OpenCore

Reference Manual (~~0.6.9~~0.7.0)

[2021.05.19]

- 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.

- 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`
- Rocket Lake (`0x0A0670`) to Comet Lake (`0x0906EB`):
  `Cpuid1Data`: `EB 06 09 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`

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
   Description: Disables `AppleIntelCpuPowerManagement`.

   Note 1: This option is a preferred alternative to `NullCpuPowerManagement.kext` for CPUs without native power management driver in macOS.

   Note 2: While this option is typically needed to disable `AppleIntelCpuPowerManagement` on unsupported platforms, it can also be used to disable this kext in other situations (e.g. with `Cpuid1Data` left blank).

4. `MaxKernel`
   Type: `plist string`
   Failsafe: Empty
   Description: Emulates CPUID and applies `DummyPowerManagement` on specified macOS version or older.

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

5. `MinKernel`
   Type: `plist string`
   Failsafe: Empty
   Description: Emulates CPUID and applies `DummyPowerManagement` on specified macOS version or newer.

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

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.

4. `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.

   This option allows integration with third-party operating system installation and upgrades (which may overwrite the `\EFI\BOOT\BOOTx64.efi` file). The BOOTx64.efi file is no longer used for bootstrapping OpenCore if a custom option is created. The custom path used for bootstrapping can be specified by using the `LauncherPath` option.

   Note 1: Some types of firmware may have NVRAM implementation flaws, no boot option support, or other incompatibilities. While unlikely, the use of this option may result in boot failures and should only be used exclusively on boards known to be compatible. Refer to acidanthera/bugtracker#1222 for some known issues affecting Haswell and other boards.

   Note 2: While NVRAM resets executed from OpenCore would not typically erase the boot option created in `Bootstrap`, executing NVRAM resets prior to loading OpenCore will erase the boot option. Therefore, for significant implementation updates, such as was the case with OpenCore 0.6.4, an NVRAM reset should be executed with `Bootstrap` disabled, after which it can be re-enabled.

5. `LauncherPath`
   Type: `plist string`
   Failsafe: `Default`
   Description: Launch path for the `LauncherOption` property.

   `Default` points to `OpenCore.efi`. User specified paths, e.g. `\EFI\SomeLauncher.efi`, can be used to provide custom loaders, which are supposed to load `OpenCore.efi` themselves.

6. `PickerAttributes`
   Type: `plist integer`
   Failsafe: `0`
   Description: Sets specific attributes for the OpenCore picker.

   Different OpenCore pickers may be configured through the attribute mask containing OpenCore-reserved (`BIT0`~`BIT15`) and OEM-specific (`BIT16`~`BIT31`) values.

   Current OpenCore values include:

   - `0x0001` — `OC_ATTR_USE_VOLUME_ICON`, provides custom icons for boot entries:
     ~~For `Tools`,~~ OpenCore will attempt loading a ~~custom icon and fallback to a default icon on failure:~~
     - ~~`ResetNVRAM` — `Resources\Image\ResetNVRAM.icns` — `ResetNVRAM.icns` from icons directory.~~
     - ~~`Tools\<TOOL_RELATIVE_PATH>.icns` — icon near the tool file with appended `.icns` extension.~~

~~For custom boot `Entries`, OpenCore will attempt loading a custom icon and fallback to the volume icon or the default icon on failure:~~
  - ~~`<ENTRY_PATH>.icns` — icon near the entry file with appended `.icns` extension.~~

~~For all other entries, OpenCore will attempt loading a volume icon~~ volume icon by searching as follows, and will fallback to the default icon on failure:
  - `.VolumeIcon.icns` file at `Preboot` volume in per-volume directory (`/System/Volumes/Preboot/{GUID}/` when mounted at the default location within macOS) for APFS (if present).
  - `.VolumeIcon.icns` file at the `Preboot` volume root (`/System/Volumes/Preboot/`, when mounted at the default location within macOS) for APFS (otherwise).
  - `.VolumeIcon.icns` file at the volume root for other filesystems.

