

OpenCore

Reference Manual (1.0.5)

[2025.03.04]

Contents

1	Introduction 1.1 Generic Terms		 	 	 		 				 		 		3 3
2	Configuration														4
_	2.1 Configuration Ter	moo ci													4
	_														
	2.2 Configuration Pro	_													4
	2.3 Configuration Str	ucture	 	 	 	• •	 		• •	• •	 	•	 • •	• •	5
3	Setup														7
	3.1 Directory Structu	re	 	 	 		 				 		 		7
	3.2 Installation and U														8
	3.3 Contribution														9
	3.4 Coding convention														10
	3.5 Debugging														11
	5.5 Debugging	· • • • • •	 	 	 		 			• •	 • •	•	 • •	• •	11
4	ACPI														13
	4.1 Introduction														13
	4.2 Properties		 	 	 		 				 		 		13
	4.3 Add Properties .		 	 	 		 				 		 		14
	4.4 Delete Properties														14
	4.5 Patch Properties .														15
	4.6 Quirks Properties														16
	4.0 Quitas i toperties		 	 • •	 	• •	 	• •			 • •	•	 • •	• •	10
5	Booter														18
	5.1 Introduction														18
	5.2 Properties		 	 	 		 				 		 		18
	5.3 MmioWhitelist Pr	roperties	 	 	 		 				 		 		19
	5.4 Patch Properties.		 	 	 		 				 		 		19
	5.5 Quirks Properties														20
6	DeviceProperties														26
Ū	6.1 Introduction														26
	6.2 Properties														26
	-														
	6.3 Common Properti	les	 	 	 		 				 	•	 	• •	26
7	Kernel														28
	7.1 Introduction														28
	7.2 Properties		 	 	 		 				 		 		28
	7.3 Add Properties .														29
	7.4 Block Properties .														30
	7.5 Emulate Propertie														31
	7.6 Force Properties .														32
															$\frac{32}{33}$
	*														
	7.8 Quirks Properties														34
	7.9 Scheme Properties	5	 	 	 		 				 	٠	 		39
8	Misc														42
	8.1 Introduction		 	 	 		 				 		 		42
	8.2 Properties														44
	8.3 Boot Properties .														45
	8.4 Debug Properties														51
	U -														
	8.5 Entry Properties .														54
	8.6 Security Properties														55
	8.7 Serial Properties .		 	 	 		 				 		 		61
9	NVRAM														64

9.1	Introduction	64
9.2	Properties	64
9.3	Mandatory Variables	65
9.4		65
9.5	Other Variables	66
10 Pl	at form Info	70
10	1 Properties	70
10	2 DataHub Properties	72
10	3 Generic Properties	74
10	4 Memory Properties	75
10	5 PlatformNVRAM Properties	78
10	6 SMBIOS Properties	78
11 U	PFI .	83
11		83
11		83
11		84
11	11	85
11	1 10	86
11		86
11		87
11	*	90
11		92
11		94
		95
		97
		99
		.00
		.03
11	16 Drivers Properties	06
11	17 Input Properties	.06
11	18 Output Properties	.08
11	19 ProtocolOverrides Properties	12
11	20 Quirks Properties	14
11	21 ReservedMemory Properties	.17
12 Tr	publeshooting 1	19
12		19
12	¥	20
12		21
12	**	22
12		23
	±	

1 Introduction

This document provides information on the format of the OpenCore user configuration file used to set up the correct functioning of the 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 to be documentation or implementation issues which should be reported via the Acidanthera Bugtracker. An errata sheet is available in OpenCorePkg repository.

This document is structured as a specification and is not meant to provide a step-by-step guide to configuring an end-user Board Support Package (BSP). The intended audience of the document is anticipated to be programmers and engineers with a basic understanding of macOS internals and UEFI functionality. For these reasons, this document is available exclusively in English, and all other sources or translations of this document are unofficial and may contain errors.

Third-party articles, utilities, books, and similar, may be more useful for a wider audience as they could provide guide-like material. However, they are subject to their authors' preferences, misinterpretations of this document, and unavoidable obsolescence. In cases of using such sources, such as Dortania's OpenCore Install Guide and related material, please refer back to this document on every decision made and re-evaluate potential implications.

Please note that regardless of the sources used, users are required to fully understand every OpenCore configuration option, and the principles behind them, before posting issues to the Acidanthera Bugtracker.

Note: Creating this document would not have been possible without the invaluable contributions from other people: Andrey1970, Goldfish64, dakanji, PMheart, and several others, with the full list available in OpenCorePkg history.

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 data, 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 multidata 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 (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.

2 Configuration

2.1 Configuration Terms

- OC config OpenCore Configuration file in plist format named config.plist. It provides an extensible way to configure OpenCore and is structured to be separated into multiple named sections situated under the root plist dictionary. These sections may have plist array or plist dictionary types and are described in corresponding sections of this document.
- valid key plist key object of OC config described in this document or its future revisions. Besides explicitly
 described valid keys, keys starting with the # symbol (e.g. #Hello) are also considered valid keys and while
 they behave as comments, effectively discarding their values, they are still required to be valid plist objects.
 All other plist keys are not valid, and their presence results in undefined behaviour.
- valid value valid plist object of OC config described in this document that matches all the additional requirements in specific plist object descriptions if any.
- invalid value valid plist object of OC config described in this document that is of other plist type, does not conform to additional requirements found in specific plist object descriptions (e.g. value range), or missing from the corresponding collection. Invalid values are read with or without an error message as any possible value of this plist object in an undetermined manner (i.e. the values may not be same across the reboots). Whilst reading an invalid value is equivalent to reading certain defined valid values, applying incompatible values to the host system may result in undefined behaviour.
- optional value valid value of OC config described in this document that reads in a certain defined manner provided in specific plist object description (instead of invalid value) when not present in OC config. All other cases of invalid value do still apply. Unless explicitly marked as optional value, any other value is required to be present and reads to invalid value if missing.
- fatal behaviour behaviour leading to boot termination. Implementations shall prevent the boot process from continuing until the host system is restarted. It is permitted, but not required, to execute cold reboots or to show warning messages in such cases.
- undefined behaviour behaviour not prescribed by this document. Implementations may take any measures including, but not limited to, measures associated with fatal behaviour, assumptions of any state or value, or disregarding any associated states or values. This is however subject to such measures not negatively impacting upon system integrity.

2.2 Configuration Processing

The OC config file is guaranteed to be processed at least once if found. Subject to the OpenCore bootstrapping mechanism, the presence of multiple OC config files may lead to the reading of any of them. It is permissible for no OC Config file to be present on disk. In such cases, if the implementation does not abort the boot process, all values shall follow the rules of invalid values and optional values.

The OC config file has restrictions on size, nesting levels, and number of keys:

- The OC config file size shall not exceed 32 MBs.
- The OC config file shall not have more than 32 nesting levels.
- The OC config file may have up to 32,768 XML nodes within each plist object.
 - One plist dictionary item is counted as a pair of nodes

Reading malformed OC config files results in undefined behaviour. Examples of malformed OC config files include the following:

- OC config files that do not conform to DTD PLIST 1.0.
- OC config files with unsupported or non-conformant plist objects found in this document.
- OC config files violating restrictions on size, nesting levels, and number of keys.

It is recommended, but not required, to abort loading malformed OC config files and to continue as if an OC config file is not present. For forward compatibility, it is recommended, but not required, for the implementation to warn about the use of invalid values.

The recommended approach to interpreting invalid values is to conform to the following convention where applicable:

Type	Value
plist string	Empty string (<string></string>)
plist data	Empty data (<data></data>)
plist integer	0 (<integer>0</integer>)
plist boolean	False (<false></false>)
plist tristate	False (<false></false>)

2.3 Configuration Structure

The OC config file is separated into subsections, as described in separate sections of this document, and is designed so as to attempt not to enable anything by default as well as to provide kill switches via an Enable property for plist dict entries that represent optional plugins and similar.

The file is structured to group related elements in subsections as follows:

- Add provides support for data addition. Existing data will not be overridden, and needs to be handled separately with Delete if necessary.
- Delete provides support for data removal.
- Patch provides support for data modification.
- Quirks provides support for specific workarounds.

Root configuration entries consist of the following:

- ACPI
- Booter
- DeviceProperties
- Kernel
- Misc
- NVRAM
- PlatformInfo
- UEFI

Basic validation of an OC config file is possible using the ocvalidate utility. Please note that the version of ocvalidate used must match the OpenCore release and that notwithstanding this, it may not detect all configuration issues present in an OC config file.

Note: To maintain system integrity, properties typically have predefined values even when such predefined values are not specified in the OC config file. However, all properties must be explicitly specified in the OC config file and this behaviour should not be relied on.

3 Setup

3.1 Directory Structure



Figure 1. Directory Structure

When directory boot is used, the directory structure used should follow the descriptions in the Directory Structure figure. Available entries include:

• BOOTx64.efi or BOOTIa32.efi

Initial bootstrap loaders, which load <code>OpenCore.efi</code>. <code>BOOTx64.efi</code> is loaded by the firmware by default consistent with the UEFI specification. However, it may also be renamed and put in a custom location to allow <code>OpenCore</code> coexist alongside operating systems, such as <code>Windows</code>, that use <code>BOOTx64.efi</code> files as their loaders. Refer to the <code>LauncherOption</code> property for details.

boot

Duet bootstrap loader, which initialises the UEFI environment on legacy BIOS firmware and loads OpenCore.efi similarly to other bootstrap loaders. A modern Duet bootstrap loader will default to OpenCore.efi on the same partition when present.

• ACPI

Directory used for storing supplemental ACPI information for the ACPI section.

• Drivers

Directory used for storing supplemental UEFI drivers for UEFI section.

Kexts

Directory used for storing supplemental kernel information for the Kernel section.

• Resources

Directory used for storing media resources such as audio files for screen reader support. Refer to the UEFI Audio Properties section for details. This directory also contains image files for graphical user interface. Refer to the OpenCanopy section for details.

• Tools

Directory used for storing supplemental tools.

• OpenCore.efi

Main booter application responsible for operating system loading. The directory OpenCore.efi resides in is called the root directory, which is set to EFI\OC by default. When launching OpenCore.efi directly or through a custom launcher however, other directories containing OpenCore.efi files are also supported.

• config.plist

OC Config.

• vault.plist

Hashes for all files potentially loadable by OC Config.

• vault.sig

Signature for vault.plist.

• SysReport

Directory containing system reports generated by SysReport option.

nvram.plist

OpenCore variable import file.

• nvram.fallback

OpenCore variable import fallback file.

• nvram.used

Renamed previous OpenCore variable import file after switch to fallback file.

• opencore-YYYY-MM-DD-HHMMSS.txt

OpenCore log file.

• panic-YYYY-MM-DD-HHMMSS.txt

Kernel panic log file.

Note: It is not guaranteed that paths longer than OC_STORAGE_SAFE_PATH_MAX (128 characters including the O-terminator) will be accessible within OpenCore.

3.2 Installation and Upgrade

To install OpenCore, replicate the Configuration Structure described in the previous section in the EFI volume of a GPT partition. While corresponding sections of this document provide some information regarding external resources such as ACPI tables, UEFI drivers, or kernel extensions (kexts), completeness of the matter is out of the scope of this document. Information about kernel extensions may be found in a separate Kext List document available in the OpenCore repository. Vaulting information is provided in the Security Properties section of this document.

The OC config file, as with any property list file, can be edited with any text editor, such as nano or vim. However, specialised software may provide a better experience. On macOS, the preferred GUI application is Xcode. The ProperTree editor is a lightweight, cross-platform and open-source alternative.

It is strongly recommended to avoid configuration creation tools that are aware of the internal configuration structure as this may result in invalid configurations (since the structure gets constantly updated). If such tools are to be used despite this warning, ensure that only stable versions of OpenCore explicitly supported by such tools are used. In such cases, the use of open-source implementations with transparent binary generation (such as OCAT) is encouraged, given that other tools may contain malware. In addition, configurations created for a specific hardware setup should never be used on different hardware setups.

For BIOS booting, a third-party UEFI environment provider is required and <code>OpenDuetPkg</code> is one such UEFI environment provider for legacy systems. To run OpenCore on such a legacy system, <code>OpenDuetPkg</code> can be installed with a dedicated tool — BootInstall (bundled with OpenCore). Third-party utilities can be used to perform this on systems other than macOS.

For upgrade purposes, refer to the Differences.pdf document which provides information about changes to the configuration (as compared to the previous release) as well as to the Changelog.md document (which contains a list of modifications across all published updates).

3.3 Contribution

OpenCore can be compiled as a standard EDK II package and requires the EDK II Stable package. The currently supported EDK II release is hosted in acidanthera/audk. Required patches for this package can be found in the Patches directory.

When updating the LaTeX documentation (e.g. Configuration.tex) please do *not* rebuild the PDF files till merging to master happens. This avoids unnecessary merge conflicts:

- External contributors using the pull-request approach should request the maintainers to handle the PDF rebuild in the pull-request message.
- Internal contributors should rebuild the documentation at merge time in the same or in a separate commit. One can ask another maintainer to rebuild the documentation when lacking the necessary tools in the pull-request message.

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

To compile with XCODE5, besides Xcode, users should also install NASM and MTOC. The latest Xcode version is recommended for use despite the toolchain name. An example command sequence is as follows:

```
git clone --depth=1 https://github.com/acidanthera/audk UDK
cd UDK
git submodule update --init --recommend-shallow
rm -rf OpenCorePkg
git clone --depth=1 https://github.com/acidanthera/OpenCorePkg
. ./edksetup.sh
make -C BaseTools
build -a X64 -b RELEASE -t XCODE5 -p OpenCorePkg/OpenCorePkg.dsc
```

Listing 1: Compilation Commands

For IDE usage Xcode projects are available in the root of the repositories. Another approach could be using Language Server Protocols. For example, Sublime Text with LSP for Sublime Text plugin. Add compile_flags.txt file with similar content to the UDK root:

- -I/UefiPackages/MdePkg
- -I/UefiPackages/MdePkg/Include
- -I/UefiPackages/MdePkg/Include/X64
- -I/UefiPackages/MdeModulePkg
- -I/UefiPackages/MdeModulePkg/Include
- -I/UefiPackages/MdeModulePkg/Include/X64

- -I/UefiPackages/OpenCorePkg/Include/AMI
- -I/UefiPackages/OpenCorePkg/Include/Acidanthera
- -I/UefiPackages/OpenCorePkg/Include/Apple
- -I/UefiPackages/OpenCorePkg/Include/Apple/X64
- -I/UefiPackages/OpenCorePkg/Include/Duet
- -I/UefiPackages/OpenCorePkg/Include/Generic
- -I/UefiPackages/OpenCorePkg/Include/Intel
- -I/UefiPackages/OpenCorePkg/Include/Microsoft
- -I/UefiPackages/OpenCorePkg/Include/Nvidia
- -I/UefiPackages/OpenCorePkg/Include/VMware
- -I/UefiPackages/OvmfPkg/Include
- -I/UefiPackages/ShellPkg/Include
- -I/UefiPackages/UefiCpuPkg/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
- -DOC_TARGET_NOOPT=1
- -DNO MSABI VA FUNCS=1

Listing 2: ECC Configuration

Note: /UefiPackages in the sample file denotes an absolute path.

Warning: Tool developers modifying config.plist or any other OpenCore files must ensure that their tools check the opencore-version NVRAM variable (see the Debug Properties section below) and warn users if the version listed is unsupported or prerelease. The OpenCore configuration may change across releases and such tools shall ensure that they carefully follow this document. Failure to do so may result in such tools being considered to be malware and blocked by any means.

3.4 Coding conventions

As with any other project, we have conventions that we follow during development. All third-party contributors are advised to adhere to the conventions listed below before submitting patches. To minimise abortive work and the potential rejection of submissions, third-party contributors should initially raise issues to the Acidanthera Bugtracker for feedback before submitting patches.

Organisation. The codebase is contained in the OpenCorePkg repository, which is the primary EDK II package.

- Whenever changes are required in multiple repositories, separate pull requests should be sent to each.
- Committing the changes should happen firstly to dependent repositories, secondly to primary repositories to avoid automatic build errors.
- Each unique commit should compile with XCODE5 and preferably with other toolchains. In the majority of the cases it can be checked by accessing the CI interface. Ensuring that static analysis finds no warnings is preferred.
- External pull requests and tagged commits must be validated. That said, commits in master may build but may not necessarily work.
- Internal branches should be named as follows: author-name-date, e.g. vit9696-ballooning-20191026.
- Commit messages should be prefixed with the primary module (e.g. library or code module) the changes were made in. For example, OcGuardLib: Add OC_ALIGNED macro. For non-library changes Docs or Build prefixes are used.

Design. The codebase is written in a subset of freestanding C11 (C17) supported by most modern toolchains used by EDK II. Applying common software development practices or requesting clarification is recommended if any particular case is not discussed below.

- Never rely on undefined behaviour and try to avoid implementation defined behaviour unless explicitly covered below (feel free to create an issue when a relevant case is not present).
- Use OcGuardLib to ensure safe integral arithmetics avoiding overflows. Unsigned wraparound should be relied on with care and reduced to the necessary amount.
- Check pointers for correct alignment with OcGuardLib and do not rely on the architecture being able to dereference unaligned pointers.
- Use flexible array members instead of zero-length or one-length arrays where necessary.
- Use static assertions (STATIC_ASSERT) for type and value assumptions, and runtime assertions (ASSERT) for precondition and invariant sanity checking. Do not use runtime assertions to check for errors as they should never alter control flow and potentially be excluded.
- Assume UINT32/INT32 to be int-sized and use %u, %d, and %x to print them.
- Assume UINTN/INTN to be of unspecified size, and cast them to UINT64/INT64 for printing with %Lu, %Ld and so
 on as normal.
- Do not rely on integer promotions for numeric literals. Use explicit casts when the type is implementation-dependent or suffixes when type size is known. Assume U for UINT32 and ULL for UINT64.
- Do ensure unsigned arithmetics especially in bitwise maths, shifts in particular.
- sizeof operator should take variables instead of types where possible to be error prone. Use ARRAY_SIZE to obtain array size in elements. Use L_STR_LEN and L_STR_SIZE macros from OcStringLib to obtain string literal sizes to ensure compiler optimisation.
- Do not use goto keyword. Prefer early return, break, or continue after failing to pass error checking instead of nesting conditionals.
- Use EFIAPI, force UEFI calling convention, only in protocols, external callbacks between modules, and functions with variadic arguments.
- Provide inline documentation to every added function, at least describing its inputs, outputs, precondition, postcondition, and giving a brief description.
- Do not use RETURN_STATUS. Assume EFI_STATUS to be a matching superset that is to be always used when BOOLEAN is not enough.
- Security violations should halt the system or cause a forced reboot.

Codestyle. The codebase follows the EDK II codestyle with a few changes and clarifications.

- Write inline documentation for the functions and variables only once: in headers, where a header prototype is available, and inline for static variables and functions.
- Use line length of 120 characters or less, preferably 100 characters.
- Use spaces after casts, e.g. (VOID *)(UINTN) Variable.
- Use two spaces to indent function arguments when splitting lines.
- Prefix public functions with either Oc or another distinct name.
- Do not prefix private static functions, but prefix private non-static functions with Internal.
- Use SPDX license headers as shown in acidanthera/bugtracker#483.

3.5 Debugging

The codebase incorporates EDK II debugging and few custom features to improve the experience.

- Use module prefixes, 2-5 letters followed by a colon (:), for debug messages. For OpenCorePkg use OC:, for libraries and drivers use their own unique prefixes.
- Do not use dots (.) in the end of debug messages and separate EFI_STATUS, printed by %r, with a hyphen (e.g. OCRAM: Allocation of %u bytes failed %r\n).
- Use DEBUG_CODE_BEGIN () and DEBUG_CODE_END () constructions to guard debug checks that may potentially reduce the performance of release builds and are otherwise unnecessary.
- Use DEBUG macro to print debug messages during normal functioning, and RUNTIME_DEBUG for debugging after EXIT_BOOT_SERVICES.
- Use DEBUG_VERBOSE debug level to leave debug messages for future debugging of the code, which are currently not necessary. By default DEBUG_VERBOSE messages are ignored even in DEBUG builds.
- Use DEBUG_INFO debug level for all non critical messages (including errors) and DEBUG_BULK_INFO for extensive messages that should not appear in NVRAM log that is heavily limited in size. These messages are ignored in

RELEASE builds.

• Use DEBUG_ERROR to print critical human visible messages that may potentially halt the boot process, and DEBUG_WARN for all other human visible errors, RELEASE builds included.

The git-bisect functionality may be useful when trying to find problematic changes. Unofficial sources of per-commit OpenCore binary builds, such as Dortania, may also be useful.

4 ACPI

4.1 Introduction

ACPI (Advanced Configuration and Power Interface) is an open standard to discover and configure computer hardware. The ACPI specification defines standard tables (e.g. DSDT, SSDT, FACS, DMAR) and various methods (e.g. _DSM, _PRW) for implementation. Modern hardware needs few changes to maintain ACPI compatibility and some options for such changes are provided as part of OpenCore.

To compile and disassemble ACPI tables, the iASL compiler developed by ACPICA can be used. A GUI front-end to iASL compiler can be downloaded from Acidanthera/MaciASL.

ACPI changes apply globally (to every operating system) with the following effective order:

- Delete is processed.
- Quirks are processed.
- Patch is processed.
- Add is processed.

Note: RebaseRegions and SyncTableIds quirks are special and are processed after all other ACPI changes since they can only be applied on the final ACPI configuration including all the patches and added tables.

Applying the changes globally resolves the problems of incorrect operating system detection (consistent with the ACPI specification, not possible before the operating system boots), operating system chainloading, and difficult ACPI debugging. Hence, more attention may be required when writing changes to _OSI.

Applying the patches early makes it possible to write so called "proxy" patches, where the original method is patched in the original table and is implemented in the patched table.

There are several sources of ACPI tables and workarounds. Commonly used ACPI tables are provided with OpenCore, VirtualSMC, VoodooPS2, and WhateverGreen releases. Besides those, several third-party instructions may be found on the AppleLife Laboratory and DSDT subforums (e.g. Battery register splitting guide). A slightly more user-friendly explanation of some tables included with OpenCore can also be found in Dortania's Getting started with ACPI guide. For more exotic cases, there are several alternatives such as daliansky's ACPI sample collection (English Translation by 5T33Z0 et al). Please note however, that suggested solutions from third parties may be outdated or may contain errors.

4.2 Properties

1. Add

Type: plist array Failsafe: Empty

Description: Load selected tables from the OC/ACPI directory.

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

2. Delete

Type: plist array Failsafe: Empty

Description: Remove selected tables from the ACPI stack.

To be filled with plist dict values, describing each delete entry. Refer to the Delete Properties section below for details.

3. Patch

Type: plist array Failsafe: Empty

Description: Perform binary patches in ACPI tables before table addition or removal.

To be filled with plist dictionary values describing each patch entry. Refer to the Patch Properties section below for details.

4. Quirks

Type: plist dict

Description: Apply individual ACPI quirks described in the Quirks Properties section below.

4.3 Add Properties

1. Comment

Type: plist string Failsafe: Empty

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

2. Enabled

Type: plist boolean Failsafe: false

Description: Set to true to add this ACPI table.

3. Path

Type: plist string Failsafe: Empty

Description: File paths meant to be loaded as ACPI tables. Example values include DSDT.aml, SubDir/SSDT-8.aml, SSDT-USBX.aml, etc.

The ACPI table load order follows the item order in the array. ACPI tables are loaded from the OC/ACPI directory.

Note: All tables apart from tables with a DSDT table identifier (determined by parsing data, not by filename) insert new tables into the ACPI stack. DSDT tables perform a replacement of DSDT tables instead.

4.4 Delete Properties

1. All

Type: plist boolean

Failsafe: false (Only delete the first matched table)

Description: Set to true to delete all ACPI tables matching the condition.

2. Comment

Type: plist string Failsafe: Empty

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

3. Enabled

Type: plist boolean Failsafe: false

Description: Set to true to remove this ACPI table.

4. OemTableId

Type: plist data, 8 bytes

Failsafe: All zero (Match any table OEM ID)

Description: Match table OEM ID equal to this value.

5. TableLength

Type: plist integer

Failsafe: 0 (Match any table size)

Description: Match table size equal to this value.

6. TableSignature

Type: plist data, 4 bytes

Failsafe: All zero (Match any table signature)

Description: Match table signature equal to this value.

Note: Do not use table signatures when the sequence must be replaced in multiple places. This is particularly relevant when performing different types of renames.

4.5 Patch Properties

1. Base

Type: plist string Failsafe: Empty (Ignored)

Description: Selects ACPI path base for patch lookup (or immediate replacement) by obtaining the offset to

the provided path.

Only fully-qualified absolute paths are supported (e.g. _SB.PCIO.LPCB.HPET). Currently supported object types are: Device, Field, Method.

Note: Use with care, not all OEM tables can be parsed. Use ACPIe utility to debug. ACPIe compiled with DEBUG=1 make command produces helpful ACPI lookup tracing.

2. BaseSkip

Type: plist integer

Failsafe: 0 (Do not skip any occurrences)

Description: Number of found Base occurrences to skip before finds and replacements are applied.

3. Comment

Type: plist string Failsafe: Empty

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

4. Count

Type: plist integer

Failsafe: 0 (Apply patch to all occurrences found) Description: Number of occurrences to patch.

5. Enabled

Type: plist boolean Failsafe: false

Description: Set to true to apply this ACPI patch.

6. Find

Type: plist data Failsafe: Empty

Description: Data to find. Must be equal to Replace in size if set.

Note: Can be empty, when Base is specified, immediate replacement after Base lookup happens in this case.

7. Limit

Type: plist integer

Failsafe: 0 (Search entire ACPI table)

Description: Maximum number of bytes to search for.

$8. \, {\tt Mask}$

Type: plist data

Failsafe: Empty (Ignored)

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

9. OemTableId

Type: plist data, 8 bytes

Failsafe: All zero (Match any table OEM ID)

Description: Match table OEM ID equal to this value.

10. Replace

Type: plist data Failsafe: Empty

Description: Replacement data of one or more bytes.

11. ReplaceMask

Type: plist data

Failsafe: Empty (Ignored)

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

12. Skip

Type: plist integer

Failsafe: 0 (Do not skip any occurrences)

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

13. TableLength

Type: plist integer

Failsafe: 0 (Match any table size)

Description: Match table size equal to this value.

14. TableSignature

Type: plist data, 4 bytes

Failsafe: All zero (Match any table signature)

Description: Match table signature equal to this value.

In most cases, ACPI patches are not useful and are harmful:

- Avoid renaming devices with ACPI patches. This may fail or perform improper renaming of unrelated devices (e.g. EC and ECO), be unnecessary, or even fail to rename devices in certain tables. For ACPI consistency it is much safer to rename devices at the I/O Registry level, as done by WhateverGreen.
- Avoid patching _OSI to support a higher feature set level whenever possible. While this enables a number of workarounds on APTIO firmware, it typically results in a need for additional patches. These are not usually needed on modern firmware and smaller patches work well on firmware that does. However, laptop vendors often rely on this method to determine the availability of functions such as modern I2C input support, thermal adjustment and custom feature additions.
- Avoid patching embedded controller event <code>_Qxx</code> just to enable brightness keys. The conventional process to find these keys typically involves significant modifications to DSDT and SSDT files and in addition, the debug kext is not stable on newer systems. Please use the built-in brightness key discovery in BrightnessKeys instead.
- Avoid making ad hoc changes such as renaming _PRW or _DSM whenever possible.

Some cases where patching is actually useful 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 AO 10 93 4F 53 46 4C 00 with A4 0A 0F A3 A3 A3 A3 A3.
- To provide a custom method implementation within an SSDT, to inject shutdown fixes on certain computers for instance, the original method can be replaced with a dummy name by patching _PTS with ZPTS and adding a callback to the original method.

The Tianocore AcpiAml.h source file may help with better understanding ACPI opcodes.

Note: Patches of different Find and Replace lengths are unsupported as they may corrupt ACPI tables and make the system unstable due to area relocation. If such changes are needed, the utilisation of "proxy" patching or the padding of NOP to the remaining area could be considered.

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 a few newer 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 flaws that result in boot crashes. Reference: Debugging AppleACPIPlatform on 10.13 by Alex James (also known as theracermaster). The issue was 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 the underlying firmware implementation. Among the position-independent code, ACPI tables may contain the physical addresses of MMIO areas used for device configuration, typically grouped by region (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 results in the shift of the addresses in the aforementioned OperationRegion constructions.

For this reason, the application of modifications to ACPI tables is extremely risky. The best approach is to make as few changes as possible to ACPI tables and to avoid replacing any tables, particularly DSDT tables. When this cannot be avoided, ensure that any custom DSDT tables are based on the most recent DSDT tables or attempt to remove reads and writes 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 is not a magic bullet however, and only works with the most typical cases. Do not use unless absolutely required as it can have the opposite effect on certain platforms and result in boot failures.

4. ResetHwSig

Type: plist boolean Failsafe: false

Description: Reset FACS table HardwareSignature value to 0.

This works around firmware that fail to maintain hardware signature across the reboots and cause issues with waking from hibernation.

5. ResetLogoStatus

Type: plist boolean Failsafe: false

Description: Reset BGRT table Displayed status field to false.

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

6. SyncTableIds

Type: plist boolean Failsafe: false

Description: Sync table identifiers with the SLIC table.

This works around patched tables becoming incompatible with the SLIC table causing licensing issues in older Windows operating systems.

5 Booter

5.1 Introduction

This section allows the application of different types of UEFI modifications to operating system bootloaders, primarily the Apple bootloader (boot.efi). The modifications currently provide various patches and environment alterations for different firmware types. Some of these features were originally implemented as part of AptioMemoryFix.efi, which is no longer maintained. Refer to the Tips and Tricks section for instructions on migration.

Booter changes apply with the following effective order:

- Quirks are processed.
- Patch is processed.

If this is used for the first time on customised firmware, the following requirements should be met before starting:

- Most up-to-date UEFI firmware (check the 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
 the ASUS WS-X299-PRO, this option results in adverse effects and must be disabled. While no other motherboards
 with the same issue are known, this option should be checked first whenever erratic boot failures are encountered.
- DisableIoMapper quirk enabled, or VT-d disabled in firmware settings if present, or ACPI DMAR table deleted.
- No 'slide' boot argument present in NVRAM or anywhere else. It is not necessary unless the system cannot be booted at all or No slide values are usable! Use custom slide! message can be seen in the log.
- CFG Lock (MSR 0xE2 write protection) disabled in firmware settings if present. Refer to the ControlMsrE2 notes for details.
- CSM (Compatibility Support Module) disabled in firmware settings if present. On NVIDIA 6xx/AMD 2xx or older, GOP ROM may have to be flashed first. Use GopUpdate (see the second post) or AMD UEFI GOP MAKER in case of any potential confusion.
- 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 Thunderbolt support, Intel SGX, and Intel Platform Trust may have to be disabled in firmware settings present.

When debugging sleep issues, Power Nap and automatic power off (which appear to sometimes cause wake to black screen or boot loop issues on older platforms) may be temporarily disabled. The specific issues may vary, but ACPI tables should typically be looked at first.

Here is an example of a defect found on 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 settings may be reset by hardware changes and in certain other circumstances. To view their current state, use the pmset -g command in Terminal.

5.2 Properties

1. MmioWhitelist

Type: plist array
Failsafe: Empty

Description: To be filled with plist dict values, describing addresses critical for particular firmware functioning when DevirtualiseMmio quirk is in use. Refer to the MmioWhitelist Properties section below for details.

2. Patch

Type: plist array Failsafe: Empty

Description: Perform binary patches in booter.

To be filled with plist dictionary values, describing each patch. Refer to the Patch Properties section below for details.

3. Quirks

Type: plist dict

Description: Apply individual booter quirks described in the Quirks Properties section below.

5.3 MmioWhitelist Properties

1. Address

Type: plist integer

Failsafe: 0

Description: Exceptional MMIO address, which memory descriptor should be left virtualised (unchanged) by <code>DevirtualiseMmio</code>. This means that the firmware will be able to directly communicate with this memory region during operating system functioning, because the region this value is in will be assigned a virtual address.

The addresses written here must be part of the memory map, have EfiMemoryMappedIO type and EFI_MEMORY_RUNTIME attribute (highest bit) set. The debug log can be used to find the list of the candidates.

2. Comment

Type: plist string Failsafe: Empty

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

3. Enabled

Type: plist boolean Failsafe: false

Description: Exclude MMIO address from the devirtualisation procedure.

5.4 Patch Properties

1. Arch

Type: plist string

Failsafe: Any (Apply to any supported architecture)

Description: Booter patch architecture (i386, x86_64).

2. Comment

Type: plist string Failsafe: Empty

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

3. Count

Type: plist integer

Failsafe: 0 (Apply to all occurrences found)

Description: Number of patch occurrences to apply.

4. Enabled

Type: plist boolean

Failsafe: false

Description: Set to true to activate this booter patch.

5. Find

Type: plist data Failsafe: Empty

Description: Data to find. Must be equal to Replace in size if set.

6. Identifier

Type: plist string

Failsafe: Any (Match any booter)

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

7. Limit

Type: plist integer

Failsafe: 0 (Search the entire booter)

Description: Maximum number of bytes to search for.

8. Mask

Type: plist data
Failsafe: Empty (Ignored)

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

9. Replace

Type: plist data Failsafe: Empty

Description: Replacement data of one or more bytes.

10. ReplaceMask

Type: plist data
Failsafe: Empty (Ignored)

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

11. Skip

Type: plist integer

Failsafe: 0 (Do not skip any occurrences)

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

5.5 Quirks Properties

1. AllowRelocationBlock

Type: plist boolean

Failsafe: false

Description: Allows booting macOS through a relocation block.

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

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

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

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

2. AvoidRuntimeDefrag

Type: plist boolean

Failsafe: false

Description: Protect from boot.efi runtime memory defragmentation.

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

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

3. ClearTaskSwitchBit

Type: plist boolean

Failsafe: false

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

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

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

4. DevirtualiseMmio

Type: plist boolean

Failsafe: false

Description: Remove runtime attribute from certain MMIO regions.

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

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

5. DisableSingleUser

Type: plist boolean

Failsafe: false

Description: Disable single user mode.

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

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

6. DisableVariableWrite

Type: plist boolean

Failsafe: false

Description: Protect from macOS NVRAM write access.

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

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

7. DiscardHibernateMap

Type: plist boolean

Failsafe: false

Description: Reuse original hibernate memory map.

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

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

8. EnableSafeModeSlide

Type: plist boolean

Failsafe: false

Description: Patch bootloader to have KASLR enabled in safe mode.

This option is relevant to users with issues booting to safe mode (e.g. by holding shift or with using the -x boot argument). By default, safe mode forces 0 slide as if the system was launched with the slide=0 boot argument.

- This quirk attempts to patch the boot.efi file to remove this limitation and to allow using other values (from 1 to 255 inclusive).
- This quirk requires enabling ProvideCustomSlide.

Note: The need for this option is dependent on the availability of safe mode. It can be enabled when booting to safe mode fails.

9. EnableWriteUnprotector

Type: plist boolean

Failsafe: false

Description: Permit write access to UEFI runtime services code.

This option bypasses W^X permissions in code pages of UEFI runtime services by removing write protection (WP) bit from CRO register during their execution. This quirk requires OC_FIRMWARE_RUNTIME protocol implemented in OpenRuntime.efi.

Note: This quirk may potentially weaken firmware security. Please use RebuildAppleMemoryMap if the firmware supports memory attributes table (MAT). Refer to the OCABC: MAT support is 1/0 log entry to determine whether MAT is supported.

10. FixupAppleEfiImages

Type: plist boolean

Failsafe: false

Description: Fix permissions and section errors in macOS boot.efi images.

