OpenCore

Reference Manual (0.8.6.7)

[2022.12.06]

Note: This option may not work well with the `System` text renderer. Setting a background different from black could help with testing GOP functionality.

2. `HibernateMode`
   Type: `plist string`
   Failsafe: `None`
   Description: Hibernation detection mode. The following modes are supported:

   - `None` — Ignore hibernation state.
   - `Auto` — Use RTC and NVRAM detection.
   - `RTC` — Use RTC detection.
   - `NVRAM` — Use NVRAM detection.

   Note: If the firmware can handle hibernation itself (valid for Mac EFI firmware), then `None` should be specified to hand-off hibernation state as is to OpenCore.

3. `HibernateSkipsPicker`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Do not show picker if waking from macOS hibernation.

   Limitations:

   - Only supports macOS hibernation wake, Windows and Linux are currently out of scope.
   - Should only be used on systems with reliable hibernation wake in macOS, otherwise users may not be able to visually see boot loops that may occur.
   - Highly recommended to pair this option with `PollAppleHotKeys`, allows to enter picker in case of issues with hibernation wake.
   - Visual indication for hibernation wake is currently out of scope.

4. `HideAuxiliary`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Set to `true` to hide auxiliary entries from the picker menu.

   An entry is considered auxiliary when at least one of the following applies:

   - Entry is macOS recovery.
   - Entry is macOS Time Machine.
   - Entry is explicitly marked as `Auxiliary`.
   - Entry is system (e.g. `Reset NVRAM`).

   To display all entries, the picker menu can be reloaded into "Extended Mode" by pressing the `Spacebar` key. Hiding auxiliary entries may increase boot performance on multi-disk systems.

5. `LauncherOption`
   Type: `plist string`
   Failsafe: `Disabled`
   Description: Register the launcher option in the firmware preferences for persistence.

   Valid values:

   - `Disabled` — do nothing.
   - `Full` — create or update the top priority boot option in UEFI variable storage at bootloader startup.
     - For this option to work, `RequestBootVarRouting` is required to be enabled.
   - `Short` — create a short boot option instead of a complete one.
     - This variant is useful for some older types of firmware, typically from Insyde, that are unable to manage full device paths.
   - `System` — create no boot option but assume specified custom option is blessed.
     - This variant is useful when relying on `ForceBooterSignature` quirk and OpenCore launcher path management happens through `bless` utilities without involving OpenCore.

loaded and connected first. Configuring the boot chime and adding this longer additional delay can also be useful in systems where fast boot time and/or slow monitor signal synchronisation may cause the boot logo not to be shown at all on some boots or reboots.

12. `Timeout`
    Type: `plist integer`, 32 bit
    Failsafe: `0`
    Description: Timeout in seconds in the OpenCore picker before automatic booting of the default boot entry. Set to `0` to disable.

13. `PickerMode`
    Type: `plist string`
    Failsafe: `Builtin`
    Description: Choose picker used for boot management.

    `PickerMode` describes the underlying boot management with an optional user interface responsible for handling boot options.

    The following values are supported:

    - `Builtin` — boot management is handled by OpenCore, a simple text-only user interface is used.
    - `External` — an external boot management protocol is used if available. Otherwise, the `Builtin` mode is used.
    - `Apple` — Apple boot management is used if available. Otherwise, the `Builtin` mode is used.

    Upon success, the `External` mode may entirely disable all boot management in OpenCore except for policy enforcement. In the `Apple` mode, it may additionally bypass policy enforcement. Refer to the OpenCanopy plugin for an example of a custom user interface.

    The OpenCore built-in picker contains a set of actions chosen during the boot process. The list of supported actions is similar to Apple BDS and typically can be accessed by holding `action hotkeys` during the boot process.

