OpenCore

Reference Manual (0.6.1.2)

[2020.09.16]

Failsafe: All zero
Description: Match table signature to be equal to this value unless all zero.

In the majority of the cases ACPI patches are not useful and harmful:

- Avoid renaming devices with ACPI patches. This may fail or perform improper renaming of unrelated devices (e.g. `EC` and `EC0`), be unnecessary, or even fail to rename devices in select tables. For ACPI consistency it is much safer to rename devices at I/O Registry level, as done by WhateverGreen.
- ~~Avoid~~ Try to avoid patching `_OSI` to support a higher level of feature sets ~~unless absolutely required~~whenever possible. Commonly this enables a number of hacks on APTIO firmwares, which result in the need to add more patches. Modern firmwares generally do not need it at all, and those that do are fine with much smaller patches. However, laptop vendors usually rely on this method to determine the availability of functions like modern I2C input support, thermal adjustment and custom feature additions.
- Avoid patching embedded controller event `_Qxx` just for enabling brightness keys. The conventional process to find these keys usually involves massive modification on DSDT and SSDTs and the debug kext is not stable on newer systems. Please switch to built-in brightness key discovery of VoodooPS2 instead.
- Try to avoid hacky changes like renaming `_PRW` or `_DSM` whenever possible.

Several cases, where patching actually does make sense, include:

- Refreshing `HPET` (or another device) method header to avoid compatibility checks by `_OSI` on legacy hardware. `_STA` method with `if ((OSFL () == Zero)) { If (HPTE) ... Return (Zero)` content may be forced to always return 0xF by replacing `A0 10 93 4F 53 46 4C 00` with `A4 0A 0F A3 A3 A3 A3 A3`.
- To provide custom method implementation with in an SSDT, for instance, to ~~report functional key presses on a laptop~~inject shutdown fix on certain computers, the original method can be replaced with a dummy name by patching `_~~Q11~~PTS` with `~~XQ11~~ZPTS` and adding a callback to original method.

Tianocore AcpiAml.h source file may help understanding ACPI opcodes.

Note: Patches of different `Find` and `Replace` lengths are unsupported as they may corrupt ACPI tables and make you system unstable due to area relocation. If you need such changes you may utilise "proxy" patching or `NOP` the remaining area.

4.6 Quirks Properties

1. `FadtEnableReset`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Provide reset register and flag in FADT table to enable reboot and shutdown.

   Mainly required on legacy hardware and few laptops. Can also fix power-button shortcuts. Not recommended unless required.

2. `NormalizeHeaders`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Cleanup ACPI header fields to workaround macOS ACPI implementation bug causing boot crashes. Reference: Debugging AppleACPIPlatform on 10.13 by Alex James aka theracermaster. The issue is fixed in macOS Mojave (10.14).

3. `RebaseRegions`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Attempt to heuristically relocate ACPI memory regions. Not recommended.

   ACPI tables are often generated dynamically by underlying firmware implementation. Among the position-independent code, ACPI tables may contain physical addresses of MMIO areas used for device configuration, usually grouped in regions (e.g. `OperationRegion`). Changing firmware settings or hardware configuration, upgrading or patching the firmware inevitably leads to changes in dynamically generated ACPI code, which sometimes lead to the shift of the addresses in aforementioned `OperationRegion` constructions.

10. `MaxKernel`
    Type: `plist string`
    Failsafe: Empty string
    Description: Patches data on specified macOS version or older.

    Note: Refer to `Add MaxKernel` description for matching logic.

11. `MinKernel`
    Type: `plist string`
    Failsafe: Empty string
    Description: Patches data on specified macOS version or newer.

    Note: Refer to `Add MaxKernel` description for matching logic.

12. `Replace`
    Type: `plist data`
    Failsafe: Empty data
    Description: Replacement data of one or more bytes.

13. `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.

14. `Skip`
    Type: `plist integer`
    Failsafe: `0`
    Description: Number of found occurrences to be skipped before replacement is done.

7.8 Quirks Properties

1. `AppleCpuPmCfgLock`
   Type: `plist boolean`
   Failsafe: `false`
   Requirement: ~~10.6 (64-bit)~~10.4
   Description: Disables `PKG_CST_CONFIG_CONTROL` (`0xE2`) MSR modification in AppleIntelCPUPowerManagement.kext, commonly causing early kernel panic, when it is locked from writing.