Mac OS X boot.efi images contain W^X permissions errors in all versions, and 10.4 and 10.5 32-bit versions also contain illegal overlapping sections. Modern, strict PE loaders will refuse to load such images unless additional mitigations are applied. The image loader which matters here is the one provided by the system firmware, or by OpenDuet if OpenDuet is providing the UEFI compatibility layer. Image loaders which enforce these stricter rules include the loader provided with current versions of OpenDuet, the loader in OVMF if compiled from audk, and possibly the image loaders of some very recent 3rd party firmware (e.g. Microsoft).

This quirk detects these issues and pre-processes such images in memory so that a stricter loader will accept them.

On a system with such a modern, stricter loader this quirk is required to load Mac OS X 10.4 to macOS 10.12, and is required for all newer macOS when SecureBootModel is set to Disabled.

Note 1: The quirk is never applied during the Apple secure boot path for newer macOS. The Apple secure boot path in OpenCore includes its own separate mitigations for boot.efi W^X issues.

Note 2: When enabled, and when not processing for Apple secure boot, this quirk is applied to:

- All images from Apple Fat binaries (32-bit and 64-bit versions in one image).
- All Apple-signed images.
- All images at \System\Library\CoreServices\boot.efi within their filesystem.

Note 3: Pre-processing in memory is incompatible with UEFI secure boot, as the image loaded is not the image on disk, so you cannot sign files which are loaded in this way based on their original disk image contents. Certain firmware will offer to register the hash of new, unknown images for future secure boot - this would still work. On the other hand, it is not particularly realistic to want to start these early, insecure images with secure boot anyway.

11. ForceBooterSignature

Type: plist boolean

Failsafe: false

Description: Set macOS boot-signature to OpenCore launcher.

Booter signature, essentially a SHA-1 hash of the loaded image, is used by Mac EFI to verify the authenticity of the bootloader when waking from hibernation. This option forces macOS to use OpenCore launcher SHA-1 hash as a booter signature to let OpenCore shim hibernation wake on Mac EFI firmware.

Note: OpenCore launcher path is determined from LauncherPath property.

12. ForceExitBootServices

 \mathbf{Type} : plist boolean

Failsafe: false

Description: Retry ExitBootServices with new memory map on failure.

Try to ensure that the ExitBootServices call succeeds. If required, an outdated MemoryMap key argument can be used by obtaining the current memory map and retrying the ExitBootServices call.

Note: The need for this quirk is determined by early boot crashes of the firmware. Do not use this option without a full understanding of the implications.

13. ProtectMemoryRegions

Type: plist boolean

Failsafe: false

Description: Protect memory regions from incorrect access.

Some types of firmware incorrectly map certain memory regions:

- The CSM region can be marked as boot services code, or data, which leaves it as free memory for the XNU kernel.
- MMIO regions can be marked as reserved memory and stay unmapped. They may however be required to be accessible at runtime for NVRAM support.

This quirk attempts to fix the types of these regions, e.g. ACPI NVS for CSM or MMIO for MMIO.

Note: The need for this quirk is determined by artifacts, sleep wake issues, and boot failures. This quirk is typically only required by very old firmware.

14. ProtectSecureBoot

 $\mathbf{Type} {:}\ \mathtt{plist}\ \mathtt{boolean}$

Failsafe: false

Description: Protect UEFI Secure Boot variables from being written.

Reports security violation during attempts to write to db, dbx, PK, and KEK variables from the operating system.

Note: This quirk attempts to avoid issues with NVRAM implementations with fragmentation issues, such as on the MacPro5,1 as well as on certain Insyde firmware without garbage collection or with defective garbage collection.

15. ProtectUefiServices

Type: plist boolean

Failsafe: false

Description: Protect UEFI services from being overridden by the firmware.

Some modern firmware, including on virtual machines such as VMware, may update pointers to UEFI services during driver loading and related actions. Consequently, this directly obstructs other quirks that affect memory management, such as DevirtualiseMmio, ProtectMemoryRegions, or RebuildAppleMemoryMap, and may also obstruct other quirks depending on the scope of such.

GRUB Shim makes similar on-the-fly changes to various UEFI image services, which are also protected against by this quirk.

Note 1: On VMware, the need for this quirk may be determined by the appearance of the "Your Mac OS guest might run unreliably with more than one virtual core." message.

Note 2: This quirk is needed for correct operation if OpenCore is chainloaded from GRUB with BIOS Secure Boot enabled.

16. ProvideCustomSlide

Type: plist boolean

Failsafe: false

Description: Provide custom KASLR slide on low memory.

This option performs memory map analysis of the firmware and checks whether all slides (from 1 to 255) can be used. As boot.efi generates this value randomly with rdrand or pseudo randomly rdtsc, there is a chance of boot failure when it chooses a conflicting slide. In cases where potential conflicts exist, this option forces macOS to select a pseudo random value from the available values. This also ensures that the slide= argument is never passed to the operating system (for security reasons).

Note: The need for this quirk is determined by the OCABC: Only N/256 slide values are usable! message in the debug log.

17. ProvideMaxSlide

Type: plist integer

Failsafe: 0

Description: Provide maximum KASLR slide when higher ones are unavailable.

This option overrides the maximum slide of 255 by a user specified value between 1 and 254 (inclusive) when ProvideCustomSlide is enabled. It is assumed that modern firmware allocates pool memory from top to bottom, effectively resulting in free memory when slide scanning is used later as temporary memory during kernel loading. When such memory is not available, this option stops the evaluation of higher slides.

Note: The need for this quirk is determined by random boot failures when ProvideCustomSlide is enabled and the randomized slide falls into the unavailable range. When AppleDebug is enabled, the debug log typically contains messages such as AAPL: [EB|'LD:LKC] } Err(0x9). To find the optimal value, append slide=X, where X is the slide value, to the boot-args and select the largest one that does not result in boot failures.

18. RebuildAppleMemoryMap

Type: plist boolean

Failsafe: false

Description: Generate macOS compatible Memory Map.

The Apple kernel has several limitations on parsing the UEFI memory map:

- The Memory map size must not exceed 4096 bytes as the Apple kernel maps it as a single 4K page. As some types of firmware can have very large memory maps, potentially over 100 entries, the Apple kernel will crash on boot.
- The Memory attributes table is ignored. EfiRuntimeServicesCode memory statically gets RX permissions while all other memory types get RW permissions. As some firmware drivers may write to global variables at runtime, the Apple kernel will crash at calling UEFI runtime services unless the driver .data section has a EfiRuntimeServicesData type.

To workaround these limitations, this quirk applies memory attribute table permissions to the memory map passed to the Apple kernel and optionally attempts to unify contiguous slots of similar types if the resulting memory map exceeds 4 KB.

Note 1: Since several types of firmware come with incorrect memory protection tables, this quirk often comes paired with SyncRuntimePermissions.

Note 2: The need for this quirk is determined by early boot failures. This quirk replaces ${\tt EnableWriteUnprotector}$ on firmware supporting Memory Attribute Tables (MAT). This quirk is typically unnecessary when using ${\tt OpenDuetPkg}$ but may be required to boot Mac OS X 10.6, and earlier, for reasons that are as yet unclear.

19. ResizeAppleGpuBars

Type: plist integer

Failsafe: -1

Description: Reduce GPU PCI BAR sizes for compatibility with macOS.

This quirk reduces GPU PCI BAR sizes for Apple macOS up to the specified value or lower if it is unsupported. The specified value follows PCI Resizable BAR spec. While Apple macOS supports a theoretical 1 GB maximum, in practice all non-default values may not work correctly. For this reason the only supported value for this quirk is the minimal supported BAR size, i.e. 0. Use -1 to disable this quirk.

For development purposes one may take risks and try other values. Consider a GPU with 2 BARs:

- BARO supports sizes from 256 MB to 8 GB. Its value is 4 GB.
- BAR1 supports sizes from 2 MB to 256 MB. Its value is 256 MB.

Example 1: Setting ResizeAppleGpuBars to 1 GB will change BARO to 1 GB and leave BAR1 unchanged.

Example 2: Setting ResizeAppleGpuBars to 1 MB will change BARO to 256 MB and BARO to 2 MB.

Example 3: Setting ResizeAppleGpuBars to 16 GB will make no changes.

Note: See ResizeGpuBars quirk for general GPU PCI BAR size configuration and more details about the technology.

20. SetupVirtualMap

Type: plist boolean

Failsafe: false

Description: Setup virtual memory at SetVirtualAddresses.

Some types of firmware access memory by virtual addresses after a SetVirtualAddresses call, resulting in early boot crashes. This quirk workarounds the problem by performing early boot identity mapping of assigned virtual addresses to physical memory.

Note: The need for this quirk is determined by early boot failures.

21. SignalAppleOS

Type: plist boolean

Failsafe: false

Description: Report macOS being loaded through OS Info for any OS.

This quirk is useful on Mac firmware, which loads different operating systems with different hardware configurations. For example, it is supposed to enable Intel GPU in Windows and Linux in some dual-GPU MacBook models.

22. SyncRuntimePermissions

Type: plist boolean

Failsafe: false

Description: Update memory permissions for the runtime environment.

Some types of firmware fail to properly handle runtime permissions:

- They incorrectly mark OpenRuntime as not executable in the memory map.
- They incorrectly mark OpenRuntime as not executable in the memory attributes table.
- They lose entries from the memory attributes table after OpenRuntime is loaded.
- They mark items in the memory attributes table as read-write-execute.

This quirk attempts to update the memory map and memory attributes table to correct this.

Note: The need for this quirk is indicated by early boot failures (note: includes halt at black screen as well as more obvious crash). Particularly likely to affect early boot of Windows or Linux (but not always both) on affected systems. Only firmware released after 2017 is typically affected.

6 DeviceProperties

6.1 Introduction

Device configuration is provided to macOS with a dedicated buffer, called EfiDevicePathPropertyDatabase. This buffer is a serialised map of DevicePaths to a map of property names and their values.

Property data can be debugged with gfxutil. To obtain current property data, use the following command in macOS:

```
ioreg -lw0 -p IODeviceTree -n efi -r -x | grep device-properties |
  sed 's/.*<//;s/>.*//' > /tmp/device-properties.hex &&
  gfxutil /tmp/device-properties.hex /tmp/device-properties.plist &&
  cat /tmp/device-properties.plist
```

Device properties are part of the IODeviceTree (gIODT) plane of the macOS I/O Registry. This plane has several construction stages relevant for the platform initialisation. While the early construction stage is performed by the XNU kernel in the IODeviceTreeAlloc method, the majority of the construction is performed by the platform expert, implemented in AppleACPIPlatformExpert.kext.

AppleACPIPlatformExpert incorporates two stages of IODeviceTree construction implemented by calling AppleACPIPlatformExpert::mergeDeviceProperties:

- 1. During ACPI table initialisation through the recursive ACPI namespace scanning by the calls to AppleACPIPlatformExpert::createDTNubs.
- 2. During IOService registration (IOServices::registerService) callbacks implemented as a part of AppleACPIPlatformExpert::platformAdjustService function and its private worker method AppleACPIPlatformExpert::platformAdjustPCIDevice specific to the PCI devices.

The application of the stages depends on the device presence in ACPI tables. The first stage applies very early but exclusively to the devices present in ACPI tables. The second stage applies to all devices much later after the PCI configuration and may repeat the first stage if the device was not present in ACPI.

For all kernel extensions that may inspect the IODeviceTree plane without probing, such as Lilu and its plugins (e.g. WhateverGreen), it is especially important to ensure device presence in the ACPI tables. A failure to do so may result in erratic behaviour caused by ignoring the injected device properties as they were not constructed at the first stage. See SSDT-IMEI.dsl and SSDT-BRGO.dsl for an example.

6.2 Properties

1. Add

Type: plist dict

Description: Sets device properties from a map (plist dict) of device paths to a map (plist dict) of variable names and their values in plist multidata format.

Note 1: Device paths must be provided in canonic string format (e.g. PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x0)).

Note 2: Existing properties will not be changed unless deleted in the DeviceProperties Delete section.

2. Delete

Type: plist dict

Description: Removes device properties from a map (plist dict) of device paths to an array (plist array) of variable names in plist string format.

Note: Currently, existing properties may only exist on firmware with DeviceProperties drivers (e.g. Apple). Hence, there is typically no reason to delete variables unless a new driver has been installed.

6.3 Common Properties

Some known properties include:

• device-id

User-specified device identifier used for I/O Kit matching. Has 4 byte data type.

- vendor-id
 - User-specified vendor identifier used for I/O Kit matching. Has 4 byte data type.
- AAPL, ig-platform-id
 - Intel GPU framebuffer identifier used for framebuffer selection on Ivy Bridge and newer. Has 4 byte data type.
- AAPL,snb-platform-id
 - Intel GPU framebuffer identifier used for framebuffer selection on Sandy Bridge. Has 4 byte data type.
- layout-id

Audio layout used for AppleHDA layout selection. Has 4 byte data type.

7 Kernel

7.1 Introduction

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

Kernel and kext changes apply with the following effective order:

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

7.2 Properties

1. Add

Type: plist array Failsafe: Empty

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

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

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

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

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

2. Block

Type: plist array Failsafe: Empty

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

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

3. Emulate

Type: plist dict

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

4. Force

Type: plist array Failsafe: Empty

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

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

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

Note 2: Force happens before Add.

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

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

5. Patch

Type: plist array Failsafe: Empty

Description: Perform binary patches in kernel and drivers prior to driver addition and removal.

To be filled with plist dictionary values, describing each patch. Refer to the Patch Properties section below for details.

6. Quirks

Type: plist dict

Description: Apply individual kernel and driver quirks described in the Quirks Properties section below.

7. Scheme

Type: plist dict

Description: Define kernelspace operation mode via parameters described in the Scheme Properties section

7.3 Add Properties

1. Arch

Type: plist string

Failsafe: Any (Apply to any supported architecture) **Description**: Kext architecture (i386, x86 64).

2. BundlePath

Type: plist string Failsafe: Empty

Description: Kext bundle path (e.g. Lilu.kext or MyKext.kext/Contents/PlugIns/MySubKext.kext).

3. Comment

Type: plist string Failsafe: Empty

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

4. Enabled

Type: plist boolean Failsafe: false

Description: Set to true to add this kernel extension.

5. ExecutablePath

Type: plist string
Failsafe: Empty

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

6. MaxKernel

Type: plist string Failsafe: Empty

Description: Adds kernel extension on specified macOS version or older.

Kernel version can be obtained with uname -r command, and should look like 3 numbers separated by dots, for example 18.7.0 is the kernel version for 10.14.6. Kernel version interpretation is implemented as follows:

 $ParseDarwinVersion(\kappa, \lambda, \mu) = \kappa \cdot 10000$ Where $\kappa \in (0, 99)$ is kernel version major $+\lambda \cdot 100$ Where $\lambda \in (0, 99)$ is kernel version minor $+\mu$ Where $\mu \in (0, 99)$ is kernel version patch

Kernel version comparison is implemented as follows:

$$\alpha = \begin{cases} ParseDarwinVersion(\texttt{MinKernel}), & \text{If MinKernel is valid} \\ 0 & Otherwise \end{cases}$$

$$\beta = \begin{cases} ParseDarwinVersion(\texttt{MaxKernel}), & \text{If MaxKernel is valid} \\ \infty & Otherwise \end{cases}$$

$$\gamma = \begin{cases} ParseDarwinVersion(FindDarwinVersion()), & \text{If valid "Darwin Kernel Version" is found} \\ 0 & Otherwise \end{cases}$$

 $f(\alpha, \beta, \gamma) = \alpha \le \gamma \le \beta$

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.

7. MinKernel

Type: plist string Failsafe: Empty

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

 $\it Note$: Refer to the Add MaxKernel description for matching logic.

8. PlistPath

Type: plist string Failsafe: Empty

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

7.4 Block Properties

1. Arch

Type: plist string

Failsafe: Any (Apply to any supported architecture)

Description: Kext block architecture (i386, x86_64).

2. Comment

Type: plist string Failsafe: Empty

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

3. Enabled

Type: plist boolean Failsafe: false

Description: Set to true to block this kernel extension.

4. Identifier

Type: plist string Failsafe: Empty

Description: Kext bundle identifier (e.g. com.apple.driver.AppleTyMCEDriver).

5. MaxKernel

Type: plist string Failsafe: Empty

Description: Blocks kernel extension on specified macOS version or older.

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

6. MinKernel

Type: plist string Failsafe: Empty

Description: Blocks kernel extension on specified macOS version or newer.

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

7. Strategy

Type: plist string

Failsafe: Disable (Forcibly make the kernel driver kmod startup code return failure)

Description: Determines the behaviour of kernel driver blocking.

Valid values:

- Disable Forcibly make the kernel driver kmod startup code return failure.
- Exclude Remove the kernel driver from the kernel cache by dropping plist entry and filling in zeroes.

Note: It is risky to Exclude a kext that is a dependency of others.

Note 2: At this moment Exclude is only applied to prelinkedkernel and newer mechanisms.

Note 3: In most cases strategy Exclude requires the new kext to be injected as a replacement.

7.5 Emulate Properties

1. Cpuid1Data

Type: plist data, 16 bytes

Failsafe: All zero

Description: Sequence of EAX, EBX, ECX, EDX values to replace CPUID (1) call in XNU kernel.

This property primarily meets three requirements:

- Enabling support for an unsupported CPU model (e.g. Intel Pentium).
- Enabling support for a CPU model not yet supported by a specific version of macOS (typically old versions).
- Enabling XCPM support for an unsupported CPU variant.

Note 1: It may also be the case that the CPU model is supported but there is no power management supported (e.g. virtual machines). In this case, MinKernel and MaxKernel can be set to restrict CPU virtualisation and dummy power management patches to the particular macOS kernel version.

Note 2: Only the value of EAX, which represents the full CPUID, typically needs to be accounted for and remaining bytes should be left as zeroes. The byte order is Little Endian. For example, C3 06 03 00 stands for CPUID 0x0306C3 (Haswell).

Note 3: For XCPM support it is recommended to use the following combinations. Be warned that one is required to set the correct frequency vectors matching the installed CPU.

- Haswell-E (0x0306F2) to Haswell (0x0306C3):
- Broadwell-E (0x0406F1) to Broadwell (0x0306D4):
- Comet Lake U62 (0x0A0660) to Comet Lake U42 (0x0806EC):
- Rocket Lake (0x0A0670) to Comet Lake (0x0A0655):
- Alder Lake (0x090672) to Comet Lake (0x0A0655):

Note 4: Be aware that the following configurations are unsupported by XCPM (at least out of the box):

- Consumer Ivy Bridge (0x0306A9) as Apple disabled XCPM for Ivy Bridge and recommends legacy power management for these CPUs. _xcpm_bootstrap should manually be patched to enforce XCPM on these CPUs instead of this option.
- Low-end CPUs (e.g. Haswell+ Pentium) as they are not supported properly by macOS. Legacy workarounds for older models can be found in the Special NOTES section of acidanthera/bugtracker#365.

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.

$3. \ {\tt DummyPowerManagement}$

Type: plist boolean
Failsafe: false

Requirement: 10.4-12

Description: Disables AppleIntelCpuPowerManagement.

Note 1: This option is a preferred alternative to NullCpuPowerManagement.kext for CPUs without native power management driver in macOS.

Note 2: While this option is typically needed to disable AppleIntelCpuPowerManagement on unsupported platforms, it can also be used to disable this kext in other situations (e.g. with Cpuid1Data left blank).

4. MaxKernel

Type: plist string Failsafe: Empty

Description: Emulates CPUID and applies DummyPowerManagement on specified macOS version or older.

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

5. MinKernel

Type: plist string Failsafe: Empty

Description: Emulates CPUID and applies DummyPowerManagement on specified macOS version or newer.

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

7.6 Force Properties

1. Arch

Type: plist string

Failsafe: Any (Apply to any supported architecture) Description: Kext architecture (i386, x86_64).

2. BundlePath

Type: plist string Failsafe: Empty

Description: Kext bundle path (e.g. System\Library \Extensions \IONetworkingFamily.kext).

3. Comment

Type: plist string Failsafe: Empty

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

4. Enabled

Type: plist boolean

Failsafe: false

Description: Set to true to load this kernel extension from the system volume when not present in the kernel cache.

5. ExecutablePath

Type: plist string Failsafe: Empty

Description: Kext executable path relative to bundle (e.g. Contents/MacOS/IONetworkingFamily).

6. Identifier

Type: plist string

Failsafe: Empty

Description: Kext identifier to perform presence checking before adding (e.g. com.apple.iokit.IONetworkingFamily). Only drivers which identifiers are not be found in the cache will be added.

7. MaxKernel

Type: plist string Failsafe: Empty

Description: Adds kernel extension on specified macOS version or older.

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

8. MinKernel

Type: plist string Failsafe: Empty

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

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

9. PlistPath

Type: plist string Failsafe: Empty

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

7.7 Patch Properties

1. Arch

Type: plist string

Failsafe: Any (Apply to any supported architecture)

Description: Kext patch architecture (i386, x86 64).

2. Base

Type: plist string
Failsafe: Empty (Ignored)

Description: Selects symbol-matched base for patch lookup (or immediate replacement) by obtaining the address of the provided symbol name.

3. Comment

Type: plist string Failsafe: Empty

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

4. Count

Type: plist integer

Failsafe: 0

Description: Number of patch occurrences to apply. 0 applies the patch to all occurrences found.

5. Enabled

Type: plist boolean Failsafe: false

Description: This kernel patch will not be used unless set to true.

 $6. \ {\tt Find}$

Type: plist data

Failsafe: Empty (Immediate replacement at Base)

Description: Data to find. Must be equal to Replace in size if set.

7. Identifier

Type: plist string Failsafe: Empty

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

8. Limit

Type: plist integer

Failsafe: 0 (Search entire kext or kernel)

Description: Maximum number of bytes to search for.

9. Mask

Type: plist data
Failsafe: Empty (Ignored)

Description: Data bitwise mask used during find comparison. Allows fuzzy search by ignoring not masked (set

to zero) bits. Must be equal to Replace in size if set.

10. MaxKernel

Type: plist string Failsafe: Empty

Description: Patches data on specified macOS version or older.

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

11. MinKernel

Type: plist string Failsafe: Empty

Description: Patches data on specified macOS version or newer.

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

12. Replace

Type: plist data Failsafe: Empty

Description: Replacement data of one or more bytes.

13. ReplaceMask

Type: plist data

Failsafe: Empty (Ignored)

Description: Data bitwise mask used during replacement. Allows fuzzy replacement by updating masked (set to

non-zero) bits. Must be equal to Replace in size if set.

14. Skip

Type: plist integer

Failsafe: 0 (Do not skip any occurrences)

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

7.8 Quirks Properties

1. AppleCpuPmCfgLock

Type: plist boolean Failsafe: false Requirement: 10.4

Description: Disables PKG_CST_CONFIG_CONTROL (0xE2) MSR modification in AppleIntelCPUPowerManage-

ment.kext, commonly causing early kernel panic, when it is locked from writing.

Note: AppleIntelCPUPowerManagement.kext is removed as of macOS 13. However, a legacy version can be injected and thus get patched using this quirk.

Some types of firmware lock the PKG_CST_CONFIG_CONTROL MSR register and the bundled ControlMsrE2 tool can be used to check its state. Note that some types of firmware only have this register locked on some cores. As modern firmware provide a CFG Lock setting that allows configuring the PKG_CST_CONFIG_CONTROL MSR register lock, this option should be avoided whenever possible.

On APTIO firmware that do not provide a CFG Lock setting in the GUI, it is possible to access the option directly:

- (a) Download UEFITool and IFR-Extractor.
- (b) Open the firmware image in UEFITool and find CFG Lock unicode string. If it is not present, the firmware may not have this option and the process should therefore be discontinued.
- (c) Extract the Setup.bin PE32 Image Section (the UEFITool found) through the 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 the 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.

On selected platforms, the ControlMsrE2 tool can also change such hidden options. Pass desired argument: lock, unlock for CFG Lock. Or pass interactive to find and modify other hidden options.

As a last resort, consider patching the BIOS (for advanced users only).

$2. \ {\tt AppleXcpmCfgLock}$

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. Refer to the AppleCpuPmCfgLock description for details.

3. AppleXcpmExtraMsrs

Type: plist boolean

Failsafe: false

Requirement: 10.8 (not required for older)

Description: Disables multiple MSR access critical for certain CPUs, which have no native XCPM support.

This is typically used in conjunction with the 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. Only certain Xeon models typically benefit from the patch.

5. CustomPciSerialDevice

Type: plist boolean Failsafe: false Requirement: 10.7

Description: Performs change of PMIO register base address on a customised PCI serial device.

The patch changes the PMIO register base address that the XNU kernel uses for serial input and output, from that of the default built-in COM1 serial port 0x3F8, to the base address stored in the first IO BAR of a specified PCI device or to a specific base address (e.g. 0x2F8 for COM2).

Note: By default, serial logging is disabled. serial=3 boot argument, which enables serial input and output, should be used for XNU to print logs to the serial port.

Note 2: In addition to this patch, kext Apple16X50PCIO should be prevented from attaching to have kprintf method working properly. This can be achieved by using PCIeSerialDisable. In addition, for certain Thunderbolt cards the IOKit personality IOPCITunnelCompatible also needs to be set to true, which can be done by the PCIeSerialThunderboltEnable.kext attached at acidanthera/bugtracker#2003.

Note 3: For this patch to be correctly applied, Override must be enabled with all keys properly set in Custom, under section Misc->Serial.

Note 4: This patch is for PMIO support and is therefore not applied if UseMmio under section Misc->Serial->Custom is false. For MMIO, there are boot arguments pcie_mmio_uart=ADDRESS and mmio_uart=ADDRESS that allow the kernel to use MMIO for serial port access.

Note 5: The serial baud rate must be correctly set in both BaudRate under section Misc->Serial->Custom and via serialbaud=VALUE boot argument, both of which should match against each other. The default baud rate is 115200.

6. CustomSMBIOSGuid

Type: plist boolean Failsafe: false Requirement: 10.4

Description: Performs GUID patching for UpdateSMBIOSMode Custom mode. Usually relevant for Dell laptops.

7. 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 1: This option is a preferred alternative to deleting DMAR ACPI table and disabling VT-d in firmware preferences, which does not obstruct VT-d support in other systems in case they need this.

Note 2: Misconfigured IOMMU in the firmware may result in broken devices such as ethernet or Wi-Fi adapters. For instance, an ethernet adapter may cycle in link-up link-down state infinitely and a Wi-Fi adapter may fail to discover networks. Gigabyte is one of the most common OEMs with these issues.

8. DisableIoMapperMapping

Type: plist boolean

Failsafe: false

Requirement: 13.3 (not required for older)

Description: Disables mapping PCI bridge device memory in IOMMU (VT-d).

This option resolves compatibility issues with Wi-Fi, Ethernet and Thunderbolt devices when AppleVTD is enabled on systems where the native DMAR table contains one or more Reserved Memory Regions and more than 16 GB memory is installed. On some systems, this quirk is only needed when iGPU is enabled.

Note 1: This quirk requires a native DMAR table that does not contain Reserved Memory Regions or a substitute SSDT-DMAR.aml in which Reserved Memory Regions have been removed.

Note 2: This option is not needed on AMD systems.

9. DisableLinkeditJettison

Type: plist boolean Failsafe: false Requirement: 11

Description: Disables __LINKEDIT jettison code.

This option lets Lilu.kext, and possibly other kexts, function in macOS Big Sur at their best performance levels without requiring the keepsyms=1 boot argument.

10. DisableRtcChecksum

Type: plist boolean

Failsafe: false Requirement: 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 AppleRtcRam protocol description if this is desired.

11. ExtendBTFeatureFlags

Type: plist boolean

Failsafe: false Requirement: 10.8-11

Description: Set FeatureFlags to 0x0F for full functionality of Bluetooth, including Continuity.

 $\it Note$: This option is a substitution for BT4LEContinuityFixup.kext, which does not function properly due to late

patching progress.

12. ExternalDiskIcons

Type: plist boolean Failsafe: false Requirement: 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 firmware typically have compatible AHCI controllers.

13. ForceAquantiaEthernet

 $\mathbf{Type} {:}\ \mathtt{plist}\ \mathtt{boolean}$

Failsafe: false

Requirement: 10.15.4

Description: Enable Aquantia AQtion based 10GbE network cards support.

This option enables Aquantia AQtion based 10GbE network cards support, which used to work natively before macOS 10.15.4.

Note: In order for Aquantia cards to properly function, DisableIoMapper must be disabled, DMAR ACPI table must not be dropped, and VT-d must be enabled in BIOS.

Note 2: While this patch should enable ethernet support for all Aquantia AQtion series, it has only been tested on AQC-107s based 10GbE network cards.

Note 3: To address AppleVTD incompatibilities after applying this quirk, the Reserved Memory Region section of the corresponding device in the DMAR ACPI table might be removed. This table should be disassembled and edited, then recompiled to AML with tool iASL. For the patched DMAR table to be added, the original one should be deleted. More details can be found at comment on commit 2441455.

14. ForceSecureBootScheme

Type: plist boolean

Failsafe: false Requirement: 11

Description: Force x86 scheme for IMG4 verification.

Note: This option is required on virtual machines when using SecureBootModel different from x86legacy.

15. IncreasePciBarSize

Type: plist boolean

Failsafe: false Requirement: 10.10

Description: Allows IOPCIFamily to boot with 2 GB PCI BARs.

Normally macOS restricts PCI BARs to 1 GB. Enabling this option (still) does not let macOS actually use PCI devices with larger BARs.

Note: This option should be avoided whenever possible. A need for this option indicates misconfigured or defective firmware.

16. LapicKernelPanic

 $\mathbf{Type} {:}\ \mathtt{plist}\ \mathtt{boolean}$

Failsafe: false

Requirement: 10.6 (64-bit)

Description: Disables kernel panic on LAPIC interrupts.

17. LegacyCommpage

Type: plist boolean

Failsafe: false

Requirement: 10.4 - 10.6

Description: Replaces the default 64-bit commpage boopy implementation with one that does not require SSSE3, useful for legacy platforms. This prevents a commpage no match for last panic due to no available 64-bit boopy functions that do not require SSSE3.

18. 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.

19. 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.

$20. \ {\tt ProvideCurrentCpuInfo}$

 $\mathbf{Type}: \mathtt{plist} \ \mathtt{boolean}$

Failsafe: false

Requirement: 10.4 (10.14)

Description: Provides current CPU info to the kernel.

This quirk works differently depending on the CPU:

- For Microsoft Hyper-V it provides the correct TSC and FSB values to the kernel, as well as disables CPU topology validation (10.8+).
- For KVM and other hypervisors it provides precomputed MSR 35h values solving kernel panic with -cpu host.
- For Intel CPUs it adds support for asymmetrical SMP systems (e.g. Intel Alder Lake) by patching core count to thread count along with the supplemental required changes (10.14+). Cache size and cache line size are also provided when using 10.4 as Intel Penryn and newer may only have cache information in CPUID leaf 0x4 which is unsupported by 10.4.

21. SetApfsTrimTimeout

Type: plist integer

Failsafe: -1

Requirement: 10.14 (not required for older)

Description: Set trim timeout in microseconds for APFS filesystems on SSDs.

The APFS filesystem is designed in a way that the space controlled via the spaceman structure is either used or free. This may be different in other filesystems where the areas can be marked as used, free, and *unmapped*. All free space is trimmed (unmapped/deallocated) at macOS startup. The trimming procedure for NVMe drives happens in LBA ranges due to the nature of the DSM command with up to 256 ranges per command. The more fragmented the memory on the drive is, the more commands are necessary to trim all the free space.

Depending on the SSD controller and the level of drive fragmenation, the trim procedure may take a considerable amount of time, causing noticeable boot slowdown. The APFS driver explicitly ignores previously unmapped areas and repeatedly trims them on boot. To mitigate against such boot slowdowns, the macOS driver introduced a timeout (9.999999 seconds) that stops the trim operation when not finished in time.

On several controllers, such as Samsung, where the deallocation process is relatively slow, this timeout can be reached very quickly. Essentially, it means that the level of fragmentation is high, thus macOS will attempt to

trim the same lower blocks that have previously been deallocated, but never have enough time to deallocate higher blocks. The outcome is that trimming on such SSDs will be non-functional soon after installation, resulting in additional wear on the flash.

One way to workaround the problem is to increase the timeout to an extremely high value, which at the cost of slow boot times (extra minutes) will ensure that all the blocks are trimmed. Setting this option to a high value, such as 4294967295 ensures that all blocks are trimmed. Alternatively, use over-provisioning, if supported, or create a dedicated unmapped partition where the reserve blocks can be found by the controller. Conversely, the trim operation can be mostly disabled by setting a very low timeout value, while 0 entirely disables it. Refer to this article for details.

Note: The failsafe value -1 indicates that this patch will not be applied, such that apfs.kext will remain untouched.

Note 2: On macOS 12.0 and above, it is no longer possible to specify trim timeout. However, trim can be disabled by setting 0.

Note 3: Trim operations are only affected at booting phase when the startup volume is mounted. Either specifying timeout, or completely disabling trim with 0, will not affect normal macOS running.

22. ThirdPartyDrives

Type: plist boolean

Failsafe: false

Requirement: 10.6 (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.

23. 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. Refer to the Legacy Apple OS section for details on how to install and troubleshoot such macOS installations.

1. CustomKernel

Type: plist boolean

Failsafe: false

Description: Use customised kernel cache from the Kernels directory located at the root of the ESP partition.

Unsupported platforms including Atom and AMD require modified versions of XNU kernel in order to boot. This option provides the possibility to using a customised kernel cache which contains such modifications from ESP partition.

2. FuzzyMatch

Type: plist boolean

Failsafe: false

Description: Use kernelcache with different checksums when available.

On Mac OS X 10.6 and earlier, kernelcache filename has a checksum, which essentially is adler32 from SMBIOS product name and EfiBoot device path. On certain firmware, the 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 Mac OS X 10.6 boot performance on several platforms.

3. KernelArch

Type: plist string

Failsafe: Auto (Choose the preferred architecture automatically)

Description: Prefer specified kernel architecture (1386, 1386-user32, x86 64) when available.

On Mac OS X 10.7 and earlier, the 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:

- 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 if supported by the operating system.
 - 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 Mac OS X 10.6. This behaviour corresponds to the -legacy kernel boot argument.
 - This option is unavailable on Mac OS X 10.4 and 10.5 when running on 64-bit firmware due to an uninitialised 64-bit segment in the XNU kernel, which causes AppleEFIRuntime to incorrectly execute 64-bit code as 16-bit code.
- $x86_64$ Use $x86_64$ (64-bit) kernel when available.

The algorithm used to determine the preferred kernel architecture is set out below.

- (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 (only on 32-bit firmware)
 - 10.6 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 Mac OS X 10.7 (where certain board identifiers are treated as i386 only machines), and Mac OS X 10.5 or earlier (where x86_64 is not supported by the macOS kernel), Mac OS X 10.6 is very special. The architecture choice on Mac OS X 10.6 depends on many factors including not only the board identifier, but also the macOS product type (client vs server), macOS point release, and amount of RAM. The detection of all these is complicated and impractical, as several point releases had implementation flaws resulting in a failure to properly execute the server detection in the first place. For this reason when Auto is set, OpenCore on Mac OS X 10.6 falls back to the x86_64 architecture when it is supported by the board, as on Mac OS X 10.7. The 32-bit KernelArch options can still be configured explicitly however.

A 64-bit Mac model compatibility matrix corresponding to actual EfiBoot behaviour on Mac OS X 10.6.8 and 10.7.5 is outlined below.

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 as they are handled by EfiBoot and hence, difficult to detect.

