OpenCore

Reference Manual (0.5~~.5~~.6)

[2020.02.08]

8 Misc

8.1 Introduction

This section contains miscellaneous configuration entries for OpenCore behaviour that does not go to any other sections

8.2 Properties

1. `Boot`
   Type: `plist dict`
   Description: Apply boot configuration described in Boot Properties section below.

2. `BlessOverride`
   Type: `plist array`
   Description: Add custom scanning paths through bless model.

   Designed to be filled with `plist string` entries containing absolute UEFI paths to customised bootloaders, for example, `\EFI\Microsoft\Boot\bootmgfw.efi` for Microsoft bootloader. This allows unusual boot paths to be automaticlly discovered by the boot picker. Designwise they are equivalent to predefined blessed path, such as `\System\Library\CoreServices\boot.efi`, but unlike predefined bless paths they have highest priority.

3. `Debug`
   Type: `plist dict`
   Description: Apply debug configuration described in Debug Properties section below.

4. `Entries`
   Type: `plist array`
   Description: Add boot entries to boot picker.

   Designed to be filled with `plist dict` values, describing each load entry. See Entry Properties section below.

5. `Security`
   Type: `plist dict`
   Description: Apply security configuration described in Security Properties section below.

6. `Tools`
   Type: `plist array`
   Description: Add tool entries to boot picker.

   Designed to be filled with `plist dict` values, describing each load entry. See Entry Properties section below.

   Note: Select tools, for example, UEFI Shell are very dangerous and **MUST NOT** appear in production configurations, especially in vaulted ones and protected with secure boot, as they may be used to easily bypass secure boot chain.

8.3 Boot Properties

1. ~~`BuiltinTextRenderer`**Type**: `plist boolean`**Failsafe**: `false`**Description**: Enables experimental builtin text renderer.~~

   ~~This option makes all text going through standard console output render through builtin text renderer bypassing firmware services. While still experimental and feature incomplete, this option effecitly avoids the need for various quirks like `ReplaceTabWithSpace` or `SanitiseClearScreen`. It should also increase the dimensions of the output area.~~

   ~~Since builtin text renderer works in graphics mode, extra care may need to be paid to `ConsoleBehaviourOs`, `ConsoleBehaviourUi`, `ConsoleControl`, and `IgnoreTextInGraphics` options. While individual for the target system, it is recommended to use `ForceGraphics` and builtin `ConsoleControl` to avoid compatibility issues.~~

   ~~*Note*: Some Macs, namely `MacPro5,1`, may have broken console output with newer GPUs, and thus enabling this option can be required for them.~~

2. ~~ConsoleMode~~~~**Type**: plist string~~~~**Failsafe**: Empty string~~~~**Description**: Sets console output mode as specified with the WxH (e.g. 80x24) formatted string. Set to empty string not to change console mode. Set to Max to try to use largest available console mode.~~

   ~~*Note*: This field is best to be left empty on most firmwares.~~

3. ~~ConsoleBehaviourOs~~~~**Type**: plist string~~~~**Failsafe**: Empty string~~~~**Description**: Set console control behaviour upon operating system load.~~

   ~~Console control is a legacy protocol used for switching between text and graphics screen output. Some firmwares do not provide it, yet select operating systems require its presence, which is what ConsoleControl UEFI protocol is for.~~

   ~~When console control is available, OpenCore can be made console control aware, and set different modes for the operating system booter (ConsoleBehaviourOs), which normally runs in graphics mode, and its own user interface (ConsoleBehaviourUi), which normally runs in text mode. Possible behaviours, set as values of these options, include:~~

   - ~~Empty string — Do not modify console control mode.~~
   - ~~Text — Switch to text mode.~~
   - ~~Graphics — Switch to graphics mode.~~
   - ~~ForceText — Switch to text mode and preserve it (requires ConsoleControl).~~
   - ~~ForceGraphics — Switch to graphics mode and preserve it (require ConsoleControl).~~

   ~~Hints:~~

   - ~~Unless empty works, firstly try to set ConsoleBehaviourOs to Graphics and ConsoleBehaviourUi to Text.~~
   - ~~On APTIO IV (Haswell and earlier) it is usually enough to have ConsoleBehaviourOs set to Graphics and ConsoleBehaviourUi set to ForceText to avoid visual glitches.~~
   - ~~On APTIO V (Broadwell and newer) ConsoleBehaviourOs set to ForceGraphics and ConsoleBehaviourUi set to ForceText usually works best.~~
   - ~~On Apple firmwares ConsoleBehaviourOs set to Graphics and ConsoleBehaviourUi set to Text is supposed to work best.~~