   Certain firmwares lock `PKG_CST_CONFIG_CONTROL` MSR register. To check its state one can use bundled `VerifyMsrE2` tool. Select firmwares have this register locked on some cores only.

   As modern firmwares provide `CFG Lock` setting, which allows configuring `PKG_CST_CONFIG_CONTROL` MSR register lock, this option should be avoided whenever possible. For several APTIO firmwares not displaying `CFG Lock` setting in the GUI it is possible to access the option directly:

   (a) Download UEFITool and IFR-Extractor.
   (b) Open your firmware image in UEFITool and find `CFG Lock` unicode string. If it is not present, your firmware may not have this option and you should stop.
   (c) Extract the `Setup.bin` PE32 Image Section (the one UEFITool found) through `Extract Body` menu option.
   (d) Run IFR-Extractor on the extracted file (e.g. `./ifrextract Setup.bin Setup.txt`).
   (e) Find `CFG Lock, VarStoreInfo (VarOffset/VarName):` in `Setup.txt` and remember the offset right after it (e.g. `0x123`).
   (f) Download and run Modified GRUB Shell compiled by brainsucker or use a newer version by datasone.
   (g) Enter `setup_var 0x123 0x00` command, where `0x123` should be replaced by your actual offset, and reboot.

   Warning: Variable offsets are unique not only to each motherboard but even to its firmware version. Never ever try to use an offset without checking.

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. See `AppleCpuPmCfgLock` description for more details.

3. `AppleXcpmExtraMsrs`
   Type: `plist boolean`
   Failsafe: `false`
   Requirement: 10.8 (not required for older)
   Description: Disables multiple MSR access critical for select CPUs, which have no native XCPM support.

   This is normally used in conjunction with `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. In general only certain Xeon models benefit from the patch.

5. `CustomSMBIOSGuid`
   Type: `plist boolean`
   Failsafe: `false`
   Requirement: ~~10.6 (64-bit)~~10.4
   Description: Performs GUID patching for `UpdateSMBIOSMode Custom` mode. Usually relevant for Dell laptops.

6. `DisableIoMapper`
   Type: `plist boolean`
   Failsafe: `false`
   Requirement: 10.8 (not required for older)
   Description: Disables `IOMapper` support in XNU (VT-d), which may conflict with the firmware implementation.

   Note: This option is a preferred alternative to deleting `DMAR` ACPI table and disabling VT-d in firmware preferences, which does not break VT-d support in other systems in case they need it.

7. `DisableLinkeditJettison`
   Type: `plist boolean`
   Failsafe: `false`
   Requirement: 11.0
   Description: Disables `__LINKEDIT` jettison code.

   This option lets `Lilu.kext` and possibly some others function in macOS Big Sur with best performance without `keepsyms=1` boot argument.

8. `DisableRtcChecksum`
   Type: `plist boolean`
   Failsafe: `false`
   Requirement: ~~10.6 (64-bit)~~10.4
   Description: Disables primary checksum (`0x58-0x59`) writing in AppleRTC.

   Note 1: This option will not protect other areas from being overwritten, see RTCMemoryFixup kernel extension if this is desired.

   Note 2: This option will not protect areas from being overwritten at firmware stage (e.g. macOS bootloader), see ~~AppleRtc~~AppleRtcRam protocol description if this is desired.

9. `DummyPowerManagement`
   Type: `plist boolean`
   Failsafe: `false`
   Requirement: ~~10.6 (64-bit)~~10.4
   Description: Disables `AppleIntelCpuPowerManagement`.

   Note: This option is a preferred alternative to `NullCpuPowerManagement.kext` for CPUs without native power management driver in macOS.

10. `ExternalDiskIcons`
    Type: `plist boolean`
    Failsafe: `false`
    Requirement: ~~10.6 (64-bit)~~10.4
    Description: Apply icon type patches to AppleAHCIPort.kext to force internal disk icons for all AHCI disks.

    Note: This option should be avoided whenever possible. Modern firmwares usually have compatible AHCI controllers.

11. `IncreasePciBarSize`
    Type: `plist boolean`
    Failsafe: `false`
    Requirement: 10.10
    Description: Increases 32-bit PCI bar size in IOPCIFamily from 1 to 4 GBs.