4. 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 prevents the use of faster kernel caching variants if slower variants are available for debugging and stability reasons. That is, by specifying Mkext, Prelinked will be disabled for e.g. 10.6 but not for 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	YES	YES (V1)	NO (V1)				
10.5	YES	YES (V1)	NO (V1)		_		
10.6	YES	YES (V2)	YES (V2)	YES	YES (V2)	YES (V2)	
10.7	YES	_	YES (V3)	YES	_	YES (V3)	_
10.8-10.9		_	_	YES	_	YES (V3)	_
10.10-10.15					_	YES (V3)	
11+	_	_	_	_	_	YES (V3)	YES

Note: The first version (V1) of the 32-bit prelinkedkernel is unsupported due to the corruption of kext symbol tables by the tools. On this version, the Auto setting will block prelinkedkernel booting. This also results in the keepsyms=1 boot argument being non-functional for kext frames on these systems.

8 Misc

8.1 Introduction

This section contains miscellaneous configuration options affecting OpenCore operating system loading behaviour in addition to other options that do not readily fit into other sections.

OpenCore broadly follows the "bless" model, also known as the "Apple Boot Policy". The primary purpose of the "bless" model is to allow embedding boot options within the file system (and be accessible through a specialised driver) as well as supporting a broader range of predefined boot paths as compared to the removable media list set out in the UEFI specification.

Partitions can only booted by OpenCore when they meet the requirements of a predefined Scan policy. This policy sets out which specific file systems a partition must have, and which specific device types a partition must be located on, to be made available by OpenCore as a boot option. Refer to the ScanPolicy property for details.

The scan process starts with enumerating all available partitions, filtered based on the Scan policy. Each partition may generate multiple primary and alternate options. Primary options represent operating systems installed on the media, while alternate options represent recovery options for the operating systems on the media.

- Alternate options may exist without primary options and vice versa.
- Options may not necessarily represent operating systems on the same partition.
- Each primary and alternate option can be an auxiliary option or not.
 - Refer to the HideAuxiliary section for details.

8.1.1 Boot Algorithm

The algorithm to determine boot options behaves as follows:

- 1. Obtain all available partition handles filtered based on the Scan policy (and driver availability).
- 2. Obtain all available boot options from the BootOrder UEFI variable.
- 3. For each boot option found:
 - Retrieve the device path of the boot option.
 - Perform fixups (e.g. NVMe subtype correction) and expansion (e.g. for Boot Camp) of the device path.
 - On failure, if it is an OpenCore custom entry device path, pre-construct the corresponding custom entry and succeed.
 - Obtain the device handle by locating the device path of the resulting device path (ignore it on failure).
 - Locate the device handle in the list of partition handles (ignore it if missing).
 - For disk device paths (not specifying a bootloader), execute "bless" (may return > 1 entry).
 - For file device paths, check for presence on the file system directly.
 - Exclude entries if there is a .contentVisibility file at a relevant location (see below) with Disabled contents (ASCII) (and if the current InstanceIentifier matches an Instance-List if present, see below).
 - Mark device handle as *used* in the list of partition handles if any.
 - Register the resulting entries as primary options and determine their types.

 The option will become auxiliary for some types (e.g. Apple HFS recovery) or if its .contentVisibility file contains Auxiliary (and if the current InstanceIentifier matches an Instance-List if present, see below).
- 4. For each partition handle:
 - If the partition handle is marked as *unused*, execute "bless" primary option list retrieval. In case a BlessOverride list is set, both standard and custom "bless" paths will be found.
 - On the OpenCore boot partition, exclude OpenCore bootstrap files using header checks.
 - Register the resulting entries as primary options and determine their types if found. The option will become auxiliary for some types (e.g. Apple HFS recovery).
 - If a partition already has any primary options of the "Apple Recovery" type, proceed to the next handle.
 - Lookup alternate entries by "bless" recovery option list retrieval and predefined paths.
 - Register the resulting entries as alternate auxiliary options and determine their types if found.
- 5. Custom entries and tools, except such pre-constructed previously, are added as primary options without any checks with respect to Auxiliary.
- 6. System entries, such as Reset NVRAM, are added as primary auxiliary options.

A .contentVisibility file may be placed next to the bootloader (such as boot.efi), or in the boot folder (for DMG folder based boot items). Example locations, as seen from within macOS, are:

- /System/Volumes/Preboot/{GUID}/System/Library/CoreServices/.contentVisibility
- /Volumes/{ESP}/EFI/BOOT/.contentVisibility

In addition a .contentVisibility file may be placed in the instance-specific (for macOS) or absolute root folders related to a boot entry, for example:

- /System/Volumes/Preboot/{GUID}/.contentVisibility
- /System/Volumes/Preboot/.contentVisibility
- /Volumes/{ESP}/.contentVisibility (not recommended)

These root folder locations are supported specifically for macOS, because non-Apple files next to the Apple bootloader are removed by macOS updates. It is supported but not recommended to place a .contentVisibility file in a non-macOS root location (such as the last location shown above), because it will hide all entries on the drive.

The .contentVisibility file, when present, may optionally target only specific instances of OpenCore. Its contents are [{Instance-List}:](Disabled|Auxiliary). If a colon (:) is present, the preceding Instance-List it is a comma separated list of InstanceIdentifier values (example: OCA,OCB:Disabled). When this list is present, the specified visibility is only applied if the InstanceIdentifier of the current instance of OpenCore is present in the list. When the list is not present, the specified visibility is applied for all instances of OpenCore.

Note 1: For any instance of OpenCore with no InstanceIdentifier value, the specified visibility from a .contentVisibility file with an Instance-List will never be applied.

Note 2: Visibilities with a visibility list will be treated as invalid, and so ignored, in earlier versions of OpenCore - which may be useful when comparing behaviour of older and newer versions.

Note 3: Avoid extraneous spaces in the .contentVisibility file: these will not be treated as whitespace, but as part of the adjacent token.

The display order of the boot options in the OpenCore picker and the boot process are determined separately from the scanning algorithm.

The display order is as follows:

- Alternate options follow corresponding primary options. That is, Apple recovery options will follow the relevant macOS option whenever possible.
- Options will be listed in file system handle firmware order to maintain an established order across reboots regardless of the operating system chosen for loading.
- Custom entries, tools, and system entries will be added after all other options.
- Auxiliary options will only be displayed upon entering "Extended Mode" in the OpenCore picker (typically by pressing the Space key).

The boot process is as follows:

- Look up the first valid primary option in the BootNext UEFI variable.
- On failure, look up the first valid primary option in the BootOrder UEFI variable.
- Mark the option as the default option to boot.
- Boot option through the picker or without it depending on the ShowPicker option.
- Show picker on failure otherwise.

Note 1: This process will only work reliably when the RequestBootVarRouting option is enabled or the firmware does not control UEFI boot options (OpenDuetPkg or custom BDS). When LauncherOption is not enabled, other operating systems may overwrite OpenCore settings and this property should therefore be enabled when planning to use other operating systems.

Note 2: UEFI variable boot options boot arguments will be removed, if present, as they may contain arguments that can compromise the operating system, which is undesirable when secure boot is enabled.

Note 3: Some operating systems, such as Windows, may create a boot option and mark it as the topmost option upon first boot or after NVRAM resets from within OpenCore. When this happens, the default boot entry choice will remain changed until the next manual reconfiguration.

8.2 Properties

1. BlessOverride

Type: plist array Failsafe: Empty

Description: Add custom scanning paths through the bless model.

To be filled with plist string entries containing absolute UEFI paths to customised bootloaders such as \EFI\debian\grubx64.efi for the Debian bootloader. This allows non-standard boot paths to be automatically discovered by the OpenCore picker. Designwise, they are equivalent to predefined blessed paths, such as \System\Library\CoreServices\boot.efi or \EFI\Microsoft\Boot\bootmgfw.efi, but unlike predefined bless paths, they have the highest priority.

2. Boot

Type: plist dict

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

3. Debug

Type: plist dict

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

4. Entries

Type: plist array Failsafe: Empty

Description: Add boot entries to OpenCore picker.

To be filled with plist dict values, describing each load entry. Refer to the Entry Properties section below for details.

5. Security

Type: plist dict

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

6. Serial

Type: plist dict

Description: Perform serial port initialisation and configure PCD values required by BaseSerialPortLib16550 for serial ports to properly function. Values are listed and described in the Serial Properties and Serial Custom Properties section below.

By enabling Init, this section ensures that the serial port is initialised when it is not done by firmware. In order for OpenCore to print logs to the serial port, bit 3 (i.e. serial logging) for Target under section Misc->Debug must be set.

When debugging with serial ports, BaseSerialPortLib16550 only recognises internal ones provided by the motherboard by default. If the option Override is enabled, this section will override the PCD values listed in BaseSerialPortLib16550.inf such that external serial ports (e.g. from a PCI card) will also function properly. Specifically, when troubleshooting macOS, in addition to overriding these PCD values, it is also necessary to turn the CustomPciSerialDevice kernel quirk on in order for the XNU to use such exterior serial ports.

Refer to MdeModulePkg.dec for the explanations of each key.

7. Tools

Type: plist array Failsafe: Empty

Description: Add tool entries to the OpenCore picker.

To be filled with plist dict values, describing each load entry. Refer to the Entry Properties section below for details.

Note: Certain UEFI tools, such as UEFI Shell, can be very dangerous and **MUST NOT** appear in production configurations, particularly in vaulted configurations as well as those protected by secure boot, as such tools can be used to bypass the secure boot chain. Refer to the UEFI section for examples of UEFI tools.

8.3 Boot Properties

1. ConsoleAttributes

Type: plist integer

Failsafe: 0

Description: Sets specific attributes for the console.

The text renderer supports colour arguments as a sum of foreground and background colours based on the UEFI specification. The value for black background and for black foreground, 0, is reserved.

List of colour values and names:

- 0x00 EFI_BLACK
- 0x01 EFI_BLUE
- 0x02 EFI_GREEN
- 0x03 EFI_CYAN
- 0x04 EFI_RED
- 0x05 EFI_MAGENTA
- 0x06 EFI BROWN
- 0x07 EFI LIGHTGRAY
- 0x08 EFI_DARKGRAY
- 0x09 EFI_LIGHTBLUE
- OxOA EFI_LIGHTGREEN
- OxOB EFI LIGHTCYAN
- OxOC EFI_LIGHTRED
- OxOD EFI_LIGHTMAGENTA
- OxOE EFI_YELLOW
- OxOF EFI_WHITE
- 0x00 EFI_BACKGROUND_BLACK
- 0x10 EFI_BACKGROUND_BLUE
- 0x20 EFI_BACKGROUND_GREEN
- 0x30 EFI_BACKGROUND_CYAN
- 0x40 EFI_BACKGROUND_RED
- 0x50 EFI_BACKGROUND_MAGENTA
- 0x60 EFI_BACKGROUND_BROWN
- 0x70 EFI BACKGROUND LIGHTGRAY

Note: This option may not work well with the System text renderer. Setting a background different from black could help with testing GOP functionality.

2. HibernateMode

Type: plist string

Failsafe: None

Description: Hibernation detection mode. The following modes are supported:

- None Ignore hibernation state.
- Auto Use RTC and NVRAM detection.
- RTC Use RTC detection.
- NVRAM Use NVRAM detection.

Note: If the firmware can handle hibernation itself (valid for Mac EFI firmware), then None should be specified to hand-off hibernation state as is to OpenCore.

3. HibernateSkipsPicker

Type: plist boolean

Failsafe: false

Description: Do not show picker if waking from macOS hibernation.

Limitations:

• Only supports macOS hibernation wake, Windows and Linux are currently out of scope.

- Should only be used on systems with reliable hibernation wake in macOS, otherwise users may not be able to visually see boot loops that may occur.
- Highly recommended to pair this option with PollAppleHotKeys, allows to enter picker in case of issues with hibernation wake.
- Visual indication for hibernation wake is currently out of scope.

4. HideAuxiliary

Type: plist boolean Failsafe: false

Description: Set to true to hide auxiliary entries from the picker menu.

An entry is considered auxiliary when at least one of the following applies:

• Entry is macOS recovery.

- Entry is macOS Time Machine.
- Entry is explicitly marked as Auxiliary.
- Entry is system (e.g. Reset NVRAM).

To display all entries, the picker menu can be reloaded into "Extended Mode" by pressing the Spacebar key. Hiding auxiliary entries may increase boot performance on multi-disk systems.

5. InstanceIdentifier

Type: plist string Failsafe: Empty

Description: An optional identifier for the current instance of OpenCore.

This should typically be a short alphanumeric string. The current use of this value is to optionally target .contentVisibility files to specific instances of OpenCore, as explained in the Boot Algorithm section.

6. LauncherOption

Type: plist string Failsafe: Disabled

Description: Register the launcher option in the firmware preferences for persistence.

Valid values:

- Disabled do nothing.
- Full create or update the top priority boot option in UEFI variable storage at bootloader startup.
 - For this option to work, RequestBootVarRouting is required to be enabled.
- Short create a short boot option instead of a complete one.
 - This variant is useful for some older types of firmware, typically from Insyde, that are unable to manage full device paths.
- System create no boot option but assume specified custom option is blessed.
 - This variant is useful when relying on ForceBooterSignature quirk and OpenCore launcher path management happens through bless utilities without involving OpenCore.

This option allows integration with third-party operating system installation and upgrades (which may overwrite the \EFI\BOOT\BOOTx64.efi file). The BOOTx64.efi file is no longer used for bootstrapping OpenCore if a custom option is created. The custom path used for bootstrapping can be specified by using the LauncherPath option.

Note 1: Some types of firmware may have NVRAM implementation flaws, no boot option support, or other incompatibilities. While unlikely, the use of this option may result in boot failures and should only be used exclusively on boards known to be compatible. Refer to acidanthera/bugtracker#1222 for some known issues affecting Haswell and other boards.

Note 2: While NVRAM resets executed from OpenCore would not typically erase the boot option created in Bootstrap, executing NVRAM resets prior to loading OpenCore will erase the boot option. Therefore, for significant implementation updates, such as was the case with OpenCore 0.6.4, an NVRAM reset should be executed with Bootstrap disabled, after which it can be re-enabled.

Note 3: Some versions of Intel Visual BIOS (e.g. on Intel NUC) have an unfortunate bug whereby if any boot option is added referring to a path on a USB drive, from then on that is the only boot option which will be shown

when any USB drive is inserted. If OpenCore is started from a USB drive on this firmware with LauncherOption set to Full or Short, this applies and only the OpenCore boot entry will be seen afterwards, when any other USB is inserted (this highly non-standard BIOS behaviour affects other software as well). The best way to avoid this is to leave LauncherOption set to Disabled or System on any version of OpenCore which will be started from a USB drive on this firmware. If the problem has already occurred the quickest reliable fix is:

- Enable the system UEFI Shell in Intel Visual BIOS
- With power off, insert an OpenCore USB
- Power up and select the system UEFI Shell
- Since the system shell does not include bcfg, use the system shell to start OpenCore's OpenShell (e.g. by entering the command FS2:\EFI\OC\Tools\OpenShell.efi , but you will need to work out which drive is correct for OpenCore and modify the drive number FS#: accordingly)
- Within OpenShell, use bcfg boot dump to display the NVRAM boot options and then use bcfg boot rm # (where # is the number of the OpenCore boot entry) to remove the OpenCore entry

It is alternatively possible to start OpenShell directly from the OpenCore boot menu, if you have a working configured OpenCore for the system. In that case, and if OpenCore has RequestBootVarRouting enabled, it will be necessary to run the command \EFI\OC\Tools\OpenControl.efi disable before using bcfg. (After OpenControl disable, it is necessary to either reboot or run OpenControl restore, before booting an operating system.) It is also possible to use efibootmgr within Linux to remove the offending entry, if you have a working version of Linux on the machine. Linux must be started either not via OpenCore, or via OpenCore with RequestBootVarRouting disabled for this to work.

7. LauncherPath

Type: plist string Failsafe: Default

Description: Launch path for the LauncherOption property.

Default points to OpenCore.efi. User specified paths, e.g. \EFI\SomeLauncher.efi, can be used to provide custom loaders, which are supposed to load OpenCore.efi themselves.

8. PickerAttributes

Type: plist integer

Failsafe: 0

Description: Sets specific attributes for the OpenCore picker.

Different OpenCore pickers may be configured through the attribute mask containing OpenCore-reserved (BIT0~BIT15) and OEM-specific (BIT16~BIT31) values.

Current OpenCore values include:

- 0x0001 0C_ATTR_USE_VOLUME_ICON, provides custom icons for boot entries: OpenCore will attempt loading a volume icon by searching as follows, and will fallback to the default icon on failure:
 - .VolumeIcon.icns file at Preboot volume in per-volume directory (/System/Volumes/Preboot/{GUID}/when mounted at the default location within macOS) for APFS (if present).
 - .VolumeIcon.icns file at the Preboot volume root (/System/Volumes/Preboot/, when mounted at the default location within macOS) for APFS (otherwise).
 - .VolumeIcon.icns file at the volume root for other filesystems.

Note 1: The Apple picker partially supports placing a volume icon file at the operating system's Data volume root, /System/Volumes/Data/, when mounted at the default location within macOS. This approach is flawed: the file is neither accessible to OpenCanopy nor to the Apple picker when FileVault 2, which is meant to be the default choice, is enabled. Therefore, OpenCanopy does not attempt supporting Apple's approach. A volume icon file may be placed at the root of the Preboot volume for compatibility with both OpenCanopy and the Apple picker, or use the Preboot per-volume location as above with OpenCanopy as a preferred alternative to Apple's approach.

Note 2: Be aware that using a volume icon on any drive overrides the normal OpenCore picker behaviour for that drive of selecting the appropriate icon depending on whether the drive is internal or external.

• 0x0002 — 0C_ATTR_USE_DISK_LABEL_FILE, use custom prerendered titles for boot entries from .disk_label (.disk_label_2x) file next to the bootloader for all filesystems. These labels can be generated via the

disklabel utility or the bless --folder {FOLDER_PATH} --label {LABEL_TEXT} command. When prerendered labels are disabled or missing, use label text in .contentDetails (or .disk_label.contentDetails) file next to bootloader if present instead, otherwise the entry name itself will be rendered.

- 0x0004 0C_ATTR_USE_GENERIC_LABEL_IMAGE, provides predefined label images for boot entries without custom entries. This may however give less detail for the actual boot entry.
- 0x0008 OC_ATTR_HIDE_THEMED_ICONS, prefers builtin icons for certain icon categories to match the theme
 style. For example, this could force displaying the builtin Time Machine icon. Requires OC_ATTR_USE_VOLUME_ICON.
- 0x0010 0C_ATTR_USE_POINTER_CONTROL, enables pointer control in the OpenCore picker when available. For example, this could make use of mouse or trackpad to control UI elements.
- 0x0020 0C_ATTR_SHOW_DEBUG_DISPLAY, enable display of additional timing and debug information, in Builtin picker in DEBUG and NOOPT builds only.
- 0x0040 0C_ATTR_USE_MINIMAL_UI, use minimal UI display, no Shutdown or Restart buttons, affects OpenCanopy and builtin picker.
- 0x0080 0C_ATTR_USE_FLAVOUR_ICON, provides flexible boot entry content description, suitable for picking the best media across different content sets:

When enabled, the entry icon in OpenCanopy and the audio assist entry sound in OpenCanopy and builtin boot picker are chosen by something called content flavour. To determine content flavour the following algorithm is used:

- For a Tool the value is read from Flavour field.
- For an automatically discovered entry, including for boot entry protocol entries such as those generated by the OpenLinuxBoot driver, it is read from the .contentFlavour file next to the bootloader, if present.
- For a custom entry specified in the Entries section it is read from the .contentFlavour file next to the bootloader if Flavour is Auto, otherwise it is specified via the Flavour value itself.
- If read flavour is Auto or there is no .contentFlavour, entry flavour is chosen based on the entry type (e.g. Windows automatically gets Windows flavour).

The Flavour value is a sequence of: separated names limited to 64 characters of printable 7-bit ASCII. This is designed to support up to approximately five names. Each name refers to a flavour, with the first name having the highest priority and the last name having the lowest priority. Such a structure allows describing an entry in a more specific way, with icons selected flexibly depending on support by the audio-visual pack. A missing audio or icon file means the next flavour should be tried, and if all are missing the choice happens based on the type of the entry. Example flavour values: BigSur:Apple, Windows10:Windows.OpenShell:UEFIShell:Shell.

Using flavours means that you can switch between icon sets easily, with the flavour selecting the best available icons from each set. E.g. specifying icon flavour Debian:Linux will use the icon Debian.icns if provided, then will try Linux.icns, then will fall back to the default for an OS, which is HardDrive.icns.

Things to keep in mind:

- For security reasons Ext<Flavour>.icns and <Flavour>.icns are both supported, and only Ext<Flavour>.icns will be used if the entry is on an external drive (followed by default fallback ExtHardDrive.icns).
- Where both apply .VolumeIcon.icns takes precedence over .contentFlavour.
- In order to allow icons and audio assist to work correctly for tools (e.g. for UEFI Shell), system default boot entry icons (see Docs/Flavours.md) specified in the Flavour setting for Tools or Entries will continue to apply even when flavour is disabled. Non-system icons will be ignored in this case. In addition, the flavours UEFIShell and NVRAMReset are given special processing, identifying their respective tools to apply correct audio-assist, default builtin labels, etc.
- A list of recommended flavours is provided in Docs/Flavours.md.
- 0x0100 0C_ATTR_USE_REVERSED_UI, reverse position of Shutdown and Restart buttons, affects Open-Canopy and builtin picker. The reversed setting matches older macOS, and since it was the previous default in OpenCore it may better match some custom backgrounds. Only applicable when OC_ATTR_USE_MINIMAL_UI is not set.
- 0x0200 OC_ATTR_REDUCE_MOTION, reduce password and menu animation in OpenCanopy, leaving only animations which communicate information not otherwise provided.

Note: These same animations, plus additional animations whose information is provided by voice-over, are automatically disabled when PickerAudioAssist is enabled.

$9.\ {\tt PickerAudioAssist}$

Type: plist boolean

Failsafe: false

Description: Enable screen reader by default in the OpenCore picker.

For the macOS bootloader, screen reader preference is set in the preferences.efires archive in the isV0Enabled.int32 file and is controlled by the operating system. For OpenCore screen reader support, this option is an independent equivalent. Toggling screen reader support in both the OpenCore picker and the macOS bootloader FileVault 2 login window can also be done by using the Command + F5 key combination.

Note: The screen reader requires working audio support. Refer to the UEFI Audio Properties section for details.

10. PickerMode

Type: plist string Failsafe: Builtin

Description: Choose picker used for boot management.

PickerMode describes the underlying boot management with an optional user interface responsible for handling boot options.

The following values are supported:

- Builtin boot management is handled by OpenCore, a simple text-only user interface is used.
- External an external boot management protocol is used if available. Otherwise, the Builtin mode is used.
- Apple Apple boot management is used if available. Otherwise, the Builtin mode is used.

Upon success, the External mode may entirely disable all boot management in OpenCore except for policy enforcement. In the Apple mode, it may additionally bypass policy enforcement. Refer to the OpenCanopy plugin for an example of a custom user interface.

The OpenCore built-in picker contains a set of actions chosen during the boot process. The list of supported actions is similar to Apple BDS and typically can be accessed by holding action hotkeys during the boot process.

The following actions are currently considered:

- Default this is the default option, and it lets the built-in OpenCore picker load the default boot option as specified in the Startup Disk preference pane.
- ShowPicker this option forces the OpenCore picker to be displayed. This can typically be achieved by holding the OPT key during boot. Setting ShowPicker to true will make ShowPicker the default option.
- BootApple this options performs booting to the first Apple operating system found unless the chosen default operating system is one from Apple. Hold the X key down to choose this option.
- BootAppleRecovery this option performs booting into the Apple operating system recovery partition. This is either that related to the default chosen operating system, or first one found when the chosen default operating system is not from Apple or does not have a recovery partition. Hold the CMD+R hotkey combination down to choose this option.

Note 1: On non-Apple firmware KeySupport, OpenUsbKbDxe, or similar drivers are required for key handling. However, not all of the key handling functions can be implemented on several types of firmware.

Note 2: In addition to OPT, OpenCore supports using both the Escape and Zero keys to enter the OpenCore picker when ShowPicker is disabled. Escape exists to support co-existence with the Apple picker (including OpenCore Apple picker mode) and to support firmware that fails to report held OPT key, as on some PS/2 keyboards. In addition, Zero is provided to support systems on which Escape is already assigned to some other pre-boot firmware feature. In systems which do not require KeySupport, pressing and holding one of these keys from after power on until the picker appears should always be successful. The same should apply when using KeySupport mode if it is correctly configured for the system, i.e. with a long enough KeyForgetThreshold. If pressing and holding the key is not successful to reliably enter the picker, multiple repeated keypresses may be tried instead.

Note 3: On Macs with problematic GOP, it may be difficult to re-bless OpenCore if its bless status is lost. The BootKicker utility can be used to work around this problem, if set up as a Tool in OpenCore with FullNvramAccess enabled. It will launch the Apple picker, which allows selection of an item to boot next (with Enter), or next and from then on until the next change (with CTRL+Enter). Note that after the selection is made, the system will reboot before the chosen entry is booted. While this behaviour might seem surprising, it can be

used both to switch which OpenCore installation is blessed, with CTRL+Enter, e.g. from a recovery OpenCore installation on CD (selected with the C key on boot) back to the main installation of OpenCore on the hard drive, if this is lost after an NVRAM reset. It can also be used, even when the native picker cannot be shown normally (unsupported GPU), to do a one-shot boot without OpenCore, e.g. to another OS or tool, or to an earlier version of macOS.

11. PickerVariant

Type: plist string

Failsafe: Auto

Description: Choose specific icon set to be used for boot management.

An icon set is a directory path relative to Resources\Image, where the icons and an optional manifest are located. It is recommended for the artists to use provide their sets in the Vendor\Set format, e.g. Acidanthera\GoldenGate.

Sample resources provided as a part of OcBinaryData repository provide the following icon set:

- Acidanthera\GoldenGate $\max OS \ 11 \ styled \ icon \ set.$
- Acidanthera\Syrah OS X 10.10 styled icon set.
- Acidanthera\Chardonnay Mac OS X 10.4 styled icon set.

For convenience purposes there also are predefined aliases:

- Auto Automatically select one set of icons based on the DefaultBackground colour: Acidanthera\GoldenGate for Syrah Black and Acidanthera\Chardonnay for Light Gray.
- Default Acidanthera\GoldenGate.

12. PollAppleHotKeys

Type: plist boolean

Failsafe: false

Description: Enable modifier hotkey handling in the OpenCore picker.

In addition to action hotkeys, which are partially described in the PickerMode section and are typically handled by Apple BDS, modifier keys handled by the operating system bootloader (boot.efi) also exist. These keys allow changing the behaviour of the operating system by providing different boot modes.

On certain firmware, using modifier keys may be problematic due to driver incompatibilities. To workaround this problem, this option allows registering certain hotkeys in a more permissive manner from within the OpenCore picker. Such extensions include support for tapping on key combinations before selecting the boot item, and for reliable detection of the Shift key when selecting the boot item, in order to work around the fact that hotkeys which are continuously held during boot cannot be reliably detected on many PS/2 keyboards.

This list of known modifier hotkeys includes:

- CMD+C+MINUS disable board compatibility checking.
- CMD+K boot release kernel, similar to kcsuffix=release.
- CMD+S single user mode.
- CMD+S+MINUS disable KASLR slide, requires disabled SIP.
- CMD+V verbose mode.
- Shift+Enter, Shift+Index safe mode, may be used in combination with CTRL+Enter, CTRL+Index.

$13. \ {\tt ShowPicker}$

Type: plist boolean

Failsafe: false

Description: Show a simple picker to allow boot entry selection.

14. TakeoffDelay

Type: plist integer, 32 bit

Failsafe: 0

Description: Delay in microseconds executed before handling the OpenCore picker startup and action hotkeys.

Introducing a delay may give extra time to hold the right action hotkey sequence to, for instance, boot into recovery mode. On most systems, the appearance of the initial boot logo is a good indication of the time from which hotkeys can be held down. Earlier than this, the key press may not be registered. On some platforms,

setting this option to a minimum of 5000-10000 microseconds is also required to access action hotkeys due to the nature of the keyboard driver.

If the boot chime is configured (see audio configuration options) then at the expense of slower startup, an even longer delay of half to one second (500000–1000000) may be used to create behaviour similar to a real Mac, where the chime itself can be used as a signal for when hotkeys can be pressed. The boot chime is inevitably later in the boot sequence in OpenCore than on Apple hardware, due to the fact that non-native drivers have to be loaded and connected first. Configuring the boot chime and adding this longer additional delay can also be useful in systems where fast boot time and/or slow monitor signal synchronisation may cause the boot logo not to be shown at all on some boots or reboots.

15. Timeout

Type: plist integer, 32 bit

Failsafe: 0

Description: Timeout in seconds in the OpenCore picker before automatic booting of the default boot entry.

Set to 0 to disable.

8.4 Debug Properties

1. AppleDebug

Type: plist boolean Failsafe: false

Description: Enable writing the boot.efi debug log to the OpenCore log.

Note: This option only applies to 10.15.4 and newer.

2. ApplePanic

Type: plist boolean Failsafe: false

Description: Save macOS kernel panic output to the OpenCore root partition.

The file is saved as panic-YYYY-MM-DD-HHMMSS.txt. It is strongly recommended to set the keepsyms=1 boot argument to see debug symbols in the panic log. In cases where it is not present, the kpdescribe.sh utility (bundled with OpenCore) may be used to partially recover the stacktrace.

Development and debug kernels produce more useful kernel panic logs. Consider downloading and installing the KernelDebugKit from developer.apple.com when debugging a problem. To activate a development kernel, the boot argument kcsuffix=development should be added. Use the uname -a command to ensure that the current loaded kernel is a development (or a debug) kernel.

In cases where the OpenCore kernel panic saving mechanism is not used, kernel panic logs may still be found in the /Library/Logs/DiagnosticReports directory.

Starting with macOS Catalina, kernel panics are stored in JSON format and thus need to be preprocessed before passing to kpdescribe.sh:

```
cat Kernel.panic | grep macOSProcessedStackshotData |
   python3 -c 'import json,sys;print(json.load(sys.stdin)["macOSPanicString"])'
```

3. DisableWatchDog

Type: plist boolean Failsafe: false

Description: Some types of firmware may not succeed in booting the operating system quickly, especially in debug mode. This results in the watchdog timer aborting the process. This option turns off the watchdog timer.

4. DisplayDelay

Type: plist integer

 ${\bf Failsafe:}\ 0$

Description: Delay in microseconds executed after every printed line visible onscreen (i.e. console).

5. DisplayLevel

Type: plist integer, 64 bit

Failsafe: 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 (bit 1) DEBUG_WARN in DEBUG, NOOPT, RELEASE.
- 0x00000040 (bit 6) DEBUG_INFO in DEBUG, NOOPT.
- 0x00400000 (bit 22) DEBUG VERBOSE in custom builds.
- 0x80000000 (bit 31) DEBUG_ERROR in DEBUG, NOOPT, RELEASE.

6. LogModules

Type: plist string

Failsafe: *

Description: Filter log entries by module.

This option filters logging generated by specific modules, both in the log and onscreen. Two modes are supported:

- + Positive filtering: Only present selected modules.
- - Negative filtering: Exclude selected modules.

When multiple log line identifiers are selected, comma (,) should be used as the splitter. For instance, +0CCPU,0CA,0CB means only 0CCPU, 0CA, 0CB should be logged, while -0CCPU,0CA,0CB indicates these modules should be filtered out (i.e. not logged). Since there may be lines in the log with no valid prefix (i.e. log lines which are not generated by parts of OpenCore, but by other loaded drivers) then the special module name question mark (?) can be included in the list to include (with positive filtering) or exclude (with negative filtering) these non-standard lines. When no + or - symbol is specified, positive filtering (+) will be used. * alone as the option value indicates all modules being logged.

Note 1: Acronyms of libraries can be found in the Libraries section below.

Note 2: Messages printed before the configuration of the log protocol cannot be filtered from the early on screen log, but on being de-buffered from the early log buffer, will be filtered as requested for other log targets.

Note 3: To avoid missing key issues, warning and error log messages are not filtered.

7. SysReport

Type: plist boolean

Failsafe: false

Description: Produce system report on ESP folder.

This option will create a SysReport directory in the ESP partition unless already present. The directory will contain ACPI, SMBIOS, and audio codec dumps. Audio codec dumps require an audio backend driver to be loaded.

Note: To maintain system integrity, the SysReport option is **not** available in RELEASE builds. Use a DEBUG build if this option is required.

8. Target

Type: plist integer

Failsafe: 0

Description: A bitmask (sum) of enabled logging targets. Logging output is hidden by default and this option must be set when such output is required, such as when debugging.

The following logging targets are supported:

- 0x01 (bit 0) Enable logging, otherwise all log is discarded.
- 0x02 (bit 1) Enable basic console (onscreen) logging.
- 0x04 (bit 2) Enable logging to Data Hub.
- 0x08 (bit 3) Enable serial port logging.
- 0x10 (bit 4) Enable UEFI variable logging.
- 0x20 (bit 5) Enable non-volatile UEFI variable logging.
- 0x40 (bit 6) Enable logging to file.
- 0x80 (bit 7) In combination with 0x40, enable faster but unsafe (see Warning 2 below) file logging.