   ~~*Note*: IgnoreTextInGraphics and SanitiseClearScreen may need to be enabled for select firmware implementations. Particularly APTIO firmwares.~~

4. ~~ConsoleBehaviourUi~~~~**Type**: plist string~~~~**Failsafe**: Empty string~~~~**Description**: Set console control behaviour upon OpenCore user interface load. Refer to ConsoleBehaviourOs description for details.~~

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

   - None — Avoid hibernation for your own good.
   - Auto — Use RTC and NVRAM detection.
   - RTC — Use RTC detection.
   - NVRAM — Use NVRAM detection.

6. HideSelf
   Type: plist boolean
   Failsafe: false
   Description: Hides own boot entry from boot picker. This may potentially hide other entries, for instance, when another UEFI OS is installed on the same volume and driver boot is used.

7. PollAppleHotKeys
   Type: plist boolean

Failsafe: `false`
Description: Enable `modifier hotkey` handling in boot picker.

In addition to `action hotkeys`, which are partially described in `UsePicker` section and are normally handled by Apple BDS, there exist modifier keys, which are handled by operating system bootloader, namely `boot.efi`. These keys allow to change operating system behaviour by providing different boot modes.

On some firmwares it may be problematic to use modifier keys due to driver incompatibilities. To workaround this problem this option allows registering select hotkeys in a more permissive manner from within boot picker. Such extensions include the support of tapping on keys in addition to holding and pressing `Shift` along with other keys instead of just `Shift` alone, which is not detectible on many PS/2 keyboards. This list of known `modifier hotkeys` includes:

- `CMD+C+MINUS` — disable board compatibility checking.
- `CMD+K` — boot release kernel, similar to `kcsuffix=release`.
- `CMD+S` — single user mode.
- `CMD+S+MINUS` — disable KASLR slide, requires disabled SIP.
- `CMD+V` — verbose mode.
- `Shift` — safe mode.

8. ~~`Resolution`**Type**: `plist string`**Failsafe**: Empty string**Description**: Sets console output screen resolution.~~

   - ~~Set to `WxH@Bpp` (e.g. `1920x1080@32`) or `WxH` (e.g. `1920x1080`) formatted string to request custom resolution from GOP if available.~~
   - ~~Set to empty string not to change screen resolution.~~
   - ~~Set to `Max` to try to use largest available screen resolution.~~

   ~~On HiDPI screens `APPLE_VENDOR_VARIABLE_GUID` `UIScale` NVRAM variable may need to be set to `02` to enable HiDPI scaling in FileVault 2 UEFI password interface and boot screen logo. Refer to section for more details.~~

   ~~*Note*: This will fail when console handle has no GOP protocol. When the firmware does not provide it, it can be added with `ProvideConsoleGop` UEFI quirk set to `true`.~~

9. `ShowPicker`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Show simple boot picker to allow boot entry selection.

10. `TakeoffDelay`
    Type: `plist integer`, 32 bit
    Failsafe: `0`
    Description: Delay in microseconds performed before handling picker startup and `action hotkeys`.

    Introducing a delay may give extra time to hold the right `action hotkey` sequence to e.g. boot to recovery mode. On some platforms setting this option to at least `5000-10000` microseconds may be necessary to access `action hotkeys` at all due to the nature of the keyboard driver.

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

12. `UsePicker`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Use OpenCore built-in boot picker for boot management.

    `UsePicker` set to `false` entirely disables all boot management in OpenCore except policy enforcement. In this case a custom user interface may utilise OcSupportPkg `OcBootManagementLib` to implement a user friendly boot picker oneself. Reference example of external graphics interface is provided in ExternalUi test driver.

    - `1` — AppleLoggingConOutOrErrSet/AppleLoggingConOutOrErrPrint (classical ConOut/StdErr)
    - `2` — AppleLoggingStdErrSet/AppleLoggingStdErrPrint (StdErr or serial?)
    - `4` — AppleLoggingFileSet/AppleLoggingFilePrint (BOOTER.LOG/BOOTER.OLD file on EFI partition)
  - `debug=VALUE`
    - `1` — enables print something to BOOTER.LOG (stripped code implies there may be a crash)
    - `2` — enables perf logging to /efi/debug-log in the device three
    - `4` — enables timestamp printing for styled printf calls
  - `level=VALUE` — Verbosity level of DEBUG output. Everything but `0x80000000` is stripped from the binary, and this is the default value.
  - `kc-read-size=VALUE` — Chunk size used for buffered I/O from network or disk for prelinkedkernel reading and related. Set to 1MB (0x100000) by default, can be tuned for faster booting.

