



OpenCore

Reference Manual (~~0.0.4~~0.5.0)

[2019.08.20]

1 Introduction

This document provides information on OpenCore user configuration file format used to setup the correct functioning of macOS operating system. It is to be read as the official clarification of expected OpenCore behaviour. All deviations, if found in published OpenCore releases, shall be considered documentation or implementation bugs, and are requested to be reported through Acidanthera Bugtracker. All other sources or translations of this document are unofficial and may contain errors.

1.1 ~~Known defects~~

~~For OpenCore issues please refer to~~ This document is structured as a specification, and is not meant to provide a step by step algorithm for configuring end-user board support package (BSP). Any third-party articles, tools, books, etc., providing such material are prone to their authors' preferences, tastes, this document misinterpretation, and essential obsolescence. In case you still use these sources, for example, Opencore Vanilla Desktop Guide, please ensure following this document for every made decision and judging its consequences. Regardless of the sources used you are required to fully understand every dedicated OpenCore configuration option and concept prior to reporting any issues in Acidanthera Bugtracker.

1.1 Generic Terms

- **plist** — Subset of ASCII Property List format written in XML, also know as XML plist format version 1. Uniform Type Identifier (UTI): `com.apple.property-list`. Plists consist of **plist objects**, which are combined to form a hierarchical structure. Due to plist format not being well-defined, all the definitions of this document may only be applied after plist is considered valid by running `plutil -lint`. External references: <https://www.apple.com/DTDs/PropertyList-1.0.dtd>, `man plutil`.
- **plist type** — plist collections (**plist array**, **plist dictionary**, **plist key**) and primitives (**plist string**, **plist data**, **plist date**, **plist boolean**, **plist integer**, **plist real**).
- **plist object** — definite realisation of **plist type**, which may be interpreted as value.
- **plist array** — array-like collection, conforms to **array**. Consists of zero or more **plist objects**.
- **plist dictionary** — map-like (associative array) collection, conforms to **dict**. Consists of zero or more **plist keys**.
- **plist key** — contains one **plist object** going by the name of **plist key**, conforms to **key**. Consists of printable 7-bit ASCII characters.
- **plist string** — printable 7-bit ASCII string, conforms to **string**.
- **plist data** — base64-encoded blob, conforms to **data**.
- **plist date** — ISO-8601 date, conforms to **date**, unsupported.
- **plist boolean** — logical state object, which is either true (1) or false (0), conforms to **true** and **false**.
- **plist integer** — possibly signed integer number in base 10, conforms to **integer**. Fits in 64-bit unsigned integer in two's complement representation, unless a smaller signed or unsigned integral type is explicitly mentioned in specific **plist object** description.
- **plist real** — floating point number, conforms to **real**, unsupported.
- **plist metadata** — value cast to data by the implementation. Permits passing **plist string**, in which case the result is represented by a null-terminated sequence of bytes (aka C string), **plist integer**, in which case the result is represented by 32-bit little endian sequence of bytes in two's complement representation, **plist boolean**, in which case the value is one byte: 01 for **true** and 00 for **false**, and **plist data** itself. All other types or larger integers invoke undefined behaviour.

