

OpenCore

Reference Manual [\(0.0.2\)](#)

[2019.05.13]

1 Introduction

This document provides information on OpenCore user configuration file format used to setup the correct functioning of macOS operating system.

1.1 Known defects

For OpenCore issues please refer to Acidanthera Bugtracker. ~~Currently this file has the following entries not completed:~~

- ~~• Not all NVRAM variables are properly described (e.g. boot-args).~~

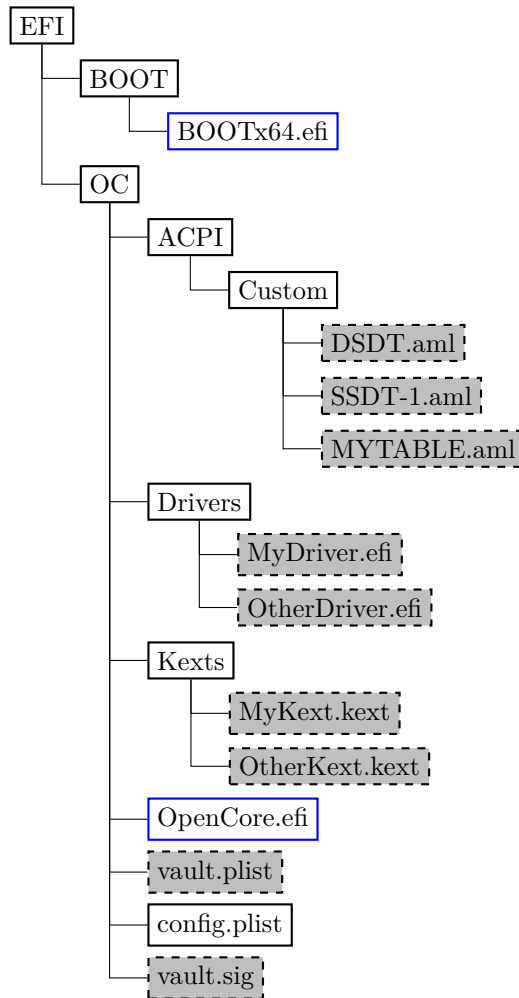


Figure 1. Directory Structure

3.5 Installation and Upgrade

To install OpenCore reflect the Configuration Structure described in the previous section on a EFI volume of a GPT partition. While corresponding sections of this document do provide some information in regards to external resources like ACPI tables, UEFI drivers, or kernel extensions (kexts), completeness of the matter is out of the scope of this document. Extra information about particular kernel extensions may be found in Lilu's Known Plugins table. Vaulting information is provided in Security Properties section of this document.

OC config, just like any property lists can be edited with any stock textual editor (e.g. nano, vim), but specialised software may provide better experience. On macOS the preferred GUI application is Xcode. For a lightweight cross-platform and open-source alternative ProperTree editor can be utilised.

For BIOS booting a third-party UEFI environment provider will have to be used. DuetPkg is one of the known UEFI environment providers for legacy systems. While it is known to be possible to run OpenCore on such a legacy system, configuration and use of DuetPkg is currently out of the scope of this document.

For upgrade purposes refer to Differences.pdf document, providing the information about the changes affecting the configuration compared to the previous release, and Changelog.md document, containing the list of modifications across all published updates.

3.6 Contribution

OpenCore can be compiled as an ordinary EDK II package with UDK 2018.

The only officially supported toolchain is XCODE5. Other toolchains might work, but are neither supported, nor recommended. Contribution of clean patches is welcome. Please do follow EDK II C Codestyle.

Required external package dependencies include EfiPkg, MacInfoPkg, and OcSupportPkg.

To compile with XCODE5, besides Xcode, one should also install NASM and MTOC. [The latest Xcode version is recommended for use despite the toolchain name.](#) Example command sequence may look as follows:

```
git clone https://github.com/tianocore/edk2 -b UDK2018 UDK
cd UDK
git clone https://github.com/acidanthera/EfiPkg
git clone https://github.com/acidanthera/MacInfoPkg
git clone https://github.com/acidanthera/OcSupportPkg
git clone https://github.com/acidanthera/OpenCorePkg
source edksetup.sh
make -C BaseTools
build -a X64 -b RELEASE -t XCODE5 -p OpenCorePkg/OpenCorePkg.dsc
```

Listing 1: Compilation Commands

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.