    Note: This option should be avoided whenever possible. In general the necessity of this option means misconfigured or broken firmware.

12. `LapicKernelPanic`
    Type: `plist boolean`
    Failsafe: `false`
    Requirement: 10.6 (64-bit)
    Description: Disables kernel panic on LAPIC interrupts.

13. `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.

14. `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 sometimes fails to wake up. For debug kernels `setpowerstate_panic=0` boot argument should be used, which is otherwise equivalent to this quirk.

15. `ThirdPartyDrives`
    Type: `plist boolean`
    Failsafe: `false`
    Requirement: 10.6 (~~64-bit,~~ not required for older)
    Description: Apply vendor patches to IOAHCIBlockStorage.kext to enable native features for third-party drives, such as TRIM on SSDs or hibernation support on 10.15 and newer.

    Note: This option may be avoided on user preference. NVMe SSDs are compatible without the change. For AHCI SSDs on modern macOS version there is a dedicated built-in utility called `trimforce`. Starting from 10.15 this utility creates `EnableTRIM` variable in `APPLE_BOOT_VARIABLE_GUID` namespace with `01 00 00 00` value.

16. `XhciPortLimit`

Type: `plist boolean`
Failsafe: `false`
Requirement: 10.11 (not required for older)
Description: Patch various kexts (AppleUSBXHCI.kext, AppleUSBXHCIPCI.kext, IOUSBHostFamily.kext) to remove USB port count limit of 15 ports.

Note: This option should be avoided whenever possible. USB port limit is imposed by the amount of used bits in locationID format and there is no possible way to workaround this without heavy OS modification. The only valid solution is to limit the amount of used ports to 15 (discarding some). More details can be found on AppleLife.ru.

7.9 Scheme Properties

These properties are particularly relevant for older macOS operating systems. For more details on how to install and troubleshoot such macOS installation refer to Legacy Apple OS.

1. `FuzzyMatch`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Use `kernelcache` with different checksums when available.

   On macOS 10.6 and earlier `kernelcache` filename has a checksum, which essentially is `adler32` from SMBIOS product name and EfiBoot device path. On certain firmwares EfiBoot device path differs between UEFI and macOS due to ACPI or hardware specifics, rendering `kernelcache` checksum as always different.

   This setting allows matching the latest `kernelcache` with a suitable architecture when the `kernelcache` without suffix is unavailable, improving macOS 10.6 boot performance on several platforms.

2. `KernelArch`
   Type: `plist string`
   Failsafe: `Auto`
   Description: Prefer specified kernel architecture (`Auto`, `i386`, `i386-user32`, `x86_64`) when available.

   On macOS 10.7 and earlier XNU kernel can boot with architectures different from the usual `x86_64`. This setting will use the specified architecture to boot macOS when it is supported by the macOS and the configuration:

   - `Auto` — Choose the preferred architecture automatically.
   - `i386` — Use `i386` (32-bit) kernel when available.
   - `i386-user32` — Use `i386` (32-bit) kernel when available and force the use of 32-bit userspace on 64-bit capable processors. On macOS 64-bit capable processors are assumed to support `SSSE3`. This is not the case for older 64-bit capable Pentium processors, which cause some applications to crash on macOS 10.6. The behaviour corresponds to `-legacy` kernel boot argument.
   - `x86_64` — Use `x86_64` (64-bit) kernel when available.

   Below is the algorithm determining the kernel architecture.

   (a) `arch` argument in image arguments (e.g. when launched via UEFI Shell) or in `boot-args` variable overrides any compatibility checks and forces the specified architecture, completing this algorithm.
   (b) OpenCore build architecture restricts capabilities to `i386` and `i386-user32` mode for the 32-bit firmware variant.
   (c) Determined EfiBoot version restricts architecture choice:
   - 10.4-10.5 — `i386` or `i386-user32`
   - 10.6 ~~10.7~~ — `i386`, `i386-user32`, or `x86_64`
   - 10.7 — `i386` or `x86_64`
   - 10.8 or newer — `x86_64`

   (d) If `KernelArch` is set to `Auto` and `SSSE3` is not supported by the CPU, capabilities are restricted to `i386-user32` if supported by EfiBoot.
   (e) Board identifier (from SMBIOS) based on EfiBoot version disables `x86_64` support on an unsupported model if any `i386` variant is supported. `Auto` is not consulted here as the list is not overridable in EfiBoot.
   (f) `KernelArch` restricts the support to the explicitly specified architecture (when not set to `Auto`) if the architecture remains present in the capabilities.
   (g) The best supported architecture is chosen in this order: `x86_64`, `i386`, `i386-user32`.

