



OpenCore

Reference Manual (0.5.~~7~~.8)

[2020.04.13]

- `.VolumeIcon.icns` file at volume root for other filesystems.
- `<TOOL_NAME>.icns` file for Tools.

Volume icons can be set in Finder. [Note, that enabling this may result in external and internal icons to be indistinguishable.](#)

- `0x0002` — `OC_ATTR_USE_DISK_LABEL_FILE`, provides custom rendered titles for boot entries:
 - `.disk_label` (`.disk_label_2x`) file near bootloader for all filesystems.
 - `<TOOL_NAME.1b1` (`<TOOL_NAME.12x`) file near tool for Tools.

Prerendered labels can be generated via `disklabel` utility or `bless` command. When disabled or missing text labels (`.contentDetails` or `.disk_label.contentDetails`) are to be rendered instead.

- `0x0004` — `OC_ATTR_USE_GENERIC_LABEL_IMAGE`, provides predefined label images for boot entries without custom entries. May give less detail for the actual boot entry.
- `0x0008` — `OC_ATTR_USE_ALTERNATE_ICONS`, changes used icon set to an alternate one if it is supported. For example, this could make a use of old-style icons with a custom background colour.

6. PickerAudioAssist

Type: plist boolean

Failsafe: false

Description: Enable screen reader by default in boot picker.

For macOS bootloader screen reader preference is set in `preferences.efi` archive in `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 OpenCore boot picker and macOS bootloader FileVault 2 login window can also be done with `Command + F5` key combination.

Note: screen reader requires working audio support, see [UEFI Audio Properties](#) section for more details.

7. PollAppleHotKeys

Type: plist boolean

Failsafe: false

Description: Enable modifier hotkey handling in boot picker.

In addition to `action hotkeys`, which are partially described in `PickerMode` section and are normally handled by Apple BDS, there exist modifier keys, which are handled by operating system bootloader, namely `boot.efi`. These keys allow to change operating system behaviour by providing different boot modes.

On some firmwares it may be problematic to use modifier keys due to driver incompatibilities. To workaround this problem this option allows registering select hotkeys in a more permissive manner from within boot picker. Such extensions include the support of tapping on keys in addition to holding and pressing `Shift` along with other keys instead of just `Shift` alone, which is not detectible 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` — safe mode.

8. ShowPicker

Type: plist boolean

Failsafe: false

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

9. TakeoffDelay

Type: plist integer, 32 bit

Failsafe: 0

Description: Delay in microseconds performed before handling picker startup and `action hotkeys`.

Introducing a delay may give extra time to hold the right `action hotkey` sequence to e.g. boot to recovery mode. On some platforms setting this option to at least 5000–10000 microseconds may be necessary to access `action hotkeys` at all due to the nature of the keyboard driver.

9 NVRAM

9.1 Introduction

Has `plist dict` type and allows to set volatile UEFI variables commonly referred as NVRAM variables. Refer to `man nvram` for more details. macOS extensively uses NVRAM variables for OS — Bootloader — Firmware intercommunication, and thus supplying several NVRAM is required for proper macOS functioning.

Each NVRAM variable consists of its name, value, attributes (refer to UEFI specification), and its GUID, representing which ‘section’ NVRAM variable belongs to. macOS uses 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)
- 8BE4DF61-93CA-11D2-AA0D-00E098032B8C (EFI_GLOBAL_VARIABLE_GUID)
- 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102 (OC_VENDOR_VARIABLE_GUID)

Note: Some of the variables may be added by PlatformNVRAM or Generic subsections of PlatformInfo section. Please ensure that variables of this section never collide with them, as behaviour is undefined otherwise.

For proper macOS functioning it is often required to use `OC_FIRMWARE_RUNTIME` protocol implementation currently offered as a part of `OpenRuntime` driver. While it brings any benefits, there are certain limitations which arise depending on the use.

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 have to be aware of `OC_FIRMWARE_RUNTIME` logic.
2. Assigned NVRAM variables are not always allowed to exceed 512 bytes.
This is true for `Boot`-prefixed variables when `RequestBootVarFallback` is used, and for overwriting volatile variables with non-volatile on UEFI 2.8 non-conformant firmwares.