  Note 1: The Apple picker partially supports placing a volume icon file at the operating system's `Data` volume root, `/System/Volumes/Data/`, when mounted at the default location within macOS. This approach is flawed: the file is neither accessible to OpenCanopy nor to the Apple picker when FileVault 2, which is meant to be the default choice, is enabled. Therefore, OpenCanopy does not attempt supporting Apple's approach. A volume icon file may be placed at the root of the `Preboot` volume for compatibility with both OpenCanopy and the Apple picker, or use the `Preboot` per-volume location as above with OpenCanopy as a preferred alternative to Apple's approach.

  Note 2: Be aware that using a volume icon on any drive overrides the normal OpenCore picker behaviour for that drive of selecting the appropriate icon depending on whether the drive is internal or external.

- `0x0002` — `OC_ATTR_USE_DISK_LABEL_FILE`, provides custom rendered titles for boot entries:
  - `.disk_label` (`.disk_label_2x`) file near bootloader for all filesystems.
  - `<TOOL_NAME>.lbl` (`<TOOL_NAME>.l2x`) file near tool for `Tools`.

  Prerendered labels can be generated via the `disklabel` utility or the `bless` command. When disabled or missing text labels, (`.contentDetails` or `.disk_label.contentDetails`) are to be rendered instead.
- `0x0004` — `OC_ATTR_USE_GENERIC_LABEL_IMAGE`, provides predefined label images for boot entries without custom entries. This may however give less detail for the actual boot entry.
- `0x0008` — `OC_ATTR_HIDE_THEMED_ICONS`, prefers builtin icons for certain icon categories to match the theme style. For example, this could force displaying the builtin Time Machine icon. Requires `OC_ATTR_USE_VOLUME_ICON`.
- `0x0010` — `OC_ATTR_USE_POINTER_CONTROL`, enables pointer control in the OpenCore picker when available. For example, this could make use of mouse or trackpad to control UI elements.
- `0x0020` — `OC_ATTR_SHOW_DEBUG_DISPLAY`, enable display of additional timing and debug information, in Builtin picker in `DEBUG` and `NOOPT` builds only.
- `0x0040` — `OC_ATTR_USE_MINIMAL_UI`, use minimal UI display, no Shutdown or Restart buttons, affects OpenCanopy and builtin picker.
- `0x0080` — `OC_ATTR_USE_FLAVOUR_ICON`, provides flexible boot entry content description, suitable for picking the best media across different content sets:
  When enabled, the entry icon in OpenCanopy and the audio assist entry sound in OpenCanopy and builtin boot picker are chosen by something called content flavour. To determine content flavour the following algorithm is used:
  - For a Tool the value is read from `Flavour` field.
  - For an automatically discovered entry it is read from the `.contentFlavour` file next to the bootloader, if present.
  - For a custom entry it is read from the `.contentFlavour` file next to the bootloader if `Flavour` is `Auto`, otherwise specified via the `Flavour` value itself.
  - If read flavour is `Auto` or there is no `.contentFlavour`, entry flavour is chosen based on the entry type (e.g. Windows automatically gets Windows flavour).

  The Flavour value is a sequence of `:` separated names limited to 64 characters of printable 7-bit ASCII. This is designed to support up to approximately five names. Each name refers to a flavour, with the first name having the highest priority and the last name having the lowest priority. Such a structure allows describing an entry in a more specific way, with icons selected flexibly depending on support by the audio-visual pack. A missing audio or icon file means the next flavour should be tried, and if all are missing the choice happens based on the type of the entry. Example flavour values: `BigSur:Apple`, `Windows10:Windows`, `OpenShell:UEFIShell:Shell`.

  Using flavours means that you can switch between icon sets easily, with the flavour selecting the best available icons from each set. E.g. specifying icon flavour `Debian:Linux` will use the icon `Debian.icns` if

provided, then will try `Linux.icns`, then will fall back to the default for an OS, which is `HardDrive.icns`.