Unlike macOS 10.7, where select boards identifiers are treated as the `i386` only machines, and macOS 10.5 or earlier, where `x86_64` is not supported by the macOS kernel, macOS 10.6 is very special. The architecture choice on macOS 10.6 depends on many factors including not only the board identifier, but also macOS product type (client vs server), macOS point release, and RAM amount. The detection of them all is complicated and not practical, because several point releases had genuine bugs and failed to properly perform the server detection in the first place. For this reason OpenCore on macOS 10.6 will fallback to `x86_64` architecture whenever it is supported by the board at all, just like on macOS 10.7. As a reference here is the 64-bit Mac model compatibility corresponding to actual EfiBoot behaviour on macOS 10.6.8 and 10.7.5.

Model	10.6 (minimal)	10.6 (client)	10.6 (server)	10.7 (any)
Macmini	4,x (Mid 2010)	5,x (Mid 2011)	4,x (Mid 2010)	3,x (Early 2009)
MacBook	Unsupported	Unsupported	Unsupported	5,x (2009/09)
MacBookAir	Unsupported	Unsupported	Unsupported	2,x (Late 2008)
MacBookPro	4,x (Early 2008)	8,x (Early 2011)	8,x (Early 2011)	3,x (Mid 2007)
iMac	8,x (Early 2008)	12,x (Mid 2011)	12,x (Mid 2011)	7,x (Mid 2007)
MacPro	3,x (Early 2008)	5,x (Mid 2010)	3,x (Early 2008)	3,x (Early 2008)
Xserve	2,x (Early 2008)	2,x (Early 2008)	2,x (Early 2008)	2,x (Early 2008)

Note: `3+2` and `6+4` hotkeys to choose the preferred architecture are unsupported due to being handled by EfiBoot and thus being hard to properly detect.

3. `KernelCache`
   Type: `plist string`
   Failsafe: `Auto`
   Description: Prefer specified kernel cache type (`Auto`, `Cacheless`, `Mkext`, `Prelinked`) when available.

   Different variants of macOS support different kernel caching variants designed to improve boot performance. This setting allows to prevent using faster kernel caching variants if slower variants are available for debugging and stability reasons. I.e. by specifying `Mkext` one will disable `Prelinked` for e.g. 10.6 but not 10.7.

   The list of available kernel caching types and its current support in OpenCore is listed below.

macOS	i386 NC	i386 MK	i386 PK	x86_64 NC	x86_64 MK	x86_64 PK	x86_64 KC
10.4	~~NO~~ YES	~~NO~~ YES (V1)	NO (V1)	—	—	—	—
10.5	~~NO~~ YES	~~NO~~ YES (V1)	NO (V1)	—	—	—	—
10.6	~~NO~~ YES	~~NO~~ YES (V2)	NO (V2)	YES	YES (V2)	YES (V2)	—
10.7	~~NO~~ YES	—	NO (V3)	YES	—	YES (V3)	—
10.8-10.9	—	—	—	YES	—	YES (V3)	—
10.10-10.15	—	—	—	—	—	YES (V3)	—
11.0+	—	—	—	—	—	YES (V3)	YES

- GSTT — GoptStop
- HDA — AudioDxe
- KKT — KeyTester
- MMDD — MmapDump
- OCPAVP — PavpProvision
- OCRST — ResetSystem
- OCUI — OpenCanopy
- OC — OpenCore main
- VMOPT — VerifyMemOpt

Libraries:

- AAPL — OcDebugLogLib, Apple EfiBoot logging
- OCABC — OcAfterBootCompatLib
- OCAE — OcAppleEventLib
- OCAK — OcAppleKernelLib
- OCAU — OcAudioLib
- OCAV — OcAppleImageVerificationLib
- OCA —- OcAcpiLib
- OCBP — OcAppleBootPolicyLib
- OCB — OcBootManagementLib
- OCCL — OcAppleChunkListLib
- OCCPU — OcCpuLib
- OCC — OcConsoleLib
- OCDC — OcDriverConnectionLib
- OCDH — OcDataHubLib
- OCDI — OcAppleDiskImageLib
- ~~OCFSQ — OcFileLib, UnblockFs quirk~~
- OCFS — OcFileLib
- OCFV — OcFirmwareVolumeLib
- OCHS — OcHashServicesLib
- OCI4 — OcAppleImg4Lib
- OCIC — OcImageConversionLib
- OCII — OcInputLib
- OCJS — OcApfsLib
- OCKM — OcAppleKeyMapLib
- OCL — OcDebugLogLib
- OCMCO — OcMachoLib
- OCME — OcHeciLib
- OCMM — OcMemoryLib
- 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

- OC_SCAN_ALLOW_DEVICE_SATA
- OC_SCAN_ALLOW_DEVICE_SASEX
- OC_SCAN_ALLOW_DEVICE_SCSI
- OC_SCAN_ALLOW_DEVICE_NVME

14. SecureBootModel
Type: plist string
Failsafe: Default
Description: Apple Secure Boot hardware model.

Sets Apple Secure Boot hardware model and policy. Specifying this value defines which operating systems will be bootable. Operating systems shipped before the specified model was released will not boot. Valid values:

- Default — Recent available model, currently set to j137.
- Disabled — No model, Secure Boot will be disabled.
- j137 — iMacPro1,1 (December 2017) minimum macOS 10.13.2 (17C2111)
- j680 — MacBookPro15,1 (July 2018) minimum macOS 10.13.6 (17G2112)
- j132 — MacBookPro15,2 (July 2018) minimum macOS 10.13.6 (17G2112)
- j174 — Macmini8,1 (October 2018) minimum macOS 10.14 (18A2063)
- j140k — MacBookAir8,1 (October 2018) minimum macOS 10.14.1 (18B2084)
- j780 — MacBookPro15,3 (May 2019) minimum macOS 10.14.5 (18F132)
- j213 — MacBookPro15,4 (July 2019) minimum macOS 10.14.5 (18F2058)
- j140a — MacBookAir8,2 (July 2019) minimum macOS 10.14.5 (18F2058)
- j152f — MacBookPro16,1 (November 2019) minimum macOS 10.15.1 (19B2093)
- j160 — MacPro7,1 (December 2019) minimum macOS 10.15.1 (19B88)
- j230k — MacBookAir9,1 (March 2020) minimum macOS 10.15.3 (19D2064)
- j214k — MacBookPro16,2 (May 2020) minimum macOS 10.15.4 (19E2269)
- j223 — MacBookPro16,3 (May 2020) minimum macOS 10.15.4 (19E2265)
- j215 — MacBookPro16,4 (June 2020) minimum macOS 10.15.5 (19F96)
- j185 — iMac20,1 (August 2020) minimum macOS 10.15.6 (19G2005)
- j185f — iMac20,2 (August 2020) minimum macOS 10.15.6 (19G2005)

PlatformInfo and SecureBootModel are independent, allowing to enabling Apple Secure Boot with any SMBIOS. Setting SecureBootModel to any valid value but Disabled is equivalent to Medium Security of Apple Secure Boot. To achieve Full Security one will need to also specify ApECID value.

Enabling Apple Secure Boot is more demanding to incorrect configurations, buggy macOS installations, and unsupported setups. Things to keep in mind:

(a) Just like on T2 Macs you will not be able to install any unsigned kernel drivers and several signed kernel drivers including NVIDIA Web Drivers.
(b) The list of cached drivers may be different, resulting in the need to change the list of Added or Forced kernel drivers. For example, IO80211Family cannot be injected in this case.
(c) System volume alterations on operating systems with sealing, like macOS 11, may result in the operating system being unbootable. Do not try to disable system volume encryption unless you disable Apple Secure Boot.
(d) If your platform requires certain settings, but they were not enabled, because the obvious issues did not trigger before, you may get boot failure. Be extra careful with IgnoreInvalidFlexRatio or HashServices.
(e) Operating systems released before Apple Secure Boot landed (e.g. macOS 10.12 or earlier) will still boot until UEFI Secure Boot is enabled. This is so, because from Apple Secure Boot point they are treated as incompatible and are assumed to be handled by the firmware just like 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) Since Default value will increase with time to support the latest major release operating system, it is not recommended to use ApECID and Default value together.