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 metadata` format. GUIDs must be provided in canonic string format in upper or lower case (e.g. 8BE4DF61-93CA-11D2-AA0D-00E098032B8C).

Created variables get `EFI_VARIABLE_BOOTSERVICE_ACCESS` and `EFI_VARIABLE_RUNTIME_ACCESS` attributes set. Variables will only be set if not present ~~and not blocked. To overwrite a variable add it to or blocked. I.e. to overwrite an existing variable value add the variable name to the~~ `Block` section. This approach enables to provide default values till the operating system takes the lead.

Note: If `plist` key does not conform to GUID format, behaviour is undefined.

2. Block

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. LegacyEnable

Type: `plist boolean`

Failsafe: `false`

Description: Enables loading of NVRAM variable file named `nvram.plist` from EFI volume root.

This file must have 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`.

Variable loading happens prior to `Block` (and `Add`) phases. Unless `LegacyOverwrite` is enabled, it will not overwrite any existing variable. Variables allowed to be set must be specified in `LegacySchema`. Third-party scripts may be used to create `nvram.plist` file. An example of such script can be found in `Utilities`. The use of

- **Overwrite** — Overwrite existing gEfiSmbiosTableGuid and gEfiSmbiosTable3Guid data if it fits new size. Abort with unspecified state otherwise.
- **Custom** — Write ~~first SMBIOS table~~ (SMBIOS tables (~~gEfiSmbiosTableGuid~~gEfiSmbios(3)TableGuid) to ~~gOeCustomSmbiosTableGuid~~gOeCustomSmbios(3)TableGuid to workaround firmwares overwriting SMBIOS contents at ExitBootServices. Otherwise equivalent to **Create**. Requires patching AppleSmbios.kext and AppleACPIPlatform.kext to read from another GUID: "EB9D2D31" - "EB9D2D35" (in ASCII), done automatically by CustomSMBIOSGuid quirk.

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

6. Generic

Type: plist dictionary

Optional: When Automatic is false

Description: Update all fields. This section is read only when Automatic is active.

7. DataHub

Type: plist dictionary

Optional: When Automatic is true

Description: Update Data Hub fields. This section is read only when Automatic is not active.

8. PlatformNVRAM

Type: plist dictionary

Optional: When Automatic is true

Description: Update platform NVRAM fields. This section is read only when Automatic is not active.

9. SMBIOS

Type: plist dictionary

Optional: When Automatic is true

Description: Update SMBIOS fields. This section is read only when Automatic is not active.

10.2 Generic Properties

1. SpoofVendor

Type: plist boolean

Failsafe: false

Description: Sets SMBIOS vendor fields to Acidanthera.

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

2. AdviseWindows

Type: plist boolean

Failsafe: false

Description: Forces Windows support in **FirmwareFeatures**.

Added bits to **FirmwareFeatures**:

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

3. SystemProductName

Type: plist string

Failsafe: MacPro6,1

Description: Refer to SMBIOS SystemProductName.

4. SystemSerialNumber

Type: plist string

Failsafe: OPENCORE_SN1

Description: Refer to SMBIOS SystemSerialNumber.

12 Troubleshooting

12.1 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, like Windows 7, might work with some extra precautions. Things to keep in mind:

- MBR (Master Boot Record) installations are legacy and will not be supported.
- To install Windows, macOS, and OpenCore on the same drive you can specify Windows bootloader path (`\EFI\Microsoft\Boot\bootmgfw.efi`) in `BlessOverride` section.
- 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 warned, on old firmwares it may be invalid, i.e. not random. In case you still have 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 you will need Windows support software from Boot Camp. 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 you may have to download and install 7-Zip 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`. In case you already have a previous version of Boot Camp installed you will have to remove it 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, sometimes you may have to address some of them 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 one is usually not needed).
- To access Apple filesystems like HFS and APFS separate software may need to be installed. Some of the known tools are: Apple HFS+ driver (hack 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 Boot Camp Startup Disk control panel?

Boot Camp control panel uses GPT partition table to obtain each boot option name. After installing Windows separately you will have to relabel the partition manually. This can be done with many tools including 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
```