

11. Replace

Type: plist data

Failsafe: Empty data

Description: Replacement data of one or more bytes.

12. ReplaceMask

Type: plist data

Failsafe: Empty data

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

13. Skip

Type: plist integer

Failsafe: 0

Description: Number of found occurrences to be skipped before replacement is done.

7.7 Quirks Properties

1. AppleCpuPmCfgLock

Type: plist boolean

Failsafe: false

Description: Disables `PKG_CST_CONFIG_CONTROL (0xE2)` MSR modification in `AppleIntelCPUPowerManagement.kext`, commonly causing early kernel panic, when it is locked from writing.

Note: This option should avoided whenever possible. Modern firmwares provide `CFG Lock` setting, disabling which is much cleaner. More details about the issue can be found in `VerifyMsrE2` notes.

2. AppleXcpmCfgLock

Type: plist boolean

8 Misc

8.1 Introduction

This section contains miscellaneous configuration entries for OpenCore behaviour that does not go to any other sections

8.2 Properties

1. Boot

Type: plist dict

Description: Apply boot configuration described in Boot Properties section below.

2. BlessOverride

Type: plist array

Description: Add custom scanning paths through bless model.

Designed to be filled with `plist string` entries containing absolute UEFI paths to customised bootloaders, for example, `\EFI\Microsoft\bootmgfw.efi` for Microsoft bootloader. This allows unusual boot paths to be automatically discovered by the boot picker. Designwise they are equivalent to predefined blessed path, such as `\System\Library\CoreServices\boot.efi`, but unlike predefined bless paths they have highest priority.

3. Debug

Type: plist dict

Description: Apply debug configuration described in Debug Properties section below.

4. Entries

Type: plist array

Description: Add boot entries to boot picker.

Designed to be filled with `plist dict` values, describing each load entry. See Entry Properties section below.

5. Security

Type: plist dict


Description: Apply security configuration described in Security Properties section below.

6. Tools

Type: plist array

Description: Add tool entries to boot picker.

Designed to be filled with `plist dict` values, describing each load entry. See Entry Properties section below.

Note: Select tools, for example, UEFI Shell  are very dangerous and **MUST NOT** appear in production configurations, especially in vaulted ones and protected with secure boot, as they may be used to easily bypass secure boot chain.

8.3 Boot Properties

1. ConsoleMode

Type: plist string

Failsafe: Empty string

Description: Sets console output mode as specified with the WxH (e.g. 80x24) formatted string. Set to empty string not to change console mode. Set to `Max` to try to use largest available console mode.

Note: This field is best to be left empty on most firmwares.

2. ConsoleBehaviourOs

Type: plist string

Failsafe: Empty string

Description: Set console control behaviour upon operating system load.

Console control is a legacy protocol used for switching between text and graphics screen output. Some firmwares do not provide it, yet select operating systems require its presence, which is what `ConsoleControl` UEFI protocol is for.

- 0x00040000 (bit 18) — OC_SCAN_ALLOW_DEVICE_SCSI, allow scanning SCSI devices.
- 0x00080000 (bit 19) — OC_SCAN_ALLOW_DEVICE_NVME, allow scanning NVMe devices.
- 0x00100000 (bit 20) — OC_SCAN_ALLOW_DEVICE_ATAPI, allow scanning CD/DVD devices.
- 0x00200000 (bit 21) — OC_SCAN_ALLOW_DEVICE_USB, allow scanning USB devices.
- 0x00400000 (bit 22) — OC_SCAN_ALLOW_DEVICE_FIREWIRE, allow scanning FireWire devices.
- 0x00800000 (bit 23) — OC_SCAN_ALLOW_DEVICE_SDCARD, allow scanning card reader devices.

Note: Given the above description, 0xF0103 value is expected to allow scanning of SATA, SAS, SCSI, and NVMe devices with APFS file system, and prevent scanning of any devices with HFS or FAT32 file systems in addition to not scanning APFS file systems on USB, CD, USB, and FireWire drives. The combination reads as:

- OC_SCAN_FILE_SYSTEM_LOCK
- OC_SCAN_DEVICE_LOCK
- OC_SCAN_ALLOW_FS_APFS
- OC_SCAN_ALLOW_DEVICE_SATA
- OC_SCAN_ALLOW_DEVICE_SASEX
- OC_SCAN_ALLOW_DEVICE_SCSI
- OC_SCAN_ALLOW_DEVICE_NVME

8.6 Entry Properties

1. Arguments

Type: plist string

Failsafe: Empty string

Description: Arbitrary ASCII string used as boot arguments (load options) of the specified entry.