Sometimes the already installed operating system may have outdated Apple Secure Boot manifests on the Preboot partition causing boot failure. If you see the "OCB: Apple Secure Boot prohibits this boot entry, enforcing!" message, it is likely the case. When this happens you can either reinstall the operating system or copy the manifests (files with .im4m extension, like boot.efi.j137.im4m) from /usr/standalone/i386 to /Volumes/Preboot/<UUID>/System/Library/CoreServices. Here <UUID> is your system volume identifier. On

- 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).
- 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 you need 10.14.
- 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.
- 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=0xFFFF5F (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)
  - keepsyms=1 (show panic log debug symbols)
  - kextlog=VALUE (kernel extension loading debug mask)
  - 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
  - slide=VALUE (manually set KASLR slide)
  - smcdebug=VALUE (AppleSMC debug mask)
  - -amd_no_dgpu_accel (alternative to WhateverGreen's -radvesa for new GPUs)
  - -nehalem_error_disable
  - -no_compat_check (disable model checking on 10.7+)
  - -s (single mode)
  - -v (verbose mode)
  - -x (safe mode)

  There are multiple external places summarising macOS argument lists: example 1, example 2.
- 7C436110-AB2A-4BBB-A880-FE41995C9F82:bootercfg
  Booter arguments, similar to boot-args but for boot.efi. Accepts a set of arguments, which are hexadecimal 64-bit values with or without 0x. At different stages boot.efi will request different debugging (logging) modes

(e.g. after `ExitBootServices` it will only print to serial). Several booter arguments control whether these requests will succeed. The list of known requests is covered below:

- `0x00` – `INIT`.
- `0x01` – `VERBOSE` (e.g. `-v`, force console logging).
- `0x02` – `EXIT`.
- `0x03` – `RESET:OK`.
- `0x04` – `RESET:FAIL` (e.g. unknown `board-id`, hibernate mismatch, panic loop, etc.).
- `0x05` – `RESET:RECOVERY`.
- `0x06` – `RECOVERY`.
- `0x07` – `REAN:START`.
- `0x08` – `REAN:END`.
- `0x09` – `DT` (can no longer log to DeviceTree).
- `0x0A` – `EXITBS:START` (forced serial only).
- `0x0B` – `EXITBS:END` (forced serial only).
- `0x0C` – `UNKNOWN`.

In 10.15 debugging support was mostly broken before 10.15.4 due to some kind of refactoring and introduction of a new debug protocol. Some of the arguments and their values below may not be valid for versions prior to 10.15.4. The list of known arguments is covered below:

- `boot-save-log=VALUE` — debug log save mode for normal boot.
  - `0`
  - `1`
  - `2` — (default).
  - `3`
  - `4` — (save to file).
- `wake-save-log=VALUE` — debug log save mode for hibernation wake.
  - `0` — disabled.
  - `1`
  - `2` — (default).
  - `3` — (unavailable).
  - `4` — (save to file, unavailable).
- `breakpoint=VALUE` — enables debug breaks (missing in production `boot.efi`).
  - `0` — disables debug breaks on errors (default).
  - `1` — enables debug breaks on errors.
- `console=VALUE` — enables console logging.
  - `0` — disables console logging.
  - `1` — enables console logging when debug protocol is missing (default).
  - `2` — enables console logging unconditionally (unavailable).
- `embed-log-dt=VALUE` — enables DeviceTree logging.
  - `0` — disables DeviceTree logging (default).
  - `1` — enables DeviceTree logging.
- `kc-read-size=VALUE` — Chunk size used for buffered I/O from network or disk for prelinkedkernel reading and related. Set to 1MB (0x100000) by default, can be tuned for faster booting.
- `log-level=VALUE` — log level bitmask.
  - `0x01` — enables trace logging (default).
- `serial=VALUE` — enables serial logging.
  - `0` — disables serial logging (default).
  - `1` — enables serial logging for `EXITBS:END` onwards.
  - ~~1~~2 — enables serial logging for `EXITBS:START` onwards.
  - `3` — enables serial logging when debug protocol is missing.
  - `4` — enables serial logging unconditionally.
- `timestamps=VALUE` — enables timestamp logging.
  - `0` — disables timestamp logging.
  - `1` — enables timestamp logging (default).
- `log=VALUE` — deprecated starting from 10.15.
  - `1` — AppleLoggingConOutOrErrSet/AppleLoggingConOutOrErrPrint (classical ConOut/StdErr)
  - `2` — AppleLoggingStdErrSet/AppleLoggingStdErrPrint (StdErr or serial?)
  - `4` — AppleLoggingFileSet/AppleLoggingFilePrint (BOOTER.LOG/BOOTER.OLD file on EFI partition)