  Note: To quickly see verbose output from `boot.efi` set this to `log=1` (currently this is broken in 10.15).
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:bootercfg-once`
  Booter arguments override removed after first launch. Otherwise equivalent to `bootercfg`.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:fmm-computer-name`
  Current saved host name. ASCII string.
- `7C436110-AB2A-4BBB-A880-FE41995C9F82:nvda_drv`
  NVIDIA Web Driver control variable. Takes ASCII digit `1` or `0` to enable or disable installed driver.

```
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
```

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

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

5. `Protocols`
   Type: `plist dict`
   Failsafe: None
   Description: Force builtin versions of select protocols described in Protocols Properties section below.

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

6. `Quirks`
   Type: `plist dict`
   Failsafe: None
   Description: Apply individual firmware quirks described in Quirks Properties section below.

11.3 Input Properties

1. `KeyForgetThreshold`
   Type: `plist integer`
   Failsafe: `0`
   Description: Remove key unless it was submitted during this timeout in milliseconds.

   `AppleKeyMapAggregator` protocol is supposed to contain a fixed length buffer of currently pressed keys. However, the majority of the drivers only report key presses as interrupts and pressing and holding the key on the keyboard results in subsequent submissions of this key with some defined time interval. As a result we use a timeout to remove once pressed keys from the buffer once the timeout expires and no new submission of this key happened.

   This option allows to set this timeout based on your platform. The recommended value that works on the majority of the platforms is `5` milliseconds. For reference, holding one key on VMware will repeat it roughly every `2` milliseconds and the same value for APTIO V is `3-4` milliseconds. Thus it is possible to set a slightly lower value on faster platforms and slightly higher value on slower platforms for more responsive input.

2. `KeyMergeThreshold`
   Type: `plist integer`
   Failsafe: `0`
   Description: Assume simultaneous combination for keys submitted within this timeout in milliseconds.

   Similarly to `KeyForgetThreshold`, this option works around the sequential nature of key submission. To be able to recognise simultaneously pressed keys in the situation when all keys arrive sequentially, we are required to set a timeout within which we assume the keys were pressed together.

   Holding multiple keys results in reports every `2` and `1` milliseconds for VMware and APTIO V respectively. Pressing keys one after the other results in delays of at least `6` and `10` milliseconds for the same platforms. The recommended value for this option is `2` milliseconds, but it may be decreased for faster platforms and increased for slower.

3. `KeySupport`
   Type: `plist boolean`

Failsafe: `false`
Description: Enable internal keyboard input translation to `AppleKeyMapAggregator` protocol.

This option activates the internal keyboard interceptor driver, based on `AppleGenericInput` aka (`AptioInputFix`), to fill `AppleKeyMapAggregator` database for input functioning. In case a separate driver is used, such as `AppleUsbKbDxe`, this option should never be enabled.

4. `KeySupportMode`
   Type: `plist string`
   Failsafe: empty string
   Description: Set internal keyboard input translation to `AppleKeyMapAggregator` protocol mode.

   - `Auto` — Performs automatic choice as available with the following preference: `AMI`, `V2`, `V1`.
   - `V1` — Uses UEFI standard legacy input protocol `EFI_SIMPLE_TEXT_INPUT_PROTOCOL`.
   - `V2` — Uses UEFI standard modern input protocol `EFI_SIMPLE_TEXT_INPUT_EX_PROTOCOL`.
   - `AMI` — Uses APTIO input protocol `AMI_EFIKEYCODE_PROTOCOL`.

5. `KeySwap`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Swap `Command` and `Option` keys during submission.

   This option may be useful for keyboard layouts with `Option` key situated to the right of `Command` key.

6. `PointerSupport`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enable internal pointer driver.

   This option implements standard UEFI pointer protocol (`EFI_SIMPLE_POINTER_PROTOCOL`) through select OEM protocols. The option may be useful on Z87 ASUS boards, where `EFI_SIMPLE_POINTER_PROTOCOL` is broken.