10.3 DataHub Properties

1. PlatformName
Type: plist string
Failsafe: Not installed
Description: Sets name in gEfiMiscSubClassGuid. Value found on Macs is platform in ASCII.
2. SystemProductName
Type: plist string
Failsafe: Not installed
Description: Sets Model in gEfiMiscSubClassGuid. Value found on Macs is equal to SMBIOS SystemProductName in Unicode.
3. SystemSerialNumber
Type: plist string
Failsafe: Not installed
Description: Sets SystemSerialNumber in gEfiMiscSubClassGuid. Value found on Macs is equal to SMBIOS SystemSerialNumber in Unicode.
4. SystemUUID
Type: plist string, GUID
Failsafe: Not installed
Description: Sets system-id in gEfiMiscSubClassGuid. Value found on Macs is equal to SMBIOS SystemUUID.
5. BoardProduct
Type: plist string
Failsafe: Not installed
Description: Sets board-id in gEfiMiscSubClassGuid. Value found on Macs is equal to SMBIOS BoardProduct in ASCII.
6. BoardRevision
Type: plist data, 1 byte
Failsafe: 0
Description: Sets board-rev in gEfiMiscSubClassGuid. Value found on Macs seems to correspond to internal board revision (e.g. 01).
7. StartupPowerEvents
Type: plist integer, 64-bit
Failsafe: 0
Description: Sets StartupPowerEvents in gEfiMiscSubClassGuid. Value found on Macs is power management state bitmask, normally 0. Known bits read by X86PlatformPlugin.kext:
 - 0x00000001 — Shutdown cause was a PWROK event (Same as GEN_PMCN_2 bit 0)
 - 0x00000002 — Shutdown cause was a SYS_PWROK event (Same as GEN_PMCN_2 bit 1)
 - 0x00000004 — Shutdown cause was a THRMTRIP# event (Same as GEN_PMCN_2 bit 3)
 - 0x00000008 — Rebooted due to a SYS_RESET# event (Same as GEN_PMCN_2 bit 4)
 - 0x00000010 — Power Failure (Same as GEN_PMCN_3 bit 1 PWR_FLR)
 - 0x00000020 — Loss of RTC Well Power (Same as GEN_PMCN_3 bit 2 RTC_PWR_STS)
 - 0x00000040 — General Reset Status (Same as GEN_PMCN_3 bit 9 GEN_RST_STS)
 - 0xffffffff80 — SUS Well Power Loss (Same as GEN_PMCN_3 bit 14)
 - 0x00010000 — Wake cause was a ME Wake event (Same as PRSTS bit 0, ME_WAKE_STS)
 - 0x00020000 — Cold Reboot was ME Induced event (Same as PRSTS bit 1 ME_HRST_COLD_STS)
 - 0x00040000 — Warm Reboot was ME Induced event (Same as PRSTS bit 2 ME_HRST_WARM_STS)
 - 0x00080000 — Shutdown was ME Induced event (Same as PRSTS bit 3 ME_HOST_PWRDN)
 - 0x00100000 — Global reset ME Watchdog Timer event (Same as PRSTS bit 6)
 - 0x00200000 — Global reset PowerManagment Watchdog Timer event (Same as PRSTS bit 15)
8. InitialTSC
Type: plist integer, 64-bit
Failsafe: 0
Description: Sets InitialTSC in gEfiProcessorSubClassGuid. Sets initial TSC value, normally 0.

9. FSBFrequency
Type: plist integer, 64-bit
Failsafe: Automatic
Description: Sets FSBFrequency in gEfiProcessorSubClassGuid.
 Sets CPU FSB frequency. [This value equals to CPU nominal frequency divided by CPU maximum bus ratio and is specified in Hz. Refer to MSR_NEHALEM_PLATFORM_INFO \(CEh\) MSR value to determine maximum bus ratio on modern Intel CPUs.](#)
10. ARTFrequency
Type: plist integer, 64-bit
Failsafe: ~~Not installed~~[Automatic](#)
Description: Sets ARTFrequency in gEfiProcessorSubClassGuid. ~~Sets-~~
[This value contains CPU ART frequency, ~~Skylake~~ also known as crystal clock frequency. Its existence is exclusive to Skylake generation and newer. The value is specified in Hz, and is normally 24 MHz for client Intel segment, 25 MHz for server Intel segment, and 19.2 MHz for Intel Atom CPUs. macOS till 10.15 inclusive assumes 24 MHz by default.](#)
11. DevicePathsSupported
Type: plist integer, 32-bit
Failsafe: Not installed
Description: Sets DevicePathsSupported in gEfiMiscSubClassGuid. Must be set to 1 for AppleACPIPlatform.kext to append SATA device paths to Boot#### and efi-boot-device-data variables. Set to 1 on all modern Macs.
12. SmcRevision
Type: plist data, 6 bytes
Failsafe: Not installed
Description: Sets REV in gEfiMiscSubClassGuid. Custom property read by VirtualSMC or FakeSMC to generate SMC REV key.
13. SmcBranch
Type: plist data, 8 bytes
Failsafe: Not installed
Description: Sets RBr in gEfiMiscSubClassGuid. Custom property read by VirtualSMC or FakeSMC to generate SMC RBr key.
14. SmcPlatform
Type: plist data, 8 bytes
Failsafe: Not installed
Description: Sets RPlt in gEfiMiscSubClassGuid. Custom property read by VirtualSMC or FakeSMC to generate SMC RPlt key.