- `TryOverwrite` — `Overwrite` if new size is <= than the page-aligned original and there are no issues with legacy region unlock. `Create` otherwise. Has issues with some firmwares.
- `Create` — Replace the tables with newly allocated EfiReservedMemoryType at AllocateMaxAddress without any fallbacks.
- `Overwrite` — Overwrite existing gEfiSmbiosTableGuid and gEfiSmbiosTable3Guid data if it fits new size. Abort with unspecified state otherwise.
- `Custom` — Write SMBIOS tables (`gEfiSmbios(3)TableGuid`) to `gOcCustomSmbios(3)TableGuid` to workaround firmwares overwriting SMBIOS contents at ExitBootServices. Otherwise equivalent to `Create`. Requires patching AppleSmbios.kext and AppleACPIPlatform.kext to read from another GUID: `"EB9D2D31"` - `"EB9D2D35"` (in ASCII), done automatically by `CustomSMBIOSGuid` quirk.

Note: A side effect of using `Custom` approach is making SMBIOS updates exclusive to macOS, avoiding a collission with existing Windows activation and custom OEM software but potentially breaking Apple-specific tools.

6. `Generic`
   Type: `plist dictonary`
   Description: Update all fields. This section is read only when `Automatic` is active.

7. `DataHub`
   Type: `plist dictonary`
   Optional: When `Automatic` is `true`
   Description: Update Data Hub fields. This section is read only when `Automatic` is not active.

8. `PlatformNVRAM`
   Type: `plist dictonary`
   Optional: When `Automatic` is `true`
   Description: Update platform NVRAM fields. This section is read only when `Automatic` is not active.

9. `SMBIOS`
   Type: `plist dictonary`
   Optional: When `Automatic` is `true`
   Description: Update SMBIOS fields. This section is read only when `Automatic` is not active.

10.2 Generic Properties

1. `SpoofVendor`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Sets SMBIOS vendor fields to `Acidanthera`.

   It is dangerous to use Apple in SMBIOS vendor fields for reasons given in `SystemManufacturer` description. However, certain firmwares may not provide valid values otherwise, which could break some software.

2. `AdviseWindows`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Forces Windows support in `FirmwareFeatures`.

   Added bits to `FirmwareFeatures`:

   - `FW_FEATURE_SUPPORTS_CSM_LEGACY_MODE` (`0x1`) - Without this bit it is not possible to reboot to Windows installed on a drive with EFI partition being not the first partition on the disk.
   - `FW_FEATURE_SUPPORTS_UEFI_WINDOWS_BOOT` (`0x20000000`) - Without this bit it is not possible to reboot to Windows installed on a drive with EFI partition being the first partition on the disk.

3. `ProcessorType` **Type**: `plist integer`
   Failsafe: Automatic
   Description: Refer to SMBIOS `ProcessorType`.

4. `SystemProductName`
   Type: `plist string`
   Failsafe: `MacPro6,1`
   Description: Refer to SMBIOS `SystemProductName`.

SMBIOS: System Enclosure or Chassis (Type 3) — Type
Description: Chassis type, refer to Table 17 — System Enclosure or Chassis Types for more details.

20. `ChassisVersion`
    Type: `plist string`
    Failsafe: OEM specified
    SMBIOS: System Enclosure or Chassis (Type 3) — Version
    Description: Should match `BoardProduct`.

21. `ChassisSerialNumber`
    Type: `plist string`
    Failsafe: OEM specified
    SMBIOS: System Enclosure or Chassis (Type 3) — Version
    Description: Should match `SystemSerialNumber`.

22. `ChassisAssetTag`
    Type: `plist string`
    Failsafe: OEM specified
    SMBIOS: System Enclosure or Chassis (Type 3) — Asset Tag Number
    Description: Chassis type name. Varies, could be empty or `MacBook-Aluminum`.