7. `PointerSupportMode`
   Type: `plist string`
   Failsafe: empty string
   Description: Set OEM protocol used for internal pointer driver.

   Currently the only supported variant is `ASUS`, using specialised protocol available on select Z87 and Z97 ASUS boards. More details can be found in `LongSoft/UefiTool#116`.

8. `TimerResolution`
   Type: `plist integer`
   Failsafe: `0`
   Description: Set architecture timer resolution.

   This option allows to update firmware architecture timer period with the specified value in `100` nanosecond units. Setting a lower value generally improves performance and responsiveness of the interface and input handling.

   The recommended value is `50000` (5 milliseconds) or slightly higher. Select ASUS Z87 boards use `60000` for the interface. Apple boards use `100000`. You may leave it as `0` in case there are issues.

11.4 Output Properties

1. `TextRenderer`
   Type: `plist string`
   Failsafe: `BuiltinGraphics`
   Description: Chooses renderer for text going through standard console output.

   Currently two renderers are supported: `Builtin` and `System`. `System` renderer uses firmware services for text rendering. `Builtin` bypassing firmware services and performs text rendering on its own. Different renderers support a different set of options. It is recommended to use `Builtin` renderer, as it supports HiDPI mode and uses full screen resolution.

   UEFI firmwares generally support `ConsoleControl` with two rendering modes: `Graphics` and `Text`. Some firmwares do not support `ConsoleControl` and rendering modes. OpenCore and macOS expect text to only be shown in `Graphics` mode and graphics to be drawn in any mode. Since this is not required by UEFI specification, exact behaviour varies.

Valid values are combinations of text renderer and rendering mode:

- `BuiltinGraphics` — Switch to `Graphics` mode and use `Builtin` renderer with custom `ConsoleControl`.
- `SystemGraphics` — Switch to `Graphics` mode and use `System` renderer with custom `ConsoleControl`.
- `SystemText` — Switch to `Text` mode and use `System` renderer with custom `ConsoleControl`.
- `SystemGeneric` — Use `System` renderer with system `ConsoleControl` assuming it behaves correctly.

The use of `BuiltinGraphics` is generally straightforward. For most platforms it is necessary to enable `ProvideConsoleGop`, set `Resolution` to `Max`, and optionally configure `Scale`.

The use of `System` protocols is more complicated. In general the preferred setting is `SystemGraphics` or `SystemText`. Enabling `ProvideConsoleGop`, setting `Resolution` to `Max`, enabling `ReplaceTabWithSpace` is useful on almost all platforms. `SanitiseClearScreen`, `IgnoreTextInGraphics`, and `ClearScreenOnModeSwitch` are more specific, and their use depends on the firmware.

Note: Some Macs, namely `MacPro5,1`, may have broken console output with newer GPUs, and thus only `BuiltinGraphics` may work for them.

2. `ConsoleMode`
Type: `plist string`
Failsafe: Empty string
Description: Sets console output mode as specified with the `WxH` (e.g. `80x24`) formatted string.

   Set to empty string not to change console mode. Set to `Max` to try to use largest available console mode. Currently `Builtin` text renderer supports only one console mode, so this option is ignored.

   Note: This field is best to be left empty on most firmwares.

3. `Resolution`
Type: `plist string`
Failsafe: Empty string
Description: Sets console output screen resolution.

   - Set to `WxH@Bpp` (e.g. `1920x1080@32`) or `WxH` (e.g. `1920x1080`) formatted string to request custom resolution from GOP if available.
   - Set to empty string not to change screen resolution.
   - Set to `Max` to try to use largest available screen resolution.

   On HiDPI screens `APPLE_VENDOR_VARIABLE_GUID` `UIScale` NVRAM variable may need to be set to `02` to enable HiDPI scaling in FileVault 2 UEFI password interface and boot screen logo. Refer to Recommended Variables section for more details.

   Note: This will fail when console handle has no GOP protocol. When the firmware does not provide it, it can be added with `ProvideConsoleGop` set to `true`.

4. `ClearScreenOnModeSwitch`
Type: `plist boolean`
Failsafe: `false`
Description: Some firmwares clear only part of screen when switching from graphics to text mode, leaving a fragment of previously drawn image visible. This option fills the entire graphics screen with black color before switching to text mode.

   Note: This option only applies to `System` renderer.