Things to keep in mind:
- For security reasons `Ext<Flavour>.icns` and `<Flavour>.icns` are both supported, and only `Ext<Flavour>.icns` will be used if the entry is on an external drive (followed by default fallback `ExtHardDrive.icns`).
- Where both apply `.VolumeIcon.icns` takes precence over `.contentFlavour`.
- In order to allow icons and audio assist to work correctly for tools (e.g. for UEFI Shell), system default boot entry icons (see `Docs/Flavours.md`) specified in the `Flavour` setting for `Tools` or `Entries` will continue to apply even when flavour is disabled. Non-system icons will be ignored in this case. In addition, the flavours `UEFIShell` and `NVRAMReset` are given special processing, identifying their respective tools to apply correct audio-assist, default builtin labels, etc.
- A list of recommended flavours is provided in `Docs/Flavours.md`.

7. `PickerAudioAssist`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enable screen reader by default in the OpenCore picker.

   For the macOS bootloader, screen reader preference is set in the `preferences.efires` archive in the `isVOEnabled.int32` file and is controlled by the operating system. For OpenCore screen reader support, this option is an independent equivalent. Toggling screen reader support in both the OpenCore picker and the macOS bootloader FileVault 2 login window can also be done by using the `Command` + `F5` key combination.

   Note: The screen reader requires working audio support. Refer to the `UEFI Audio Properties` section for details.

8. `PollAppleHotKeys`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enable `modifier hotkey` handling in the OpenCore picker.

   In addition to `action hotkeys`, which are partially described in the `PickerMode` section and are typically handled by Apple BDS, modifier keys handled by the operating system bootloader (`boot.efi`) also exist. These keys allow changing the behaviour of the operating system by providing different boot modes.

   On certain firmware, using modifier keys may be problematic due to driver incompatibilities. To workaround this problem, this option allows registering certain hotkeys in a more permissive manner from within the OpenCore picker. Such extensions include support for tapping on key combinations before selecting the boot item, and for reliable detection of the `Shift` key when selecting the boot item, in order to work around the fact that hotkeys which are continuously held during boot cannot be reliably detected 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+Enter`, `Shift+Index` — safe mode, may be used in combination with `CTRL+Enter`, `CTRL+Index`.

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

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

    Introducing a delay may give extra time to hold the right `action hotkey` sequence to, for instance, boot into recovery mode. On some platforms, setting this option to a minimum of `5000-10000` microseconds may be required to access `action hotkeys` due to the nature of the keyboard driver.

Type: `plist string`
Failsafe: `Auto`
Description: Choose specific icon set to be used for boot management.

~~The following values are supported~~An icon set is a directory path relative to `Resources\Image`, where the icons and an optional manifest are located. It is recommended for the artists to use provide their sets in the `Vendor\Set` format, e.g. `Acidanthera\GoldenGate`.

Sample resources provided as a part of OcBinaryData repository provide the following icon set:

- ~~`Auto`~~`Acidanthera\GoldenGate` — ~~Automatically select one setof icons based on the `DefaultBackground` colour~~macOS 11 styled icon set.
- ~~`Default`~~`Acidanthera\Syrah` — ~~Normal icon set(without prefix)~~macOS 10.10 styled icon set.
- ~~`Old`~~`Acidanthera\Chardonnay` — ~~Vintage icon set(`Old` filename prefix).~~ macOS 10.4 styled icon set.

For convenience purposes there also are predefined aliases:

- ~~`Modern`~~`Auto` — ~~Nouveau icon set (~~Automatically select one set of icons based on the ~~`Modern`~~`DefaultBackground` ~~filename prefix)~~colour: `Acidanthera\GoldenGate` for Syrah Black and `Acidanthera\Chardonnay` for Light Gray.
- ~~Other value~~ `Default` — ~~Custom icon set if supported by installed resources~~`Acidanthera\GoldenGate`.

8.4 Debug Properties

1. `AppleDebug`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Enable writing the `boot.efi` debug log to the OpenCore log.

   Note: This option only applies to 10.15.4 and newer.

2. `ApplePanic`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Save macOS kernel panic output to the OpenCore root partition.

