OpenCore

Reference Manual (0.8.~~1~~.2)

[2022.06.22]

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

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

7.5 Emulate Properties

1. `Cpuid1Data`
   Type: `plist data`, 16 bytes
   Failsafe: All zero
   Description: Sequence of `EAX`, `EBX`, `ECX`, `EDX` values to replace `CPUID (1)` call in XNU kernel.

   This property primarily meets three requirements:

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

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

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

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

   - Haswell-E (`0x0306F2`) to Haswell (`0x0306C3`):
     `Cpuid1Data`: `C3 06 03 00 00 00 00 00 00 00 00 00 00 00 00 00`
     `Cpuid1Mask`: `FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
   - Broadwell-E (`0x0406F1`) to Broadwell (`0x0306D4`):
     `Cpuid1Data`: `D4 06 03 00 00 00 00 00 00 00 00 00 00 00 00 00`
     `Cpuid1Mask`: `FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
   - Comet Lake U62 (`0x0A0660`) to Comet Lake U42 (`0x0806EC`):
     `Cpuid1Data`: `EC 06 08 00 00 00 00 00 00 00 00 00 00 00 00 00`
     `Cpuid1Mask`: `FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
   - Rocket Lake (`0x0A0670`) to Comet Lake (`0x0A0655`):
     `Cpuid1Data`: `55 06 0A 00 00 00 00 00 00 00 00 00 00 00 00 00`
     `Cpuid1Mask`: `FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`
   - Alder Lake (`0x090672`) to Comet Lake (`0x0A0655`):
     `Cpuid1Data`: `55 06 0A 00 00 00 00 00 00 00 00 00 00 00 00 00`
     `Cpuid1Mask`: `FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00`

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

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

2. `Cpuid1Mask`
   Type: `plist data`, 16 bytes
   Failsafe: All zero
   Description: Bit mask of active bits in `Cpuid1Data`.

   When each `Cpuid1Mask` bit is set to 0, the original CPU bit is used, otherwise set bits take the value of `Cpuid1Data`.

3. `DummyPowerManagement`
   Type: `plist boolean`
   Failsafe: `false`
   Requirement: 10.4-12
   Description: Disables `AppleIntelCpuPowerManagement`.

Failsafe: Empty
Description: Patches data on specified macOS version or older.

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

11. `MinKernel`
Type: `plist string`
Failsafe: Empty
Description: Patches data on specified macOS version or newer.

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

12. `Replace`
Type: `plist data`
Failsafe: Empty
Description: Replacement data of one or more bytes.

13. `ReplaceMask`
Type: `plist data`
Failsafe: Empty (Ignored)
Description: Data bitwise mask used during replacement. Allows fuzzy replacement by updating masked (set to non-zero) bits. Must be equal to `Replace` in size if set.

14. `Skip`
Type: `plist integer`
Failsafe: `0` (Do not skip any occurrences)
Description: Number of found occurrences to skip before replacements are applied.

7.8 Quirks Properties

1. `AppleCpuPmCfgLock`
Type: `plist boolean`
Failsafe: `false`
Requirement: 10.4-12
Description: Disables `PKG_CST_CONFIG_CONTROL` (`0xE2`) MSR modification in AppleIntelCPUPowerManagement.kext, commonly causing early kernel panic, when it is locked from writing.

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

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

(a) Download UEFITool and IFR-Extractor.
(b) Open the firmware image in UEFITool and find `CFG Lock` unicode string. If it is not present, the firmware may not have this option and the process should therefore be discontinued.
(c) Extract the `Setup.bin` PE32 Image Section (the UEFITool found) through the `Extract Body` menu option.
(d) Run IFR-Extractor on the extracted file (e.g. `./ifrextract Setup.bin Setup.txt`).
(e) Find `CFG Lock, VarStoreInfo (VarOffset/VarName):` in `Setup.txt` and remember the offset right after it (e.g. `0x123`).
(f) Download and run Modified GRUB Shell compiled by brainsucker or use a newer version by datasone.
(g) Enter `setup_var 0x123 0x00` command, where `0x123` should be replaced by the actual offset, and reboot.

Warning: Variable offsets are unique not only to each motherboard but even to its firmware version. Never ever try to use an offset without checking.

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

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

sometimes fails to wake up. For debug kernels `setpowerstate_panic=0` boot argument should be used, which is otherwise equivalent to this quirk.

19. `ProvideCurrentCpuInfo`
    Type: `plist boolean`
    Failsafe: `false`
    Requirement: 10.8 (10.14)
    Description: Provides current CPU info to the kernel.

    This quirk works differently depending on the CPU:

    - For Microsoft Hyper-V it provides the correct TSC and FSB values to the kernel, as well as disables CPU topology validation (10.8+).
    - For KVM and other hypervisors it provides precomputed MSR 35h values solving kernel panic with `-cpu host`.
    - For Intel CPUs it adds support for asymmetrical SMP systems (e.g. Intel Alder Lake) by patching core count to thread count along with the supplemental required changes (10.14+).

20. `SetApfsTrimTimeout`
    Type: `plist integer`
    Failsafe: `-1`
    Requirement: 10.14 (not required for older)
    Description: Set trim timeout in microseconds for APFS filesystems on SSDs.

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

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

    On several controllers, such as Samsung, where the deallocation process is relatively slow, this timeout can be reached very quickly. Essentially, it means that the level of fragmentation is high, thus macOS will attempt to trim the same lower blocks that have previously been deallocated, but never have enough time to deallocate higher blocks. The outcome is that trimming on such SSDs will be non-functional soon after installation, resulting in additional wear on the flash.

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

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

    Note 2: On macOS 12.0 and above, it is no longer possible to specify trim timeout. However, ~~it~~ 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.

21. `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.

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.

11.9 Properties

1. `APFS`
   Type: `plist dict`
   Failsafe: None
   Description: Provide APFS support as configured in the APFS Properties section below.

2. `Audio`
   Type: `plist dict`
   Failsafe: None
   Description: Configure audio backend support described in the `Audio Properties` section below.

   Unless documented otherwise (e.g. `ResetTrafficClass`) settings in this section are for UEFI audio support only (e.g. OpenCore generated boot chime and audio assist) and are unrelated to any configuration needed for OS audio support (e.g. `AppleALC`).

   UEFI audio support provides a way for upstream protocols to interact with the selected audio hardware and resources. All audio resources should reside in `\EFI\OC\Resources\Audio` directory. Currently the supported audio file formats are MP3 and WAVE PCM. While it is driver-dependent which audio stream format is supported, most common audio cards support 16-bit signed stereo audio at 44100 or 48000 Hz.

   Audio file path is determined by audio type, audio localisation, and audio path. Each filename looks as follows: `[audio type]_[audio localisation]_[audio path].[audio ext]`. For unlocalised files filename does not include the language code and looks as follows: `[audio type]_[audio path].[audio ext]`. Audio extension can either be `mp3` or `wav`.

   - Audio type can be `OCEFIAudio` for OpenCore audio files or `AXEFIAudio` for macOS bootloader audio files.
   - Audio localisation is a two letter language code (e.g. `en`) with an exception for Chinese, Spanish, and Portuguese. Refer to `APPLE_VOICE_OVER_LANGUAGE_CODE` definition for the list of all supported localisations.
   - Audio path is the base filename corresponding to a file identifier. For macOS bootloader audio paths refer to `APPLE_VOICE_OVER_AUDIO_FILE` definition. For OpenCore audio paths refer to `OC_VOICE_OVER_AUDIO_FILE` definition. The only exception is OpenCore boot chime file, which is `OCEFIAudio_VoiceOver_Boot.mp3`.