5. `IgnoreTextInGraphics`
Type: `plist boolean`
Failsafe: `false`
Description: Select firmwares output text onscreen in both graphics and text mode. This is normally unexpected, because random text may appear over graphical images and cause UI corruption. Setting this option to `true` will discard all text output when console control is in mode different from `Text`.

   Note: This option only applies to `System` renderer.

6. `ReplaceTabWithSpace`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Some firmwares do not print tab characters or even everything that follows them, causing difficulties or inability to use the UEFI Shell 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.

7. `ProvideConsoleGop`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Ensure GOP (Graphics Output Protocol) on console handle.

   macOS bootloader requires GOP to be present on console handle, yet the exact location of GOP is not covered by the UEFI specification. This option will ensure GOP is installed on console handle if it is present.

   Note: This option will also replace broken GOP protocol on console handle, which may be the case on `MacPro5,1` with newer GPUs.

8. `ReconnectOnResChange`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Reconnect console controllers after changing screen resolution.

   On some firmwares when screen resolution is changed via GOP, it is required to reconnect the controllers, which produce the console protocols (simple text out). 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.

9. `SanitiseClearScreen`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Some firmwares reset screen resolution to a failsafe value (like `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 `System` renderer. On all known affected systems `ConsoleMode` had to be set to empty string for this to work.

10. `Scale`
    Type: `plist integer`
    Failsafe: `100`
    Description: Sets text renderer HiDPI scaling in percents.

    Currently only `100` and `200` values are supported.

    Note: This option only applies to `Builtin` renderer.

11.5 Protocols Properties

1. `AppleBootPolicy`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Reinstalls Apple Boot Policy protocol with a builtin version. This may be used to ensure APFS compatibility on VMs or legacy Macs.

   Note: Some Macs, namely `MacPro5,1`, do have APFS compatibility, but their Apple Boot Policy protocol contains recovery detection issues, thus using this option is advised on them as well.

2. `AppleEvent`
   Type: `plist boolean`
   Failsafe: `false`

Description: Reinstalls Apple Event protocol with a builtin version. This may be used to ensure File Vault 2 compatibility on VMs or legacy Macs.

3. `AppleImageConversion`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Reinstalls Apple Image Conversion protocol with a builtin version.

4. `AppleKeyMap`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Reinstalls Apple Key Map protocols with builtin versions.

5. `AppleSmcIo`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Reinstalls Apple SMC I/O protocol with a builtin version.

   This protocol replaces legacy `VirtualSmc` UEFI driver, and is compatible with any SMC kernel extension. However, in case `FakeSMC` kernel extension is used, manual NVRAM key variable addition may be needed.

6. `AppleUserInterfaceTheme`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Reinstalls Apple User Interface Theme protocol with a builtin version.

7. ~~`ConsoleControl`**Type**: `plist boolean`**Failsafe**: `false`**Description**: Replaces Console Control protocol with a builtin version.~~

   ~~macOS bootloader requires console control protocol for text output, which some firmwares miss. This option is required to be set when the protocol is already available in the firmware, and other console control options are used, such as `IgnoreTextInGraphics`, `SanitiseClearScreen`, and sometimes `ConsoleBehaviourOs` with `ConsoleBehaviourUi`).~~

8. `DataHub`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Reinstalls Data Hub protocol with a builtin version. This will drop all previous properties if the protocol was already installed.

9. `DeviceProperties`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Reinstalls Device Property protocol with a builtin version. This will drop all previous properties if it was already installed. This may be used to ensure full compatibility on VMs or legacy Macs.

10. `FirmwareVolume`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Forcibly wraps Firmware Volume protocols or installs new to support custom cursor images for File Vault 2. Should be set to `true` to ensure File Vault 2 compatibility on everything but VMs and legacy Macs.

11. `HashServices`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Forcibly reinstalls Hash Services protocols with builtin versions. Should be set to `true` to ensure File Vault 2 compatibility on platforms providing broken SHA-1 hashing. Can be diagnosed by invalid cursor size with `UIScale` set to `02`, in general platforms prior to APTIO V (Haswell and older) are affected.

12. `OSInfo`
    Type: `plist boolean`
    Failsafe: `false`

Description: Forcibly reinstalls OS Info protocol with builtin versions. This protocol is generally used to receive notifications from macOS bootloader, by the firmware or by other applications.