    The following actions are currently considered:

    - `Default` — this is the default option, and it lets the built-in OpenCore picker load the default boot option as specified in the Startup Disk preference pane.
    - `ShowPicker` — this option forces the OpenCore picker to be displayed. This can typically be achieved by holding the `OPT` key during boot. Setting `ShowPicker` to `true` will make `ShowPicker` the default option.
    - `BootApple` — this options performs booting to the first Apple operating system found unless the chosen default operating system is one from Apple. Hold the `X` key down to choose this option.
    - `BootAppleRecovery` — this option performs booting into the Apple operating system recovery partition. This is either that related to the default chosen operating system, or first one found when the chosen default operating system is not from Apple or does not have a recovery partition. Hold the `CMD+R` hotkey combination down to choose this option.

    Note 1: On non-Apple firmware `KeySupport`, `OpenUsbKbDxe`, or similar drivers are required for key handling. However, not all of the key handling functions can be implemented on several types of firmware.

    Note 2: In addition to `OPT`, OpenCore supports using both the `Escape` and `Zero` keys to enter the OpenCore picker when `ShowPicker` is disabled. `Escape` exists to support co-existence with the Apple picker (including OpenCore `Apple` picker mode) and to support firmware that fails to report held `OPT` key, as on some PS/2 keyboards. In addition, `Zero` is provided to support systems on which `Escape` is already assigned to some other pre-boot firmware feature. In systems which do not require `KeySupport`, pressing and holding one of these keys from after power on until the picker appears should always be successful. The same should apply when using `KeySupport` mode if it is correctly configured for the system, i.e. with a long enough `KeyForgetThreshold`. If pressing and holding the key is not successful to reliably enter the picker, multiple repeated keypresses may be tried instead.

    Note 3: On Macs with problematic GOP, it may be difficult to ~~access the Apple picker~~re-bless OpenCore if its bless status is lost. The `BootKicker` utility can be ~~blessed to workaround this problemeven without loading OpenCore . On some Macs however, the~~ used to work around this problem, if set up as a Tool in OpenCore (e.g. on a CDROM) with `FullNvramAccess` enabled. It will launch the Apple picker, which allows selection of an item to

OpenVariableRuntimeDxe*	OpenCore plugin offering emulated NVRAM support. OpenDuet already includes this driver.
Ps2KeyboardDxe*	PS/2 keyboard driver from MdeModulePkg. OpenDuetPkg and some types of firmware may not include this driver, but it is necessary for PS/2 keyboard to work. Note, unlike OpenUsbKbDxe this driver has no AppleKeyMapAggregator support and thus requires KeySupport to be enabled.
Ps2MouseDxe*	PS/2 mouse driver from MdeModulePkg. Some very old laptop firmware may not include this driver but it is necessary for the touchpad to work in UEFI graphical interfaces such as OpenCanopy.
OpenHfsPlus*	HFS file system driver with bless support. This driver is an alternative to a closed source HfsPlus driver commonly found in Apple firmware. While it is feature complete, it is approximately 3 times slower and is yet to undergo a security audit.
ResetNvramEntry*	OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL to add a configurable Reset NVRAM entry to the boot picker menu.
ToggleSipEntry*	OpenCore plugin implementing OC_BOOT_ENTRY_PROTOCOL to add a configurable Toggle SIP entry to the boot picker menu.
UsbMouseDxe*	USB mouse driver from MdeModulePkg. Some virtual machine firmware such as OVMF may not include this driver but it is necessary for the mouse to work in UEFI graphical interfaces such as OpenCanopy.
XhciDxe*	XHCI USB controller support driver from MdeModulePkg. This driver is included in most types of firmware starting with the Sandy Bridge generation. For earlier firmware or legacy systems, it may be used to support external USB 3.0 PCI cards.

Driver marked with * are bundled with OpenCore. To compile the drivers from UDK (EDK II) the same command used for OpenCore compilation can be taken, but choose a corresponding package:

```
git clone https://github.com/acidanthera/audk UDK
cd UDK
source edksetup.sh
make -C BaseTools
build -a X64 -b RELEASE -t XCODE5 -p FatPkg/FatPkg.dsc
build -a X64 -b RELEASE -t XCODE5 -p MdeModulePkg/MdeModulePkg.dsc
```

11.3 Tools and Applications

Standalone tools may help to debug firmware and hardware. Some of the known tools are listed below. While some tools can be launched from within OpenCore (Refer to the Tools subsection for more details), most should be run separately either directly or from Shell.