For IDE usage Xcode projects are available in the root of the repositories. Another approach could be Sublime Text with EasyClangComplete plugin. Add .clang_complete file with similar content to your UDK root:

```
-I/UefiPackages/MdePkg
-I/UefiPackages/MdePkg/Include
-I/UefiPackages/MdePkg/Include/X64
-I/UefiPackages/EfiPkg
-I/UefiPackages/EfiPkg/Include
-I/UefiPackages/EfiPkg/Include/X64
-I/UefiPackages/AptioFixPkg/Include
-I/UefiPackages/AppleSupportPkg/Include
-I/UefiPackages/OpenCorePkg/Include
-I/UefiPackages/OcSupportPkg/Include
-I/UefiPackages/IntelFrameworkPkg/Include
-I/UefiPackages/MacInfoPkg/Include
-IInclude
-include
/UefiPackages/MdePkg/Include/Uefi.h
-fshort-wchar
-Wall
-Wextra
-Wno-unused-parameter
-Wno-missing-braces
-Wno-missing-field-initializers
-Wno-tautological-compare
-Wno-sign-compare
-Wno-varargs
-Wno-unused-const-variable
```

Listing 2: ECC Configuration

Warning: Tool developers modifying config.plist or any other OpenCore files must ensure that their tool checks for opencore-version NVRAM variable (see Debug Properties section below) and warn the user if the version listed is unsupported or prerelease. OpenCore configuration may change across the releases and the tool shall ensure that it carefully follows this document. Failure to do so may result in this tool to be considered as malware and blocked with all possible means.

Default value: false

Description: Provide reset register and flag in FADT table to enable reboot and shutdown on legacy hardware. Not recommended unless required.

2. IgnoreForWindows

Type: plist boolean

Default value: false

Description: Disable all sorts of ACPI modifications when booting Windows operating system.

This flag implements a quick workaround for those, who made their ACPI tables incompatible with Windows, but need it right now. Not recommended, as ACPI tables must be compatible with any operating system regardless of the changes.

Note: This option may be removed in the future.

3. NormalizeHeaders

Type: plist boolean

Default value: 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).

4. RebaseRegions

Type: plist boolean

Default value: 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.

For this reason it is very dangerous to apply any kind of modifications to ACPI tables. The most reasonable approach is to make as few as possible changes to ACPI and try to not replace any tables, especially DSDT. When this is not possible, then at least attempt to ensure that custom DSDT is based on the most recent DSDT or remove writes and reads for the affected areas.

When nothing else helps this option could be tried to avoid stalls at `PCI Configuration Begin` phase of macOS booting by attempting to fix the ACPI addresses. It does not do magic, and only works with most common cases. Do not use unless absolutely required.

5. ResetLogoStatus

Type: plist boolean

Default value: false

Description: Reset BGRT table `Displayed` status field to false.

This works around firmwares that provide BGRT table but fail to handle screen updates afterwards.

4. HideSelf
Type: plist boolean
Default value: false
Description: Hides own boot entry from boot picker. This may potentially hide other entries, for instance, when another UEFI OS is installed on the same volume and driver boot is used.
5. Resolution
Type: plist string
Default value: Empty string
Description: Sets console output screen resolution ~~as specified with the~~ [~](#)
 - [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 empty string not to change screen resolution.
 - Set to Max to try to use largest available screen resolution.

[On HiDPI screens APPLE_VENDOR_VARIABLE_GUID UIScale NVRAM variable may need to be set to 02 to enable HiDPI scaling in FileVault 2 UEFI password interface and boot screen logo. Refer to Recommended Variables section for more 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 UEFI quirk set to true.
6. ShowPicker
Type: plist boolean
Default value: false
Description: Show simple boot picker to allow boot entry selection.
7. Timeout
Type: plist integer, 32 bit
Default value: 0
Description: Timeout in seconds in boot picker before automatic booting of the default boot entry.