13. `UnicodeCollation`
Type: `plist boolean`
Failsafe: `false`
Description: Forcibly reinstalls unicode collation services with builtin version. Should be set to `true` to ensure UEFI Shell compatibility on platforms providing broken unicode collation. In general legacy Insyde and APTIO platforms on Ivy Bridge and earlier are affected.

11.6 Quirks Properties

1. ~~`AvoidHighAlloc`**Type**: `plist boolean`**Failsafe**: `false`**Description**: Advises allocators to avoid allocations above first 4 GBs of RAM.~~

   ~~This is a workaround for select board firmwares, namely GA-Z77P-D3 (rev. 1.1), failing to properly access higher memory in UEFI Boot Services. On these boards this quirk is required for booting entries that need to allocate large memory chunks, such as macOS DMG recovery entries. On unaffected boards it may cause boot failures, and thus strongly not recommended. For known issues refer to .~~

2. ~~`ClearScreenOnModeSwitch`**Type**: `plist boolean`**Failsafe**: `false`**Description**: Some firmwares clear only part of screen when switching from graphics to text mode, leaving a fragment of previously drawn image visible. This option fills the entire graphics screen with black color before switching to text mode.~~

   ~~*Note*: `ConsoleControl` should be set to `true` for this to work.~~

3. `ExitBootServicesDelay`
Type: `plist integer`
Failsafe: `0`
Description: Adds delay in microseconds after `EXIT_BOOT_SERVICES` event.

   This is a very ugly quirk to circumvent "Still waiting for root device" message on select APTIO IV firmwares, namely ASUS Z87-Pro, when using FileVault 2 in particular. It seems that for some reason they execute code in parallel to `EXIT_BOOT_SERVICES`, which results in SATA controller being inaccessible from macOS. A better approach should be found in some future. Expect 3-5 seconds to be enough in case the quirk is needed.

4. `IgnoreInvalidFlexRatio`
Type: `plist boolean`
Failsafe: `false`
Description: Select firmwares, namely APTIO IV, may contain invalid values in `MSR_FLEX_RATIO` (`0x194`) MSR register. These values may cause macOS boot failure on Intel platforms.

   Note: While the option is not supposed to induce harm on unaffected firmwares, its usage is not recommended when it is not required.

5. ~~`IgnoreTextInGraphics`**Type**: `plist boolean`**Failsafe**: `false`**Description**: Select firmwares output text onscreen in both graphics and text mode. This is normally unexpected, because random text may appear over graphical images and cause UI corruption. Setting this option to `true` will discard all text output when console control is in mode different from `Text`.~~

   ~~*Note*: While the option is not supposed to induce harm on unaffected firmwares, its usage is not recommended when it is not required. This option may hide onscreen error messages. `ConsoleControl` may need to be set to `true` for this to work.~~

6. ~~`ReplaceTabWithSpace`**Type**: `plist boolean`**Failsafe**: `false`**Description**: Some firmwares do not print tab characters or even everything that follows them, causing difficulties or inability to use the UEFI Shell builtin text editor to edit property lists and other documents. This option makes the console output spaces instead of tabs.~~

   ~~*Note*: `ConsoleControl` may need to be set to `true` for this to work.~~

7. ~~`ProvideConsoleGop`**Type**: `plist boolean`**Failsafe**: `false`**Description**: Ensure GOP (Graphics Output Protocol) on console handle.~~

~~macOS bootloader requires GOP to be present on console handle, yet the exact location of GOP is not covered by the UEFI specification. This option will ensure GOP is installed on console handle if it is present.~~

~~*Note*: This option will also replace broken GOP protocol on console handle, which may be the case on `MacPro5,1` with newer GPUs.~~

8. ~~`ReconnectOnResChange`**Type**: `plist boolean`**Failsafe**: `false`**Description**: Reconnect console controllers after changing screen resolution.~~

   ~~On some firmwares when screen resolution is changed via GOP, it is required to reconnect the controllers, which produce the console protocols (simple text out). 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.~~

9. `ReleaseUsbOwnership`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Attempt to detach USB controller ownership from the firmware driver. While most firmwares manage to properly do that, or at least have an option for, select firmwares do not. As a result, operating system may freeze upon boot. Not recommended unless required.

10. `RequestBootVarFallback`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Request fallback of some `Boot` prefixed variables from `OC_VENDOR_VARIABLE_GUID` to `EFI_GLOBAL_VARIABLE_GUID`.