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

To obtain Data Hub logs, use the following command in macOS (Note that Data Hub logs do not log kernel and kext patches):

```
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. To maintain system integrity, the log size is limited to 32 kilobytes. Some types of firmware may truncate it much earlier or drop completely if they have no memory. Using the non-volatile flag will cause the log to be written to NVRAM flash after every printed line.

To obtain UEFI variable logs, 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 1: Certain firmware appear to have defective NVRAM garbage collection. As a result, they may not be able to always free space after variable deletion. Do not enable non-volatile NVRAM logging on such devices unless specifically required.

While the OpenCore boot log already contains basic version information including build type and date, this information may also be found in the opencore-version NVRAM variable even when boot logging is disabled.

File logging will create a file named opencore-YYYY-MM-DD-HHMMSS.txt (in UTC) under the EFI volume root with log contents (the upper case letter sequence is replaced with date and time from the firmware). Please be warned that some file system drivers present in firmware are not reliable and may corrupt data when writing files through UEFI. Log writing is attempted in the safest manner and thus, is very slow. Ensure that DisableWatchDog is set to true when a slow drive is used. Try to avoid frequent use of this option when dealing with flash drives as large I/O amounts may speed up memory wear and render the flash drive unusable quicker.

Warning 2: It is possible to enable fast file logging, which requires a fully compliant firmware FAT32 driver. On drivers with incorrect FAT32 write support (e.g. APTIO IV, but maybe others) this setting can result in corruption up to and including an unusable ESP filesystem, therefore be prepared to recreate the ESP partition and all of its contents if testing this option. This option can increase logging speed significantly on some suitable firmware, but may make little speed difference on some others.

When interpreting the log, note that the lines are prefixed with a tag describing the relevant location (module) of the log line allowing better attribution of the line to the functionality.

The list of currently used tags is as follows.

Drivers and tools:

- BMF OpenCanopy, bitmap font
- BS Bootstrap
- GSTT GoptStop
- HDA AudioDxe
- KKT KevTester
- LNX OpenLinuxBoot
- NETB OpenNetworkBoot
- MMDD MmapDump
- OCPAVP PavpProvision
- \bullet OCRST ResetSystem
- OCUI OpenCanopy
- OC OpenCore main, also OcMainLib
- VMOPT VerifyMemOpt

Libraries:

• AAPL — OcLogAggregatorLib, Apple EfiBoot logging

- \bullet OCABC OcAfterBootCompatLib
- OCAE OcAppleEventLib
- OCAK OcAppleKernelLib
- OCAU OcAudioLib
- OCA OcAcpiLib
- OCBP OcAppleBootPolicyLib
- OCB OcBootManagementLib
- OCLBT OcBlitLib
- OCCL OcAppleChunkListLib
- OCCPU OcCpuLib
- OCC OcConsoleLib
- OCDC OcDriverConnectionLib
- OCDH OcDataHubLib
- OCDI OcAppleDiskImageLib
- ullet OCDM OcDeviceMiscLib
- OCFS OcFileLib
- OCFV OcFirmwareVolumeLib
- OCHS OcHashServicesLib
- OCI4 OcAppleImg4Lib
- $\bullet \ \ \mathtt{OCIC} \\ \mathrm{OcImageConversionLib}$
- OCII OcInputLib
- OCJS OcApfsLib
- OCKM OcAppleKeyMapLib
- OCL OcLogAggregatorLib
- OCM OcMiscLib
- OCMCO OcMachoLib
- OCME OcHeciLib
- OCMM OcMemoryLib
- OCPE OcPeCoffLib, OcPeCoffExtLib
- OCPI OcFileLib, partition info
- OCPNG OcPngLib
- OCRAM OcAppleRamDiskLib
- OCRTC OcRtcLib
- OCSB OcAppleSecureBootLib
- OCSMB OcSmbiosLib
- ullet OCSMC OcSmcLib
- OCST OcStorageLib
- OCTPL OcTemplateLib
- $\bullet \ \ \mathtt{OCUC} \mathrm{OcUnicodeCollationLib} \\$
- OCUT OcAppleUserInterfaceThemeLib
- OCVAR OcVariableLib
- OCXML OcXmlLib

8.5 Entry Properties

1. Arguments

Type: plist string Failsafe: Empty

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

2. Auxiliary

Type: plist boolean

Failsafe: false

Description: Set to true to hide this entry when HideAuxiliary is also set to true. Press the Spacebar key to enter "Extended Mode" and display the entry when hidden.

3. Comment

Type: plist string

Failsafe: Empty

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

4. Enabled

Type: plist boolean Failsafe: false

Description: Set to true activate this entry.

5. Flavour

Type: plist string Failsafe: Auto

Description: Specify the content flavour for this entry. See OC_ATTR_USE_FLAVOUR_ICON flag for documentation.

6. FullNvramAccess

Type: plist boolean Failsafe: false

Description: Disable OpenRuntime NVRAM protection during usage of a tool.

This disables all of the NVRAM protections provided by OpenRuntime.efi, during the time a tool is in use. It should normally be avoided, but may be required for instance if a tool needs to access NVRAM directly without the redirections put in place by RequestBootVarRouting.

Note: This option is only valid for Tools and cannot be specified for Entries (is always false).

7. Name

Type: plist string Failsafe: Empty

Description: Human readable entry name displayed in the OpenCore picker.

8 Path

Type: plist string Failsafe: Empty

Description: Entry location depending on entry type.

- Entries specify external boot options, and therefore take device paths in the Path key. Care should be exercised as these values are not checked. Example: PciRoot(0x0)/Pci(0x1,0x1)/.../\EFI\COOL.EFI
- Tools specify internal boot options, which are part of the bootloader vault, and therefore take file paths relative to the OC/Tools directory. Example: OpenShell.efi.

9. RealPath

Type: plist boolean Failsafe: false

Description: Pass full path to the tool when launching.

This should typically be disabled as passing the tool directory may be unsafe with tools that accidentally attempt to access files without checking their integrity. Reasons to enable this property may include cases where tools cannot work without external files or may need them for enhanced functionality such as memtest86 (for logging and configuration), or Shell (for automatic script execution).

Note: This option is only valid for Tools and cannot be specified for Entries (is always true).

10. TextMode

Type: plist boolean

Failsafe: false

Description: Run the entry in text mode instead of graphics mode.

This setting may be beneficial for some older tools that require text output as all the tools are launched in graphics mode by default. Refer to the Output Properties section below for information on text modes.

8.6 Security Properties

1. AllowSetDefault

Type: plist boolean

Failsafe: false

Description: Allow CTRL+Enter and CTRL+Index handling to set the default boot option in the OpenCore picker.

Note 1: May be used in combination with Shift+Enter or Shift+Index when PollAppleHotKeys is enabled.

Note 2: In order to support systems with unresponsive modifiers during preboot (which includes V1 and V2 KeySupport mode on some firmware) OpenCore also allows holding the =/+ key in order to trigger 'set default' mode.

2. ApECID

Type: plist integer, 64 bit

Failsafe: 0

Description: Apple Enclave Identifier.

Setting this value to any non-zero 64-bit integer will allow using personalised Apple Secure Boot identifiers. To use this setting, generate a random 64-bit number with a cryptographically secure random number generator. As an alternative, the first 8 bytes of SystemUUID can be used for ApECID, this is found in macOS 11 for Macs without the T2 chip.

With this value set and SecureBootModel valid (and not Disabled), it is possible to achieve Full Security of Apple Secure Boot.

To start using personalised Apple Secure Boot, the operating system must be reinstalled or personalised. Until the operating system is personalised, only macOS DMG recovery can be loaded. In cases where DMG recovery is missing, it can be downloaded by using the macrecovery utility and saved in com.apple.recovery.boot as explained in the Tips and Tricks section. Note that DMG loading needs to be set to Signed to use any DMG with Apple Secure Boot.

To personalise an existing operating system, use the bless command after loading to macOS DMG recovery. Mount the system volume partition, unless it has already been mounted, and execute the following command:

```
bless --folder "/Volumes/Macintosh HD/System/Library/CoreServices" \
   --bootefi --personalize
```

On macOS 11 and newer the dedicated x86legacy model always uses ApECID. When this configuration setting is left as 0 first 8 bytes of system-id variable are used instead.

On macOS versions before macOS 11, which introduced a dedicated x861egacy model for models without the T2 chip, personalised Apple Secure Boot may not work as expected. When reinstalling the operating system, the macOS Installer from macOS 10.15 and older will often run out of free memory on the /var/tmp partition when trying to install macOS with the personalised Apple Secure Boot. Soon after downloading the macOS installer image, an Unable to verify macOS error message will appear.

To workaround this issue, allocate a dedicated RAM disk of 2 MBs for macOS personalisation by entering the following commands in the macOS recovery terminal before starting the installation:

```
disk=$(hdiutil attach -nomount ram://4096)
diskutil erasevolume HFS+ SecureBoot $disk
diskutil unmount $disk
mkdir /var/tmp/OSPersonalizationTemp
diskutil mount -mountpoint /var/tmp/OSPersonalizationTemp $disk
```

3. AuthRestart

 $\mathbf{Type} \colon \mathtt{plist} \ \mathtt{boolean}$

Failsafe: false

Description: Enable VirtualSMC-compatible authenticated restart.

Authenticated restart is a way to reboot FileVault 2 enabled macOS without entering the password. A dedicated terminal command can be used to perform authenticated restarts: sudo fdesetup authrestart. It is also used when installing operating system updates.

VirtualSMC performs authenticated restarts by splitting and saving disk encryption keys between NVRAM and RTC, which despite being removed as soon as OpenCore starts, may be considered a security risk and thus is optional.

4. BlacklistAppleUpdate Type: plist boolean

Failsafe: false

Description: Ignore boot options trying to update Apple peripheral firmware (e.g. MultiUpdater.efi).

Note: Certain operating systems, such as macOS Big Sur, are incapable of disabling firmware updates by using the run-efi-updater NVRAM variable.

5. DmgLoading

Type: plist string Failsafe: Signed

Description: Define Disk Image (DMG) loading policy used for macOS Recovery.

Valid values:

- Disabled loading DMG images will fail. The Disabled policy will still let the macOS Recovery load in most cases as typically, there are boot.efi files compatible with Apple Secure Boot. Manually downloaded DMG images stored in com.apple.recovery.boot directories will not load, however.
- Signed only Apple-signed DMG images will load. Due to the design of Apple Secure Boot, the Signed policy will let any Apple-signed macOS Recovery load regardless of the Apple Secure Boot state, which may not always be desired. While using signed DMG images is more desirable, verifying the image signature may slightly slow the boot time down (by up to 1 second).
- Any any DMG images will mount as normal filesystems. The Any policy is strongly discouraged and will result in boot failures when Apple Secure Boot is active.

6. EnablePassword

Type: plist boolean

Failsafe: false

Description: Enable password protection to facilitate sensitive operations.

Password protection ensures that sensitive operations such as booting a non-default operating system (e.g. macOS recovery or a tool), resetting NVRAM storage, trying to boot into a non-default mode (e.g. verbose mode or safe mode) are not allowed without explicit user authentication by a custom password. Currently, password and salt are hashed with 5000000 iterations of SHA-512.

Note: This functionality is still under development and is not ready for production environments.

7. ExposeSensitiveData

Type: plist integer

Failsafe: 0x6

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

- 0x01 Expose the printable booter path as a UEFI variable.
- 0x02 Expose the OpenCore version as a UEFI variable.
- 0x04 Expose the OpenCore version in the OpenCore picker menu title.
- 0x08 Expose OEM information as a set of UEFI variables.

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

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

To use a booter path to mount a booter volume, use the following command in macOS:

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

To obtain the current OpenCore version, use the following command in macOS:

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

If the OpenCore version is not exposed the variable will contain UNK-000-0000-00 sequence.

To obtain OEM information, use the following commands in macOS:

```
nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:oem-product # SMBIOS Type1 ProductName
nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:oem-vendor # SMBIOS Type2 Manufacturer
nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:oem-board # SMBIOS Type2 ProductName
```

8. HaltLevel

Type: plist integer, 64 bit

Failsafe: 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.

Note 1: A halt will only occur if bit 0 (i.e. enable logging) for Target under section Misc->Debug is set.

Note 2: A halt will only occur after the configuration is loaded and logging is configured. If any log messages occur at the specified halt level in early log (i.e. before this), they will cause a halt when they are flushed to the log once it has been configured.

9. PasswordHash

Type: plist data 64 bytes

Failsafe: all zero

Description: Password hash used when EnablePassword is set.

10. PasswordSalt

Type: plist data Failsafe: empty

Description: Password salt used when EnablePassword is set.

11. ScanPolicy

Type: plist integer, 32 bit

Failsafe: 0x10F0103

Description: Define operating system detection policy.

This value allows preventing scanning (and booting) untrusted sources based on a bitmask (sum) of a set of flags. As it is not possible to reliably detect every file system or device type, this feature cannot be fully relied upon in open environments, and additional measures are to be applied.

Third party drivers may introduce additional security (and performance) consideratons following the provided scan policy. The active Scan policy is exposed in the scan-policy variable of 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102 GUID for UEFI Boot Services only.

- 0x00000001 (bit 0) 0C_SCAN_FILE_SYSTEM_LOCK, restricts scanning to only known file systems defined as a part of this policy. File system drivers may not be aware of this policy. Hence, to avoid mounting of undesired file systems, drivers for such file systems should not be loaded. This bit does not affect DMG mounting, which may have any file system. Known file systems are prefixed with OC_SCAN_ALLOW_FS_.
- 0x00000002 (bit 1) 0C_SCAN_DEVICE_LOCK, restricts scanning to only known device types defined as a part of this policy. It is not always possible to detect protocol tunneling, so be aware that on some systems, it may be possible for e.g. USB HDDs to be recognised as SATA instead. Cases like this must be reported. Known device types are prefixed with OC_SCAN_ALLOW_DEVICE_.
- 0x00000100 (bit 8) OC_SCAN_ALLOW_FS_APFS, allows scanning of APFS file system.
- 0x00000200 (bit 9) 0C_SCAN_ALLOW_FS_HFS, allows scanning of HFS file system.
- 0x00000400 (bit 10) 0C_SCAN_ALLOW_FS_ESP, allows scanning of EFI System Partition file system.
- 0x00000800 (bit 11) 0C_SCAN_ALLOW_FS_NTFS, allows scanning of NTFS (Msft Basic Data) file system.
- 0x00001000 (bit 12) 0C_SCAN_ALLOW_FS_LINUX_ROOT, allows scanning of Linux Root file systems.
- 0x00002000 (bit 13) 0C_SCAN_ALLOW_FS_LINUX_DATA, allows scanning of Linux Data file systems.
- 0x00004000 (bit 14) 0C_SCAN_ALLOW_FS_XB00TLDR, allows scanning the Extended Boot Loader Partition as defined by the Boot Loader Specification.
- 0x00010000 (bit 16) OC_SCAN_ALLOW_DEVICE_SATA, allow scanning SATA devices.
- 0x00020000 (bit 17) 0C_SCAN_ALLOW_DEVICE_SASEX, allow scanning SAS and Mac NVMe devices.
- 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 and old SATA.
- 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.
- 0x01000000 (bit 24) 0C_SCAN_ALLOW_DEVICE_PCI, allow scanning devices directly connected to PCI bus (e.g. VIRTIO).

Note: Given the above description, a value of 0xF0103 is expected to do the following:

- Permit scanning SATA, SAS, SCSI, and NVMe devices with APFS file systems.
- Prevent scanning any devices with HFS or FAT32 file systems.
- Prevent scanning APFS file systems on USB, CD, 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

12. 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 Matching model for current SMBIOS.
- 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)
- x86legacy Macs without T2 chip and VMs. Minimum macOS 11.0.1 (20B29)

Warning: Not all Apple Secure Boot models are supported on all hardware configurations.

Apple Secure Boot appeared in macOS 10.13 on models with T2 chips. Prior to macOS 12 PlatformInfo and SecureBootModel were independent, allowing Apple Secure Boot can be used with any SMBIOS with and without T2. Starting with macOS 12 SecureBootModel must match the SMBIOS Mac model. Default model derives the model based on SMBIOS board identifier, either set automatically via the Generic section or set manually via the SMBIOS section. If there is no board identifier override the model will be derived heuristically from OEM SMBIOS.

Setting SecureBootModel to any valid value but Disabled is equivalent to Medium Security of Apple Secure Boot. The ApECID value must also be specified to achieve Full Security. Check ForceSecureBootScheme when using Apple Secure Boot on a virtual machine.

Note that enabling Apple Secure Boot is demanding on invalid configurations, faulty macOS installations, and on unsupported setups.

Things to consider:

- (a) As with T2 Macs, all unsigned kernel extensions as well as several signed kernel extensions, including NVIDIA Web Drivers, cannot be installed.
- (b) The list of cached kernel extensions may be different, resulting in a need to change the list of Added or Forced kernel extensions. For example, IO80211Family cannot be injected in this case.
- (c) System volume alterations on operating systems with sealing, such as macOS 11, may result in the operating system being unbootable. Do not try to disable system volume encryption unless Apple Secure Boot is disabled.
- (d) Boot failures might occur when the platform requires certain settings, but they have not been enabled because the associated issues were not discovered earlier. Be extra careful with IgnoreInvalidFlexRatio or HashServices.
- (e) Operating systems released before Apple Secure Boot was released (e.g. macOS 10.12 or earlier), will still boot until UEFI Secure Boot is enabled. This is so because Apple Secure Boot treats these as incompatible and they are then handled by the firmware (as 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) As the Default value will increase with time to support the latest major released operating system, it is not recommended to use the ApECID and the Default settings together.
- (h) Installing macOS with Apple Secure Boot enabled is not possible while using HFS+ target volumes. This may include HFS+ formatted drives when no spare APFS drive is available.

The installed operating system may have sometimes outdated Apple Secure Boot manifests on the Preboot partition, resulting in boot failures. This is likely to be the case when an "OCB: Apple Secure Boot prohibits this boot entry, enforcing!" message is logged.

When this happens, either reinstall the operating system or copy the manifests (files with .im4m extension, such as boot.efi.j137.im4m) from /usr/standalone/i386 to /Volumes/Preboot/<UUID>/System/Library/CoreServices. Here, <UUID> is the system volume identifier. On HFS+ installations, the manifests should be copied to /System/Library/CoreServices on the system volume.

For more details on how to configure Apple Secure Boot with UEFI Secure Boot, refer to the UEFI Secure Boot section.

13. Vault

Type: plist string Failsafe: Secure

Description: Enables the OpenCore vaulting mechanism.

Valid values:

- Optional require nothing, no vault is enforced, insecure.
- Basic require vault.plist file present in OC directory. This provides basic filesystem integrity verification and may protect from unintentional filesystem corruption.
- Secure require vault.sig signature file for vault.plist in OC directory. This includes Basic integrity checking but also attempts to build a trusted bootchain.

The vault.plist file should contain SHA-256 hashes for all files used by OpenCore. The presence of this file is highly recommended to ensure that unintentional file modifications (including filesystem corruption) do not go unnoticed. To create this file automatically, use the create_vault.sh script. Notwithstanding the underlying file system, the path names and cases between config.plist and vault.plist must match.

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

To embed the public key, either one of the following should be performed:

- 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.

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

The steps to binary patch OpenCore.efi are:

- 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.

A script to do this is privided in OpenCore releases:

/Utilities/CreateVault/sign.command /Volumes/EFI/EFI/OC

Note 1: While it may appear obvious, an external method is required to verify <code>OpenCore.efi</code> and <code>BOOTx64.efi</code> for secure boot path. For this, it is recommended to enable UEFI SecureBoot using a custom certificate and to sign <code>OpenCore.efi</code> and <code>BOOTx64.efi</code> with a custom key. More details on customising secure boot on modern firmware can be found in the Taming UEFI SecureBoot paper (in Russian).

Note 2: Regardless of this option, vault.plist is always used when present, and both vault.plist and vault.sig are used and required when a public key is embedded into OpenCore.efi, and errors will abort the boot process in either case. Setting this option allows OpenCore to warn the user if the configuration is not as required to achieve an expected higher security level.

8.7 Serial Properties

1. Custom

Type: plist dict

Description: Update serial port properties in BaseSerialPortLib16550.

This section lists the PCD values that are used by the BaseSerialPortLib16550. When option Override is set to false, this dictionary is optional.

2. Init

Type: plist boolean Failsafe: false

Description: Perform serial port initialisation.

This option will perform serial port initialisation within OpenCore prior to enabling (any) debug logging.

Refer to the Debugging section for details.

3. Override

Type: plist boolean Failsafe: false

Description: Override serial port properties. When this option is set to false, no keys from Custom will be overridden.

This option will override serial port properties listed in the Serial Custom Properties section below.

8.7.1 Serial Custom Properties

1. BaudRate

Type: plist integer Failsafe: 115200

Description: Set the baud rate for serial port.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialBaudRate defined in Mde-ModulePkg.dec.

2. ClockRate

Type: plist integer Failsafe: 1843200

Description: Set the clock rate for serial port.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialClockRate defined in Mde-ModulePkg.dec.

3. DetectCable

Type: plist boolean Failsafe: false

Description: Enable serial port cable detection.

This option will override the value of <code>gEfiMdeModulePkgTokenSpaceGuid.PcdSerialDetectCable</code> defined in <code>MdeModulePkg.dec</code>.

4. ExtendedTxFifoSize

Type: plist integer

Failsafe: 64

Description: Set the extended transmit FIFO size for serial port.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialExtendedTxFifoSize defined in MdeModulePkg.dec.

5. FifoControl

Type: plist integer

Failsafe: 0x07

Description: Configure serial port FIFO Control settings.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialFifoControl defined in MdeModulePkg.dec.

6. LineControl

Type: plist integer

Failsafe: 0x07

Description: Configure serial port Line Control settings.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialLineControl defined in MdeModulePkg.dec.

7. PciDeviceInfo

Type: plist data Failsafe: 0xFF

Description: Set PCI serial device information.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialPciDeviceInfo defined in MdeModulePkg.dec.

Note: The maximum allowed size of this option is 41 bytes. Refer to a cidanthera/bugtracker#1954 for more details.

Note 2: This option can be set by running the FindSerialPort tool.

8. RegisterAccessWidth

Type: plist integer

Failsafe: 8

Description: Set serial port register access width.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialRegisterAccessWidth defined in MdeModulePkg.dec.

9. RegisterBase

Type: plist integer Failsafe: 0x03F8

Description: Set the base address of serial port registers.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialRegisterBase defined in MdeModulePkg.dec.

10. RegisterStride

Type: plist integer

Failsafe: 1

Description: Set the serial port register stride in bytes.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialRegisterStride defined in MdeModulePkg.dec.

11. UseHardwareFlowControl

Type: plist boolean

Failsafe: false

Description: Enable serial port hardware flow control.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialUseHardwareFlowControl defined in MdeModulePkg.dec.

12. UseMmio

Type: plist boolean

Failsafe: false

Description: Indicate whether the serial port registers are in MMIO space.

This option will override the value of gEfiMdeModulePkgTokenSpaceGuid.PcdSerialUseMmio defined in Mde-ModulePkg.dec.

9 NVRAM

9.1 Introduction

This section allows setting non-volatile UEFI variables commonly described as NVRAM variables. Refer to man nvram for details. The macOS operating system extensively uses NVRAM variables for OS — Bootloader — Firmware intercommunication. Hence, the supply of several NVRAM variables is required for the proper functioning of macOS.

Each NVRAM variable consists of its name, value, attributes (refer to UEFI specification), and its GUID, representing which 'section' the NVRAM variable belongs to. The macOS operating system makes use of several GUIDs, including but not limited to:

- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14 (APPLE_VENDOR_VARIABLE_GUID)
- 7C436110-AB2A-4BBB-A880-FE41995C9F82 (APPLE_BOOT_VARIABLE_GUID)
- 5EDDA193-A070-416A-85EB-2A1181F45B18 (Apple Hardware Configuration Storage for MacPro7,1)
- 8BE4DF61-93CA-11D2-AAOD-00E098032B8C (EFI_GLOBAL_VARIABLE_GUID)
- 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102 (OC_VENDOR_VARIABLE_GUID)

Note: Some of the variables may be added by the PlatformNVRAM or Generic subsections of the PlatformInfo section. Please ensure that variables set in this section do not conflict with items in those subsections as the implementation behaviour is undefined otherwise.

The OC_FIRMWARE_RUNTIME protocol implementation, currently offered as a part of the OpenRuntime driver, is often required for macOS to function properly. While this brings many benefits, there are some limitations that should be considered for certain use cases.

1. Not all tools may be aware of protected namespaces.

When RequestBootVarRouting is used, Boot-prefixed variable access is restricted and protected in a separate namespace. To access the original variables, tools must be aware of the OC_FIRMWARE_RUNTIME logic.

9.2 Properties

1. Add

Type: plist dict

Description: Sets NVRAM variables from a map (plist dict) of GUIDs to a map (plist dict) of variable names and their values in plist multidata format. GUIDs must be provided in canonic string format in upper or lower case (e.g. 8BE4DF61-93CA-11D2-AAOD-00E098032B8C).

The EFI_VARIABLE_BOOTSERVICE_ACCESS and EFI_VARIABLE_RUNTIME_ACCESS attributes of created variables are set. Variables will only be set if not present or deleted. That is, to overwrite an existing variable value, add the variable name to the Delete section. This approach enables the provision of default values until the operating system takes the lead.

Note: The implementation behaviour is undefined when the plist key does not conform to the GUID format.

2. Delete

Type: plist dict

Description: Removes NVRAM variables from a map (plist dict) of GUIDs to an array (plist array) of variable names in plist string format.

3. LegacyOverwrite

Type: plist boolean

Failsafe: false

Description: Permits overwriting firmware variables from nvram.plist.

Note: Only variables accessible from the operating system will be overwritten.

4. LegacySchema

Type: plist dict

Description: Allows setting certain NVRAM variables from a map (plist dict) of GUIDs to an array (plist array) of variable names in plist string format.

* value can be used to accept all variables for certain GUID.

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

5. WriteFlash

Type: plist boolean

Failsafe: false

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

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

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

A continuously updated variable list can be found in a corresponding document: NVRAM Variables.

9.3 Mandatory Variables

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

The following variables are mandatory for macOS functioning:

- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeatures 32-bit FirmwareFeatures. Present on all Macs to avoid extra parsing of SMBIOS tables.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeaturesMask 32-bit FirmwareFeaturesMask. Present on all Macs to avoid extra parsing of SMBIOS tables.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:MLB
 BoardSerialNumber. Present on newer Macs (2013+ at least) to avoid extra parsing of SMBIOS tables, especially in boot.efi.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ROM

 Primary network adapter MAC address or replacement value. Present on newer Macs (2013+ at least) to avoid accessing special memory region, especially in boot.efi.

9.4 Recommended Variables

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

- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:BridgeOSHardwareModel
 Bridge OS hardware model variable used to propagate to IODT bridge-model by EfiBoot. Read by hw.target
 sysctl, used by SoftwareUpdateCoreSupport.
- 7C436110-AB2A-4BBB-A880-FE41995C9F82:csr-active-config 32-bit System Integrity Protection bitmask. Declared in XNU source code in csr.h.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14: ExtendedFirmwareFeatures
 Combined FirmwareFeatures and ExtendedFirmwareFeatures. Present on newer Macs to avoid extra parsing of SMBIOS tables.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeaturesMask

 Combined FirmwareFeaturesMask and ExtendedFirmwareFeaturesMask. Present on newer Macs to avoid extra parsing of SMBIOS tables.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_BID

Hardware BoardProduct (e.g. Mac-35C1E88140C3E6CF). Not present on real Macs, but used to avoid extra parsing of SMBIOS tables, especially in boot.efi.

- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_MLB
 - Hardware BoardSerialNumber. Override for MLB. Present on newer Macs (2013+ at least).
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_ROM
 - Hardware ROM. Override for ROM. Present on newer Macs (2013+ at least).
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:SSN
 - Serial number. Present on newer Macs (2013+ at least).
- 7C436110-AB2A-4BBB-A880-FE41995C9F82:prev-lang:kbd

ASCII string defining default keyboard layout. Format is lang-COUNTRY:keyboard, e.g. ru-RU:252 for Russian locale and ABC keyboard. Also accepts short forms: ru:252 or ru:0 (U.S. keyboard, compatible with 10.9). Full

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

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:security-mode

ASCII string defining FireWire security mode. Legacy, can be found in IOFireWireFamily source code in IOFireWireController.cpp. It is recommended not to set this variable, which may speedup system startup. Setting to full is equivalent to not setting the variable and none disables FireWire security.

- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:UIScale
 - One-byte data defining boot.efi user interface scaling. Should be **01** for normal screens and **02** for HiDPI screens.
- 7C436110-AB2A-4BBB-A880-FE41995C9F82:ForceDisplayRotationInEFI 32-bit integer defining display rotation. Can be **0** for no rotation or any of 90, 180, 270 for matching rotation in degrees.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14: DefaultBackgroundColor
 Four-byte BGRA data defining boot.efi user interface background colour. Standard colours include BF BF BF 00 (Light Gray) and 00 00 00 (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)
- ioaccel_debug=VALUE (IOAccelerator debug mask)
- keepsyms=1 (show panic log debug symbols)
- kextlog=VALUE (kernel extension loading debug mask)
- nvram-log=1 (enables AppleEFINVRAM logs)
- nv_disable=1 (disables NVIDIA GPU acceleration)
- nvda_drv=1 (legacy way to enable NVIDIA web driver, removed in 10.12)
- npci=0x2000 (legacy, disables kIOPCIConfiguratorPFM64)
- lapic_dont_panic=1 (disable lapic spurious interrupt panic on AP cores)
- panic_on_display_hang=1 (trigger panic on display hang)
- panic_on_gpu_hang=1 (trigger panic on GPU hang)
- serial=VALUE (configure serial logging mode) The following bits are used by XNU:
 - * 0x01 (SERIALMODE_OUTPUT, bit 0) Enable serial output.
 - * 0x02 (SERIALMODE_INPUT, bit 1) Enable serial input.
 - * 0x04 (SERIALMODE_SYNCDRAIN, bit 2) Enable serial drain synchronisation.
 - * 0x08 (SERIALMODE_BASE_TTY, bit 3) Load Base/Recovery/FVUnlock TTY.
 - * 0x10 (SERIALMODE_NO_IOLOG, bit 4) Prevent IOLogs writing to serial.
- slide=VALUE (manually set KASLR slide)
- smcdebug=VALUE (AppleSMC debug mask)
- spin_wait_for_gpu=1 (reduces GPU timeout on high load)
- -amd_no_dgpu_accel (alternative to WhateverGreen's -radvesa for new GPUs)
- -nehalem_error_disable (disables the AppleTyMCEDriver)
- -no_compat_check (disable model checking on 10.7+)
- -s (single mode)
- v (verbose mode)
- -x(safe mode)

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 Ox. 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: 0K.
- 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).

    OxOC - UNKNOWN.
```