To boot into OpenShell or any other tool directly save OpenShell.efi under the name of EFI\BOOT\BOOTX64.EFI on a FAT32 partition. It is typically unimportant whether the partition scheme is GPT or MBR.

While the previous approach works both on Macs and other computers, an alternative Mac-only approach to bless the tool on an HFS+ or APFS volume:

```
sudo bless --verbose --file /Volumes/VOLNAME/DIR/OpenShell.efi \
  --folder /Volumes/VOLNAME/DIR/ --setBoot
```

Listing 3: Blessing tool

Note 1: /System/Library/CoreServices/BridgeVersion.bin should be copied to /Volumes/VOLNAME/DIR.
Note 2: To be able to use the bless command, disabling System Integrity Protection is necessary.
Note 3: To be able to boot Secure Boot might be disabled if present.

Some of the known tools are listed below (builtin tools are marked with *):

Failsafe: `false`
Description: Enable AVX vector acceleration of SHA-512 and SHA-384 hashing algorithms.

Note: This option may cause issues on certain laptop firmwares, including Lenovo.

3. `EnableVmx`
Type: `plist boolean`
Failsafe: `false`
Description: Enable Intel virtual machine extensions.

Note: Required to allow virtualization in Windows on some Mac hardware. VMX is enabled or disabled and locked by BIOS before OpenCore starts on most firmware. Use BIOS to enable virtualization where possible.

4. `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.

5. `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.

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

7. `ForgeUefiSupport`
Type: `plist boolean`
Failsafe: `false`
Description: Implement partial UEFI 2.x support on EFI 1.x firmware.

This setting allows running some software written for UEFI 2.x firmware~~like~~, such as NVIDIA GOP Option ROMs, on hardware with older EFI 1.x firmware ~~like~~ (e.g. `MacPro5,1`).

8. `IgnoreInvalidFlexRatio`
Type: `plist boolean`
Failsafe: `false`
Description: Some types of firmware (such as APTIO IV) may contain invalid values in the `MSR_FLEX_RATIO` (`0x194`) MSR register. These values may cause macOS boot failures on Intel platforms.

Note: While the option is not expected to harm unaffected firmware, its use is recommended only when specifically required.

9. `ReleaseUsbOwnership`
Type: `plist boolean`
Failsafe: `false`

Note: On several motherboards (and possibly USB UART dongles) PIN naming may be incorrect. It is very common to have GND swapped with RX, thus, motherboard "TX" must be connected to USB UART GND, and motherboard "GND" to USB UART RX.

Remember to enable COM port in firmware settings, and never use USB cables longer than 1 meter to avoid output corruption. To additionally enable XNU kernel serial output debug=0x8 boot argument is needed.

12.5 Tips and Tricks

1. **How do I debug boot failures?**

   Obtaining the actual error message is usually adequate. For this, ensure that:

   - A DEBUG or NOOPT version of OpenCore is used.
   - Logging is enabled (1) and shown onscreen (2): Misc → Debug → Target = 3.
   - Logged messages from at least DEBUG_ERROR (0x80000000), DEBUG_WARN (0x00000002), and DEBUG_INFO (0x00000040) levels are visible onscreen: Misc → Debug → DisplayLevel = 0x80000042.
   - Critical error messages, such as DEBUG_ERROR, stop booting: Misc → Security → HaltLevel = 0x80000000.
   - Watch Dog is disabled to prevent automatic reboot: Misc → Debug → DisableWatchDog = true.
   - Boot Picker (entry selector) is enabled: Misc → Boot → ShowPicker = true.

   If there is no obvious error, check the available workarounds in the Quirks sections one by one. For early boot troubleshooting, for instance, when OpenCore menu does not appear, using UEFI Shell (bundled with OpenCore) may help to see early debug messages.

