OpenCore

Reference Manual (0.7.~~5~~.6)

[2021.11.05]

14. `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.

15. `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.

16. `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 10.6, and earlier, for reasons that are as yet unclear.

17. `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. ~~Use 0 for 1 MB, 1 for 2 MB, 2 for 4 MB, and so on up to 19 for 512 GB.~~ While Apple macOS supports a theoretical 1 GB maximum, ~~which is 10.~~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.

Note1: See `ResizeGpuBars` quirk for general GPU PCI BAR size configuration and more details about the technology.

~~*Note 2*: Certain GPU drivers do not support non-standard BAR sizes, causing sleep wake issues, for this reason for macOS it is recommended to use minimal supported BAR sizes, i.e. specify `0` (1 MB).~~

18. `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.

19. `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.

20. `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.

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.

10. `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. `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.

12. `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.

13. `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.

14. `UgaPassThrough`
Type: `plist boolean`
Failsafe: `false`
Description: Provide UGA protocol instances on top of GOP protocol instances.

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. `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.

11.15 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.

3. `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.

4. `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.

5. `ForceOcWriteFlash`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enables writing to flash memory for all ~~OpenCore~~ 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.