In 10.15, debugging support was defective up to the 10.15.4 release due to refactoring issues as well as the 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.
    * 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).
```

* 2 — enables serial logging for EXITBS:START onwards.

* 1 — enables serial logging for EXITBS: END 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)
- debug=VALUE deprecated starting from 10.15.
 - * 1 enables print something to BOOTER.LOG (stripped code implies there may be a crash)
 - * 2 enables perf logging to /efi/debug-log in the device three
 - * 4 enables timestamp printing for styled printf calls
- level=VALUE deprecated starting from 10.15. Verbosity level of DEBUG output. Everything but 0x80000000 is stripped from the binary, and this is the default value.

Note: Enable the AppleDebug option to display verbose output from boot.efi on modern macOS versions. This will save the log to the general OpenCore log file. For versions before 10.15.4, set bootercfg to log=1. This will print verbose output onscreen.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:bootercfg-once

Booter arguments override removed after first launch. Otherwise equivalent to bootercfg.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:csr-data

Specify sources of kexts which will be approved regardless of SIP CSR_ALLOW_UNAPPROVED_KEXTS value. Example contents:

<dict><key>kext-allowed-teams</key><array><string>{DEVELOPER-TEAM-ID}</string></array></dict>%00

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:efiboot-perf-record

Enable performance log saving in boot.efi. Performance log is saved to physical memory and is pointed to by the efiboot-perf-record-data and efiboot-perf-record-size variables. Starting from 10.15.4, it can also be saved to the OpenCore log by setting the AppleDebug option.

- $\bullet \ \ \texttt{7C436110-AB2A-4BBB-A880-FE41995C9F82:fmm-computer-name}$
 - Current saved host name. ASCII string.
- 7C436110-AB2A-4BBB-A880-FE41995C9F82:nvda_drv

NVIDIA Web Driver control variable. Takes ASCII digit 1 or 0 to enable or disable installed driver.

- 7C436110-AB2A-4BBB-A880-FE41995C9F82:run-efi-updater
 - Override EFI firmware updating support in macOS (MultiUpdater, ThorUtil, and so on). Setting this to No or alternative boolean-castable value will prevent any firmware updates in macOS starting with 10.10 at least.
- 7C436110-AB2A-4BBB-A880-FE41995C9F82:StartupMute

Mute startup chime sound in firmware audio support. 8-bit integer. The value of 0x00 means unmuted. Missing variable or any other value means muted.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:SystemAudioVolume

System audio volume level for firmware audio support. 8-bit unsigned integer. The bit of 0x80 means muted. The remaining bits are used to encode the raw amplifier gain setting to be applied to the audio amplifier in use. Exactly what this value means depends on the codec (and potentially on the specific amplifier within the codec). This value is capped by macOS to the MaximumBootBeepVolume AppleHDA layout value, to avoid over-loud audio playback in the firmware.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:SystemAudioVolumeDB

Current system audio volume level in decibels (dB). 8-bit signed integer. The value represents the audio offset (gain if positive, attenuation if negative) in dB relative to the amplifier reference value of 0 dB. Exactly which volume level is represented by 0 dB depends on the codec (and potentially on the specific amplifier within the codec) but it is normally at or near the maximum available amplifier volume. Typical values of this variable range from approximately -60 (the exact value depends on the audio hardware) up to exactly 0. On non-typical audio hardware, the value may go above zero.

Note: Unlike SystemAudioVolume, this value is not capped.

• 5EDDA193-A070-416A-85EB-2A1181F45B18:PEXConf

PCI expansion slot configuration for MacPro7,1. 8-byte sequence describing default PCI slot configuration. Each byte refers to a configuration for a dedicated PCI slot.

- Slot 1 resides at IOService:/AppleACPIPlatformExpert/PC01@0/AppleACPIPCI/BR1A@0 and its path is hardcoded. This slot is not behind a muxer.
- Slot 3 resides at IOService:/AppleACPIPlatformExpert/PC03@0/AppleACPIPCI/BR3A@0 and its path is hardcoded. This slot is not behind a muxer.
- Slots 2, 4-8 are dynamic and are matched based on AAPL, slot-name property with Slot-N value, where N is the slot number. All these slots are behind the muxer.

Refer to the support page for more details on how MacPro7,1 slots are configured.

User PCI expansion slot configuration for MacPro7,1. 8-byte sequence describing user PCI slot configuration.

• 5EDDA193-A070-416A-85EB-2A1181F45B18:SlotUtilPEXConf

10 PlatformInfo

Platform information consists of several identification fields generated or filled manually to be compatible with macOS services. The base part of the configuration may be obtained from AppleModels, which itself generates a set of interfaces based on a database in YAML format. These fields are written to three 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 all the values are specified (the default), and semi-automatic, where (Automatic) only certain values are specified, and later used for system configuration.

The dmidecode utility can be used to inspect SMBIOS contents and a version with macOS specific enhancements can be downloaded from Acidanthera/dmidecode.

10.1 Properties

1. Automatic

Type: plist boolean Failsafe: false

Description: Generate PlatformInfo based on the Generic section instead of using values from the 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.
- When disabled Generic section is unused.

Warning: Setting this option to false is strongly discouraged when intending to update platform information. A false setting is typically only valid for minor corrections to SMBIOS values on legacy Apple hardware. In all other cases, setting Automatic to false may lead to hard-to-debug errors resulting from inconsistent or invalid settings.

2. CustomMemory

Type: plist boolean Failsafe: false

Description: Use custom memory configuration defined in the Memory section. This completely replaces any existing memory configuration in SMBIOS, and is only active when UpdateSMBIOS is set to true.

 $3.\ {\tt DataHub}$

Type: plist dictionary

Description: Update Data Hub fields in non-Automatic mode.

Note: This section is ignored and may be removed when Automatic is true.

4. Generic

Type: plist dictionary

Description: Update all fields in Automatic mode.

Note: This section is ignored but may not be removed when Automatic is false.

5. Memory

Type: plist dictionary

Description: Define custom memory configuration.

Note: This section is ignored and may be removed when CustomMemory is false.

6. PlatformNVRAM

Type: plist dictionary

Description: Update platform NVRAM fields in non-Automatic mode.

Note: This section is ignored and may be removed when Automatic is true.

7. SMBIOS

Type: plist dictionary

Description: Update SMBIOS fields in non-Automatic mode.

Note: This section is ignored and may be removed when Automatic is true.

8. UpdateDataHub

Type: plist boolean

Failsafe: false

Description: Update Data Hub fields. These fields are read from the Generic or DataHub sections depending on the setting of the Automatic property.

Note: The implementation of the Data Hub protocol in EFI firmware on virtually all systems, including Apple hardware, means that existing Data Hub entries cannot be overridden. New entries are added to the end of the Data Hub instead, with macOS ignoring old entries. This can be worked around by replacing the Data Hub protocol using the ProtocolOverrides section. Refer to the DataHub protocol override description for details.

9. UpdateNVRAM

Type: plist boolean

Failsafe: false

Description: Update NVRAM fields related to platform information.

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

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

$10. \ {\tt UpdateSMBIOS}$

Type: plist boolean

Failsafe: false

Description: Update SMBIOS fields. These fields are read from the **Generic** or **SMBIOS** sections depending on the setting of the **Automatic** property.

11. UpdateSMBIOSMode

Type: plist string Failsafe: Create

Description: Update SMBIOS fields approach:

- TryOverwrite Overwrite if new size is <= than the page-aligned original and there are no issues with legacy region unlock. Create otherwise. Has issues on some types of firmware.
- 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 firmware 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 the Custom approach that it makes SMBIOS updates exclusive to macOS, avoiding a collision with existing Windows activation and custom OEM software but potentially obstructing the operation of Apple-specific tools.

12. UseRawUuidEncoding

Type: plist boolean

Failsafe: false

Description: Use raw encoding for SMBIOS UUIDs.

Each UUID AABBCCDD-EEFF-GGHH-IIJJ-KKLLMMNNOOPP is essentially a hexadecimal 16-byte number. It can be encoded in two ways:

- Big Endian by writing all the bytes as they are without making any order changes ({AA BB CC DD EE FF GG HH II JJ KK LL MM NN OO PP}). This method is also known as RFC 4122 encoding or Raw encoding.
- Little Endian by interpreting the bytes as numbers and using Little Endian byte representation ({DD CC BB AA FF EE HH GG II JJ KK LL MM NN 00 PP}).

The SMBIOS specification did not explicitly specify the encoding format for the UUID up to SMBIOS 2.6, where it stated that Little Endian encoding shall be used. This led to the confusion in both firmware implementations and system software as different vendors used different encodings prior to that.

- Apple uses the Big Endian format everywhere but it ignores SMBIOS UUID within macOS.
- dmidecode uses the Big Endian format for SMBIOS 2.5.x or lower and the Little Endian format for 2.6 and newer. Acidanthera dmidecode prints all three.
- Windows uses the Little Endian format everywhere, but this only affects the visual representation of the values.

OpenCore always sets a recent SMBIOS version (currently 3.2) when generating the modified DMI tables. If UseRawUuidEncoding is enabled, the Big Endian format is used to store the SystemUUID data. Otherwise, the Little Endian format is used.

Note: This preference does not affect UUIDs used in DataHub and NVRAM as they are not standardised and are added by Apple. Unlike SMBIOS, they are always stored in the Big Endian format.

10.2 DataHub Properties

1. ARTFrequency

Type: plist integer, 64-bit Failsafe: 0 (Automatic)

Description: Sets ARTFrequency in gEfiProcessorSubClassGuid.

This value contains CPU ART frequency, also known as crystal clock frequency. Its existence is exclusive to the Skylake generation and newer. The value is specified in Hz, and is normally 24 MHz for the client Intel segment, 25 MHz for the server Intel segment, and 19.2 MHz for Intel Atom CPUs. macOS till 10.15 inclusive assumes 24 MHz by default.

Note: On Intel Skylake X ART frequency may be a little less (approx. 0.25%) than 24 or 25 MHz due to special EMI-reduction circuit as described in Acidanthera Bugtracker.

2. BoardProduct

Type: plist string

Failsafe: Empty (Not installed)

Description: Sets board-id in gEfiMiscSubClassGuid. The value found on Macs is equal to SMBIOS

BoardProduct in ASCII.

 $3. \; {\tt BoardRevision}$

Type: plist data, 1 byte

Failsafe: 0

Description: Sets board-rev in gEfiMiscSubClassGuid. The value found on Macs seems to correspond to internal board revision (e.g. 01).

4. DevicePathsSupported

Type: plist integer, 32-bit Failsafe: 0 (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.

5. FSBFrequency

Type: plist integer, 64-bit Failsafe: 0 (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.

Note: This value is not used on Skylake and newer but is still provided to follow suit.

6. InitialTSC

Type: plist integer, 64-bit

Failsafe: 0

Description: Sets Initial TSC in gEfiProcessorSubClassGuid. Sets initial TSC value, normally 0.

7. PlatformName

Type: plist string

Failsafe: Empty (Not installed)

Description: Sets name in gEfiMiscSubClassGuid. The value found on Macs is platform in ASCII.

8. SmcBranch

Type: plist data, 8 bytes Failsafe: Empty (Not installed)

 $\textbf{Description} : \textbf{Sets} \ \mathtt{RBr} \ \textbf{in} \ \mathtt{gEfiMiscSubClassGuid}. \ \textbf{Custom} \ \textbf{property} \ \textbf{read} \ \textbf{by} \ \mathtt{VirtualSMC} \ \textbf{or} \ \mathtt{FakeSMC} \ \textbf{to} \ \textbf{generate}$

SMC RBr key.

9. SmcPlatform

Type: plist data, 8 bytes Failsafe: Empty (Not installed)

Description: Sets RPlt in gEfiMiscSubClassGuid. Custom property read by VirtualSMC or FakeSMC to

generate SMC RPlt key.

10. SmcRevision

Type: plist data, 6 bytes Failsafe: Empty (Not installed)

Description: Sets REV in gEfiMiscSubClassGuid. Custom property read by VirtualSMC or FakeSMC to generate

SMC REV key.

11. StartupPowerEvents

 $\mathbf{Type}: \ \mathtt{plist} \ \mathtt{integer}, \ 64 ext{-bit}$

Failsafe: 0

Description: Sets StartupPowerEvents in gEfiMiscSubClassGuid. The 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_PMCON_2 bit 0)
- 0x00000002 Shutdown cause was a SYS_PWROK event (Same as GEN_PMCON_2 bit 1)
- 0x00000004 Shutdown cause was a THRMTRIP# event (Same as GEN_PMCON_2 bit 3)
- 0x00000008 Rebooted due to a SYS RESET# event (Same as GEN_PMCON_2 bit 4)
- 0x00000010 Power Failure (Same as GEN_PMCON_3 bit 1 PWR_FLR)
- 0x00000020 Loss of RTC Well Power (Same as GEN_PMCON_3 bit 2 RTC_PWR_STS)
- 0x00000040 General Reset Status (Same as GEN PMCON 3 bit 9 GEN RST STS)
- Oxffffff80 SUS Well Power Loss (Same as GEN_PMCON_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 PowerManagement Watchdog Timer event (Same as PRSTS bit 15)

12. SystemProductName

Type: plist string

Failsafe: Empty (Not installed)

Description: Sets Model in gEfiMiscSubClassGuid. The value found on Macs is equal to SMBIOS SystemProductName in Unicode.

13. SystemSerialNumber

Type: plist string

Failsafe: Empty (Not installed)

Description: Sets SystemSerialNumber in gEfiMiscSubClassGuid. The value found on Macs is equal to SMBIOS SystemSerialNumber in Unicode.

14. SystemUUID

Type: plist string, GUID Failsafe: Empty (Not installed)

Description: Sets system-id in gEfiMiscSubClassGuid. The value found on Macs is equal to SMBIOS

SystemUUID (with swapped byte order).

10.3 Generic Properties

1. AdviseFeatures

 $\mathbf{Type}:$ plist boolean

Failsafe: false

Description: Updates FirmwareFeatures with supported bits.

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 an EFI partition that is 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 an EFI partition that is the first partition on the disk.
- FW_FEATURE_SUPPORTS_APFS (0x00080000) Without this bit, it is not possible to install macOS on an APFS disk.
- FW_FEATURE_SUPPORTS_LARGE_BASESYSTEM (0x800000000) Without this bit, it is not possible to install macOS versions with large BaseSystem images, such as macOS 12.

Note: On most newer firmwares these bits are already set, the option may be necessary when "upgrading" the firmware with new features.

2. MLB

Type: plist string

Failsafe: Empty (OEM specified or not installed)

Description: Refer to SMBIOS BoardSerialNumber.

Specify special string value OEM to extract current value from NVRAM (MLB variable) or SMBIOS and use it throughout the sections. This feature can only be used on Mac-compatible firmware.

3. MaxBIOSVersion

Type: plist boolean Failsafe: false

Description: Sets BIOSVersion to 9999.999.999.999, recommended for legacy Macs when using Automatic PlatformInfo, to avoid BIOS updates in unofficially supported macOS versions.

$4. \ {\tt ProcessorType}$

Type: plist integer
Failsafe: 0 (Automatic)

Description: Refer to SMBIOS ProcessorType.

5. R.O.M

Type: plist multidata, 6 bytes

Failsafe: Empty (OEM specified or not installed)

Description: Refer to 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ROM.

Specify special string value <code>OEM</code> to extract current value from NVRAM (ROM variable) and use it throughout the sections. This feature can only be used on Mac-compatible firmware.

6. SpoofVendor

 $\mathbf{Type}:$ plist boolean

Failsafe: false

Description: Sets SMBIOS vendor fields to Acidanthera.

It can be dangerous to use "Apple" in SMBIOS vendor fields for reasons outlined in the SystemManufacturer description. However, certain firmware may not provide valid values otherwise, which could obstruct the operation of some software.

7. SystemMemoryStatus

Type: plist string

Failsafe: Auto

Description: Indicates whether system memory is upgradable in PlatformFeature. This controls the visibility of the Memory tab in "About This Mac".

Valid values:

- Auto use the original PlatformFeature value.
- Upgradable explicitly unset PT_FEATURE_HAS_SOLDERED_SYSTEM_MEMORY (0x2) in PlatformFeature.
- $\bullet \ \, {\tt Soldered-explicitly \ set \ PT_FEATURE_HAS_SOLDERED_SYSTEM_MEMORY \ (0x2) \ in \ PlatformFeature}.$

Note: On certain Mac models, such as the MacBookPro10,x and any MacBookAir, SPMemoryReporter.spreporter will ignore PT_FEATURE_HAS_SOLDERED_SYSTEM_MEMORY and assume that system memory is non-upgradable.

$8. \; {\tt SystemProductName}$

Type: plist string

Failsafe: Empty (OEM specified or not installed)

Description: Refer to SMBIOS SystemProductName.

$9. \; {\tt SystemSerialNumber}$

Type: plist string

Failsafe: Empty (OEM specified or not installed)

Description: Refer to SMBIOS SystemSerialNumber.

Specify special string value OEM to extract current value from NVRAM (SSN variable) or SMBIOS and use it throughout the sections. This feature can only be used on Mac-compatible firmware.

10. SystemUUID

Type: plist string, GUID

Failsafe: Empty (OEM specified or not installed)
Description: Refer to SMBIOS SystemUUID.

Specify special string value OEM to extract current value from NVRAM (system-id variable) or SMBIOS and use it throughout the sections. Since not every firmware implementation has valid (and unique) values, this feature is not applicable to some setups, and may provide unexpected results. It is highly recommended to specify the UUID explicitly. Refer to UseRawUuidEncoding to determine how SMBIOS value is parsed.

10.4 Memory Properties

1. DataWidth

Type: plist integer, 16-bit Failsafe: OxFFFF (unknown)

SMBIOS: Memory Device (Type 17) — Data Width

Description: Specifies the data width, in bits, of the memory. A DataWidth of 0 and a TotalWidth of 8 indicates that the device is being used solely to provide 8 error-correction bits.

2. Devices

Type: plist array Failsafe: Empty

Description: Specifies the custom memory devices to be added.

To be filled with plist dictionary values, describing each memory device. Refer to the Memory Devices Properties section below. This should include all memory slots, even if unpopulated.

3. ErrorCorrection

Type: plist integer, 8-bit

Failsafe: 0x03

SMBIOS: Physical Memory Array (Type 16) — Memory Error Correction

Description: Specifies the primary hardware error correction or detection method supported by the memory.

- 0x01 Other
- 0x02 Unknown
- 0x03 None
- 0x04 Parity
- 0x05 Single-bit ECC
- 0x06 Multi-bit ECC
- 0x07 CRC

4. FormFactor

Type: plist integer, 8-bit

Failsafe: 0x02

SMBIOS: Memory Device (Type 17) — Form Factor

Description: Specifies the form factor of the memory. On Macs, this should typically be DIMM or SODIMM. Commonly used form factors are listed below.

When CustomMemory is false, this value is automatically set based on Mac product name.

When Automatic is true, the original value from the the corresponding Mac model will be set if available. Otherwise, the value from OcMacInfoLib will be set. When Automatic is false, a user-specified value will be set if available. Otherwise, the original value from the firmware will be set. If no value is provided, the failsafe value will be set.

- 0x01 Other
- 0x02 Unknown
- 0x09 DIMM
- 0x0D SODIMM
- 0x0F FB-DIMM
- 5. MaxCapacity

Type: plist integer, 64-bit

Failsafe: 0

SMBIOS: Physical Memory Array (Type 16) — Maximum Capacity

Description: Specifies the maximum amount of memory, in bytes, supported by the system.

6. TotalWidth

Type: plist integer, 16-bit Failsafe: OxFFFF (unknown)

SMBIOS: Memory Device (Type 17) — Total Width

Description: Specifies the total width, in bits, of the memory, including any check or error-correction bits. If there are no error-correction bits, this value should be equal to DataWidth.

7. Type

Type: plist integer, 8-bit

Failsafe: 0x02

SMBIOS: Memory Device (Type 17) — Memory Type

Description: Specifies the memory type. Commonly used types are listed below.

- 0x01 Other
- 0x02 Unknown
- 0x0F SDRAM
- 0x12 DDR
- 0x13 DDR2
- 0x14 DDR2 FB-DIMM
- 0x18 DDR3
- 0x1A DDR4
- 0x1B LPDDR
- 0x1C LPDDR2
- 0x1D LPDDR3
- 0x1E LPDDR4
- 8. TypeDetail

Type: plist integer, 16-bit

Failsafe: 0x4

SMBIOS: Memory Device (Type 17) — Type Detail

Description: Specifies additional memory type information.

- Bit 0 Reserved, set to 0
- Bit 1 Other
- Bit 2 Unknown
- Bit 7 Synchronous
- Bit 13 Registered (buffered)
- Bit 14 Unbuffered (unregistered)

10.4.1 Memory Device Properties

1. AssetTag

Type: plist string Failsafe: Unknown

SMBIOS: Memory Device (Type 17) — Asset Tag

Description: Specifies the asset tag of this memory device.

2. BankLocator

Type: plist string Failsafe: Unknown

SMBIOS: Memory Device (Type 17) — Bank Locator

Description: Specifies the physically labeled bank where the memory device is located.

$3. \ {\tt DeviceLocator}$

Type: plist string Failsafe: Unknown

SMBIOS: Memory Device (Type 17) — Device Locator

Description: Specifies the physically-labeled socket or board position where the memory device is located.

$4.\ {\tt Manufacturer}$

Type: plist string Failsafe: Unknown

SMBIOS: Memory Device (Type 17) — Manufacturer

Description: Specifies the manufacturer of this memory device.

For empty slot this must be set to NO DIMM for macOS System Profiler to correctly display memory slots on certain Mac models, e.g. MacPro7,1 imposes additional requirements on the memory layout:

- The amount of installed sticks must one of the following: 4, 6, 8, 10, 12. Using any different value will cause an error in the System Profiler.
- The amount of memory slots must equal to 12. Using any different value will cause an error in the System Profiler.
- Memory sticks must be installed in dedicated memory slots as explained on the support page. SMBIOS memory devices are mapped to the following slots: 8, 7, 10, 9, 12, 11, 5, 6, 3, 4, 1, 2.

5. PartNumber

Type: plist string Failsafe: Unknown

SMBIOS: Memory Device (Type 17) — Part Number

Description: Specifies the part number of this memory device.

6. SerialNumber

Type: plist string Failsafe: Unknown

SMBIOS: Memory Device (Type 17) — Serial Number

Description: Specifies the serial number of this memory device.

7. Size

Type: plist integer, 32-bit

Failsafe: 0

SMBIOS: Memory Device (Type 17) — Size

Description: Specifies the size of the memory device, in megabytes. 0 indicates this slot is not populated.

8. Speed

Type: plist integer, 16-bit

Failsafe: 0

SMBIOS: Memory Device (Type 17) — Speed

indicates an unknown speed.

10.5 PlatformNVRAM Properties

1. BID

Type: plist string

Failsafe: Empty (Not installed)

Description: Specifies the value of NVRAM variable 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_BID.

2. FirmwareFeatures

Type: plist data, 8 bytes Failsafe: Empty (Not installed)

Description: This variable comes in pair with FirmwareFeaturesMask. Specifies the values of NVRAM variables:

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeatures

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeatures

3. FirmwareFeaturesMask

Type: plist data, 8 bytes Failsafe: Empty (Not installed)

Description: This variable comes in pair with FirmwareFeatures. Specifies the values of NVRAM variables:

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeaturesMask

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeaturesMask

4. MLB

Type: plist string

Failsafe: Empty (Not installed)

 $\textbf{Description} : \textbf{Specifies the values of NVRAM variables 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_MLB and a substitution of the value of NVRAM variables and the value of NVRAM variables and the value of NVRAM variables are substituted as a substitution of NVRAM variables and the value of NVRAM variables are substituted as a substitution of NVRAM variables and the value of NVRAM variables are substituted as a substitution of NVRAM variables and the value of NVRAM variables are substituted as a substitution of NVRAM variables are substituted as a substitute of$

4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:MLB.

5. ROM

Type: plist data, 6 bytes Failsafe: Empty (Not installed)

Description: Specifies the values of NVRAM variables 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_ROM and 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ROM.

4D1EDE03-30C7-4A0A-9CC0-4DCCA0D30C

6. SystemSerialNumber

Type: plist string

Failsafe: Empty (Not installed)

Description: Specifies the values of NVRAM variables 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_SSN and

4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:SSN.

7. SystemUUID

Type: plist string

Failsafe: Empty (Not installed)

Description: Specifies the value of NVRAM variable 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:system-id for boot services only. The value found on Macs is equal to SMBIOS SystemUUID.

10.6 SMBIOS Properties

1. BIOSReleaseDate

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: BIOS Information (Type 0) — BIOS Release Date

Description: Firmware release date. Similar to BIOSVersion. May look like 12/08/2017.

2. BIOSVendor

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: BIOS Information (Type 0) — Vendor

Description: BIOS Vendor. All rules of SystemManufacturer do apply.

$3. \; {\tt BIOSVersion}$

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: BIOS Information (Type 0) — BIOS Version

Description: Firmware version. This value gets updated and takes part in update delivery configuration and macOS version compatibility. This value could look like MM71.88Z.0234.B00.1809171422 in older firmware and is described in BiosId.h. In newer firmware, it should look like 236.0.0.0.0 or 220.230.16.0.0 (iBridge: 16.16.2542.0.0,0). iBridge version is read from BridgeOSVersion variable, and is only present on macs with T2.

Apple ROM Version

BIOS ID: MBP151.88Z.F000.B00.1811142212

Model: MBP151
EFI Version: 220.230.16.0.0
Built by: root@quinoa

Date: Wed Nov 14 22:12:53 2018

Revision: 220.230.16 (B&I)

ROM Version: F000_B00

Build Type: Official Build, RELEASE

Compiler: Apple LLVM version 10.0.0 (clang-1000.2.42)

UUID: E5D1475B-29FF-32BA-8552-682622BA42E1 UUID: 151B0907-10F9-3271-87CD-4BF5DBECACF5

4. BoardAssetTag

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Baseboard (or Module) Information (Type 2) — Asset Tag

Description: Asset tag number. Varies, may be empty or Type2 - Board Asset Tag.

$5. \ {\tt BoardLocationInChassis}$

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Baseboard (or Module) Information (Type 2) — Location in Chassis

Description: Varies, may be empty or Part Component.

6. BoardManufacturer

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Baseboard (or Module) Information (Type 2) - Manufacturer

Description: Board manufacturer. All rules of SystemManufacturer do apply.

7. BoardProduct

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Baseboard (or Module) Information (Type 2) - Product

Description: Mac Board ID (board-id). May look like Mac-7BA5B2D9E42DDD94 or Mac-F221BEC8 in older models.

8. BoardSerialNumber

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Baseboard (or Module) Information (Type 2) — Serial Number

Description: Board serial number in defined format. Known formats are described in macserial.

9. BoardType

Type: plist integer Failsafe: 0 (OEM specified)

SMBIOS: Baseboard (or Module) Information (Type 2) — Board Type

Description: Either 0xA (Motherboard (includes processor, memory, and I/O) or 0xB (Processor/Memory

Module). Refer to Table 15 – Baseboard: Board Type for details.

10. BoardVersion

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Baseboard (or Module) Information (Type 2) - Version

Description: Board version number. Varies, may match SystemProductName or SystemProductVersion.

11. ChassisAssetTag

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Enclosure or Chassis (Type 3) — Asset Tag Number

Description: Chassis type name. Varies, could be empty or MacBook-Aluminum.

12. ChassisManufacturer

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Enclosure or Chassis (Type 3) — Manufacturer

Description: Board manufacturer. All rules of SystemManufacturer do apply.

13. ChassisSerialNumber

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Enclosure or Chassis (Type 3) — Version

Description: Should match SystemSerialNumber.

14. ChassisType

Type: plist integer Failsafe: 0 (OEM specified)

SMBIOS: System Enclosure or Chassis (Type 3) — Type

Description: Chassis type. Refer to Table 17 — System Enclosure or Chassis Types for details.

15. ChassisVersion

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Enclosure or Chassis (Type 3) — Version

Description: Should match BoardProduct.

16. FirmwareFeatures

Type: plist data, 8 bytes

Failsafe: 0 (OEM specified on Apple hardware, 0 otherwise)

 ${\bf SMBIOS: APPLE_SMBIOS_TABLE_TYPE128 - Firmware Features} \ {\bf and} \ {\bf ExtendedFirmware Features}$

Description: 64-bit firmware features bitmask. Refer to AppleFeatures.h for details. Lower 32 bits match

FirmwareFeatures. Upper 64 bits match ExtendedFirmwareFeatures.

17. FirmwareFeaturesMask

Type: plist data, 8 bytes

Failsafe: 0 (OEM specified on Apple hardware, 0 otherwise)

SMBIOS: APPLE_SMBIOS_TABLE_TYPE128 - FirmwareFeaturesMask and ExtendedFirmwareFeaturesMask Description: Supported bits of extended firmware features bitmask. Refer to AppleFeatures.h for details. Lower

32 bits match FirmwareFeaturesMask. Upper 64 bits match ExtendedFirmwareFeaturesMask.

18. PlatformFeature

Type: plist integer, 32-bit

Failsafe: 0xFFFFFFF (OEM specified on Apple hardware, do not provide the table otherwise)

SMBIOS: APPLE_SMBIOS_TABLE_TYPE133 - PlatformFeature

Description: Platform features bitmask (Missing on older Macs). Refer to AppleFeatures.h for details.

19. ProcessorType

Type: plist integer, 16-bit Failsafe: 0 (Automatic)

SMBIOS: APPLE_SMBIOS_TABLE_TYPE131 - ProcessorType **Description**: Combined of Processor Major and Minor types.

Automatic value generation attempts to provide the most accurate value for the currently installed CPU. When this fails, please raise 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 the Apple SMBIOS definitions header here.

20. SmcVersion

Type: plist data, 16 bytes

Failsafe: All zero (OEM specified on Apple hardware, do not provide the table otherwise)

SMBIOS: APPLE_SMBIOS_TABLE_TYPE134 - Version

Description: ASCII string containing SMC version in upper case. Missing on T2 based Macs.

21. SystemFamily

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Information (Type 1) — Family **Description**: Family name. May look like iMac Pro.

22. SystemManufacturer

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Information (Type 1) — Manufacturer

Description: OEM manufacturer of the particular board. Use failsafe unless strictly required. Do not override to contain Apple Inc. on non-Apple hardware, as this confuses numerous services present in the operating system, such as firmware updates, eficheck, as well as kernel extensions developed in Acidanthera, such as Lilu and its plugins. In addition it will also make some operating systems such as Linux unbootable.

23. SystemProductName

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Information (Type 1), Product Name

Description: Preferred Mac model used to mark the device as supported by the operating system. This value must be specified by any configuration for later automatic generation of the related values in this and other SMBIOS tables and related configuration parameters. If SystemProductName is not compatible with the target operating system, -no_compat_check boot argument may be used as an override.

Note: If SystemProductName is unknown, and related fields are unspecified, default values should be assumed as being set to MacPro6,1 data. The list of known products can be found in AppleModels.

24. SystemSKUNumber

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Information (Type 1) — SKU Number

Description: Mac Board ID (board-id). May look like Mac-7BA5B2D9E42DDD94 or Mac-F221BEC8 in older models. Sometimes it can be just empty.

25. SystemSerialNumber

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Information (Type 1) — Serial Number

Description: Product serial number in defined format. Known formats are described in macserial.

26. SystemUUID

Type: plist string, GUID

Failsafe: Empty (OEM specified)

SMBIOS: System Information (Type 1) — UUID

Description: A UUID is an identifier that is designed to be unique across both time and space. It requires no central registration process.

$27. \; {\tt SystemVersion}$

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Information (Type 1) — Version

Description: Product iteration version number. May look like 1.1.

11 UEFI

11.1 Introduction

UEFI (Unified Extensible Firmware Interface) is a specification that defines a software interface between an operating system and platform firmware. This section allows loading additional UEFI modules as well as applying tweaks to the onboard firmware. To inspect firmware contents, apply modifications and perform upgrades UEFITool and supplementary utilities can be used.

11.2 Drivers

OpenRuntime*
OpenLegacyBoot*

Depending on the firmware, a different set of drivers may be required. Loading an incompatible driver may lead the system to unbootable state or even cause permanent firmware damage. Some of the known drivers are listed below:

AudioDxe*	HDA audio support driver in UEFI firmware for most Intel and some other analog audio controllers. Staging driver, refer to acidanthera/bugtracker#740 for known issues in AudioDxe.
btrfs_x64	Open source BTRFS file system driver, required for booting with OpenLinuxBoot from a file system which is now quite commonly used with Linux.
BiosVideo*	CSM video driver implementing graphics output protocol based on VESA and legacy BIOS interfaces. Used for UEFI firmware with fragile GOP support (e.g. low resolution). Requires ReconnectGraphicsOnConnect. Included in OpenDuet out of the box.
CrScreenshotDxe*	Screenshot making driver saving images to the root of OpenCore partition (ESP) or any available writeable filesystem upon pressing F10. Accepts optional driver argumentenable-mouse-click to additionally take screenshot on mouse click. (It is recommended to enable this option only if a keypress would prevent a specific screenshot, and disable it again after use.) This is a modified version of CrScreenshotDxe driver by Nikolaj Schlej.
<pre>EnableGop{Direct}*</pre>	Early beta release firmware-embeddable driver providing pre-OpenCore non-native GPU support on MacPro5,1. Installation instructions can be found in the Utilities/EnableGop directory of the OpenCore release zip file - proceed with caution.
ExFatDxe	Proprietary ExFAT file system driver for Bootcamp support commonly found in Apple firmware. For Sandy Bridge and earlier CPUs, the ExFatDxeLegacy driver should be used due to the lack of RDRAND instruction support.
ext4_x64	Open source EXT4 file system driver, required for booting with OpenLinuxBoot from the file system most commonly used with Linux.
FirmwareSettings*	OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL to add an entry to the boot picker menu which reboots into UEFI firmware settings, when this is supported by the firmware.
HfsPlus	Recommended. Proprietary HFS file system driver with bless support commonly found in Apple firmware. For Sandy Bridge and earlier CPUs, the HfsPlusLegacy driver should be used due to the lack of RDRAND instruction support.
HiiDatabase*	HII services support driver from MdeModulePkg. This driver is included in most types of firmware starting with the Ivy Bridge generation. Some applications with GUI, such as UEFI Shell, may need this driver to work properly.
EnhancedFatDxe	FAT filesystem driver from FatPkg. This driver is embedded in all UEFI firmware and cannot be used from OpenCore. Several types of firmware have defective FAT support implementation that may lead to corrupted filesystems on write attempts. Embedding this driver within the firmware may be required in case writing to the EFI partition is needed during the boot process.
NvmExpressDxe*	NVMe support driver from MdeModulePkg. This driver is included in most firmware starting with the Broadwell generation. For Haswell and earlier, embedding it within the firmware may be more favourable in case a NVMe SSD drive is installed.
OpenCanopy*	OpenCore plugin implementing graphical interface.

CSM.

OpenCore plugin implementing OC_FIRMWARE_RUNTIME protocol.

OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL to allow detection and booting of legacy operating systems from OpenCore on Macs, OpenDuet and systems with a

OpenLinuxBoot* OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL to allow direct detection and

booting of Linux distributions from OpenCore, without chainloading via GRUB.

OpenNetworkBoot* OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL to show available PXE and

HTTP(S) boot options on the OpenCore boot menu.

OpenNtfsDxe* New Technologies File System (NTFS) read-only driver. NTFS is the primary file system

for Microsoft Windows versions that are based on Windows NT.

OpenUsbKbDxe* USB keyboard driver adding support for AppleKeyMapAggregator protocols on top

of a custom USB keyboard driver implementation. This is an alternative to builtin

KeySupport, which may work better or worse depending on the firmware.

OpenPartitionDxe* Partition management driver with Apple Partitioning Scheme support. This driver

can be used to support loading older DMG recoveries such as OS X 10.9 using Apple Partitioning Scheme, or for loading other macOS Installers where these were created using the Apple Partitioning Scheme (creating macOS Installers using GPT avoids the

need for this). OpenDuet already includes this driver.

OpenVariableRuntimeDxe* OpenCore plugin offering emulated NVRAM support. OpenDuet already includes this

driver.

Ps2KeyboardDxe* PS/2 keyboard driver from MdeModulePkg. OpenDuetPkg and some types of firmware

may not include this driver, but it is necessary for PS/2 keyboard to work. Note, unlike OpenUsbKbDxe this driver has no AppleKeyMapAggregator support and thus requires

KeySupport to be enabled.

Ps2MouseDxe* PS/2 mouse driver from MdeModulePkg. Some very old laptop firmware may not include

this driver but it is necessary for the touchpad to work in UEFI graphical interfaces

such as OpenCanopy.

OpenHfsPlus* HFS file system driver with bless support. This driver is an alternative to a closed source

HfsPlus driver commonly found in Apple firmware. While it is feature complete, it is

approximately 3 times slower and is yet to undergo a security audit.

ResetNvramEntry* OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL to add a configurable Reset

NVRAM entry to the boot picker menu.

ToggleSipEntry* OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL to add a configurable Toggle

SIP entry to the boot picker menu.

USB mouse driver from MdeModulePkg. Some virtual machine firmware such as OVMF

may not include this driver but it is necessary for the mouse to work in UEFI graphical

interfaces such as OpenCanopy.

XhciDxe* XHCI USB controller support driver from MdeModulePkg. This driver is included in

most types of firmware starting with the Sandy Bridge generation. For earlier firmware

or legacy systems, it may be used to support external USB 3.0 PCI cards.

Driver marked with * are bundled with OpenCore. To compile the drivers from UDK (EDK II) the same command used for OpenCore compilation can be taken, but choose a corresponding package:

git clone https://github.com/acidanthera/audk 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

11.3 Tools and Applications

Standalone tools may help to debug firmware and hardware. Some of the known tools are listed below. While some tools can be launched from within OpenCore (Refer to the Tools subsection for more details), most should be run separately either directly or from Shell.

To boot into OpenShell or any other tool directly save OpenShell.efi under the name of EFI\BOOT\BOOTX64.EFI on a FAT32 partition. It is typically unimportant whether the partition scheme is GPT or MBR.

While the previous approach works both on Macs and other computers, an alternative Mac-only approach to bless the tool on an HFS+ or APFS volume:

```
sudo bless --verbose --file /Volumes/VOLNAME/DIR/OpenShell.efi \
    --folder /Volumes/VOLNAME/DIR/ --setBoot
```

Listing 3: Blessing tool

Note 1: /System/Library/CoreServices/BridgeVersion.bin should be copied to /Volumes/VOLNAME/DIR.

Note 2: To be able to use the bless command, disabling System Integrity Protection is necessary.

Note 3: To be able to boot Secure Boot might be disabled if present.

Some of the known tools are listed below (builtin tools are marked with *):

BootKicker* Display Apple BootPicker menu (for Macs with compatible firmware).

ChipTune* Test BeepGen protocol and generate audio signals of different style and length.

CleanNvram* Reset NVRAM alternative bundled as a standalone tool.

CsrUtil* Simple implementation of SIP-related features of Apple csrutil.

FontTester* Render the console font pages which the Builtin renderer provides.

GopStop* Test GraphicsOutput protocol with a simple scenario.

KeyTester* Test keyboard input in SimpleText mode.

Memory testing utility.

OpenControl* Unlock and lock back NVRAM protection for other tools to be able to get full NVRAM

access when launching from OpenCore.

OpenShell* OpenCore-configured UEFI Shell for compatibility with a broad range of firmware.

PaypProvision Perform EPID provisioning (requires certificate data configuration).

ResetSystem* Utility to perform system reset. Takes reset type as an argument: coldreset, firmware,

shutdown, warmreset. Defaults to coldreset.

RtcRw* Utility to read and write RTC (CMOS) memory.

ControlMsrE2* Check CFG Lock (MSR 0xE2 write protection) consistency across all cores and change such

hidden options on selected platforms.

TpmInfo* Check Intel PTT (Platform Trust Technology) capability on the platform, which allows using

fTPM 2.0 if enabled. The tool does not check whether fTPM 2.0 is actually enabled.

11.4 OpenCanopy

OpenCanopy is a graphical OpenCore user interface that runs in External PickerMode and relies on OpenCorePkg OcBootManagementLib similar to the builtin text interface.

OpenCanopy requires graphical resources located in Resources directory to run. Sample resources (fonts and images) can be found in OcBinaryData repository. Customised icons can be found over the internet (e.g. here or there).

OpenCanopy provides full support for PickerAttributes and offers a configurable builtin icon set. The chosen icon set may depend on the DefaultBackgroundColor variable value. Refer to PickerVariant for more details.

Predefined icons are saved in the PickerVariant-derived subdirectory of the \EFI\OC\Resources\Image directory. A full list of supported icons (in .icns format) is provided below. When optional icons are missing, the closest available icon will be used. External entries will use Ext-prefixed icon if available (e.g. OldExtHardDrive.icns).

Note: In the following all dimensions are normative for the 1x scaling level and shall be scaled accordingly for other levels.

- Cursor Mouse cursor (mandatory, up to 144x144).
- Selected Selected item (mandatory, 144x144).
- Selector Selecting item (mandatory, up to 144x40).
- SetDefault Selecting default (mandatory, up to 144x40; must be same width as Selector).
- Left Scrolling left (mandatory, 40x40).
- Right Scrolling right (mandatory, 40x40).
- HardDrive Generic OS (mandatory, 128x128).
- Background Centred background image.
- Apple Apple OS (128x128).
- AppleRecv Apple Recovery OS (128x128).
- AppleTM Apple Time Machine (128x128).
- Windows Windows (128x128).

- Other Custom entry (see Entries, 128x128).
- ResetNVRAM Reset NVRAM system action or tool (128x128).
- Shell Entry with UEFI Shell name for e.g. OpenShell (128x128).
- Tool Any other tool (128x128).

Predefined labels are saved in the \EFI\OC\Resources\Label directory. Each label has .1b1 or .12x suffix to represent the scaling level. Full list of labels is provided below. All labels are mandatory.

- EFIBoot Generic OS.
- Apple Apple OS.
- AppleRecv Apple Recovery OS.
- AppleTM Apple Time Machine.
- Windows Windows.
- Other Custom entry (see Entries).
- ResetNVRAM Reset NVRAM system action or tool.
- SIPDisabled Toggle SIP tool with SIP disabled.
- SIPEnabled Toggle SIP tool with SIP enabled.
- Shell Entry with UEFI Shell name (e.g. OpenShell).
- FirmwareSettings Firmware settings menu entry.
- Tool Any other tool.

Note: All labels must have a height of exactly 12 px. There is no limit for their width.

Label and icon generation can be performed with bundled utilities: disklabel and icnspack. Font is Helvetica 12 pt times scale factor.

Font format corresponds to AngelCode binary BMF. While there are many utilities to generate font files, currently it is recommended to use dpFontBaker to generate bitmap font (using CoreText produces best results) and fonverter to export it to binary format.

11.5 OpenRuntime

OpenRuntime is an OpenCore plugin implementing OC_FIRMWARE_RUNTIME protocol. This protocol implements multiple features required for OpenCore that are otherwise not possible to implement in OpenCore itself as they are needed to work in runtime, i.e. during operating system functioning. Feature highlights:

- NVRAM namespaces, allowing to isolate operating systems from accessing select variables (e.g. RequestBootVarRouting or ProtectSecureBoot).
- Read-only and write-only NVRAM variables, enhancing the security of OpenCore, Lilu, and Lilu plugins, such as VirtualSMC, which implements AuthRestart support.
- NVRAM isolation, allowing to protect all variables from being written from an untrusted operating system (e.g. DisableVariableWrite).
- UEFI Runtime Services memory protection management to workaround read-only mapping (e.g. EnableWriteUnprotector).

11.6 OpenLegacyBoot

OpenLegacyBoot is an OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL. It aims to detect and boot legacy installed operating systems on supported systems, such as OpenDuet and Mac models capable of legacy booting.

Usage:

- Add OpenLegacyBoot.efi and also optionally (see below) OpenNtfsDxe.efi to the config.plist Drivers section.
- Install Windows or another legacy operating system as normal if this has not been done earlier OpenLegacyBoot is not involved in this stage and may be unable to boot from installation media such as a USB device.
- Reboot into OpenCore: the installed legacy operating system should appear and boot directly from OpenCore when selected.

OpenLegacyBoot does not require any additional filesystem drivers such as <code>OpenNtfsDxe.efi</code> to be loaded for base functionality, but loading them will enable the use of <code>.contentDetails</code> and <code>.VolumeIcon.icns</code> files for boot entry customisation.

11.6.1 Configuration

No additional configuration should work well in most circumstances, but if required the following options for the driver may be specified in UEFI/Drivers/Arguments:

• --hide-devices - String value, no default.

When this option is present and has one or more values separated by semicolons (e.g. --hide-devices=PciRoot(0x0)/Pci(0x1F,0x2)/Sata(0x0,0xFFFF,0x0)/HD(2,GPT,...)), it disables scanning the specified disks for legacy operating system boot sectors.

11.7 OpenLinuxBoot

OpenLinuxBoot is an OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL. It aims to automatically detect and boot most Linux distros without additional configuration.

Usage is as follows:

- Add OpenLinuxBoot.efi and also typically (see below) ext4_x64.efi to the config.plist Drivers section.
- Make sure RequestBootVarRouting and LauncherOption are enabled in config.plist; it is also recommended to enable HideAuxiliary in order to hide older Linux kernels except when required (they are added as auxiliary entries and so may then be shown by pressing the Spacebar key in the OpenCore boot menu).
- Install Linux as normal if this has not been done earlier OpenLinuxBoot is not involved in this stage.
- Reboot into OpenCore: the installed Linux distribution should just appear and boot directly from OpenCore when selected, which it does without chainloading via GRUB.

If OpenCore has already been manually set up to boot Linux, e.g. via BlessOverride or via Entries then then these settings may be removed so that the Linux distribution is not displayed twice in the boot menu.

It is recommended to install Linux with its default bootloader, even though this will not be actively used when booting via OpenLinuxBoot. This is because OpenLinuxBoot has to detect the correct kernel options to use, and does so by looking in files left by the default bootloader. If no bootloader was installed (or these options cannot be found) booting is still possible, but the correct boot options must be manually specified before OpenLinuxBoot will attempt to start the distro.

OpenLinuxBoot typically requires filesystem drivers that are not available in firmware, such as EXT4 and BTRFS drivers. These drivers can be obtained from external sources. Drivers tested in basic scenarios can be downloaded from OcBinaryData. Be aware that these drivers are not tested for reliability in all scenarios, nor did they undergo tamper-resistance testing, therefore they may carry potential security or data-loss risks.

Most Linux distros require the ext4_x64 driver, a few may require the btrfs_x64 driver, and a few may require no additional file system driver: it depends on the filesystem of the boot partition of the installed distro, and on what filesystems are already supported by the system's firmware. LVM is not currently supported - this is because it is not believed that there is currently a stand-alone UEFI LVM filesystem driver.

Be aware of the SyncRuntimePermissions quirk, which may need to be set to avoid early boot failure (typically halting with a black screen) of the Linux kernel, due to a firmware bug of some firmware released after 2017. When present and not mitigated by this quirk, this affects booting via OpenCore with or without OpenLinuxBoot.

After installing OpenLinuxBoot, it is recommended to compare the options shown in the OpenCore debug log when booting (or attempting to boot) a given distro against the options seen using the shell command cat /proc/cmdline when the same distro has been booted via its native bootloader. In general (for safety and security of the running distro) these options should match, and if they do not it is recommended to use the driver arguments below (in particular LINUX_BOOT_ADD_RO, LINUX_BOOT_ADD_RW, autoopts:{PARTUUID} and autoopts) to modify the options as required. Note however that the following differences are normal and do not need to be fixed:

- If the default bootloader is GRUB then the options generated by OpenLinuxBoot will not contain a BOOT_IMAGE=... value where the GRUB options do, and will contain an initrd=... value where the GRUB options do not.
- OpenLinuxBoot uses PARTUUID rather than filesystem UUID to identify the location of initrd, this is by design as UEFI filesystem drivers do not make Linux filesystem UUID values available.
- Less important graphics handover options (such as discussed in the Ubuntu example given in autoopts below) will not match exactly, this is not important as long as distro boots successfully.

If using OpenLinuxBoot with Secure Boot, users may wish to install a user built, user signed Shim bootloader giving SBAT and MOK integration, as explained in OpenCore ShimUtils.

11.7.1 Configuration

The default parameter values should work well with no changes under most circumstances, but if required the following options for the driver may be specified in UEFI/Drivers/Arguments:

- flags Default: all flags are set except the following:
 - LINUX_BOOT_ADD_RW,
 - LINUX BOOT LOG VERBOSE,
 - LINUX_BOOT_LOG_GRUB_VARS and
 - LINUX_BOOT_ADD_DEBUG_INFO.

Available flags are:

- 0x00000001 (bit 0) LINUX_BOOT_SCAN_ESP, Allows scanning for entries on EFI System Partition.
- 0x00000002 (bit 1) LINUX_BOOT_SCAN_XBOOTLDR, Allows scanning for entries on Extended Boot Loader Partition.
- 0x00000004 (bit 2) LINUX_BOOT_SCAN_LINUX_ROOT, Allows scanning for entries on Linux Root filesystems.
- 0x00000008 (bit 3) LINUX_BOOT_SCAN_LINUX_DATA, Allows scanning for entries on Linux Data filesystems.
- 0x00000080 (bit 7) LINUX_BOOT_SCAN_OTHER, Allows scanning for entries on file systems not matched by any of the above.

The following notes apply to all of the above options:

- Note 1: Apple filesystems APFS and HFS are never scanned.
- Note 2: Regardless of the above flags, a file system must first be allowed by Misc/Security/ScanPolicy before it can be seen by OpenLinuxBoot or any other OC_BOOT_ENTRY_PROTOCOL driver.
- Note 3: It is recommended to enable scanning LINUX_ROOT and LINUX_DATA in both OpenLinuxBoot flags and Misc/Security/ScanPolicy in order to be sure to detect all valid Linux installs, since Linux boot filesystems are very often marked as LINUX_DATA.
- 0x00000100 (bit 8) LINUX_BOOT_ALLOW_AUTODETECT, If set allows autodetecting and linking vmlinuz* and init* ramdisk files when loader/entries files are not found.
- 0x00000200 (bit 9) LINUX_BOOT_USE_LATEST, When a Linux entry generated by OpenLinuxBoot is selected as the default boot entry in OpenCore, automatically switch to the latest kernel when a new version is installed.

When this option is set, an internal menu entry id is shared between kernel versions from the same install of Linux. Linux boot options are always sorted highest kernel version first, so this means that the latest kernel version of the same install always shows as the default, with this option set.

Note: This option is recommended on all systems.

- 0x00000400 (bit 10) LINUX_BOOT_ADD_RO, This option applies to autodetected Linux only (i.e. not to BLSpec or Fedora-style distributions which have /loader/entries/*.conf files). Some distributions run a filesystem check on loading which requires the root filesystem to initially be mounted read-only via the ro kernel option, which requires this option to be added to the autodetected options. Set this bit to add this option on autodetected distros; should be harmless but very slightly slow down boot time (due to required remount as read-write) on distros which do not require it. When there are multiple distros and it is required to specify this option for specific distros only, use autoopts:{PARTUUID}+=ro to manually add the option where required, instead of using this flag.
- 0x00000800 (bit 11) LINUX_BOOT_ADD_RW, Like LINUX_BOOT_ADD_RO, this option applies to autodetected Linux only. It is not required for most distros (which usually require either ro or nothing to be added to detected boot options), but is required on some Arch-derived distros, e.g. EndeavourOS. When there are multiple distros and it is required to specify this option for specific distros only, use autoopts:{PARTUUID}+=rw to manually add the option where required, instead of using this flag. If this option and LINUX_BOOT_ADD_RO are both specified, only this option is applied and LINUX_BOOT_ADD_RO is ignored.
- 0x00002000 (bit 13) LINUX_BOOT_ALLOW_CONF_AUTO_ROOT, In some instances of BootLoaderSpecByDefault in combination with ostree, the /loader/entries/*.conf files do not specify a required root=... kernel option it is added by GRUB. If this bit is set and this situation is detected, then automatically add this option. (Required for example by Endless OS.)

- 0x00004000 (bit 14) LINUX_BOOT_LOG_VERBOSE, Add additional debug log info about files encountered and autodetect options added while scanning for Linux boot entries.
- 0x00008000 (bit 15) LINUX_BOOT_ADD_DEBUG_INFO, Adds a human readable file system type, followed by the first eight characters of the partition's unique partition unid, to each generated entry name. Can help with debugging the origin of entries generated by the driver when there are multiple Linux installs on one system.
- 0x00010000 (bit 16) LINUX_BOOT_LOG_GRUB_VARS, When a BootLoaderSpecByDefault setup is detected, log available GRUB variables found in grub2/grubenv and grub2/grub.cfg.
- 0x00020000 (bit 17) LINUX_BOOT_FIX_TUNED, In some circumstances, such as after upgrades which add TuneD to existing systems, the TuneD system tuning plugin may add its GRUB variables to loader/entries/*.conf files but not initialise them in grub2/grub.cfg. In order to avoid incorrect boots, OpenLinuxBoot treats used, non-initialised GRUB variables as an error. When this flag is set, empty values are added for the TuneD variables tuned_params and tuned_initrd if they are not present. This is required for OpenLinuxBoot on TuneD systems with this problem, and harmless otherwise.

Flag values can be specified in hexadecimal beginning with 0x or in decimal, e.g. flags=0x80 or flags=128. It is also possible to specify flags to add or remove, using syntax such as flags+=0xC000 to add all debugging options or flags-=0x400 to remove the LINUX_BOOT_ADD_RO option.

• autoopts:{PARTUUID}[+]="{options}" - Default: not set.

Allows manually specifying kernel options to use in autodetect mode for a given partition only. Replace the text {PARTUUID} with the specific partition UUID on which the kernels are stored (in normal layout, the partition which contains /boot), e.g. autoopts:11223344-5566-7788-99aa-bbccddeeff00+="vt.handoff=7". If specified with += then these options are used in addition to any autodetected options, if specified with = they are used instead. Used for autodetected Linux only - values specified here are never used for entries created from /loader/entries/*.conf files.

Note: The PARTUUID value to be specified here is typically the same as the PARTUUID seen in root=PARTUUID=... in the Linux kernel boot options (view using cat /proc/cmdline). Alternatively, and for more advanced scenarios, it is possible to examine how the distro's partitions are mounted using the Linux mount command, and then find out the partuuid of relevant mounted partitions by examining the output of ls -l /dev/disk/by-partuuid.

• autoopts[+]="{options}" - Default: None specified.

Allows manually specifying kernel options to use in autodetect mode. The alternative format autoopts:{PARTUUID} is more suitable where there are multiple distros, but autoopts with no PARTUUID required may be more convenient for just one distro. If specified with += then these are used in addition to autodetected options, if specified with = they are used instead. Used for autodetected Linux only - values specified here are never used for entries created from /loader/entries/*.conf files.

As example usage, it is possible to use += format to add a vt.handoff options, such as autoopts+="vt.handoff=7" or autoopts+="vt.handoff=3" (check cat /proc/cmdline when booted via the distro's default bootloader) on Ubuntu and related distros, in order to add the vt.handoff option to the auto-detected GRUB defaults, and avoid a flash of text showing before the distro splash screen.

11.7.2 Additional information

OpenLinuxBoot can detect the loader/entries/*.conf files created according to the Boot Loader Specification or the closely related Fedora BootLoaderSpecByDefault. The former is specific to systemd-boot and is used by Arch Linux, the latter applies to most Fedora-related distros including Fedora itself, RHEL and variants.

Where the above files are not present, OpenLinuxBoot can autodetect and boot {boot}/vmlinuz* kernel files directly. It links these automatically – based on the kernel version in the filename – to their associated {boot}/init* ramdisk files. This applies to most Debian-related distros, including Debian itself, Ubuntu and variants.

When autodetecting in /boot as part of the root filesystem, OpenLinuxBoot looks in /etc/default/grub for kernel boot options and /etc/os-release for the distro name. When autodetecting in a standalone boot partition (i.e. when /boot has its own mount point), OpenLinuxBoot cannot autodetect kernel arguments and all kernel arguments except initrd=... must be fully specified by hand using autoopts=... or autoopts:{partuuid}=... (+= variants of these options will not work, as these only add additional arguments).

Fedora BootLoaderSpecByDefault (but not pure Boot Loader Specification) can expand GRUB variables in the *.conf files – and this is used in practice in certain distros such as CentOS. In order to handle this correctly, when this situation is detected OpenLinuxBoot extracts all variables from {boot}/grub2/grubenv and also any unconditionally set variables from {boot}/grub2/grub.cfg, and then expands these where required in *.conf file entries.

The only currently supported method of starting Linux kernels from OpenLinuxBoot relies on their being compiled with EFISTUB. This applies to almost all modern distros, particularly those which use systemd. Note that most modern distros use systemd as their system manager, even though most do not use systemd-boot as their bootloader.

systemd-boot users (probably almost exclusively Arch Linux users) should be aware that OpenLinuxBoot does not support the systemd-boot-specific Boot Loader Interface; therefore efibootmgr rather than bootctl must be used for any low-level Linux command line interaction with the boot menu.

11.8 OpenNetworkBoot

OpenNetworkBoot is an OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL. It enables PXE and HTTP(S) Boot options in the OpenCore menu if these are supported by the underlying firmware, or if the required network boot drivers have been loaded using OpenCore.

It has additional support for loading .dmg files and their associated .chunklist file over HTTP(S) Boot, allowing macOS recovery to be started over HTTP(S) Boot: if either extension is seen in the HTTP(S) Boot URI then the other file of the pair is automatically loaded as well, and both are passed to OpenCore to verify and boot from the DMG file.

PXE Boot is already supported on most firmware, so in most cases PXE Boot entries should appear as soon as the driver is loaded. Using the additional network boot drivers provided with OpenCore, when needed, HTTP(S) Boot should be available on most firmware even if not natively supported.

Detailed information about the available network boot drivers and how to configure PXE and HTTP(S) Boot is provided on this page.

The below configuration options may be specified in the Arguments section for this driver.

Note: There is no problem if configuration options within <arguments>...</arguments> are given on multiple lines, and option values enclosed within quotes can also span multiple lines. This applies to all drivers which use OpenCore argument parsing, and can be particularly convenient when multiple long options such as uri, static4 and particularly enroll-cert may be needed.

• aux - Boolean flag, enabled if present.

If specified the driver will generate auxiliary boot entries.

• delete-all-certs[:{OWNER_GUID}] - Default: not set.

If specified, delete all certificates present for OWNER_GUID. OWNER_GUID is optional, and will default to all zeros if not specified.

• delete-cert[:{OWNER GUID}]="{cert-text}" - Default: not set.

If specified, delete the given certificate(s) for HTTPS Boot. The certificate(s) can be specified as a multi-line PEM value between double quotes. A single PEM file can contain one or more certicates. Multiple instances of this option can be used to delete multiple different PEM files, if required. OWNER_GUID is optional, and will default to all zeros if not specified.

• enroll-cert[:{OWNER_GUID}]="{cert-text}" - Default: not set.

If specified, enroll the given certificate(s) for HTTPS Boot. The certificate(s) can be specified as a multi-line PEM value between double quotes. A single PEM file can contain one or more certicates. Multiple instances of this option can be used to enroll multiple different PEM files, if required. OWNER_GUID is optional, and will default to all zeros if not specified.

• http - Boolean flag, enabled if present.

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

• https - Boolean flag, enabled if present.

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

• ipv4 - Boolean flag, enabled if present.

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

• ipv6 - Boolean flag, enabled if present.

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

pxe - Boolean flag, enabled if present.

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

• static4:{MAC_ADDR}[\VLAN_ID][="{IP},{MASK},{GATEWAY}[,{DNS}]"] - String value.

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

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

MAC_ADDR is not optional.

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

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

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

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

Note 3: If Ip4Dxe is loaded before OpenCore then any setting here will only take effect after one reboot (during which network boot is not attempted). If Ip4Dxe is loaded after OpenNetworkBoot the setting will take effect immediately.

Note 4: In the majority of cases this option is not required, and the default DHCP behaviour should be preferred, since IP address conflicts are automatically avoided, and any IP address assigned by DHCP during network boot will normally automatically match the IP address assigned in-OS, as the same MAC address is used in both cases.

• uri - String value, no default.

If present, specify the URI to use for HTTP(S) Boot. If not present then DHCP boot options must be enabled on the network in order for HTTP(S) Boot to know what to boot.

11.8.1 OpenNetworkBoot Certificate Management

Certificates are enrolled to NVRAM storage, therefore once a certificate has been enrolled, it will remain enrolled even if the enroll-cert config option is removed. delete-cert or delete-all-certs should be used to remove enrolled certificates.

Checking for certificate presence by the enroll-cert and delete-cert options uses the simple algorithm of matching by exact file contents, not by file meaning. The intended usage is to leave an enroll-cert option present in the config file until it is time to delete it, e.g. after another more up-to-date enroll-cert option has been added and tested. At this point the user can change enroll-cert to delete-cert for the old certificate.

Certificate options are processed one at a time, in order, and each will potentially make changes to the certificate NVRAM storage. However each option will not change the NVRAM store if it is already correct for the option at that point in time (e.g. will not enroll a certificate if it is already enrolled). Avoid combinations such as delete-all-certs followed by enroll-cert, as this will modify the NVRAM certificate storage twice on every boot. However a combination such as delete-cert="{certA-text}" followed by enroll-cert="{certB-text}" (with certA-text and certB-text different) is safe, because certA will only be deleted if it is present and certB will only be added if it is not present, therefore no NVRAM changes will be made on the second and subsequent boots with these options.

In some cases (such as OVMF with https:// boot support) the OpenNetworkBoot certificate configuration options manage the same certificates as those seen in the firmware UI. In other cases of vendor customised HTTPS Boot firmware, the certificates managed by this driver will be separate from those managed by firmware.

When using the debug version of this driver, the OpenCore debug log includes NETB: entries that show which certificates are enrolled and removed by these options, and which certificates are present after all certificate configuration options have been processed.

11.9 Other Boot Entry Protocol drivers

In addition to the OpenLinuxBoot and OpenNetworkBoot plugins, the following OC_BOOT_ENTRY_PROTOCOL plugins are made available to add optional, configurable boot entries to the OpenCore boot picker.

11.9.1 ResetNvramEntry

Adds a menu entry which resets NVRAM and immediately restarts. Additionally adds support for hotkey CMD+0PT+P+R to perform the same action. Note that on some combinations of firmware and drivers, the TakeoffDelay option must be configured in order for this and other builtin hotkeys to be reliably detected.

Note 1: It is known that some Lenovo laptops have a firmware bug, which makes them unbootable after performing NVRAM reset. Refer to acidanthera/bugtracker#995 for details.

Note 2: If LauncherOption is set to Full or Short then the OpenCore boot entry is protected. Resetting NVRAM will normally erase any other boot options not specified via BlessOverride, for example Linux installations to custom locations and not using the OpenLinuxBoot driver, or user-specified UEFI boot menu entries. To obtain reset NVRAM functionality which does not remove other boot options, it is possible to use the --preserve-boot option (though see the warning specified).

The following configuration options may be specified in the Arguments section for this driver:

• --preserve-boot - Boolean flag, enabled if present.

If enabled, BIOS boot entries are not cleared during NVRAM reset. This option should be used with caution, as some boot problems can be fixed by clearing these entries.

• --apple - Boolean flag, enabled if present.

On Apple firmware only, this performs a system NVRAM reset. This can result in additional, desirable operations such as NVRAM garbage collection. This is achieved by setting the ResetNVRam NVRAM variable. Where available, this has the same effect as pressing CMD+0PT+P+R during native boot, although note that if accessed from the menu entry only one boot chime will be heard.

Note 1: Due to using system NVRAM reset, this option is not compatible with the --preserve-boot option and will override it, therefore all BIOS boot entries will be removed.

Note 2: Due to using system NVRAM reset, the OpenCore boot option cannot be preserved and OpenCore will have to either be reselected in the native boot picker or re-blessed.

Note 3: On non-Apple hardware, this option will still set this variable but the variable will not be recognised by the firmware and no NVRAM reset will happen.

11.9.2 ToggleSipEntry

Provides a boot entry for enabling and disabling System Integrity Protection (SIP) in OpenCore picker.

While macOS is running, SIP involves multiple configured software protection systems, however all the information about which of these protections to enable is stored in the single Apple NVRAM variable csr-active-config. As long as this variable is set before macOS startup, SIP will be fully configured, so setting the variable using this boot option (or in any other way, before macOS starts) has exactly the same end result as configuring SIP using the csrutil command in macOS Recovery.

csr-active-config will be toggled between 0 for enabled, and a user-specified or default value for disabled.

Options for the driver should be specified as plain text values separated by whitespace in the Arguments section of Driver entry. Available options are:

• --show-csr - Boolean flag, enabled if present.

If enabled, show the current hexadecimal value of csr-active-config in the boot entry name. This option will not work in OpenCanopy when used in combination with OC_ATTR_USE_GENERIC_LABEL_IMAGE in PickerAttributes.

• Numerical value - Default value 0x27F.

Specify the csr-active-config value to use to disabled SIP. This can be specified as hexadecimal, beginning with 0x, or as decimal. For more info see Note 2 below.

Note 1: It is recommended not to run macOS with SIP disabled. Use of this boot option may make it easier to quickly disable SIP protection when genuinely needed - it should be re-enabled again afterwards.

Note 2: The default value for disabling SIP with this boot entry is 0x27F. For comparison, csrutil disable with no other arguments on macOS Big Sur and Monterey sets 0x7F, and on Catalina it sets 0x77. The OpenCore default value of 0x27F is a variant of the Big Sur and Monterey value, chosen as follows:

- CSR_ALLOW_UNAPPROVED_KEXTS (0x200) is included in the default value, since it is generally useful, in the case where you need to have SIP disabled anyway, to be able to install unsigned kexts without manual approval in System Preferences.
- CSR_ALLOW_UNAUTHENTICATED_ROOT (0x800) is not included in the default value, as it is very easy when using it to inadvertently break OS seal and prevent incremental OTA updates.
- If unsupported bits from a later OS are specified in csr-active-config (e.g. specifying 0x7F on Catalina) then csrutil status will report that SIP has a non-standard value, however protection will be functionally the same.

11.9.3 FirmwareSettings

Adds a menu entry which will reboot into UEFI firmware settings, when supported. No menu entry added and logs a warning when not supported.

11.10 AudioDxe

High Definition Audio (HDA) support driver in UEFI firmware for most Intel and some other analog audio controllers. Note: AudioDxe is a staging driver, refer to acidanthera/bugtracker#740 for known issues.

11.10.1 Configuration

Most UEFI audio configuration is handled via the UEFI Audio Properties section, but in addition some of the following configuration options may be required in order to allow AudioDxe to correctly drive certain devices. All options are specified as text strings, separated by space if more than one option is required, in the Arguments property for the driver within the UEFI/Drivers section:

• --codec-setup-delay - Integer value, default 0.

Amount of time in milliseconds to wait for all widgets to come fully on, applied per codec during driver connection phase. In most systems this should not be needed and a faster boot will be achieved by using Audio section SetupDelay if any audio setup delay is required. Where required, values of up to one second may be needed.

• --force-codec - Integer value, no default.

Force use of an audio codec, this value should be equal to Audio section AudioCodec. Can result in faster boot especially when used in conjunction with --force-device.