2. **How do I debug macOS boot failures?**

   - Refer to boot-args values such as debug=0x100, keepsyms=1, -v, and similar.
   - Do not forget about AppleDebug and ApplePanic properties.
   - For macOS to correctly recognise and set up serial ports, the CustomPciSerialDevice quirk may be enabled when a PCI serial port card is installed.
   - Take care of Booter, Kernel, and UEFI quirks.
   - Consider using serial port to inspect early kernel boot failures. For this debug=0x108, serial=5, and msgbuf=1048576 boot arguments are needed. Refer to the patches in Sample.plist when dying before serial init.
   - Always read the logs carefully.

3. **How do I customise boot entries?**

   OpenCore follows standard Apple Bless model and extracts the entry name from .contentDetails and .disk_label.contentDetails files in the booter directory if present. These files contain an ASCII string with an entry title, which may then be customised by the user.

4. **How do I choose the default boot entry?**

   OpenCore uses the primary UEFI boot option to select the default entry. This choice can be altered from UEFI Setup, with the macOS Startup Disk preference, or the Windows Boot Camp Control Panel. Since choosing OpenCore's BOOTx64.EFI as a primary boot option limits this functionality in addition to several types of firmware deleting incompatible boot options, potentially including those created by macOS, users are strongly encouraged to use the RequestBootVarRouting quirk, which will preserve the selection made in the operating system within the OpenCore variable space. Note, that RequestBootVarRouting requires a separate driver for functioning.

5. **What is the simplest way to install macOS?**

   Copy online recovery image (*.dmg and *.chunklist files) to com.apple.recovery.boot directory on a FAT32 partition with OpenCore. Load the OpenCore picker and choose the entry, it will have a (dmg) suffix. Custom name may be created by providing .contentDetails file.

   To download recovery online macrecovery.py can be used.

   For offline installation refer to How to create a bootable installer for macOS article. Apart from App Store and softwareupdate utility there also are third-party utilities to download an offline image.

6. **Why do online recovery images (*.dmg) fail to load?**

   This may be caused by missing HFS+ driver, as all presently known recovery volumes have HFS+ filesystem.

7. **Can I use this on Apple hardware or virtual machines?**

   ~~Sure,~~ Yes. Virtual machines and most relatively modern Mac models~~including~~ , as far back as the ~~MacPro5~~`MacPro3,1`~~and virtual machines~~ , are fully supported. ~~Even though there are little to none specific details~~ While specific detail relevant to Mac hardware is often limited, some ongoing instructions can be found on MacRumors.com.

8. **Why must Find&Replace patches be equal in size?**

   For machine code (x86 code) it is not possible to do differently sized replacements due to relative addressing. For ACPI code this is risky, and is technically equivalent to ACPI table replacement, thus not implemented. More detailed explanation can be found on AppleLife.ru or in the ACPI section of this document.

9. **How can I decide which `Booter` quirks to use?**

   These quirks originate from `AptioMemoryFix` driver but provide a wider set of changes specific to modern systems. Note, that `OpenRuntime` driver is required for most configurations. To get a configuration similar to `AptioMemoryFix` the following set of quirks should be enabled:

   - `ProvideConsoleGop` (UEFI quirk)
   - `AvoidRuntimeDefrag`
   - `DiscardHibernateMap`
   - `EnableSafeModeSlide`
   - `EnableWriteUnprotector`
   - `ForceExitBootServices`
   - `ProtectMemoryRegions`
   - `ProvideCustomSlide`
   - `RebuildAppleMemoryMap`
   - `SetupVirtualMap`

   However, as of today, such set is strongly discouraged as some of these quirks are not necessary to be enabled or need additional quirks. For example, `DevirtualiseMmio` and `ProtectUefiServices` are often required, while `DiscardHibernateMap` and `ForceExitBootServices` are rarely necessary.

   Unfortunately for some quirks such as `RebuildAppleMemoryMap`, `EnableWriteUnprotector`, `ProtectMemoryRegions`, `SetupVirtualMap`, and `SyncRuntimePermissions` there is no definite approach even on similar systems, so trying all their combinations may be required for optimal setup. Refer to individual quirk descriptions in this document for details.