23. `PlatformFeature`
    Type: `plist integer`, 32-bit
    Failsafe: `0xFFFFFFFF`
    SMBIOS: `APPLE_SMBIOS_TABLE_TYPE133` - `PlatformFeature`
    Description: Platform features bitmask. Refer to AppleFeatures.h for more details. Use `0xFFFFFFFF` value to not provide this table.

24. `SmcVersion`
    Type: `plist data`, 16 bytes
    Failsafe: All zero
    SMBIOS: `APPLE_SMBIOS_TABLE_TYPE134` - `Version`
    Description: ASCII string containing SMC version in upper case. Missing on T2 based Macs. Ignored when zero.

25. `FirmwareFeatures`
    Type: `plist data`, 8 bytes
    Failsafe: `0`
    SMBIOS: `APPLE_SMBIOS_TABLE_TYPE128` - `FirmwareFeatures` and `ExtendedFirmwareFeatures`
    Description: 64-bit firmware features bitmask. Refer to AppleFeatures.h for more details. Lower 32 bits match `FirmwareFeatures`. Upper 64 bits match `ExtendedFirmwareFeatures`.

26. `FirmwareFeaturesMask`
    Type: `plist data`, 8 bytes
    Failsafe: `0`
    SMBIOS: `APPLE_SMBIOS_TABLE_TYPE128` - `FirmwareFeaturesMask` and `ExtendedFirmwareFeaturesMask`
    Description: Supported bits of extended firmware features bitmask. Refer to AppleFeatures.h for more details. Lower 32 bits match `FirmwareFeaturesMask`. Upper 64 bits match `ExtendedFirmwareFeaturesMask`.

27. `ProcessorType`
    Type: `plist integer`, 16-bit
    Failsafe: Automatic
    SMBIOS: `APPLE_SMBIOS_TABLE_TYPE131` - `ProcessorType`
    Description: Combined of Processor Major and Minor types.

    Automatic value generation tries to provide most accurate value for the currently installed CPU. When this fails please make sure to create an issue and provide `sysctl machdep.cpu` and `dmidecode` output. For a full list of available values and their limitations (the value will only apply if the CPU core count matches) refer to Apple SMBIOS definitions header here.

28. `MemoryFormFactor`
    Type: `plist integer`, 8-bit

```
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 4: 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.

12.4 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 OpenCore 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. To initialise serial within OpenCore use `SerialInit` configuration option. For macOS your best choice are CP2102-based UART devices. Connect motherboard `TX` to USB UART `RX`, and motherboard `GND` to USB UART `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.

12.5 Tips and Tricks

1. **How to debug boot failure?**

   Normally it is enough to obtain the actual error message. For this ensure that:

   - You have a `DEBUG` or `NOOPT` version of OpenCore.
   - Logging is enabled (`1`) and shown onscreen (`2`): `Misc` → `Debug` → `Target` = `3`.
   - Logged messages from at least `DEBUG_ERROR` (`0x80000000`), `DEBUG_WARN` (`0x00000002`), and `DEBUG_INFO` (`0x00000040`) levels are visible onscreen: `Misc` → `Debug` → `DisplayLevel` = `0x80000042`.
   - Critical error messages, like `DEBUG_ERROR`, stop booting: `Misc` → `Security` → `HaltLevel` = `0x80000000`.
   - Watch Dog is disabled to prevent automatic reboot: `Misc` → `Debug` → `DisableWatchDog` = `true`.
   - Boot Picker (entry selector) is enabled: `Misc` → `Boot` → `ShowPicker` = `true`.

   If there is no obvious error, check the available hacks in `Quirks` sections one by one. For early boot troubleshooting, for instance, when OpenCore menu does not appear, using `UEFI Shell` (bundled with OpenCore) may help to see early debug messages.

2. **How to debug macOS boot failure?**

   - Refer to `boot-args` values like `debug=0x100`, `keepsyms=1`, `-v`, and similar.
   - Do not forget about `AppleDebug` and `ApplePanic` properties.
   - Take care of `Booter`, `Kernel`, and `UEFI` quirks.
   - Consider using serial port to inspect early kernel boot failures. For this you may need `debug=0x108`, `serial=5`, and `msgbuf=1048576` boot arguments. Refer to the patches in Sample.plist when dying before serial init.
   - Always read the logs carefully.