    This quirk requires `RequestBootVarRouting` to be enabled and therefore `OC_FIRMWARE_RUNTIME` protocol implemented in `FwRuntimeServices.efi`.

    By redirecting `Boot` prefixed variables to a separate GUID namespace we achieve multiple goals:

    - Operating systems are jailed and only controlled by OpenCore boot environment to enhance security.
    - Operating systems do not mess with OpenCore boot priority, and guarantee fluent updates and hibernation wakes for cases that require reboots with OpenCore in the middle.
    - Potentially incompatible boot entries, such as macOS entries, are not deleted or anyhow corrupted.

    However, some firmwares do their own boot option scanning upon startup by checking file presence on the available disks. Quite often this scanning includes non-standard locations, such as Windows Bootloader paths. Normally it is not an issue, but some firmwares, ASUS firmwares on APTIO V in particular, have bugs. For them scanning is implemented improperly, and firmware preferences may get accidentally corrupted due to `BootOrder` entry duplication (each option will be added twice) making it impossible to boot without cleaning NVRAM.

    To trigger the bug one should have some valid boot options (e.g. OpenCore) and then install Windows with `RequestBootVarRouting` enabled. As Windows bootloader option will not be created by Windows installer, the firmware will attempt to create it itself, and then corrupt its boot option list.

    This quirk forwards all UEFI specification valid boot options, that are not related to macOS, to the firmware into `BootF###` and `BootOrder` variables upon write. As the entries are added to the end of `BootOrder`, this does not break boot priority, but ensures that the firmware does not try to append a new option on its own after Windows installation for instance.

11. `RequestBootVarRouting`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Request redirect of all `Boot` prefixed variables from `EFI_GLOBAL_VARIABLE_GUID` to `OC_VENDOR_VARIABLE_GUID`.

    This quirk requires `OC_FIRMWARE_RUNTIME` protocol implemented in `FwRuntimeServices.efi`. The quirk lets default boot entry preservation at times when firmwares delete incompatible boot entries. Simply said, you are required to enable this quirk to be able to reliably use Startup Disk preference pane in a firmware that is not compatible with macOS boot entries by design.

12. ~~`SanitiseClearScreen`**Type**: `plist boolean`**Failsafe**: `false`**Description**: Some firmwares reset screen resolution to a failsafe value (like `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*: `ConsoleControl` may need to be set to `true` for this to work. On all known affected systems `ConsoleMode` had to be set to empty string for this to work.~~

13. `UnblockFsConnect`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Some firmwares block partition handles by opening them in By Driver mode, which results in File System protocols being unable to install.

    Note: The quirk is mostly relevant for select HP laptops with no drives listed.

- 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, like `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 hacks in `Quirks` sections one by one. For early boot troubleshooting, for instance, when OpenCore menu does not appear, using UEFI Shell may help to see early debug messages.

2. **How to 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.

3. **How to 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 firmwares deleting incompatible boot options, potentially including those created by macOS, you are strongly encouraged to use the `RequestBootVarRouting` quirk, which will preserve your selection made in the operating system within the OpenCore variable space. Note, that `RequestBootVarRouting` requires a separate driver for functioning.

4. **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 OpenCore Boot Picker and choose the entry, it will have a `(dmg)` suffix. Custom name may be created by providing `.contentDetails` file.

To download recovery online you may use macrecovery.py tool from MacInfoPkg.

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 tools to download an offline image.

5. **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~~. Another cause may be buggy firmware allocator, which can be worked around with `AvoidHighAlloc` UEFI quirk~~.

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

Sure, most relatively modern Mac models including `MacPro5,1` and virtual machines are fully supported. Even though there are little to none specific details relevant to Mac hardware, some ongoing instructions can be found in acidanthera/bugtracker#377.

7. **Why do Find&Replace patches must equal in length?**

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.

8. **How can I migrate from `AptioMemoryFix`?**

Behaviour similar to that of `AptioMemoryFix` can be obtained by installing `FwRuntimeServices` driver and enabling the quirks listed below. Please note, that most of these are not necessary to be enabled. Refer to their individual descriptions in this document for more details.

- `ProvideConsoleGop` (UEFI quirk)
- `AvoidRuntimeDefrag`
- `DiscardHibernateMap`
- `EnableSafeModeSlide`
- `EnableWriteUnprotector`
- `ForceExitBootServices`
- `ProtectCsmRegion`