OpenCore

Reference Manual (0.8.3.4)

[2022.08.25]

Copyright ©2018-2022 vit9696

Failsafe: 0 (Search entire kext or kernel)
Description: Maximum number of bytes to search for.

9. 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 Replace in size if set.

10. MaxKernel
Type: plist string
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).

mode) are not allowed without explicit user authentication by a custom password. Currently, password and salt are hashed with 5000000 iterations of SHA-512.

Note: This functionality is still under development and is not ready for production environments.

7. `ExposeSensitiveData`
   Type: `plist integer`
   Failsafe: `0x6`
   Description: Sensitive data exposure bitmask (sum) to operating system.

   - `0x01` — Expose the printable booter path as a UEFI variable.
   - `0x02` — Expose the OpenCore version as a UEFI variable.
   - `0x04` — Expose the OpenCore version in the OpenCore picker menu title.
   - `0x08` — Expose OEM information as a set of UEFI variables.

   The exposed booter path points to OpenCore.efi or its booter depending on the load order. To obtain the booter path, use the following command in macOS:

   ```
   nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-path
   ```

   To use a booter path to mount a booter volume, use the following command in macOS:

   ```
   u=$(nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-path | sed 's/.*GPT,\([^,]*\),.*/\1/'); \
     if [ "$u" != "" ]; then sudo diskutil mount $u ; fi
   ```

   To obtain the current OpenCore version, use the following command in macOS:

   ```
   nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:opencore-version
   ```

   If the OpenCore version is not exposed the variable will contain `UNK-000-0000-00-00` sequence.

   To obtain OEM information, use the following commands in macOS:

   ```
   nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:oem-product # SMBIOS Type1 ProductName
   nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:oem-vendor  # SMBIOS Type2 Manufacturer
   nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:oem-board   # SMBIOS Type2 ProductName
   ```

8. `HaltLevel`
   Type: `plist integer`, 64 bit
   Failsafe: `0x80000000` (`DEBUG_ERROR`)
   Description: EDK II debug level bitmask (sum) causing CPU to halt (stop execution) after obtaining a message of `HaltLevel`. Possible values match `DisplayLevel` values.

9. `PasswordHash`
   Type: `plist data` 64 bytes
   Failsafe: all zero
   Description: Password hash used when ~~`EnabledPassword`~~`EnablePassword` is set.

10. `PasswordSalt`
    Type: `plist data`
    Failsafe: empty
    Description: Password salt used when ~~`EnabledPassword`~~`EnablePassword` is set.

11. `Vault`
    Type: `plist string`
    Failsafe: `Secure`
    Description: Enables the OpenCore vaulting mechanism.

    Valid values:

    - `Optional` — require nothing, no vault is enforced, insecure.
    - `Basic` — require `vault.plist` file present in `OC` directory. This provides basic filesystem integrity verification and may protect from unintentional filesystem corruption.

2. `Auxiliary`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Set to `true` to hide this entry when `HideAuxiliary` is also set to `true`. Press the `Spacebar` key to enter "Extended Mode" and display the entry when hidden.

3. `Comment`
   Type: `plist string`
   Failsafe: Empty
   Description: Arbitrary ASCII string used to provide a human readable reference for the entry. Whether this value is used is implementation defined.

4. `Enabled`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Set to `true` activate this entry.

5. `Flavour`
   Type: `plist string`
   Failsafe: `Auto`
   Description: Specify the content flavour for this entry. See `OC_ATTR_USE_FLAVOUR_ICON` flag for documentation.

6. `FullNvramAccess`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Disable `OpenRuntime` NVRAM protection during usage of a tool.

   This disables all of the NVRAM protections provided by `OpenRuntime.efi`, during the time a tool is in use. It should normally be avoided, but may be required for instance if a tool needs to access NVRAM directly without the redirections put in place by `RequestBootVarRouting`.

   Note: This option is only valid for `Tools` and cannot be specified for `Entries` (is always `false`).

7. `Name`
   Type: `plist string`
   Failsafe: Empty
   Description: Human readable entry name displayed in the OpenCore picker.

8. `Path`
   Type: `plist string`
   Failsafe: Empty
   Description: Entry location depending on entry type.

   - `Entries` specify external boot options, and therefore take device paths in the `Path` key. Care should be exercised as these values are not checked. Example: `PciRoot(0x0)/Pci(0x1,0x1)/.../\EFI\COOL.EFI`
   - `Tools` specify internal boot options, which are part of the bootloader vault, and therefore take file paths relative to the `OC/Tools` directory. Example: `OpenShell.efi`.

9. `RealPath`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Pass full path to the tool when launching.

   This should typically be disabled as passing the tool directory may be unsafe with tools that accidentally attempt to access files without checking their integrity. Reasons to enable this property may include cases where tools cannot work without external files or may need them for enhanced functionality such as `memtest86` (for logging and configuration), or `Shell` (for automatic script execution).

   Note: This ~~property~~ option is only valid for `Tools` and cannot be specified for `Entries` (is always `true`).

10. `TextMode`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Run the entry in text mode instead of graphics mode.

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 restarts will be to this entry).

Note: 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.

11.10 Properties

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

2. `AppleInput`
   Type: `plist dict`
   ~~**Failsafe**: None~~**Description**: Configure the re-implementation of the Apple Event protocol described in the AppleInput Properties section below.

3. `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`.

Audio localisation is determined separately for macOS bootloader and OpenCore. For macOS bootloader it is set in `preferences.efires` archive in `systemLanguage.utf8` file and is controlled by the operating system. For OpenCore the value of `prev-lang:kbd` variable is used. When native audio localisation of a particular file is missing, English language (`en`) localisation is used. Sample audio files can be found in OcBinaryData repository.

4. `ConnectDrivers`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Perform UEFI controller connection after driver loading.

   This option is useful for loading drivers following UEFI driver model as they may not start by themselves. Examples of such drivers are filesystem or audio drivers. While effective, this option may not be necessary for drivers performing automatic connection, and may slightly slowdown the boot.

   Note: Some types of firmware, particularly those made by Apple, only connect the boot drive to speed up the boot process. Enable this option to be able to see all the boot options when running multiple drives.

5. `Drivers`
   Type: `plist array`
   Failsafe: Empty
   Description: Load selected drivers from `OC/Drivers` directory.

   To be filled with `plist dict` values, describing each driver. Refer to the Drivers Properties section below.

6. `Input`
   Type: `plist dict`
   ~~**Failsafe**: None~~**Description**: Apply individual settings designed for input (keyboard and mouse) in the Input Properties section below.

7. `Output`
   Type: `plist dict`
   ~~**Failsafe**: None~~**Description**: Apply individual settings designed for output (text and graphics) in the Output Properties section below.

8. `ProtocolOverrides`
   Type: `plist dict`
   ~~**Failsafe**: None~~**Description**: Force builtin versions of certain protocols described in the ProtocolOverrides Properties section below.

   Note: all protocol instances are installed prior to driver loading.

9. `Quirks`
   Type: `plist dict`~~**Failsafe**: None~~
   Description: Apply individual firmware quirks described in the Quirks Properties section below.

10. `ReservedMemory`
    Type: `plist array`
    Failsafe: Empty
    Description: To be filled with `plist dict` values, describing memory areas exclusive to specific firmware and hardware functioning, which should not be used by the operating system. Examples of such memory regions could be the second 256 MB corrupted by the Intel HD 3000 or an area with faulty RAM. Refer to the ReservedMemory Properties section below for details.

11.11 APFS Properties

1. `EnableJumpstart`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Load embedded APFS drivers from APFS containers.