10.4 PlatformNVRAM Properties

1. BID
Type: plist string
Failsafe: Not installed
Description: Specifies the value of NVRAM variable 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_BID.
2. ROM
Type: plist data, 6 bytes
Failsafe: Not installed
Description: Specifies the values of NVRAM variables 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_ROM and 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ROM.
3. MLB
Type: plist string
Failsafe: Not installed
Description: Specifies the values of NVRAM variables 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_MLB and 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:MLB.

- 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.

2. How to customise boot entries?

OpenCore follows standard Apple Bless model and extracts the entry name from `.contentDetails` and `.disk_label.contentDetails` files in the booter directory if present. These files contain an ASCII string with an entry title, which may then be customised by the user.

3. [How to choose the default boot entry?](#)

[OpenCore uses the primary UEFI boot option to select the default entry. This choice can be altered from UEFI Setup, with the macOS Startup Disk preference, or the Windows Boot Camp Control Panel. Since choosing OpenCore's B00Tx64.EFI as a primary boot option limits this functionality in addition to several firmwares deleting incompatible boot options, potentially including those created by macOS, you are strongly encouraged to use the RequestBootVarRouting quirk, which will preserve your selection made in the operating system within the OpenCore variable space. Note, that RequestBootVarRouting requires a separate driver for functioning.](#)

4. What is the simplest way to install macOS?

Copy online recovery image (`*.dmg` and `*.chunklist` files) to `com.apple.recovery.boot` directory on a FAT32 partition with OpenCore. Load OpenCore Boot Picker and choose the entry, it will have a `(dmg)` suffix. Custom name may be created by providing `.contentDetails` file.

To download recovery online you may use Recovery tool from `OcSupportPkg`.

[For offline installation refer to](#) [How to create a bootable installer for macOS article.](#)

5. Why do online recovery images (`*.dmg`) fail to load?

This may be caused by missing HFS+ driver, as all presently known recovery volumes have HFS+ filesystem. Another cause may be buggy firmware allocator, which can be worked around with `AvoidHighAlloc` UEFI quirk.

6. Can I use this on Apple hardware or virtual machines?

Sure, most relatively modern Mac models including `MacPro5,1` and virtual machines are fully supported. Even though there are little to none specific details relevant to Mac hardware, some ongoing instructions can be found in [acidanthera/bugtracker#377](#).

7. Why do Find&Replace patches must equal in length?

For machine code (x86 code) it is not possible to do such replacements due to relative addressing. For ACPI code this is risky, and is technically equivalent to ACPI table replacement, thus not implemented. More detailed explanation can be found on [AppleLife.ru](#).

8. How can I migrate from AptioMemoryFix?

Behaviour similar to that of `AptioMemoryFix` can be obtained by installing `FwRuntimeServices` driver and enabling the quirks listed below. Please note, that most of these are not necessary to be enabled. Refer to their individual descriptions in this document for more details.

- `ProvideConsoleGop` (UEFI quirk)
- `AvoidRuntimeDefrag`
- `DiscardHibernateMap`
- `EnableSafeModeSlide`
- `EnableWriteUnprotector`
- `ForceExitBootServices`
- `ProtectCsmRegion`
- `ProvideCustomSlide`
- `SetupVirtualMap`
- `ShrinkMemoryMap`