• --force-device - String value, no default.

When this option is present and has a value (e.g. --force-device=PciRoot(0x0)/Pci(0x1f,0x3)), it forces AudioDxe to connect to the specified PCI device, even if the device does not report itself as an HDA audio controller.

During driver connection, AudioDxe automatically provides audio services on all supported codecs of all available HDA controllers. However, if the relevant controller is misreporting its identity (typically, it will be reporting itself as a legacy audio device instead of an HDA controller) then this argument may be required.

Applies if the audio device can be made to work in macOS, but shows no sign of being detected by AudioDxe (e.g. when including DEBUG_INFO in DisplayLevel and using a DEBUG build of AudioDxe, no controller and codec layout information is displayed during the Connecting drivers... phase of OpenCore log).

• --gpio-setup - Default value is 0 (GPIO setup disabled) if argument is not provided, or 7 (all GPIO setup stages stages enabled) if the argument is provided with no value.

Available values, which may be combined by adding, are:

- 0x00000001 (bit 0) GPIO_SETUP_STAGE_DATA, set GPIO pin data high on specified pins. Required e.g. on MacBookPro10,2 and MacPro5,1.
- 0x00000002 (bit 1) GPIO_SETUP_STAGE_DIRECTION, set GPIO data direction to output on specified pins. Required e.g. on MacPro5,1.
- 0x00000004 (bit 2) GPIO_SETUP_STAGE_ENABLE, enable specified GPIO pins. Required e.g. on MacPro5,1.

If audio appears to be 'playing' on the correct codec, e.g. based on the debug log, but no sound is heard on any channel, it is suggested to use --gpio-setup (with no value) in the AudioDxe driver arguments. If specified with no value, all stages will be enabled (equivalent of specifying 7). If this produces sound, it is then possible to try fewer bits, e.g. --gpio-setup=1, --gpio-setup=3, to find out which stages are actually required.

Note: Value 7 (all flags enabled) of this option – as required for the MacPro5,1 – is compatible with most systems, but is known to cause problems with sound (previous sounds are not allowed to finish before new sounds start) on a small number of other systems, hence this option is not enabled by default.

• --gpio-pins - Default: 0, auto-detect.

Specifies which GPIO pins should be operated on by --gpio-setup. This is a bit mask, with possible values from 0x0 to 0xFF. The usable maximum depends on the number if available pins on the audio out function group of the codec in use, e.g. it is 0x3 (lowest two bits) if two GPIO pins are present, 0x7 if three pins are present, etc.

When --gpio-setup is enabled (i.e. non-zero), then 0 is a special value for --gpio-pins, meaning that the pin mask will be auto-generated based on the reported number of GPIO pins on the specified codec (see AudioCodec), e.g. if the codec's audio out function group reports 4 GPIO pins, a mask of 0xF will be used. The value in use can be seen in the debug log in a line such as:

HDA: GPIO setup on pins 0x0F - Success

Values for driver parameters can be specified in hexadecimal beginning with 0x or in decimal, e.g. --gpio-pins=0x12 or --gpio-pins=18.

• --restore-nosnoop - Boolean flag, enabled if present.

AudioDxe clears the Intel HDA No Snoop Enable (NSNPEN) bit. On some systems, this change must be reversed on exit in order to avoid breaking sound in Windows or Linux. If so, this flag should be added to AudioDxe driver arguments. Not enabled by default, since restoring the flag can prevent sound from working in macOS on some other systems.

• --use-conn-none - Boolean flag, enabled if present.

On some sound cards enabling this option will enable additional usable audio channels (e.g. the bass or treble speaker of a pair, where only one is found without it).

Note: Enabling this option may increase the available channels, in which case any custom setting of AudioOutMask may need to be changed to match the new channel list.

11.11 OpenVariableRuntimeDxe

Provides in-memory emulated NVRAM implementation. This can be useful on systems with fragile (e.g. MacPro5,1, see discussion linked from this forum post) or incompatible NVRAM implementations. This driver is included by default in OpenDuet.

In addition to installing emulated NVRAM, this driver additionally installs an OpenCore compatible protocol enabling the following:

- NVRAM values are loaded from NVRAM/nvram.plist (or from NVRAM/nvram.fallback if it is present and NVRAM/nvram.plist is missing) on boot
- The Reset NVRAM option installed by the ResetNvramEntry driver removes the above files instead of affecting underlying NVRAM
- CTRL+Enter in the OpenCore bootpicker updates or creates NVRAM/nvram.plist

Recommended configuration settings for this driver:

- OpenVariableRuntimeDxe.efi loaded using LoadEarly=true. OpenDuet users should not load this driver, as it is included in OpenDuet.
- OpenRuntime.efi specified after OpenVariableRuntimeDxe.efi (when applicable), also loaded using LoadEarly=true for correct operation of RequestBootVarRouting.
 - RequestBootVarRouting is never strictly needed while using emulated NVRAM, but it can be convenient to leave it set on a system which needs to switch between real and emulated NVRAM.
 - RequestBootVarRouting is never required on an OpenDuet system, since there are no BIOS-managed boot entries to protect, therefore on OpenDuet recommended settings are LoadEarly=false for OpenRuntime.efi and RequestBootVarRouting=false.
- LegacySchema populated.
 - For simpler testing (allows arbitrary test variables), and future-proofing against changes in the variables required by macOS updates, use <string>*</string> settings, as described in notes below.
 - For increased security, populate sections with known required keys only, as shown in OpenCore's sample .plist files.
- ExposeSensitiveData with at least bit 0x1 set to make boot-path variable containing the OpenCore EFI partition UUID available to the Launchd.command script.

Variable loading happens prior to the NVRAM Delete (and Add) phases. Unless LegacyOverwrite is enabled, it will not overwrite any existing variable. Variables allowed for loading and for saving with CTRL+Enter must be specified in LegacySchema.

In order to allow changes to NVRAM within macOS to be captured and saved, an additional script must be installed. An example of such script can be found in Utilities/LogoutHook/Launchd.command.

Note 1: This driver requires working FAT write support in firmware, and sufficient free space on the OpenCore EFI partition for up to three saved NVRAM files.

Note 2: The nvram.plist (and nvram.fallback if present) files must have a root plist dictionary type and contain two fields:

- Version plist integer, file version, must be set to 1.
- Add plist dictionary, equivalent to Add from config.plist.

Note 3: When setting up legacy NVRAM, it can be convenient to set <string>*</string> as the value for the following three GUID keys in LegacySchema:

- 36C28AB5-6566-4C50-9EBD-CBB920F83843
- 7C436110-AB2A-4BBB-A880-FE41995C9F82
- 8BE4DF61-93CA-11D2-AA0D-00E098032B8C

This enables all variables saved by Launchd.command to be saved to nvram.plist, therefore it allows all arbitrary user test variables (e.g. as set by sudo nvram foo=bar) to be saved. Using this permissive policy is also future-proof against any changes in the variables which need to be passed from macOS update setup to the macOS Installer stage, in order for it to succeed. Nevertheless, once emulated NVRAM is set up, only allowing known strictly required variables (as shown in OpenCore's sample .plist files) is considerably more secure. See also the following warning about the overall security of loading NVRAM variables from a non-vaulted file.

Warning: The ability to load NVRAM from a file on disk can be dangerous, as it passes unprotected data to firmware variable services. Only use when no hardware NVRAM implementation is provided by the firmware or when the NVRAM implementation available in firmware is incompatible or dangerously fragile (e.g. in a state where excessive use may cause bricked hardware).

11.11.1 Managing macOS updates when using emulated NVRAM

OpenCore combined with OpenVariableRuntimeDxe will only use a given saved nvram.plist file once, if it is used to launch a macOS Installer boot entry. After that the used settings are moved to nvram.used and fallback settings, if any, from nvram.fallback are used instead. Launchd.command itself always copies the previous NVRAM settings to fallback, each time it saves new settings.

This strategy is used to work round the limitation that the Launchd.command script is not running, and therefore cannot save NVRAM changes (particularly default boot entry changes), during the second and subsequent restarts of the macOS installer.

In brief, this fallback strategy allows full or incremental OTA updates of recent macOS, which are started from within an existing macOS (with the Launchd.command script installed), to proceed without manual intervention.

However, for full installs, there can be more than one full restart back to the macOS Installer entry. In this case the fallback strategy will lose track of the correct startup item (i.e. macOS Installer) from the second reboot onwards. Equally, if installing to a drive other than the current default boot partition, this will not be automatically selected after the installer completes, as it would be when using non-emulated NVRAM. (This behaviour remains preferable to not having the fallback strategy, in which case a macOS Installer entry would be continually recreated in the picker menu, even once it no longer exists).

In both the above two cases it is recommended to use the following settings, to make it easy to manually control which boot entry is selected during the installer process:

- Set ShowPicker=true.
- Set Timeout=0.
- Set DisableWatchdog=true.
- If possible, start from a situation where there are no other pending macOS Installer entries in the boot menu (to avoid potential confusion as to which is relevant).

The first reboot should correctly select macOS Installer. For second and subsequent reboots, if a macOS Installer entry is still present it should be manually selected (using just Enter, not CTRL+Enter). Once a macOS Installer entry is no longer present, the entry for the new OS will still be automatically selected if it was the previous boot default. If

not, it should be manually selected (at this point, CTRL+Enter is a good idea as any final remaining installation restarts will be to this entry).

Note 1: When using emulated NVRAM but not installing from within an existing installed macOS (i.e. when installing from within macOS Recovery, or from an installation USB), please refer to this forum post (in Russian) for additional options.

Note 2: After upgrading from an earlier macOS version to macOS Sonoma, the Launchd.command script should be reinstalled, as a different strategy is required in order for NVRAM to be saved successfully.

Note 3: In macOS Sonoma the following additional constraints apply to the ESP partition on which OpenCore is installed, in order for the Launchd.command script to work successfully:

- It must not be set to automount.
- It must be unmounted again before shutdown or restart if it was manually mounted.

11.12 Properties

1. APFS

Type: plist dict

Description: Provide APFS support as configured in the APFS Properties section below.

2. AppleInput

Type: plist dict

Description: Configure the re-implementation of the Apple Event protocol described in the AppleInput Properties section below

3. Audio

Type: plist dict

Description: Configure audio backend support described in the Audio Properties section below.

Unless documented otherwise (e.g. ResetTrafficClass) settings in this section are for UEFI audio support only (e.g. OpenCore generated boot chime and audio assist) and are unrelated to any configuration needed for OS audio support (e.g. AppleALC).

UEFI audio support provides a way for upstream protocols to interact with the selected audio hardware and resources. All audio resources should reside in \EFI\OC\Resources\Audio directory. Currently the supported audio file formats are MP3 and WAVE PCM. While it is driver-dependent which audio stream format is supported, most common audio cards support 16-bit signed stereo audio at 44100 or 48000 Hz.

Audio file path is determined by audio type, audio localisation, and audio path. Each filename looks as follows: [audio type]_[audio localisation]_[audio path]. [audio ext]. For unlocalised files filename does not include the language code and looks as follows: [audio type]_[audio path]. [audio ext]. Audio extension can either be mp3 or wav.

- Audio type can be OCEFIAudio for OpenCore audio files or AXEFIAudio for macOS bootloader audio files.
- Audio localisation is a two letter language code (e.g. en) with an exception for Chinese, Spanish, and Portuguese. Refer to APPLE_VOICE_OVER_LANGUAGE_CODE definition for the list of all supported localisations.
- Audio path is the base filename corresponding to a file identifier. For macOS bootloader audio paths refer to APPLE_VOICE_OVER_AUDIO_FILE definition. For OpenCore audio paths refer to OC_VOICE_OVER_AUDIO_FILE definition. The only exception is OpenCore boot chime file, which is OCEFIAudio_VoiceOver_Boot.mp3.

Audio localisation is determined separately for macOS bootloader and OpenCore. For macOS bootloader it is set in preferences.efires archive in systemLanguage.utf8 file and is controlled by the operating system. For OpenCore the value of prev-lang:kbd variable is used. When native audio localisation of a particular file is missing, English language (en) localisation is used. Sample audio files can be found in OcBinaryData repository.

4. ConnectDrivers

Type: plist boolean

Failsafe: false

Description: Perform UEFI controller connection after driver loading.

This option is useful for loading drivers following UEFI driver model as they may not start by themselves. Examples of such drivers are filesystem or audio drivers. While effective, this option may not be necessary for

drivers performing automatic connection, and may slightly slowdown the boot.

Note: Some types of firmware, particularly those made by Apple, only connect the boot drive to speed up the boot process. Enable this option to be able to see all the boot options when running multiple drives.

5. Drivers

Type: plist array Failsafe: Empty

Description: Load selected drivers from OC/Drivers directory.

To be filled with plist dict values, describing each driver. Refer to the Drivers Properties section below.

6. Input

Type: plist dict

Description: Apply individual settings designed for input (keyboard and mouse) in the Input Properties section below.

7. Output

Type: plist dict

Description: Apply individual settings designed for output (text and graphics) in the Output Properties section

8. ProtocolOverrides

Type: plist dict

Description: Force builtin versions of certain protocols described in the ProtocolOverrides Properties section below.

Note: all protocol instances are installed prior to driver loading.

9. Quirks

Type: plist dict

Description: Apply individual firmware quirks described in the Quirks Properties section below.

10. ReservedMemory

Type: plist array Failsafe: Empty

Description: To be filled with plist dict values, describing memory areas exclusive to specific firmware and hardware functioning, which should not be used by the operating system. Examples of such memory regions could be the second 256 MB corrupted by the Intel HD 3000 or an area with faulty RAM. Refer to the ReservedMemory Properties section below for details.

11. Unload

Type: plist array Failsafe: Empty

Description: Unload specified firmware drivers.

To be filled with plist string entries containing the names of firmware drivers to unload before loading the Drivers section. This setting is typically only required if a user-provided driver is a variant of an existing system firmware driver, and if the new driver would detect itself as partially loaded, or otherwise fail to operate correctly, if the old driver is not unloaded first.

Warning: Unloading system firmware drivers is usually not required and not recommended. Poorly written drivers may crash when unloaded, or cause subsequent crashes (e.g by allowing themselves to be unloaded even though they have active dependencies). However standard UEFI network stack drivers should unload cleanly.

Note 1: Drivers specified in this option will be unloaded in the reverse of the order in which they were loaded, regardless of the order in which they are specified here.

Note 2: See SysReport/Drivers/DriverImageNames.txt for the list of drivers which this option can attempt to unload. The relevant name is the driver component name. Drivers are only listed if they implement DriverBindingProtocol and LoadedImageProtocol, and have an available component name.

Note 3: The NVRAM Lang and PlatformLang variables are ignored when determining the driver component names recognised by this option, and listed in the SysReport file. This is in order to make unloading images stable across changes in these variables. The UEFI Shell dh command takes account of these variables, so in

some circumstances may display different driver component names from those listed for this option, unless these variables are cleared.

11.13 APFS Properties

1. EnableJumpstart

Type: plist boolean Failsafe: false

Description: Load embedded APFS drivers from APFS containers.

An APFS EFI driver is bundled in all bootable APFS containers. This option performs the loading of signed APFS drivers (consistent with the ScanPolicy). Refer to the "EFI Jumpstart" section of the Apple File System Reference for details.

2. GlobalConnect

Type: plist boolean Failsafe: false

Description: Perform full device connection during APFS loading.

Every handle is connected recursively instead of the partition handle connection typically used for APFS driver loading. This may result in additional time being taken but can sometimes be the only way to access APFS partitions on certain firmware, such as those on older HP laptops.

3. HideVerbose

Type: plist boolean Failsafe: false

Description: Hide verbose output from APFS driver.

APFS verbose output can be useful for debugging.

4. JumpstartHotPlug

Type: plist boolean

Failsafe: false

Description: Load APFS drivers for newly connected devices.

Permits APFS USB hot plug which enables loading APFS drivers, both at OpenCore startup and during OpenCore picker display. Disable if not required.

5. MinDate

Type: plist integer

Failsafe: 0

Description: Minimal allowed APFS driver date.

The APFS driver date connects the APFS driver with the calendar release date. Apple ultimately drops support for older macOS releases and APFS drivers from such releases may contain vulnerabilities that can be used to compromise a computer if such drivers are used after support ends. This option permits restricting APFS drivers to current macOS versions.

- 0 require the default supported release date of APFS in OpenCore. The default release date will increase with time and thus this setting is recommended. Currently set to 2021/01/01.
- -1 permit any release date to load (strongly discouraged).
- Other use custom minimal APFS release date, e.g. 20200401 for 2020/04/01. APFS release dates can be found in OpenCore boot log and OcApfsLib.
- 6. MinVersion

Type: plist integer

Failsafe: 0

Description: Minimal allowed APFS driver version.

The APFS driver version connects the APFS driver with the macOS release. Apple ultimately drops support for older macOS releases and APFS drivers from such releases may contain vulnerabilities that can be used to compromise a computer if such drivers are used after support ends. This option permits restricting APFS drivers to current macOS versions.

- 0 require the default supported version of APFS in OpenCore. The default version will increase with time and thus this setting is recommended. Currently set to allow macOS Big Sur and newer (160000000000000).
- -1 permit any version to load (strongly discouraged).
- Other use custom minimal APFS version, e.g. 1412101001000000 from macOS Catalina 10.15.4. APFS versions can be found in OpenCore boot log and OcApfsLib.

11.14 AppleInput Properties

1. AppleEvent

Type: plist string Failsafe: Auto

Description: Determine whether the OpenCore builtin or the OEM Apple Event protocol is used.

This option determines whether the OEM Apple Event protocol is used (where available), or whether OpenCore's reversed engineered and updated re-implementation is used. In general OpenCore's re-implementation should be preferred, since it contains updates such as noticeably improved fine mouse cursor movement and configurable key repeat delays.

- Auto Use the OEM Apple Event implementation if available, connected and recent enough to be used, otherwise use the OpenCore re-implementation. On non-Apple hardware, this will use the OpenCore builtin implementation. On some Macs such as Classic Mac Pros, this will prefer the Apple implementation but on both older and newer Mac models than these, this option will typically use the OpenCore re-implementation instead. On older Macs, this is because the implementation available is too old to be used while on newer Macs, it is because of optimisations added by Apple which do not connect the Apple Event protocol except when needed e.g. except when the Apple boot picker is explicitly started. Due to its somewhat unpredictable results, this option is not typically recommended.
- Builtin Always use OpenCore's updated re-implementation of the Apple Event protocol. Use of this setting is recommended even on Apple hardware, due to improvements (better fine mouse control, configurable key delays) made in the OpenCore re-implementation of the protocol.
- OEM Assume Apple's protocol will be available at driver connection. On all Apple hardware where a recent enough Apple OEM version of the protocol is available whether or not connected automatically by Apple's firmware this option will reliably access the Apple implementation. On all other systems, this option will result in no keyboard or mouse support. For the reasons stated, Builtin is recommended in preference to this option in most cases.
- 2. CustomDelays

Type: plist boolean Failsafe: false

Description: Enable custom key repeat delays when using the OpenCore re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see AppleEvent setting).

- true The values of KeyInitialDelay and KeySubsequentDelay are used.
- false Apple default values of 500ms (50) and 50ms (5) are used.
- 3. GraphicsInputMirroring

Type: plist boolean

Failsafe: false

Description: Apple's own implementation of AppleEvent prevents keyboard input during graphics applications from appearing on the basic console input stream.

With the default setting of false, OpenCore's builtin implementation of AppleEvent replicates this behaviour.

On non-Apple hardware this can stop keyboard input working in graphics-based applications such as Windows BitLocker which use non-Apple key input methods.

The recommended setting on all hardware is true.

Note: AppleEvent's default behaviour is intended to prevent unwanted queued keystrokes from appearing after exiting graphics-based UEFI applications; this issue is already handled separately within OpenCore.

- true Allow keyboard input to reach graphics mode apps which are not using Apple input protocols.
- false Prevent key input mirroring to non-Apple protocols when in graphics mode.

4. KeyInitialDelay

Type: plist integer

Failsafe: 50 (500ms before first key repeat)

Description: Configures the initial delay before keyboard key repeats in the OpenCore re-implementation of the

Apple Event protocol, in units of 10ms.

The Apple OEM default value is 50 (500ms).

Note 1: On systems not using KeySupport, this setting may be freely used to configure key repeat behaviour.

Note 2: On systems using KeySupport, but which do not show the 'two long delays' behavior (see Note 3) and/or which always show a solid 'set default' indicator (see KeyForgetThreshold) then this setting may also be freely used to configure key repeat initial delay behaviour, except that it should never be set to less than KeyForgetThreshold to avoid uncontrolled key repeats.

Note 3: On some systems using KeySupport, you may find that you see one additional slow key repeat before normal speed key repeat starts, when holding a key down. If so, you may wish to configure KeyInitialDelay and KeySubsequentDelay according to the instructions at Note 3 of KeySubsequentDelay.

5. KeySubsequentDelay

Type: plist integer

Failsafe: 5 (50ms between subsequent key repeats)

Description: Configures the gap between keyboard key repeats in the OpenCore re-implementation of the Apple Event protocol, in units of 10ms.

The Apple OEM default value is 5 (50ms). 0 is an invalid value for this option (will issue a debug log warning and use 1 instead).

Note 1: On systems not using KeySupport, this setting may be freely used to configure key repeat behaviour.

Note 2: On systems using KeySupport, but which do not show the 'two long delays' behaviour (see Note 3) and/or which always show a solid 'set default' indicator (see KeyForgetThreshold) (which should apply to many/most systems using AMI KeySupport mode) then this setting may be freely used to configure key repeat subsequent delay behaviour, except that it should never be set to less than KeyForgetThreshold to avoid uncontrolled key repeats.

Note 3: On some systems using KeySupport, particularly KeySupport in non-AMI mode, you may find that after configuring KeyForgetThreshold you get one additional slow key repeat before normal speed key repeat starts, when holding a key down. On systems where this is the case, it is an unavoidable artefect of using KeySupport to emulate raw keyboard data, which is not made available by UEFI. While this 'two long delays' issue has minimal effect on overall usability, nevertheless you may wish to resolve it, and it is possible to do so as follows:

- Set CustomDelays to true
- Set KeyInitialDelay to 0
- Set KeySubsequentDelay to at least the value of your KeyForgetThreshold setting

The above procedure works as follows:

- Setting KeyInitialDelay to 0 cancels the Apple Event initial repeat delay (when using the OpenCore builtin Apple Event implementation with CustomDelays enabled), therefore the only long delay you will see is the non-configurable and non-avoidable initial long delay introduced by the BIOS key support on these machines.
- Key-smoothing parameter KeyForgetThreshold effectively acts as the shortest time for which a key can appear to be held, therefore a key repeat delay of less than this will guarantee at least one extra repeat for every key press, however quickly the key is physically tapped.
- In the unlikely event that you still get frequent, or occasional, double key responses after setting KeySubsequentDelay equal to your system's value of KeyForgetThreshold, then increase KeySubsequentDelay by one or two more until this effect goes away.

6. PointerDwellClickTimeout

Type: plist integer

Failsafe: 0

Description: Configure pointer dwell-clicking single left click timeout in milliseconds in the OpenCore re-

implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see AppleEvent setting).

When the timeout expires, a single left click is issued at the current position. 0 indicates the timeout is disabled.

7. PointerDwellDoubleClickTimeout

Type: plist integer

Failsafe: 0

Description: Configure pointer dwell-clicking single left double click timeout in milliseconds in the OpenCore re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see AppleEvent setting).

When the timeout expires, a single left double click is issued at the current position. 0 indicates the timeout is disabled.

8. PointerDwellRadius

Type: plist integer

Failsafe: 0

Description: Configure pointer dwell-clicking tolerance radius in pixels in the OpenCore re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see AppleEvent setting).

The radius is scaled by UIScale. When the pointer leaves this radius, the timeouts for PointerDwellClickTimeout and PointerDwellDoubleClickTimeout are reset and the new position is the centre for the new dwell-clicking tolerance radius.

9. PointerPollMask

Type: plist integer, 32 bit

Failsafe: -1

Description: Configure indices of polled pointers.

Selects pointer devices to poll for AppleEvent motion events. -1 implies all devices. A bit sum is used to determine particular devices. E.g. to enable devices 0, 2, 3 the value will be 1+4+8 (corresponding powers of two). A total of 32 configurable devices is supported.

Certain pointer devices can be present in the firmware even when no corresponding physical devices are available. These devices usually are placeholders, aggregate devices, or proxies. Gathering information from these devices may result in inaccurate motion activity in the user interfaces and even cause performance issues. Disabling such pointer devices is recommended for laptop setups having issues of this kind.

The amount of pointer devices available in the system can be found in the log. Refer to Found N pointer devices message for more details.

Note: Has no effect when using the OEM Apple implementation (see AppleEvent setting).

10. PointerPollMax

Type: plist integer

Failsafe: 0

Description: Configure maximum pointer polling period in ms.

This is the maximum period the OpenCore builtin AppleEvent driver polls pointer devices (e.g. mice, trackpads) for motion events. The period is increased up to this value as long as the devices do not respond in time. The current implementation defaults to 80 ms. Setting 0 leaves this default unchanged.

Certain trackpad drivers often found in Dell laptops can be very slow to respond when no physical movement happens. This can affect OpenCanopy and FileVault 2 user interface responsiveness and loading times. Increasing the polling periods can reduce the impact.

Note: The OEM Apple implementation uses a polling rate of 2 ms.

11. PointerPollMin

Type: plist integer

Failsafe: 0

Description: Configure minimal pointer polling period in ms.

This is the minimal period the OpenCore builtin AppleEvent driver polls pointer devices (e.g. mice, trackpads) for motion events. The current implementation defaults to 10 ms. Setting 0 leaves this default unchanged.

Note: The OEM Apple implementation uses a polling rate of 2 ms.

12. PointerSpeedDiv

Type: plist integer

Failsafe: 1

Description: Configure pointer speed divisor in the OpenCore re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see AppleEvent setting).

Configures the divisor for pointer movements. The Apple OEM default value is 1. 0 is an invalid value for this option.

Note: The recommended value for this option is 1. This value may optionally be modified in combination with PointerSpeedMul, according to user preference, to achieve customised mouse movement scaling.

13. PointerSpeedMul

Type: plist integer

Failsafe: 1

Description: Configure pointer speed multiplier in the OpenCore re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see AppleEvent setting).

Configures the multiplier for pointer movements. The Apple OEM default value is 1.

Note: The recommended value for this option is 1. This value may optionally be modified in combination with PointerSpeedDiv, according to user preference, to achieve customised mouse movement scaling.

11.15 Audio Properties

1. AudioCodec

Type: plist integer

Failsafe: 0

Description: Codec address on the specified audio controller for audio support.

This typically contains the first audio codec address on the builtin analog audio controller (HDEF). Audio codec addresses, e.g. 2, can be found in the debug log (marked in bold-italic):

OCAU: 1/3 PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x1)/VenMsg(<redacted>,00000000) (4 outputs)

OCAU: 2/3 PciRoot(0x0)/Pci(0x3,0x0)/VenMsg(<redacted>,00000000) (1 outputs)
OCAU: 3/3 PciRoot(0x0)/Pci(0x1B,0x0)/VenMsg(<redacted>,02000000) (7 outputs)

As an alternative, this value can be obtained from ${\tt IOHDACodecDevice}$ class in ${\tt I/O}$ Registry containing it in ${\tt IOHDACodecAddress}$ field.

2. AudioDevice

Type: plist string Failsafe: Empty

Description: Device path of the specified audio controller for audio support.

This typically contains builtin analog audio controller (HDEF) device path, e.g. PciRoot(0x0)/Pci(0x1b,0x0). The list of recognised audio controllers can be found in the debug log (marked in bold-italic):

OCAU: 1/3 PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x1)/VenMsg(<redacted>,00000000) (4 outputs)

OCAU: 2/3 PciRoot(0x0)/Pci(0x3,0x0)/VenMsg(<redacted>,00000000) (1 outputs)
OCAU: 3/3 PciRoot(0x0)/Pci(0x1B,0x0)/VenMsg(<redacted>,02000000) (7 outputs)

If using AudioDxe, the available controller device paths are also output on lines formatted like this:

HDA: Connecting controller - PciRoot(0x0)/Pci(0x1B,0x0)

Finally, gfxutil -f HDEF command can be used in macOS to obtain the device path.

Specifying an empty device path results in the first available codec and audio controller being used. The value of AudioCodec is ignored in this case. This can be a convenient initial option to try to get UEFI audio working. Manual settings as above will be required when this default value does not work.

AudioOutMask

Type: plist integer

Failsafe: -1

Description: Bit field indicating which output channels to use for UEFI sound.

Audio mask is 1 « audio output (equivalently 2 ^ audio output). E.g. for audio output 0 the bitmask is 1, for output 3 it is 8, and for outputs 0 and 3 it is 9.

The number of available output nodes (N) for each HDA codec is shown in the debug log (marked in bold-italic), audio outputs 0 to N-1 may be selected:

```
OCAU: 1/3 PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x1)/VenMsg(<redacted>,00000000) (4 outputs)
```

OCAU: 2/3 PciRoot(0x0)/Pci(0x3,0x0)/VenMsg(<redacted>,00000000) (1 outputs)
OCAU: 3/3 PciRoot(0x0)/Pci(0x1B,0x0)/VenMsg(<redacted>,02000000) (7 outputs)

When AudioDxe is used then additional information about each output channel is logged during driver binding, including the bitmask for each output. The bitmask values for the desired outputs should be added together to obtain the AudioOutMask value:

```
HDA: | Port widget @ 0x9 is an output (pin defaults 0x2B4020) (bitmask 1)
HDA: | Port widget @ 0xA is an output (pin defaults 0x90100112) (bitmask 2)
HDA: | Port widget @ 0xB is an output (pin defaults 0x90100110) (bitmask 4)
HDA: | Port widget @ 0x10 is an output (pin defaults 0x4BE030) (bitmask 8)
```

Further information on the available output channels may be found from a Linux codec dump using the command:

cat /proc/asound/card{n}/codec#{m}

Using AudioOutMask, it is possible to play sound to more than one channel (e.g. main speaker plus bass speaker; headphones plus speakers) as long as all the chosen outputs support the sound file format in use; if any do not then no sound will play and a warning will be logged.

When all available output channels on the codec support the available sound file format then a value of -1 will play sound to all channels simultaneously. If this does not work it will usually be quickest to try each available output channel one by one, by setting AudioOutMask to 1, 2, 4, etc., up to 2 ^ N - 1, in order to work out which channel(s) produce sound.

4. AudioSupport

Type: plist boolean

Failsafe: false

Description: Activate audio support by connecting to a backend driver.

Enabling this setting routes audio playback from builtin protocols to specified (AudioOutMask) dedicated audio ports of the specified codec (AudioCodec), located on the specified audio controller (AudioDevice).

5. DisconnectHda

Type: plist boolean

Failsafe: false

Description: Disconnect HDA controller before loading drivers.

May be required on some systems (e.g. Apple hardware, VMware Fusion guest) to allow a UEFI sound driver (such as AudioDxe) to take control of the audio hardware.

Note: In addition to this option, most Apple hardware also requires the --gpio-setup driver argument which is dealt with in the AudioDxe section.

6. MaximumGain

Type: plist integer

Failsafe: -15

Description: Maximum gain to use for UEFI audio, specified in decibels (dB) with respect to amplifier reference level of 0 dB (see note 1).

All UEFI audio will use this gain setting when the system amplifier gain read from the SystemAudioVolumeDB NVRAM variable is higher than this. This is to avoid over-loud UEFI audio when the system volume is set very high, or the SystemAudioVolumeDB NVRAM value has been misconfigured.

Note 1: Decibels (dB) specify gain (positive values; increase in volume) or attenuation (negative values; decrease in volume) compared to some reference level. When you hear the sound level of a jet plane expressed as 120 decibels, say, the reference level is the sound level just audible to an average human. However generally in acoustic science and computer audio any reference level can be specified. Intel HDA and macOS natively use decibels to specify volume level. On most Intel HDA hardware the reference level of 0 dB is the *loudest* volume of the hardware, and all lower volumes are therefore negative numbers. The quietest volume on typical sound hardware is around -55 dB to -60 dB.

Note 2: Matching how macOS handles decibel values, this value is converted to a signed byte; therefore values outside -128 dB to +127 dB (which are well beyond physically plausible volume levels) are not allowed.

Note 3: Digital audio output – which does not have a volume slider in-OS – ignores this and all other gain settings, only mute settings are relevant.

7. MinimumAssistGain

Type: plist integer

Failsafe: -30

Description: Minimum gain in decibels (dB) to use for picker audio assist.

The screen reader will use this amplifier gain if the system amplifier gain read from the SystemAudioVolumeDB NVRAM variable is lower than this.

Note 1: In addition to this setting, because audio assist must be audible to serve its function, audio assist is not muted even if the OS sound is muted or the StartupMute NVRAM variable is set.

Note 2: See MaximumGain for an explanation of decibel volume levels.

8. MinimumAudibleGain

Type: plist integer

Failsafe: -128

Description: Minimum gain in decibels (dB) at which to attempt to play any sound.

The boot chime will not play if the system amplifier gain level in the SystemAudioVolumeDB NVRAM variable is lower than this.

Note 1: This setting is designed to save unnecessary pauses due to audio setup at inaudible volume levels, when no sound will be heard anyway. Whether there are inaudible volume levels depends on the hardware. On some hardware (including Apple) the audio values are well enough matched to the hardware that the lowest volume levels available are very quiet but audible, whereas on some other hardware combinations, the lowest part of the volume range may not be audible at all.

Note 2: See MaximumGain for an explanation of decibel volume levels.

9. PlayChime

Type: plist string

Failsafe: Auto

Description: Play chime sound at startup.

Enabling this setting plays the boot chime using the builtin audio support. The volume level is determined by the SystemAudioVolumeDB NVRAM variable. Supported values are:

- Auto Enables chime when StartupMute NVRAM variable is not present or set to 00.
- Enabled Enables chime unconditionally.
- Disabled Disables chime unconditionally.

Note 1: Enabled can be used separately from the StartupMute NVRAM variable to avoid conflicts when the firmware is able to play the boot chime.

Note 2: Regardless of this setting, the boot chime will not play if system audio is muted, i.e. if the SystemAudioVolume NVRAM variable has bit 0x80 set.

10. ResetTrafficClass

Type: plist boolean

Failsafe: false

Description: Set HDA Traffic Class Select Register to TCO.

AppleHDA kext will function correctly only if TCSEL register is configured to use TC0 traffic class. Refer to Intel I/O Controller Hub 9 (ICH9) Family Datasheet (or any other ICH datasheet) for more details about this register.

Note: This option is independent from AudioSupport. If AppleALC is used it is preferred to use AppleALC alctcsel property instead.

11. SetupDelay

Type: plist integer

Failsafe: 0

Description: Audio codec reconfiguration delay in milliseconds.

Some codecs require a vendor-specific delay after the reconfiguration (e.g. volume setting). This option makes it configurable. A typical delay can be up to 0.5 seconds.

11.16 Drivers Properties

1. Arguments

Type: plist string Failsafe: Empty

Description: Some OpenCore plugins accept optional additional arguments which may be specified as a string

here.

2. Comment

Type: plist string Failsafe: Empty

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

3. Enabled

Type: plist boolean Failsafe: false

Description: If false this driver entry will be ignored.

4. LoadEarly

Type: plist boolean Failsafe: false

Description: Load the driver early in the OpenCore boot process, before NVRAM setup.

Note: Do not enable this option unless specifically recommended to do so for a given driver and purpose.

5. Path

Type: plist string Failsafe: Empty

Description: Path of file to be loaded as a UEFI driver from OC/Drivers directory.

11.17 Input Properties

1. KeyFiltering

Type: plist boolean Failsafe: false

Description: Enable keyboard input sanity checking.

Apparently some boards such as the GA Z77P-D3 may return uninitialised data in EFI_INPUT_KEY with all input protocols. This option discards keys that are neither ASCII, nor are defined in the UEFI specification (see tables 107 and 108 in version 2.8).

2. KeyForgetThreshold

Type: plist integer

Failsafe: 0

Description: Treat duplicate key presses as held keys if they arrive during this timeout, in 10 ms units. Only applies to systems using KeySupport.

AppleKeyMapAggregator protocol is supposed to contain a fixed length buffer of currently pressed keys. However, the majority of the drivers which require KeySupport report key presses as interrupts, with automatically generated key repeat behaviour with some defined initial and subsequent delay. As a result, to emulate the raw key behaviour required by several Apple boot systems, we use a timeout to merge multiple repeated keys which are submitted within a small timeout window.

This option allows setting this timeout based on the platform. The recommended value for the majority of platforms is from 5 (50 milliseconds) to 7 (70 milliseconds), although values up to 9 (90 milliseconds) have been observed to be required on some PS/2 systems. For reference, holding a key on VMware will repeat roughly every 20 milliseconds and the equivalent value for APTIO V is 30-40 milliseconds. KeyForgetThreshold should be configured to be longer than this. Thus, it is possible to configure a lower KeyForgetThreshold value on platforms with a faster native driver key repeat rate, for more responsive input, and it is required to set a higher value on slower platforms.

Pressing keys one after the other results in delays of at least 60 and 100 milliseconds for the same platforms. Ideally, KeyForgetThreshold should remain lower than this value, to avoid merging real key presses.

Tuning the value of KeyForgetThreshold is necessary for accurate and responsive keyboard input on systems on which KeySupport is enabled, and it is recommended to follow the instructions below to tune it correctly for your system.

Note 1: To tune KeyForgetThreshold, you may use the 'set default' indicator within either OpenCanopy or the builtin picker. If KeyForgetThreshold is too low then the 'set default' indicator will continue to flicker while CTRL or =/+ is held down. You should configure the lowest value which avoids this flicker. On some systems (e.g. Aptio IV and potentially other systems using AMI KeySupport mode) you will be able to find a minimum KeyForgetThreshold value at which the 'set default' indicator goes on and stays on with no flicker at all - if so, use this value. On most other systems using KeySupport, you will find that the 'set default' indicator will flicker once, when first pressing and holding the CTRL or =/+ key, and then after a further very brief interval will go on and stay on. On such systems, you should chose the lowest value of KeyForgetThreshold at which you see only one initial flicker and then no subsequent flickering. (Where this happens, it is an unavoidable artefect on those systems of using KeySupport to emulate raw keyboard data, which is not made available by UEFI.)

Note 2: KeyForgetThreshold should never need to be more than about 9 or 10 at most. If it is set to a value much higher than this, it will result in noticeably unresponsive keyboard input. Therefore, for overall key responsiveness, it is strongly recommended to configure a relatively lower value, at which the 'set default' indicator flickers once and then does not flicker, rather than using a much higher value (i.e. significantly greater than 10), which you may be able to find but should not use, where the 'set default' indicator does not flicker at all.

3. KeySupport

Type: plist boolean Failsafe: false

Description: Enable internal keyboard input translation to AppleKeyMapAggregator protocol.

This option activates the internal keyboard interceptor driver, based on AppleGenericInput, also known as AptioInputFix, to fill the AppleKeyMapAggregator database for input functioning. In cases where a separate driver such as OpenUsbKbDxe is used, this option should never be enabled. Additionally, this option is not required and should not be enabled with Apple firmware.

4. KeySupportMode

Type: plist string

Failsafe: Auto

Description: Set internal keyboard input translation to AppleKeyMapAggregator protocol mode.

- Auto Performs automatic choice as available with the following preference: AMI, V2, V1.
- V1 Uses UEFI standard legacy input protocol EFI_SIMPLE_TEXT_INPUT_PROTOCOL.
- V2 Uses UEFI standard modern input protocol EFI_SIMPLE_TEXT_INPUT_EX_PROTOCOL.
- AMI Uses APTIO input protocol AMI EFIKEYCODE PROTOCOL.

Note: Currently V1, V2, and AMI unlike Auto only do filtering of the particular specified protocol. This may change in the future versions.

5. KeySwap

Type: plist boolean

Failsafe: false

Description: Swap Command and Option keys during submission.

This option may be useful for keyboard layouts with Option key situated to the right of Command key.

6. PointerSupport

Type: plist boolean Failsafe: false

Description: Enable internal pointer driver.

This option implements standard UEFI pointer protocol (EFI_SIMPLE_POINTER_PROTOCOL) through certain OEM protocols. The option may be useful on Z87 ASUS boards, where EFI_SIMPLE_POINTER_PROTOCOL is defective.

7. PointerSupportMode

Type: plist string
Failsafe: Empty

Description: Set OEM protocol used for internal pointer driver.

Currently the only supported variant is ASUS, using specialised protocol available on certain Z87 and Z97 ASUS boards. More details can be found in LongSoft/UefiTool#116. The value of this property cannot be empty if PointerSupport is enabled.

8. TimerResolution

Type: plist integer

Failsafe: 0

Description: Set architecture timer resolution.

This option allows updating the firmware architecture timer period with the specified value in 100 nanosecond units. Setting a lower value typically improves performance and responsiveness of the interface and input handling.

The recommended value is 50000 (5 milliseconds) or slightly higher. Select ASUS Z87 boards use 60000 for the interface. Apple boards use 100000. In case of issues, this option can be left as 0 to not change the timer resolution.

11.18 Output Properties

1. ClearScreenOnModeSwitch

Type: plist boolean

Failsafe: false

Description: Some types of firmware only clear part of the screen when switching from graphics to text mode, leaving a fragment of previously drawn images visible. This option fills the entire graphics screen with black colour before switching to text mode.

Note: This option only applies to System renderer.

2. ConsoleFont

Type: plist string

Failsafe: Empty (use OpenCore builtin console font)

Description: Specify the console font to use for OpenCore Builtin text renderer.

The font file must be located in EFI/OC/Resources/Font/{font-name}.hex and must be 8x16 resolution. Various console fonts can be found online in either .bdf or .hex format. .bdf can be converted to .hex format using gbdfed (available for Linux or macOS).

There is often no need to change console font, the main use-case being to provide an extended character set for those relatively rare EFI applications which have multi-lingual support (e.g. memtest86).

The OcBinaryData repository includes:

- Terminus A font with extensive character support suitable for applications such as the above.
- TerminusCore A lightly modified version of the Terminus font, making some glyphs (@KMRSTVWimrsw) more similar to the free ISO Latin font used in XNU and OpenCore.

Terminus and TerminusCore are provided under the SIL Open Font License, Version 1.1. Some additional GPL licensed fonts from the EPTO Fonts library, converted to the required .hex format, can be found here.

Note 1: On many newer systems the **System** text renderer already provides a full set of international characters, in which case this can be used without requiring the **Builtin** renderer and a custom font.

Note 2: This option only affects the Builtin text renderer and only takes effect from the point at which the Builtin renderer is configured. When console output is visible before this point, it is using the system console font.

3. ConsoleMode

Type: plist string

Failsafe: Empty (Maintain current console mode)

Description: Sets console output mode as specified with the WxH (e.g. 80x24) formatted string.

Set to Max to attempt using the largest available console mode.

Note: This field is best left empty on most types of firmware.

$4. \ {\tt DirectGopRendering}$

Type: plist boolean

Failsafe: false

Description: Use builtin graphics output protocol renderer for console.

On certain firmware, such as on the MacPro5,1, this may provide better performance or fix rendering issues. However, this option is not recommended unless there is an obvious benefit as it may result in issues such as slower scrolling.

This renderer fully supports AppleEg2Info protocol and will provide screen rotation for all EFI applications. In order to provide seamless rotation compatibility with EfiBoot, builtin AppleFramebufferInfo should also be used, i.e. it may need to be overridden on Mac EFI.

5. ForceResolution

Type: plist boolean

Failsafe: false

Description: Forces Resolution to be set in cases where the desired resolution is not available by default, such as on legacy Intel GMA and first generation Intel HD Graphics (Ironlake/Arrandale). Setting Resolution to Max will try to pull the largest available resolution from the connected display's EDID.

Note: This option depends on the OC_FORCE_RESOLUTION_PROTOCOL protocol being present. This protocol is currently only supported by OpenDuetPkg. The OpenDuetPkg implementation currently only supports Intel iGPUs and certain ATI GPUs.

6. GopBurstMode

Type: plist boolean

Failsafe: false

Description: Enable write-combining (WC) caching for GOP memory, if system firmware has not already enabled it.

Some older firmware (e.g. EFI-era Macs) fails to set write-combining caching (aka burst mode) for GOP memory, even though the CPU supports it. Setting this can give a considerable speed-up for GOP operations, especially on systems which require DirectGopRendering.

Note 1: This quirk takes effect whether or not DirectGopRendering is set, and in some cases may give a noticeable speed-up to GOP operations even when DirectGopRendering is false.

Note 2: On most systems from circa 2013 onwards, write-combining caching is already applied by the firmware to GOP memory, in which case GopBurstMode is unnecessary. On such systems enabling the quirk should normally be harmless, producing an OCC: debug log entry indicating that burst mode is already started.

Note 3: Some caution should be taken when enabling this quirk, as it has been observed to cause hangs on a few systems. Since additional guards have been added to try to prevent this, please log a bugtracker issue if such a system is found.

7. GopPassThrough

Type: plist string Failsafe: Disabled

Description: Provide GOP protocol instances on top of UGA protocol instances.

This option provides the GOP protocol via a UGA-based proxy for firmware that do not implement the protocol. The supported values for the option are as follows:

• Enabled — provide GOP for all UGA protocols.

- Apple provide GOP for AppleFramebufferInfo-enabled protocols.
- Disabled do not provide GOP.

Note: This option requires ProvideConsoleGop to be enabled.

$8. \ {\tt IgnoreTextInGraphics}$

Type: plist boolean

Failsafe: false

Description: Some types of firmware output text onscreen in both graphics and text mode. This is typically unexpected as random text may appear over graphical images and cause UI corruption. Setting this option to true will discard all text output if console control is not in Text mode.

Note: This option only applies to the System renderer.

9. InitialMode

Type: plist string

Failsafe: Auto

Description: Selects the internal ConsoleControl mode in which TextRenderer will operate.

Available values are Auto, Text and Graphics. Text and Graphics specify the named mode. Auto uses the current mode of the system ConsoleControl protocol when one exists, defaulting to Text mode otherwise.

UEFI firmware typically supports ConsoleControl with two rendering modes: Graphics and Text. Some types of firmware do not provide a native ConsoleControl and rendering modes. OpenCore and macOS expect text to only be shown in Text mode but graphics to be drawn in any mode, and this is how the OpenCore Builtin renderer behaves. Since this is not required by the UEFI specification, behaviour of the system ConsoleControl protocol, when it exists, may vary.

$10.\ {\tt ProvideConsoleGop}$

Type: plist boolean

Failsafe: false

Description: Ensure GOP (Graphics Output Protocol) on console handle.

macOS bootloader requires GOP or UGA (for 10.4 EfiBoot) to be present on console handle, yet the exact location of the graphics protocol is not covered by the UEFI specification. This option will ensure GOP and UGA, if present, are available on the console handle.

Note: This option will also replace incompatible implementations of GOP on the console handle, as may be the case on the MacPro5,1 when using modern GPUs.

11. ReconnectGraphicsOnConnect

Type: plist boolean Failsafe: false

Description: Reconnect all graphics drivers during driver connection.

On certain firmware, it may be desirable to use an alternative graphics driver, for example BiosVideo.efi, providing better screen resolution options on legacy machines, or a driver supporting ForceResolution. This option attempts to disconnect all currently connected graphics drivers before connecting newly loaded drivers.

Note: This option requires ConnectDrivers to be enabled.

12. ReconnectOnResChange

Type: plist boolean

Failsafe: false

Description: Reconnect console controllers after changing screen resolution.

On certain firmware, the controllers that produce the console protocols (simple text out) must be reconnected when the screen resolution is changed via GOP. Otherwise, they will not produce text based on the new resolution.

Note: On several boards this logic may result in black screen when launching OpenCore from Shell and thus it is optional. In versions prior to 0.5.2 this option was mandatory and not configurable. Please do not use this unless

required.

13. ReplaceTabWithSpace Type: plist boolean

Failsafe: false

Description: Some types of firmware do not print tab characters or everything that follows them, causing difficulties in using the UEFI Shell's builtin text editor to edit property lists and other documents. This option makes the console output spaces instead of tabs.

Note: This option only applies to System renderer.

14. Resolution

Type: plist string

Failsafe: Empty (Maintain current screen resolution) Description: Sets console output screen resolution.

- Set to WxH@Bpp (e.g. 1920x1080@32) or WxH (e.g. 1920x1080) formatted string to request custom resolution from GOP if available.
- Set to Max to attempt using the largest available screen resolution. When set to Max all available resolutions will be listed in lines starting OCC: Mode in the debug log.

On HiDPI screens APPLE_VENDOR_VARIABLE_GUID UIScale NVRAM variable may need to be set to 02 to enable HiDPI scaling in Builtin text renderer, FileVault 2 UEFI password interface, and boot screen logo. Refer to the Recommended Variables section for details.

Note: This will fail when console handle has no GOP protocol. When the firmware does not provide it, it can be added with ProvideConsoleGop set to true.

15. SanitiseClearScreen

Type: plist boolean

Failsafe: false

Description: Some types of firmware reset screen resolutions to a failsafe value (such as 1024x768) on the attempts to clear screen contents when large display (e.g. 2K or 4K) is used. This option attempts to apply a workaround.

Note: This option only applies to the System renderer. On all known affected systems, ConsoleMode must be set to an empty string for this option to work.

16. TextRenderer

Type: plist string Failsafe: BuiltinGraphics

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

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

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

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

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

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

and defective laptop firmware, which can only draw in Text mode.

The use of System protocols is more complicated. Typically, the preferred setting is SystemGraphics or SystemText. Enabling ProvideConsoleGop, setting Resolution to Max, enabling ReplaceTabWithSpace is useful on almost all platforms. SanitiseClearScreen, IgnoreTextInGraphics, and ClearScreenOnModeSwitch are more specific, and their use depends on the firmware.

Note: Some Macs, such as the MacPro5,1, may have incompatible console output when using modern GPUs, and thus only BuiltinGraphics may work for them in such cases. NVIDIA GPUs may require additional firmware upgrades.

17. UIScale

Type: plist integer, 8 bit

Failsafe: -1

Description: User interface scaling factor.

Corresponds to 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:UIScale variable.

- 1 1x scaling, corresponds to normal displays.
- 2 2x scaling, corresponds to HiDPI displays.
- -1 leaves the current variable unchanged.
- 0 automatically chooses scaling based on the current resolution.

Note 1: Automatic scale factor detection works on the basis of total pixel area and may fail on small HiDPI displays, in which case the value may be manually managed using the NVRAM section.

 $Note \ 2$: When switching from manually specified NVRAM variable to this preference an NVRAM reset may be needed.

18. UgaPassThrough

 $\mathbf{Type}:$ plist boolean

Failsafe: false

Description: Provide UGA protocol instances on top of GOP protocol instances.

Some types of firmware do not implement the legacy UGA protocol but this may be required for screen output by older EFI applications such as EfiBoot from 10.4.

11.19 ProtocolOverrides Properties

1. AppleAudio

 $\mathbf{Type} {:}\ \mathtt{plist}\ \mathtt{boolean}$

Failsafe: false

Description: Replaces Apple audio protocols with builtin versions.

Apple audio protocols allow OpenCore and the macOS bootloader to play sounds and signals for screen reading or audible error reporting. Supported protocols are beep generation and VoiceOver. The VoiceOver protocol is only provided natively by Gibraltar machines (T2), however versions of macOS which support VoiceOver will see and use the implementation provided by OpenCore, on screens such as FileVault 2 unlock. VoiceOver is not supported before macOS High Sierra (10.13). Older macOS versions use the AppleHDA protocol (which is not currently implemented) instead.

Only one set of audio protocols can be available at a time, so this setting should be enabled in order to enable audio playback in the OpenCore user interface on Mac systems implementing some of these protocols.

Note: The backend audio driver needs to be configured in UEFI Audio section for these protocols to be able to stream audio.

2. AppleBootPolicy

Type: plist boolean

Failsafe: false

Description: Replaces the Apple Boot Policy protocol with a builtin version. This may be used to ensure APFS compatibility on VMs and legacy Macs.

Note: This option is advisable on certain Macs, such as the MacPro5,1, that are APFS compatible but on which the Apple Boot Policy protocol has recovery detection issues.

3. AppleDebugLog

Type: plist boolean

Failsafe: false

Description: Replaces the Apple Debug Log protocol with a builtin version.

4. AppleEg2Info

Type: plist boolean Failsafe: false

Description: Replaces the Apple EFI Graphics 2 protocol with a builtin version.

Note 1: This protocol allows newer EfiBoot versions (at least 10.15) to expose screen rotation to macOS. Refer to ForceDisplayRotationInEFI variable description on how to set screen rotation angle.

Note 2: On systems without native support for ForceDisplayRotationInEFI, DirectGopRendering=true is also required for this setting to have an effect.

5. AppleFramebufferInfo

Type: plist boolean

Failsafe: false

Description: Replaces the Apple Framebuffer Info protocol with a builtin version. This may be used to override framebuffer information on VMs and legacy Macs to improve compatibility with legacy EfiBoot such as the one in Mac OS X 10.4.

Note: The current implementation of this property results in it only being active when GOP is available (it is always equivalent to false otherwise).

6. AppleImageConversion

 $\mathbf{Type}:$ plist boolean

Failsafe: false

Description: Replaces the Apple Image Conversion protocol with a builtin version.

7. AppleImg4Verification

Type: plist boolean

Failsafe: false

Description: Replaces the Apple IMG4 Verification protocol with a builtin version. This protocol is used to verify im4m manifest files used by Apple Secure Boot.

8. AppleKeyMap

Type: plist boolean

Failsafe: false

Description: Replaces Apple Key Map protocols with builtin versions.

9. AppleRtcRam

Type: plist boolean

Failsafe: false

Description: Replaces the Apple RTC RAM protocol with a builtin version.

Note: Builtin version of Apple RTC RAM protocol may filter out I/O attempts to certain RTC memory addresses. The list of addresses can be specified in 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:rtc-blacklist variable as a data array.

10. AppleSecureBoot

Type: plist boolean

Failsafe: false

Description: Replaces the Apple Secure Boot protocol with a builtin version.

11. AppleSmcIo

Type: plist boolean

Failsafe: false

Description: Replaces the Apple SMC I/O protocol with a builtin version.

This protocol replaces the legacy VirtualSmc UEFI driver, and is compatible with any SMC kernel extension. However, in case the FakeSMC kernel extension is used, manual NVRAM key variable addition may be needed.

12. AppleUserInterfaceTheme

Type: plist boolean

Failsafe: false

Description: Replaces the Apple User Interface Theme protocol with a builtin version.

13. DataHub

Type: plist boolean Failsafe: false

Description: Replaces the Data Hub protocol with a builtin version.

Note: This will discard all previous entries if the protocol was already installed, so all properties required for the safe operation of the system must be specified in the configuration file.

14. DeviceProperties

Type: plist boolean Failsafe: false

Description: Replaces the Device Property protocol with a builtin version. This may be used to ensure full compatibility on VMs and legacy Macs.

Note: This will discard all previous entries if the protocol was already installed, so all properties required for safe operation of the system must be specified in the configuration file.

15. FirmwareVolume

Type: plist boolean Failsafe: false

Description: Wraps Firmware Volume protocols, or installs a new version, to support custom cursor images for FileVault 2. Set to **true** to ensure FileVault 2 compatibility on anything other than on VMs and legacy Macs.

Note: Several virtual machines, including VMware, may have corrupted cursor images in HiDPI mode and thus, may also require enabling this setting.

16. HashServices

Type: plist boolean Failsafe: false

Description: Replaces Hash Services protocols with builtin versions. Set to true to ensure FileVault 2 compatibility on platforms with defective SHA-1 hash implementations. This can be determined by an invalid cursor size when UIScale is set to 02. Platforms earlier than APTIO V (Haswell and older) are typically affected.

17. OSInfo

Type: plist boolean Failsafe: false

Description: Replaces the OS Info protocol with a builtin version. This protocol is typically used by the firmware and other applications to receive notifications from the macOS bootloader.

18. PciIo

Type: plist boolean Failsafe: false

Description: Replaces functions in CpuIo and PciRootBridgeIo with 64-bit MMIO compatible ones to fix Invalid Parameter when using 4G Decoding. This affects UEFI drivers such as AudioDxe which access 64-bit MMIO devices. Platforms earlier than APTIO V (Haswell and older) are typically affected.

19. UnicodeCollation

Type: plist boolean

Failsafe: false

Description: Replaces unicode collation services with builtin versions. Set to **true** to ensure UEFI Shell compatibility on platforms with defective unicode collation implementations. Legacy Insyde and APTIO platforms on Ivy Bridge, and earlier, are typically affected.

11.20 Quirks Properties

$1. \ {\tt ActivateHpetSupport}$

Type: plist boolean

Failsafe: false

Description: Activates HPET support.

Older boards like ICH6 may not always have HPET setting in the firmware preferences, this option tries to force enable it.

DisableSecurityPolicy Type: plist boolean

Failsafe: false

Description: Disable platform security policy.

Note: This setting disables various security features of the firmware, defeating the purpose of any kind of Secure

Boot. Do NOT enable if using UEFI Secure Boot.

3. EnableVectorAcceleration

 $\mathbf{Type}:$ plist boolean

Failsafe: false

Description: Enable AVX vector acceleration of SHA-512 and SHA-384 hashing algorithms.

Note: This option may cause issues on certain laptop firmwares, including Lenovo.

4. EnableVmx

Type: plist boolean Failsafe: false

Description: Enable Intel virtual machine extensions.

Note: Required to allow virtualization in Windows on some Mac hardware. VMX is enabled or disabled and locked by BIOS before OpenCore starts on most firmware. Use BIOS to enable virtualization where possible.

5. ExitBootServicesDelay

Type: plist integer

Failsafe: 0

Description: Adds delay in microseconds after EXIT BOOT SERVICES event.

This is a very rough workaround to circumvent the Still waiting for root device message on some APTIO IV firmware (ASUS Z87-Pro) particularly when using FileVault 2. It appears that for some reason, they execute code in parallel to EXIT_BOOT_SERVICES, which results in the SATA controller being inaccessible from macOS. A better approach is required and Acidanthera is open to suggestions. Expect 3 to 5 seconds to be adequate when this quirk is needed.

6. ForceOcWriteFlash

Type: plist boolean

Failsafe: false

Description: Enables writing to flash memory for all OpenCore-managed NVRAM system variables.

Note: This value should be disabled on most types of firmware but is left configurable to account for firmware that may have issues with volatile variable storage overflows or similar. Boot issues across multiple OSes can be observed on e.g. Lenovo Thinkpad T430 and T530 without this quirk. Apple variables related to Secure Boot and hibernation are exempt from this for security reasons. Furthermore, some OpenCore variables are exempt for different reasons, such as the boot log due to an available user option, and the TSC frequency due to timing issues. When toggling this option, a NVRAM reset may be required to ensure full functionality.

7. ForgeUefiSupport

Type: plist boolean

Failsafe: false

Description: Implement partial UEFI 2.x support on EFI 1.x firmware.

This setting allows running some software written for UEFI 2.x firmware, such as NVIDIA GOP Option ROMs, on hardware with older EFI 1.x firmware (e.g. MacPro5,1).

$8. \ {\tt IgnoreInvalidFlexRatio}$

Type: plist boolean

Failsafe: false

Description: Some types of firmware (such as APTIO IV) may contain invalid values in the MSR_FLEX_RATIO (0x194) MSR register. These values may cause macOS boot failures on Intel platforms.

Note: While the option is not expected to harm unaffected firmware, its use is recommended only when specifically required.

9. ReleaseUsbOwnership

 $\mathbf{Type}:$ plist boolean

Failsafe: false

Description: Attempt to detach USB controller ownership from the firmware driver. While most types of firmware manage to do this properly, or at least have an option for this, some do not. As a result, the operating system may freeze upon boot. Not recommended unless specifically required.

10. ReloadOptionRoms

Type: plist boolean Failsafe: false

Description: Query PCI devices and reload their Option ROMs if available.

For example, this option allows reloading NVIDIA GOP Option ROM on older Macs after the firmware version is upgraded via ForgeUefiSupport.

11. RequestBootVarRouting

 $\mathbf{Type} \colon \mathtt{plist} \ \mathtt{boolean}$

Failsafe: false

 $\textbf{Description} \colon \text{Request redirect of all Boot prefixed variables from $\tt EFI_GLOBAL_VARIABLE_GUID$ to $\tt Constraints of the prefixed variables from {\tt EFI_GLOBAL_VARIABLE_GUID$} and {\tt Constraints} are also constraints of the prefixed variables from {\tt EFI_GLOBAL_VARIABLE_GUID$} and {\tt Constraints} are also constraints of the prefixed variables from {\tt EFI_GLOBAL_VARIABLE_GUID$} and {\tt Constraints} are also constraints of {\tt Constraints} and {\tt Constraints} are also constraints of {\tt Constraints} and {\tt Constraints} are also constraints are also constraints and {\tt Constraints} are also constraints and {\tt Constraints} are also constraints are also constraints and {\tt Constraints} are also constraints are also constraints are also constraints are also constraints and {\tt Constraints} are also constraints ar$

OC VENDOR VARIABLE GUID.

This quirk requires OC_FIRMWARE_RUNTIME protocol implemented in OpenRuntime.efi. The quirk lets default boot entry preservation at times when the firmware deletes incompatible boot entries. In summary, this quirk is required to reliably use the Startup Disk preference pane in firmware that is not compatible with macOS boot entries by design.

By redirecting Boot prefixed variables to a separate GUID namespace with the help of RequestBootVarRouting quirk we achieve multiple goals:

- Operating systems are jailed and only controlled by OpenCore boot environment to enhance security.
- Operating systems do not mess with OpenCore boot priority, and guarantee fluent updates and hibernation wakes for cases that require reboots with OpenCore in the middle.
- Potentially incompatible boot entries, such as macOS entries, are not deleted or corrupted in any way.

12. ResizeGpuBars

Type: plist integer

Failsafe: -1

Description: Configure GPU PCI BAR sizes.

This quirk sets GPU PCI BAR sizes as specified or chooses the largest available below the ResizeGpuBars value. The specified value follows PCI Resizable BAR spec. Use 0 for 1 MB, 1 for 2 MB, 2 for 4 MB, and so on up to 19 for 512 GB.

Resizable BAR technology allows to ease PCI device programming by mapping a configurable memory region, BAR, into CPU address space (e.g. VRAM to RAM) as opposed to a fixed memory region. This technology is necessary, because one cannot map the largest memory region by default, for the reasons of backwards compatibility with older hardware not supporting 64-bit BARs. Consequentially devices of the last decade use BARs up to 256 MB by default (4 remaining bits are used by other data) but generally allow resizing them to both smaller and larger powers of two (e.g. from 1 MB up to VRAM size).

Operating systems targeting x86 platforms generally do not control PCI address space, letting UEFI firmware decide on the BAR addresses and sizes. This illicit practice resulted in Resizable BAR technology being unused up until 2020 despite being standardised in 2008 and becoming widely available in the hardware soon after.

Modern UEFI firmware allow the use of Resizable BAR technology but generally restrict the configurable options to failsafe default (OFF) and maximum available (ON). This quirk allows to fine-tune this value for testing and development purposes.

Consider a GPU with 2 BARs:

- BARO supports sizes from 256 MB to 8 GB. Its value is 4 GB.
- $\bullet\,$ BAR1 supports sizes from 2 MB to 256 MB. Its value is 256 MB.

Example 1: Setting ResizeGpuBars to 1 GB will change BAR0 to 1 GB and leave BAR1 unchanged.

Example 2: Setting ResizeGpuBars to 1 MB will change BARO to 256 MB and BARO to 2 MB.

Example 3: Setting ResizeGpuBars to 16 GB will change BARO to 8 GB and leave BAR1 unchanged.

Note 1: This quirk shall not be used to workaround macOS limitation to address BARs over 1 GB. ResizeAppleGpuBars should be used instead.

Note 2: While this quirk can increase GPU PCI BAR sizes, this will not work on most firmware as is, because the quirk does not relocate BARs in memory, and they will likely overlap. In most cases it is best to either update the firmware to the latest version or customise it with a specialised driver like ReBarUEFI.

13. ResizeUsePciRbIo

Type: plist boolean

Failsafe: false

 $\textbf{Description} : \ Use \ PciRootBridgeIo \ for \ \texttt{ResizeGpuBars} \ and \ \texttt{ResizeAppleGpuBars}$

The quirk makes ResizeGpuBars and ResizeAppleGpuBars use PciRootBridgeIo instead of PciIo. This is needed on systems with a buggy PciIo implementation where trying to configure Resizable BAR results in Capability I/O Error. Typically this is required on older systems which have been modified with ReBarUEFI.

14. ShimRetainProtocol

Type: plist boolean

Failsafe: false

Description: Request Linux Shim to keep protocol installed for subsequent image loads.

This option is only required if chaining OpenCore from Shim. It must be set in order to allow OpenCore to launch items which are verified by certificates present in Shim, but not in the system Secure Boot database.

15. TscSyncTimeout

Type: plist integer

Failsafe: 0

Description: Attempts to perform TSC synchronisation with a specified timeout.

The primary purpose of this quirk is to enable early bootstrap TSC synchronisation on some server and laptop models when running a debug XNU kernel. For the debug kernel the TSC needs to be kept in sync across the cores before any kext could kick in rendering all other solutions problematic. The timeout is specified in microseconds and depends on the amount of cores present on the platform, the recommended starting value is 500000.

This is an experimental quirk, which should only be used for the aforementioned problem. In all other cases, the quirk may render the operating system unstable and is not recommended. The recommended solution in the other cases is to install a kernel extension such as VoodooTSCSync, TSCAdjustReset, or CpuTscSync (a more specialised variant of VoodooTSCSync for newer laptops).

Note: This quirk cannot replace the kernel extension because it cannot operate in ACPI S3 (sleep wake) mode and because the UEFI firmware only provides very limited multicore support which prevents precise updates of the MSR registers.

16. UnblockFsConnect

Type: plist boolean

Failsafe: false

Description: Some types of firmware block partition handles by opening them in By Driver mode, resulting in an inability to install File System protocols.

Note: This quirk is useful in cases where unsuccessful drive detection results in an absence of boot entries.

11.21 ReservedMemory Properties

1. Address

Type: plist integer

Failsafe: 0

Description: Start address of the reserved memory region, which should be allocated as reserved effectively marking the memory of this type inaccessible to the operating system.

The addresses written here must be part of the memory map, have a EfiConventional Memory type, and be page-aligned (4 KBs).

Note: Some types of firmware may not allocate memory areas used by S3 (sleep) and S4 (hibernation) code unless CSM is enabled causing wake failures. After comparing the memory maps with CSM disabled and enabled, these areas can be found in the lower memory and can be fixed up by doing the reservation. Refer to the Sample.plist file for details.

2. Comment

Type: plist string Failsafe: Empty

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

3. Enabled

Type: plist boolean Failsafe: false

Description: This region will not be reserved unless set to true.

4. Size

Type: plist integer

Failsafe: 0

Description: Size of the reserved memory region, must be page-aligned (4 KBs).

5. Type

Type: plist string Failsafe: Reserved

Description: Memory region type matching the UEFI specification memory descriptor types. Mapping:

- Reserved EfiReservedMemoryType
- LoaderCode EfiLoaderCode
- LoaderData EfiLoaderData
- BootServiceCode EfiBootServicesCode
- BootServiceData EfiBootServicesData
- RuntimeCode EfiRuntimeServicesCode
- $\bullet \ \ {\tt RuntimeData} {\tt EfiRuntimeServicesData}$
- Available EfiConventionalMemory
- $\bullet \ \ {\tt Persistent} {\tt EfiPersistentMemory}$
- $\bullet \ \ {\tt Unusable Memory} \ -\! \ {\tt EfiUnusable Memory}$
- ACPIReclaimMemory EfiACPIReclaimMemory
- ACPIMemoryNVS EfiACPIMemoryNVS
- MemoryMappedIO EfiMemoryMappedIO
- MemoryMappedIOPortSpace EfiMemoryMappedIOPortSpace
- PalCode EfiPalCode

12 Troubleshooting

12.1 Legacy Apple OS

Older operating systems may be more complicated to install, but are sometimes necessary for various reasons. While a compatible board identifier and CPUID are the obvious requirements for proper functioning of an older operating system, there are many other less obvious things to consider. This section covers a common set of issues relevant to installing older macOS operating systems.

While newer operating systems can be downloaded over the internet, older operating systems did not have installation media for every minor release. For compatible distributions of such, download a device-specific image and modify it if necessary. Visit this archived Apple Support article for a list of the bundled device-specific builds for legacy operating systems. However, as this may not always be accurate, the latest versions are listed below.

12.1.1 OS X 10.8 and 10.9

- Disk images on these systems use the Apple Partitioning Scheme and require the OpenPartitionDxe driver to run DMG recovery and installation (included in OpenDuet). It is possible to set DmgLoading to Disabled to run the recovery without DMG loading avoiding the need for OpenPartitionDxe.
- Cached kernel images often do not contain family drivers for networking (IONetworkingFamily) or audio (IOAudioFamily) requiring the use of Force loading in order to inject networking or audio drivers.

12.1.2 Mac OS X 10.7

- All previous issues apply.
- SSSE3 support (not to be confused with SSE3 support) is a hard requirement for Mac OS X 10.7 kernel.
- Many kexts, including Lilu when 32-bit kernel is used and a lot of Lilu plugins, are unsupported on Mac OS X 10.7 and older as they require newer kernel APIs, which are not part of the Mac OS X 10.7 SDK.
- Prior to OS X 10.8 KASLR sliding is not supported, which will result in memory allocation failures on firmware that utilise lower memory for their own purposes. Refer to acidanthera/bugtracker#1125 for tracking.

12.1.3 Mac OS X 10.6

- All previous issues apply.
- SSSE3 support is a requirement for Mac OS X 10.6 kernel with 64-bit userspace enabled. This limitation can mostly be lifted by enabling the LegacyCommpage quirk.
- Last released installer images for Mac OS X 10.6 are Mac OS X 10.6.7 builds 10J3250 (for MacBookPro8,x) and 10J4139 (for iMac12,x), without Xcode). These images are limited to their target model identifiers and have no -no_compat_check boot argument support. Modified images (with ACDT suffix) without model restrictions can be found here (MEGA Mirror), assuming Mac OS X 10.6 is legally owned. Refer to the DIGEST.txt file for details. Note that these are the earliest tested versions of Mac OS X 10.6 with OpenCore.

Model checking may also be erased by editing OSInstall.mpkg with e.g. Flat Package Editor by making Distribution script to always return true in hwbeModelCheck function. Since updating the only file in the image and not corrupting other files can be difficult and may cause slow booting due to kernel cache date changes, it is recommended to script image rebuilding as shown below:

```
#!/bin/bash
# Original.dmg is original image, OSInstall.mpkg is patched package
mkdir RO
hdiutil mount Original.dmg -noverify -noautoopen -noautoopenrw -noautofsck -mountpoint RO
cp RO/.DS_Store DS_STORE
hdiutil detach RO -force
rm -rf RO
hdiutil convert Original.dmg -format UDRW -o ReadWrite.dmg
mkdir RW
xattr -c OSInstall.mpkg
```

hdiutil mount ReadWrite.dmg -noverify -noautoopen -noautoopenrw -noautofsck -mountpoint RW cp OSInstall.mpkg RW/System/Installation/Packages/OSInstall.mpkg killall Finder fseventsd rm -rf RW/.fseventsd cp DS_STORE RW/.DS_Store hdiutil detach RW -force rm -rf DS_STORE RW hdiutil convert ReadWrite.dmg -format UDZO -o ReadOnly.dmg

12.1.4 Mac OS X 10.5

- All previous issues apply.
- This macOS version does not support x86_64 kernel and requires i386 kernel extensions and patches.
- This macOS version uses the first (V1) version of prelinkedkernel, which has kext symbol tables corrupted by the kext tools. This nuance renders prelinkedkernel kext injection impossible in OpenCore. Mkext kext injection will still work without noticeable performance drain and will be chosen automatically when KernelCache is set to Auto.
- Last released installer image for Mac OS X 10.5 is Mac OS X 10.5.7 build 9J3050 (for MacBookPro5,3). Unlike the others, this image is not limited to the target model identifiers and can be used as is. The original 9J3050 image can be found here (MEGA Mirror), assuming Mac OS X 10.5 is legally owned. Refer to the DIGEST.txt file for details. Note that this is the earliest tested version of Mac OS X 10.5 with OpenCore.

12.1.5 Mac OS X 10.4

- All previous issues apply.
- This macOS version has a hard requirement to access all the optional packages on the second DVD disk installation media, requiring either two disks or USB media installation.
- Last released installer images for Mac OS X 10.4 are Mac OS X 10.4.10 builds 8R4061a (for MacBookPro3,1) and 8R4088 (for iMac7,1)). These images are limited to their target model identifiers as on newer macOS versions. Modified 8R4088 images (with ACDT suffix) without model restrictions can be found here (MEGA Mirror), assuming Mac OS X 10.4 is legally owned. Refer to the DIGEST.txt file for details. Note that these are the earliest tested versions of Mac OS X 10.4 with OpenCore.

12.2 UEFI Secure Boot

OpenCore is designed to provide a secure boot chain between firmware and operating system. On most x86 platforms trusted loading is implemented via UEFI Secure Boot model. Not only OpenCore fully supports this model, but it also extends its capabilities to ensure sealed configuration via vaulting and provide trusted loading to the operating systems using custom verification, such as Apple Secure Boot. Proper secure boot chain requires several steps and careful configuration of certain settings as explained below:

- 1. Enable Apple Secure Boot by setting SecureBootModel to run macOS. Note, that not every macOS is compatible with Apple Secure Boot and there are several other restrictions as explained in Apple Secure Boot section.
- 2. Disable DMG loading by setting DmgLoading to Disabled if users have concerns of loading old vulnerable DMG recoveries. This is **not** required, but recommended. For the actual tradeoffs see the details in DMG loading section
- 3. Make sure that APFS JumpStart functionality restricts the loading of old vulnerable drivers by setting MinDate and MinVersion to 0. More details are provided in APFS JumpStart section. An alternative is to install apfs.efi driver manually.
- 4. Make sure that Force driver loading is not needed and all the operating systems are still bootable.
- 5. Make sure that ScanPolicy restricts loading from undesired devices. It is a good idea to prohibit all removable drivers or unknown filesystems.

- 6. Sign all the installed drivers and tools with the private key. Do not sign tools that provide administrative access to the computer, such as UEFI Shell.
- 7. Vault the configuration as explained Vaulting section.
- 8. Sign all OpenCore binaries (BOOTX64.efi, BOOTIa32.efi, OpenCore.efi, custom launchers) used on this system with the same private key.
- 9. Sign all third-party operating system (not made by Microsoft or Apple) bootloaders if needed. For Linux there is an option to install a user built, user signed Shim bootloader giving SBAT and MOK integration, as explained in the /Utilities/ShimUtils directory of OpenCore source or releases.
- 10. Enable UEFI Secure Boot in firmware preferences and install the certificate with a private key. Details on how to generate a certificate can be found in various articles, such as this one, and are out of the scope of this document. If Windows is needed one will also need to add the Microsoft Windows Production CA 2011. To launch option ROMs or to use signed Linux drivers if not using a user build of Shim, Microsoft UEFI Driver Signing CA will also be needed.
- 11. Password-protect changing firmware settings to ensure that UEFI Secure Boot cannot be disabled without the user's knowledge.

12.3 Windows support

Can I install Windows?

While no official Windows support is provided, 64-bit UEFI Windows installations (Windows 8 and above) prepared with Boot Camp are supposed to work. Third-party UEFI installations as well as systems partially supporting UEFI boot, such as Windows 7, might work with some extra precautions. Things to consider:

- MBR (Master Boot Record) installations are legacy and are only supported with the OpenLegacyBoot driver on legacy systems.
- All the modifications applied (to ACPI, NVRAM, SMBIOS, etc.) are supposed to be operating system agnostic, i.e. apply equally regardless of the OS booted. This enables Boot Camp software experience on Windows.
- macOS requires the first partition to be EFI System Partition, and does not support the default Windows layout. While OpenCore does have a workaround for this, it is highly recommend not to rely on it and install properly.
- Windows may need to be reactivated. To avoid it consider setting SystemUUID to the original firmware UUID. Be aware that it may be invalid on old firmware, i.e., not random. If there still are issues, consider using HWID or KMS38 license or making the use Custom UpdateSMBIOSMode. Other nuances of Windows activation are out of the scope of this document and can be found online.

What additional software do I need?

To enable operating system switching and install relevant drivers in the majority of cases Windows support software from Boot Camp is required. For simplicity of the download process or when configuring an already installed Windows version a third-party utility, Brigadier, can be used successfully. Note, that 7-Zip may be downloaded and installed prior to using Brigadier.

Remember to always use the latest version of Windows support software from Boot Camp, as versions prior to 6.1 do not support APFS, and thus will not function correctly. To download newest software pass most recent Mac model to Brigadier, for example ./brigadier.exe -m iMac19,1. To install Boot Camp on an unsupported Mac model afterwards run PowerShell as Administrator and enter msiexec /i BootCamp.msi. If there is a previous version of Boot Camp installed it should be removed first by running msiexec /x BootCamp.msi command. BootCamp.msi file is located in BootCamp/Drivers/Apple directory and can be reached through Windows Explorer.

While Windows support software from Boot Camp solves most of compatibility problems, the rest may still have to be addressed manually:

- To invert mouse wheel scroll direction FlipFlopWheel must be set to 1 as explained on SuperUser.
- RealTimeIsUniversal must be set to 1 to avoid time desync between Windows and macOS as explained on SuperUser (this is typically not required).

• To access Apple filesystems such as HFS+ and APFS, separate software may need to be installed. Some of the known utilities are: Apple HFS+ driver (workaround for Windows 10), HFSExplorer, MacDrive, Paragon APFS, Paragon HFS+, TransMac, etc. Remember to never ever attempt to modify Apple file systems from Windows as this often leads to irrecoverable data loss.

Why do I see Basic data partition in the Boot Camp Startup Disk control panel?

The Boot Camp control panel uses the GPT partition table to obtain each boot option name. After installing Windows separately, the partition has to be relabelled manually. This can be done with many utilities including the open-source gdisk utility. Reference example:

```
PS C:\gdisk> .\gdisk64.exe \\.\physicaldrive0
GPT fdisk (gdisk) version 1.0.4

Command (? for help): p
Disk \\.\physicaldrive0: 419430400 sectors, 200.0 GiB
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 \.\ physicaldriveO. Disk synchronization succeeded! The computer should now use the new partition table. The operation has completed successfully.
```

Listing 4: Relabeling Windows volume

How do I 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 disrupt certain macOS functionality, including the 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, version 7.3 or newer is needed, and Debugging the XNU Kernel with IDA Pro may also help.

To obtain the log during boot serial port debugging can be used. Serial port debugging is enabled in Target, e.g. 0xB for onscreen with serial. To initialise serial within OpenCore use Init configuration option under Misc->Serial with other values properly set. For macOS the best choice is 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, motherboard "TX" must be connected 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 debug=0x8 boot argument is needed.

12.5 Tips and Tricks

1. How do I debug boot failures?

Obtaining the actual error message is usually adequate. For this, ensure that:

- A DEBUG or NOOPT version of OpenCore is used.
- Logging is enabled (1) and shown onscreen (2): $Misc \rightarrow Debug \rightarrow Target = 3$.
- Logged messages from at least DEBUG_ERROR (0x80000000), DEBUG_WARN (0x00000002), and DEBUG_INFO (0x000000040) levels are visible onscreen: $Misc \rightarrow Debug \rightarrow DisplayLevel = 0x80000042$.
- Critical error messages, such as DEBUG_ERROR, stop booting: Misc → Security → HaltLevel = 0x80000000.
- Watch Dog is disabled to prevent automatic reboot: $\texttt{Misc} \to \texttt{Debug} \to \texttt{DisableWatchDog} = \texttt{true}$.
- Boot Picker (entry selector) is enabled: $\mathtt{Misc} \to \mathtt{Boot} \to \mathtt{ShowPicker} = \mathtt{true}$.

If there is no obvious error, check the available workarounds in the 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 do I debug macOS boot failures?

- Refer to boot-args values such as debug=0x100, keepsyms=1, -v, and similar.
- Do not forget about AppleDebug and ApplePanic properties.
- For macOS to correctly recognise and set up serial ports, the CustomPciSerialDevice quirk may be enabled when a PCI serial port card is installed.
- Take care of Booter, Kernel, and UEFI quirks.
- Consider using serial port to inspect early kernel boot failures. For this debug=0x108, serial=5, and msgbuf=1048576 boot arguments are needed. Refer to the patches in Sample.plist when dying before serial init.
- Always read the logs carefully.

3. How do I 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.

4. How do I 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 BOOTx64.EFI as a primary boot option limits this functionality in addition to several types of firmware deleting incompatible boot options, potentially including those created by macOS, users are strongly encouraged to use the RequestBootVarRouting quirk, which will preserve the selection made in the operating system within the OpenCore variable space. Note, that RequestBootVarRouting requires a separate driver for functioning.

5. 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 the OpenCore picker and choose the entry, it will have a (dmg) suffix. Custom name may be created by providing .contentDetails file.

To download recovery online macrecovery.py can be used.

For offline installation refer to How to create a bootable installer for macOS article. Apart from App Store and softwareupdate utility there also are third-party utilities to download an offline image.

6. 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.

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

Yes. Virtual machines and most relatively modern Mac models, as far back as the MacPro3,1, are fully supported. While specific detail relevant to Mac hardware is often limited, some ongoing instructions can be found on MacRumors.com.

8. Why must Find&Replace patches be equal in size?

For machine code (x86 code) it is not possible to do differently sized 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 or in the ACPI section of this document.

9. How can I decide which Booter quirks to use?

These quirks originate from AptioMemoryFix driver but provide a wider set of changes specific to modern systems. Note, that OpenRuntime driver is required for most configurations. To get a configuration similar to AptioMemoryFix the following set of quirks should be enabled:

- ProvideConsoleGop (UEFI quirk)
- AvoidRuntimeDefrag
- DiscardHibernateMap
- EnableSafeModeSlide
- EnableWriteUnprotector
- ForceExitBootServices
- ProtectMemoryRegions
- ProvideCustomSlide
- RebuildAppleMemoryMap
- SetupVirtualMap

However, as of today, such set is strongly discouraged as some of these quirks are not necessary to be enabled or need additional quirks. For example, DevirtualiseMmio and ProtectUefiServices are often required, while DiscardHibernateMap and ForceExitBootServices are rarely necessary.

Unfortunately for some quirks such as RebuildAppleMemoryMap, EnableWriteUnprotector, ProtectMemoryRegions, SetupVirtualMap, and SyncRuntimePermissions there is no definite approach even on similar systems, so trying all their combinations may be required for optimal setup. Refer to individual quirk descriptions in this document for details.