OpenCore

Reference Manual (0.9.2.3)

[2023.06.11]

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).

   ~~*Note 1:* 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 iGPU is enabled 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.

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

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 argument `--enable-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.
`EnableGop{Direct}*`	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.
`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.
`OpenRuntime*`	OpenCore plugin implementing `OC_FIRMWARE_RUNTIME` protocol.
`OpenLinuxBoot*`	OpenCore plugin implementing `OC_BOOT_ENTRY_PROTOCOL` to allow direct detection and booting of Linux distributiuons from OpenCore, without chainloading via GRUB.
`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.

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.
`MemTest86`	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.
`PavpProvision`	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).

11.8.1 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 conjuction 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.

11.9 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`.

renderer behaves. Since this is not required by the UEFI specification, behaviour of the system `ConsoleControl` protocol, when it exists, may vary.

2. `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.

3. `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.

4. `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.

5. `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.

   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`.

6. `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.

7. `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.

8. `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.

9. `GopBurstMode`
   Type: `plist boolean`
   Failsafe: `false`

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.

11.18 Quirks Properties

1. `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.

2. `EnableVectorAcceleration`
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.

3. `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.

4. `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.

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`