7.4 Debug Properties

1. DisableWatchDog
Type: plist boolean
Default value: NO
Description: Select firmwares may not succeed in quickly booting the operating system, especially in debug mode, which results in watch dog timer aborting the process. This option turns off watch dog timer.
2. DisplayDelay
Type: plist integer
Default value: 0
Description: Delay in microseconds performed after every printed line visible onscreen (i.e. console).
3. DisplayLevel
Type: plist integer, 64 bit
Default value: 0
Description: EDK II debug level bitmask (sum) showed onscreen. Unless **Target** enables console (onscreen) printing, onscreen debug output will not be visible. The following levels are supported (discover more in DebugLib.h):
 - 0x00000002 — DEBUG_WARN in DEBUG, NOOPT, RELEASE.
 - 0x00000040 — DEBUG_INFO in DEBUG, NOOPT.
 - 0x00400000 — DEBUG_VERBOSE in custom builds.
 - 0x80000000 — DEBUG_ERROR in DEBUG, NOOPT, RELEASE.
4. ~~ExposeBootPath~~
Type: ~~plist boolean~~
Default value: ~~false~~
Description: ~~Expose printable booter path to OpenCore.efi or its booter (depending on the load order) as an UEFI variable.~~

~~To obtain booter path use the following command in macOS:-~~

```
nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-path
```

To use booter path for mounting booter volume use the following command in macOS:-

```
u=$(nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-path | sed 's/.*GPT,\([^,]*\)\\.*/\1/');\  
—if [ "$u" != "" ]; then sudo diskutil mount $u ; fi
```

5. Target

Type: plist integer

Default value: 0

Description: A bitmask (sum) of enabled logging targets. By default all the logging output is hidden, so this option is required to be set when debugging is necessary.

The following logging targets are supported:

- 0x01 — Enable logging, otherwise all log is discarded.
- 0x02 — Enable basic console (onscreen) logging.
- 0x04 — Enable logging to Data Hub.
- 0x08 — Enable serial port logging.
- 0x10 — Enable UEFI variable logging.
- 0x20 — Enable non-volatile UEFI variable logging.
- 0x40 — Enable logging to file.

Console logging prints less than all the other variants. Depending on the build type (RELEASE, DEBUG, or NOOPT) different amount of logging may be read (from least to most).

Data Hub log will not log kernel and kext patches. To obtain Data Hub log use the following command in macOS:

```
ioreg -lw0 -p IODeviceTree | grep boot-log | sort | sed 's/.*<\(.*\)>\\.*/\1/' | xxd -r -p
```

UEFI variable log does not include some messages and has no performance data. For safety reasons log size is limited to 32 kilobytes. Some firmwares may truncate it much earlier or drop completely if they have no memory. Using non-volatile flag will write the log to NVRAM flash after every printed line. To obtain UEFI variable log use the following command in macOS:

```
nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-log |  
awk '{gsub(/%0d%0a%00/, "");gsub(/%0d%0a/, "\n")}'1'
```

Warning: Some firmwares are reported to have broken NVRAM garbage collection. This means that they may not be able to always free space after variable deletion. Do not use non-volatile NVRAM logging without extra need on such devices.

While OpenCore boot log already contains basic version information with build type and date, this data may also be found in NVRAM in `opencore-version` variable even with boot log disabled:-

```
nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:opencore-version
```

~

File logging will create a file named `opencore.log` at EFI volume root with log contents. Please be warned that some file system drivers present in firmwares are not reliable, and may corrupt data when writing files through UEFI. Log is attempted to be written in the safest manner, and thus is very slow. Ensure that `DisableWatchDog` is set to `true` when you use a slow drive.

7.5 Security Properties

1. ExposeSensitiveData

Type: plist integer

Default value: 2

Description: Sensitive data exposure bitmask (sum) to operating system.

- 0x01 — Expose printable booter path as an UEFI variable.
- 0x02 — Expose OpenCore version as an UEFI variable.

Exposed booter path points to OpenCore.efi or its booter depending on the load order. To obtain booter path use the following command in macOS:

```
nvrnm 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:booter-path
```

To use booter path for mounting booter volume use the following command in macOS:

```
u=$(nvrnm 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:booter-path | sed 's/.*GPT,\([^\]*\)\\.*/\1/');\
if [ "$u" != "" ]; then sudo diskutil mount $u ; fi
```

To obtain OpenCore version use the following command in macOS:

```
nvrnm 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:opencore-version
```

2. HaltLevel

Type: plist integer, 64 bit

Default value: 0x80000000 (DEBUG_ERROR)

Description: EDK II debug level bitmask (sum) causing CPU to halt (stop execution) after obtaining a message of HaltLevel. Possible values match DisplayLevel values.

3. RequireSignature

Type: plist boolean

Default value: true

Description: Require vault.sig signature file for vault.plist in OC directory.

