OpenCore

Reference Manual (~~0.9.9~~1.0.0)

[2024.04.17]

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 ~~macOS~~ Mac OS X 10.7 or older)

   This quirk typically requires `ProvideCustomSlide` and `AvoidRuntimeDefrag` to be enabled ~~and typically also requires enabling `AvoidRuntimeDefrag`~~ 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.

8. `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 `CR0` 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.

9. `FixupAppleEfiImages`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Fix errors in early Mac OS X boot.efi images.

   Modern secure PE loaders will refuse to load `boot.efi` images from ~~macOS~~ Mac OS X 10.4 to macOS 10.12 due to these files containing `W^X` errors (in all versions) and illegal overlapping sections (in 10.4 and 10.5 32-bit versions only).

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

   Pre-processing in memory is incompatible with 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 - 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.

   Note 1: The quirk is never applied during the Apple secure boot path for newer macOS. The Apple secure boot path 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: This quirk is needed for ~~macOS~~ Mac OS X 10.4 to macOS 10.12 (and higher, if Apple secure boot is not enabled), but only when the firmware itself includes a modern, more secure PE COFF image loader. This applies to current builds of OpenDuet, and to OVMF if built from audk source code.

10. `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.

11. `ForceExitBootServices`
    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.

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.

17. `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 `EnableWriteUnprotector` on firmware supporting Memory Attribute Tables (MAT). This quirk is typically unnecessary when using `OpenDuetPkg` but may be required to boot ~~macOS~~ Mac OS X 10.6, and earlier, for reasons that are as yet unclear.

18. `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:

    - `BAR0` 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 `BAR0` to 1 GB and leave `BAR1` unchanged.
    Example 2: Setting `ResizeAppleGpuBars` to 1 MB will change `BAR0` to 256 MB and `BAR0` 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.

19. `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

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 ~~macOS~~ 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 ~~macOS~~ 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 (`i386`, `i386-user32`, `x86_64`) when available.

   On ~~macOS~~ 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 ~~macOS~~Mac OS X 10.6. This behaviour corresponds to the `-legacy` kernel boot argument.
     - This option is unavailable on ~~macOS~~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 ~~macOS~~Mac OS X 10.7 (where certain board identifiers are treated as `i386` only machines), and ~~macOS~~Mac OS X 10.5 or earlier (where `x86_64` is not supported by the macOS kernel), ~~macOS~~Mac OS X 10.6 is very special. The architecture choice on ~~macOS~~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 ~~macOS~~Mac OS X 10.6 falls back to the `x86_64` architecture when it is supported by the board, as on ~~macOS~~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 ~~macOS~~ 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)

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` — macOS 11 styled icon set.
- `Acidanthera\Syrah` — ~~macOS~~ OS X 10.10 styled icon set.
- `Acidanthera\Chardonnay` — ~~macOS~~ 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`.

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`
   Failsafe: `0`
   Description: Delay in microseconds executed after every printed line visible onscreen (i.e. console).

5. `DisplayLevel`
   Type: `plist integer`, 64 bit

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.
`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.
`OpenRuntime*`	OpenCore plugin implementing `OC_FIRMWARE_RUNTIME` protocol.
`OpenLegacyBoot*`	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 CSM.

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 ~~installion~~ 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 ~~ealier~~ 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 ~~paritition~~ 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.

11.11 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`).

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 ~~macOS~~ 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`
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.

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 ~~macOS~~ 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 ~~macOS~~ Mac OS X 10.7

- All previous issues apply.
- `SSSE3` support (not to be confused with `SSE3` support) is a hard requirement for ~~macOS~~ 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 ~~macOS~~Mac OS X 10.7 and older as they require newer kernel APIs, which are not part of the ~~macOS~~Mac OS X 10.7 SDK.
- Prior to ~~macOS~~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 ~~macOS~~ Mac OS X 10.6

- All previous issues apply.
- `SSSE3` support is a requirement for ~~macOS~~ 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 ~~macOS~~Mac OS X 10.6 are ~~macOS~~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 ~~macOS~~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 ~~macOS~~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 ~~macOS~~ 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 ~~macOS~~Mac OS X 10.5 is ~~macOS~~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 ~~macOS~~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 ~~macOS~~Mac OS X 10.5 with OpenCore.

12.1.5 ~~macOS~~ 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 ~~macOS~~Mac OS X 10.4 are ~~macOS~~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 ~~macOS~~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 ~~macOS~~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.