2. Comment

Type: plist string

Failsafe: Empty string

Description: Arbitrary ASCII string used to provide human readable reference for the entry. It is implementation defined whether this value is used.

3. Enabled

Type: plist boolean

Failsafe: false

Description: This entry will not be listed unless set to **true**.

4. Name

Type: plist string

Failsafe: Empty string

Description: Human readable entry name displayed in boot picker.

5. Path

Type: plist string

Failsafe: Empty string

Description: Entry location depending on entry type.

- **Entries** specify external boot options, and therefore take device paths in **Path** key. These values are not checked, thus be extremely careful. Example: `PciRoot(0x0)/Pci(0x1,0x1)/.../EFI\COOL.EFI`
- **Tools** specify internal boot options, which are part of bootloader vault, and therefore take file paths relative to OC/Tools directory. Example: `CleanNvramShell.efi`.


```

Sector size (logical): 512 bytes
Disk identifier (GUID): DEC57EB1-B3B5-49B2-95F5-3B8C4D3E4E12
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 419430366
Partitions will be aligned on 2048-sector boundaries
Total free space is 4029 sectors (2.0 MiB)

```

Number	Start (sector)	End (sector)	Size	Code	Name
1	2048	1023999	499.0 MiB	2700	Basic data partition
2	1024000	1226751	99.0 MiB	EF00	EFI system partition
3	1226752	1259519	16.0 MiB	0C01	Microsoft reserved ...
4	1259520	419428351	199.4 GiB	0700	Basic data partition

```

Command (? for help): c
Partition number (1-4): 4
Enter name: BOOTCAMP

```

```

Command (? for help): w

```

```

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING PARTITIONS!!

```

```

Do you want to proceed? (Y/N): Y
OK; writing new GUID partition table (GPT) to \\.\\physicaldrive0.
Disk synchronization succeeded! The computer should now use the new partition table.
The operation has completed successfully.

```

Listing 3: Relabeling Windows volume

How to choose Windows BOOTCAMP with custom NTFS drivers?

Third-party drivers providing NTFS support, such as NTFS-3G, Paragon NTFS, Tuxera NTFS or Seagate Paragon Driver break certain macOS functionality, including Startup Disk preference pane normally used for operating system selection. While the recommended option remains not to use such drivers as they commonly corrupt the filesystem, and prefer the driver bundled with macOS with optional write support (command or GUI), there still exist vendor-specific workarounds for their products: Tuxera, Paragon, etc.

12.2 Debugging

Similar to other projects working with hardware OpenCore supports auditing and debugging. The use of NOOPT or DEBUG build modes instead of RELEASE can produce a lot more debug output. With NOOPT source level debugging with GDB or IDA Pro is also available. For GDB check OcSupport Debug page. For IDA Pro you will need IDA Pro 7.3 or newer, refer to Debugging the XNU Kernel with IDA Pro for more details.

To obtain the log during boot you can make the use of serial port debugging. Serial port debugging is enabled in Target, e.g. 0xB for onscreen with serial. OpenCore uses 115200 baud rate, 8 data bits, no parity, and 1 stop bit. For macOS your best choice are CP2102-based UART devices. Connect motherboard TX to USB UART ~~GND~~RX, and motherboard GND to USB UART ~~RX~~GND. Use screen utility to get the output, or download GUI software, such as CoolTerm.

Note: On several motherboards (and possibly USB UART dongles) PIN naming may be incorrect. It is very common to have GND swapped with RX, thus you have to connect motherboard “TX” to USB UART GND, and motherboard “GND” to USB UART RX.

Remember to enable COM port in firmware settings, and never use USB cables longer than 1 meter to avoid output corruption. To additionally enable XNU kernel serial output you will need debug=0x8 boot argument.

12.3 Tips and Tricks

1. How to debug boot failure?