This file should contain a raw 256 byte RSA-2048 signature from SHA-256 hash of vault.plist. The signature is verified against the public key embedded into OpenCore.efi.

To embed the public key you should do either of the following:

- Provide public key during the OpenCore.efi compilation in OpenCoreVault.c file.
- Binary patch OpenCore.efi replacing zeroes with the public key between =BEGIN OC VAULT= and ==END OC VAULT== ASCII markers.

RSA public key 520 byte format description can be found in Chromium OS documentation. To convert public key from X.509 certificate or from PEM file use RsaTool.

Note: vault.sig is used regardless of this option when public key is embedded into OpenCore.efi. Setting it to true will only ensure configuration sanity, and abort the boot process when public key is not set but was supposed to be used for verification.

4. RequireVault

Type: plist boolean

Default value: true

Description: Require vault.plist file present in OC directory.

This file should contain SHA-256 hashes for all files used by OpenCore. Presence of this file is highly recommended to ensure that unintentional file modifications (including filesystem corruption) do not happen unnoticed. To create this file automatically use create_vault.sh script.

Regardless of the underlying filesystem, path name and case must match between config.plist and vault.plist.

Note: vault.plist is tried to be read regardless of the value of this option, but setting it to true will ensure configuration sanity, and abort the boot process.

The complete set of commands to:

- Create vault.plist.
- Create a new RSA key (always do this to avoid loading old configuration).
- Embed RSA key into OpenCore.efi.
- Create vault.sig.

8.4 Recommended Variables

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

- `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).
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:prev-lang:kbd`
ASCII string defining default keyboard layout. Format is `lang-COUNTRY:keyboard`, e.g. `ru-RU:19456252` for [Mac Russian locale and ABC keyboard](#). Also accepts short forms: `ru:19456252` or `ru:0`. [Full decoded list of keyboards in \(U.S. keyboard, compatible with 10.9\). Full decoded keyboard list from AppleKeyboardLayouts-L.dat can be found on AppleLife. Using non-latin keyboard on 10.14 will not enable ABC keyboard, unlike previous macOS versions, and is thus not recommended.](#)
- `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`
8-bit integer defining `boot.efi` user interface scaling. Should be 1 for normal screens and 2 for HDPI screens.

8.5 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:](#)
 - ~~FIXME: document several known values! debug, keepsyms, slide, -v, -s, -x, cpus=x, io=x, kextlog=x, nehalen_error_disable, no_compat_check, nvda_drv=1, etc?~~ `acpi_layer=0xFFFFFFFF`
 - `acpi_level=0xFFFF5F` (implies `ACPI_ALL_COMPONENTS`)
 - `cpus=VALUE`
 - `debug=VALUE`
 - `io=VALUE`
 - `keepsyms=1`
 - `kextlog=VALUE`
 - `nvda_drv=1`
 - `slide=VALUE`
 - `-nehalen_error_disable`
 - `-no_compat_check`
 - `-s`
 - `-v`
 - `-x`
- `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` prefix primarily for logging control:
 - `log=VALUE`

9 PlatformInfo

Platform information is comprised of several identification fields generated or filled manually to be compatible with macOS services. The base part of the configuration may be obtained from `MacInfoPkg` package, which itself generates a set of interfaces based on a database in YAML format. These fields are written to three select destinations:

- SMBIOS
- Data Hub
- NVRAM

Most of the fields specify the overrides in SMBIOS, and their field names conform to EDK2 `SmBios.h` header file. However, several important fields reside in Data Hub and NVRAM. Some of the values can be found in more than one field and/or destination, so there are two ways to control their update process: manual, where one specifies all the values (the default), and semi-automatic, where (`Automatic`) only select values are specified, and later used for system configuration.

9.1 Properties

1. `Automatic`

Type: plist boolean

Default value: false

Description: Generate PlatformInfo based on `Generic` section instead of using values from `DataHub`, `NVRAM`, and `SMBIOS` sections.

Enabling this option is useful when `Generic` section is flexible enough. When enabled `SMBIOS`, `DataHub`, and `PlatformNVRAM` data is unused.

2. `UpdateDataHub`

Type: plist boolean

Default value: false

Description: Update DataHub fields. These fields are read from `Generic` or `DataHub` sections depending on `Automatic` value.

3. `UpdateNVRAM`

Type: plist boolean

Default value: false

Description: Update NVRAM fields related to platform information.

These fields are read from `Generic` or `PlatformNVRAM` sections depending on `Automatic` value. All the other fields are to be specified with `NVRAM` section.

If `UpdateNVRAM` is set to `false` the aforementioned variables can be updated with `NVRAM` section. If `UpdateNVRAM` is set to `true` the behaviour is undefined when any of the fields are present in `NVRAM` section.

4. `UpdateSMBIOS`

Type: plist boolean

Default value: false

Description: Update SMBIOS fields. These fields are read from `Generic` or `SMBIOS` sections depending on `Automatic` value.

5. `UpdateSMBIOSMode`

Type: plist string

Default value: `AutoCreate`

Description: Update SMBIOS fields approach:

- `AutoTryOverwrite` — Overwrite if new size is \leq 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 first SMBIOS table (`gEfiSmbiosTableGuid`) to `gOcCustomSmbiosTableGuid` 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).

6. Generic
Type: plist dictionary
Optional: When Automatic is false
Description: Update all fields. This section is read only when Automatic is active.
7. DataHub
Type: plist dictionary
Optional: When Automatic is true
Description: Update Data Hub fields. This section is read only when Automatic is not active.
8. PlatformNVRAM
Type: plist dictionary
Optional: When Automatic is true
Description: Update platform NVRAM fields. This section is read only when Automatic is not active.
9. SMBIOS
Type: plist dictionary
Optional: When Automatic is true
Description: Update SMBIOS fields. This section is read only when Automatic is not active.

9.2 Generic Properties

1. SpoofVendor
Type: plist boolean
Default value: 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. SystemProductName
Type: plist string
Default value: MacPro6,1
Description: Refer to SMBIOS SystemProductName.
3. SystemSerialNumber
Type: plist string
Default value: OPENCORE_SN1
Description: Refer to SMBIOS SystemSerialNumber.
4. SystemUUID
Type: plist string, GUID
Default value: OEM specified
Description: Refer to SMBIOS SystemUUID.
5. MLB
Type: plist string
Default value: OPENCORE_MLB_SN11
Description: Refer to SMBIOS BoardSerialNumber.
6. ROM
Type: plist data, 6 bytes
Default value: all zero
Description: Refer to 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ROM.

9.3 DataHub Properties

1. PlatformName
Type: plist string

10 UEFI

10.1 Introduction

UEFI (Unified Extensible Firmware Interface) is a specification that defines a software interface between an operating system and platform firmware. This section allows to load additional UEFI modules and/or apply tweaks for the onboard firmware.

10.2 Properties

1. ConnectDrivers

Type: plist boolean

Default value: NO

Description: Perform UEFI controller connection after driver loading. This option is useful for loading filesystem drivers, which usually follow UEFI driver model, and may not start by themselves. While effective, this option is not necessary with e.g. APFS loader driver, and may slightly slowdown the boot.

2. Drivers

Type: plist array

Default value: None

Description: Load selected drivers from `OC/Drivers` directory.

Designed to be filled with string filenames meant to be loaded as UEFI drivers. Depending on the firmware a different set of drivers may be required. Loading an incompatible driver may lead your system to unbootable state or even cause permanent firmware damage. Some of the known drivers include:

- **ApfsDriverLoader** — APFS file system bootstrap driver adding the support of embedded APFS drivers in bootable APFS containers in UEFI firmwares.
- **AppleUiSupport** — Apple-specific user interface support driver. This driver brings the support for FileVault 2 GUI, hotkey parsing (shift, cmd+v, etc.), language collation support, and certain other features important for normal macOS functioning. For hotkey support **AppleKeyMapAggregator**-compatible driver is required.
- **AptioInputFix** — user input driver adding the support of **AppleKeyMapAggregator** protocols on top of different UEFI input protocols. Additionally resolves mouse input issues on select firmwares. This is an alternative to **UsbKbDxe**, which may work better or worse depending on the firmware.
- **AptioMemoryFix** — a set of quirks for various firmwares. While it primarily targets APTIO firmwares, other firmwares may be compatible as well. Among the resolved issues are hibernation support, KASLR, Lilu NVRAM security enhancements, NVRAM, and UEFI Boot entry preservation.
- **DataHubDxe** — ~~Intel DataHub driver from IntelFrameworkModulePkg. It is believed that this driver is included in all UEFI firmwares, and is not required.~~
- ~~—~~NVRAM emulation driver from **MdeModulePkg**. NVRAM is supported by most modern firmwares. For firmwares with macOS incompatible NVRAM implementation an emulated driver may be used. This driver will not preserve NVRAM contents across the reboots.
- **EnglishDxe** — Unicode collation driver from **MdeModulePkg**. This driver is a lightweight alternative to **AppleUiSupport**, which contains no Apple-specific code, and only provides unicode collation support. The driver is not recommended for use on any hardware but few original Macs.
- **EnhancedFatDxe** — FAT filesystem driver from **FatPkg**. This driver is embedded in all UEFI firmwares, and cannot be used from OpenCore. It is known that multiple firmwares have a bug in their FAT support implementation, which leads to corrupted filesystems on write attempt. Embedding this driver within the firmware may be required in case writing to EFI partition is needed during the boot process.
- **NvmExpressDxe** — NVMe support driver from **MdeModulePkg**. This driver is included in most firmwares starting with Broadwell generation. For Haswell and earlier embedding it within the firmware may be more favourable in case a NVMe SSD drive is installed.
- **UsbKbDxe** — USB keyboard driver adding the support of **AppleKeyMapAggregator** protocols on top of a custom USB keyboard driver implementation. This is an alternative to **AptioInputFix**, which may work better or worse depending on the firmware.
- **VirtualSmc** — UEFI SMC driver, required for proper FileVault 2 functionality and potentially other macOS specifics. An alternative, named **SMCHelper**, is not compatible with **VirtualSmc** and OpenCore, which is unaware of its specific interfaces. In case **FakeSMC** kernel extension is used, manual NVRAM variable addition may be needed and **VirtualSmc** driver should still be used.

- VBoxHfs — HFS file system driver with bless support. This driver is an alternative to a closed source HFSPPlus driver commonly found in Apple firmwares. While it is feature complete, it is approximately 3 times slower and is yet to undergo a security audit.
- XhciDxe — [XHCI USB controller support driver from MdeModulePkg. This driver is included in most firmwares starting with Sandy Bridge generation. For earlier firmwares or legacy systems it may be used to support external USB 3.0 PCI cards.](#)

To compile the drivers from TianoCore UDK use the same command you do normally use for OpenCore compilation, but choose a corresponding package:

```
git clone https://github.com/tianocore/edk2 -b UDK2018 UDK
cd UDK
source edksetup.sh
make -C BaseTools
build -a X64 -b RELEASE -t XCODE5 -p FatPkg/FatPkg.dsc
build -a X64 -b RELEASE -t XCODE5 -p MdeModulePkg/MdeModulePkg.dsc
build -a X64 -b RELEASE -t XCODE5 -p IntelFrameworkModulePkg/IntelFrameworkModulePkg.dsc
```

3. Protocols

Type: plist dict
Default value: None
Description: Force builtin versions of select protocols described in Protocols Properties section below.
4. Quirks

Type: plist dict
Default value: None
Description: Apply individual firmware quirks described in Quirks Properties section below.

10.3 Protocols Properties

1. AppleBootPolicy

Type: plist boolean
Default value: false
Description: Reinstalls apple boot policy protocol with a builtin version. This may be used to ensure APFS compatibility on VMs or legacy Macs.
2. DeviceProperties

Type: plist boolean
Default value: false
Description: Reinstalls device property protocol with a builtin version. This will drop all previous properties if it was already installed. This may be used to ensure full compatibility on VMs or legacy Macs.

10.4 Quirks Properties

1. ExitBootServicesDelay

Type: plist integer
Default value: 0
Description: Adds delay in microseconds after EXIT_BOOT_SERVICES event.

This is a very ugly quirk to circumvent "Still waiting for root device" message on select APTIO IV firmwares, namely ASUS Z87-Pro, when using FileVault 2 in particular. It seems that for some reason they execute code in parallel to EXIT_BOOT_SERVICES, which results in SATA controller being inaccessible from macOS. A better approach should be found in some future. Expect 3-5 seconds to be enough in case the quirk is needed.

2. IgnoreInvalidFlexRatio

Type: plist boolean
Default value: NO
Description: Select firmwares, namely APTIO IV, may contain invalid values in MSR_FLEX_RATIO (0x194) MSR register. These values may cause macOS boot failure on Intel platforms.