   The file is saved as `panic-YYYY-MM-DD-HHMMSS.txt`. It is strongly recommended to set the `keepsyms=1` boot argument to see debug symbols in the panic log. In cases where it is not present, the `kpdescribe.sh` utility (bundled with OpenCore) may be used to partially recover the stacktrace.

   Development and debug kernels produce more useful kernel panic logs. Consider downloading and installing the `KernelDebugKit` from developer.apple.com when debugging a problem. To activate a development kernel, the boot argument `kcsuffix=development` should be added. Use the `uname -a` command to ensure that the current loaded kernel is a development (or a debug) kernel.

   In cases where the OpenCore kernel panic saving mechanism is not used, kernel panic logs may still be found in the `/Library/Logs/DiagnosticReports` directory.

   Starting with macOS Catalina, kernel panics are stored in JSON format and thus need to be preprocessed before passing to `kpdescribe.sh`:

   ```
   cat Kernel.panic | grep macOSProcessedStackshotData |
     python -c 'import json,sys;print(json.load(sys.stdin)["macOSPanicString"])'
   ```

3. `DisableWatchDog`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Some types of firmware may not succeed in booting the operating system quickly, especially in debug mode. This results in the watchdog timer aborting the process. This option turns off the watchdog timer.

4. `DisplayDelay`
   Type: `plist integer`
   Failsafe: `0`
   Description: Delay in microseconds executed after every printed line visible onscreen (i.e. console).

UEFI variable log does not include some messages and has no performance data. To maintain system integrity, the log size is limited to 32 kilobytes. Some types of firmware may truncate it much earlier or drop completely if they have no memory. Using the `non-volatile` flag will cause the log to be written to NVRAM flash after every printed line.

To obtain UEFI variable logs, use the following command in macOS:

```
nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-log |
  awk '{gsub(/%0d%0a%00/,"");gsub(/%0d%0a/,"\n")}1'
```

Warning: Certain firmware appear to have defective NVRAM garbage collection. As a result, they may not be able to always free space after variable deletion. Do not enable `non-volatile` NVRAM logging on such devices unless specifically required.

While the OpenCore boot log already contains basic version information including build type and date, this information may also be found in the `opencore-version` NVRAM variable even when boot logging is disabled.

File logging will create a file named `opencore-YYYY-MM-DD-HHMMSS.txt` (in UTC) under the EFI volume root with log contents (the upper case letter sequence is replaced with date and time from the firmware). Please be warned that some file system drivers present in firmware are not reliable and may corrupt data when writing files through UEFI. Log writing is attempted in the safest manner and thus, is very slow. Ensure that `DisableWatchDog` is set to `true` when a slow drive is used. Try to avoid frequent use of this option when dealing with flash drives as large I/O amounts may speed up memory wear and render the flash drive unusable quicker.

When interpreting the log, note that the lines are prefixed with a tag describing the relevant location (module) of the log line allowing better attribution of the line to the functionality.

The list of currently used tags is as follows.

Drivers and tools:

- `BMF` — OpenCanopy, bitmap font
- `BS` — Bootstrap
- `GSTT` — GoptStop
- `HDA` — AudioDxe
- `KKT` — KeyTester
- `MMDD` — MmapDump
- `OCPAVP` — PavpProvision
- `OCRST` — ResetSystem
- `OCUI` — OpenCanopy
- `OC` — OpenCore main, also OcMainLib
- `VMOPT` — VerifyMemOpt

Libraries:

- `AAPL` — OcDebugLogLib, Apple EfiBoot logging
- `OCABC` — OcAfterBootCompatLib
- `OCAE` — OcAppleEventLib
- `OCAK` — OcAppleKernelLib
- `OCAU` — OcAudioLib
- `OCA` —- OcAcpiLib
- `OCBP` — OcAppleBootPolicyLib
- `OCB` — OcBootManagementLib
- `OCLBT` — OcBlitLib
- `OCCL` — OcAppleChunkListLib
- `OCCPU` — OcCpuLib
- `OCC` — OcConsoleLib
- `OCDC` — OcDriverConnectionLib
- `OCDH` — OcDataHubLib
- `OCDI` — OcAppleDiskImageLib
- `OCDM` — OcDeviceMiscLib
- `OCFS` — OcFileLib

- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeatures
  Combined FirmwareFeatures and ExtendedFirmwareFeatures. Present on newer Macs to avoid extra parsing of SMBIOS tables.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeaturesMask
  Combined FirmwareFeaturesMask and ExtendedFirmwareFeaturesMask. Present on newer Macs to avoid extra parsing of SMBIOS tables.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_BID
  Hardware BoardProduct (e.g. Mac-35C1E88140C3E6CF). Not present on real Macs, but used to avoid extra parsing of SMBIOS tables, especially in boot.efi.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_MLB
  Hardware BoardSerialNumber. Override for MLB. Present on newer Macs (2013+ at least).
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_ROM
  Hardware ROM. Override for ROM. Present on newer Macs (2013+ at least).
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:SSN
  Serial number. Present on newer Macs (2013+ at least).
- 7C436110-AB2A-4BBB-A880-FE41995C9F82:prev-lang:kbd
  ASCII string defining default keyboard layout. Format is lang-COUNTRY:keyboard, e.g. ru-RU:252 for Russian locale and ABC keyboard. Also accepts short forms: ru:252 or ru:0 (U.S. keyboard, compatible with 10.9). Full decoded keyboard list from AppleKeyboardLayouts-L.dat can be found here. Using non-latin keyboard on 10.14 will not enable ABC keyboard, unlike previous and subsequent macOS versions, and is thus not recommended in case 10.14 is needed.
- 7C436110-AB2A-4BBB-A880-FE41995C9F82:security-mode
  ASCII string defining FireWire security mode. Legacy, can be found in IOFireWireFamily source code in IOFireWireController.cpp. It is recommended not to set this variable, which may speedup system startup. Setting to full is equivalent to not setting the variable and none disables FireWire security.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:UIScale
  One-byte data defining boot.efi user interface scaling. Should be **01** for normal screens and **02** for HiDPI screens.
- 7C436110-AB2A-4BBB-A880-FE41995C9F82:ForceDisplayRotationInEFI
  32-bit integer defining display rotation. Can be **0** for no rotation or any of 90, 180, 270 for matching rotation in degrees.
- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:DefaultBackgroundColor
  Four-byte BGRA data defining boot.efi user interface background colour. Standard colours include **BF BF BF 00** (Light Gray) and **00 00 00 00** (Syrah Black). Other colours may be set at user's preference.

9.5 Other Variables

The following variables may be useful for certain configurations or troubleshooting:

- 7C436110-AB2A-4BBB-A880-FE41995C9F82:boot-args
  Kernel arguments, used to pass configuration to Apple kernel and drivers. There are many arguments, which may be found by looking for the use of PE_parse_boot_argn function in the kernel or driver code. Some of the known boot arguments include:
  - acpi_layer=0xFFFFFFFF
  - acpi_level=0xFFFF5F (implies ACPI_ALL_COMPONENTS)
  - arch=i386 (force kernel architecture to i386, see KernelArch)
  - batman=VALUE (AppleSmartBatteryManager debug mask)
  - batman-nosmc=1 (disable AppleSmartBatteryManager SMC interface)
  - cpus=VALUE (maximum number of CPUs used)
  - debug=VALUE (debug mask)
  - io=VALUE (IOKit debug mask)
  - ioaccel_debug=VALUE (IOAccelerator debug mask)
  - keepsyms=1 (show panic log debug symbols)
  - kextlog=VALUE (kernel extension loading debug mask)
  - nvram-log=1 (enables AppleEFINVRAM logs)
  - nv_disable=1 (disables NVIDIA GPU acceleration)
  - nvda_drv=1 (legacy way to enable NVIDIA web driver, removed in 10.12)
  - npci=0x2000 (legacy, disables kIOPCIConfiguratorPFM64)

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 *):

`BootKicker*`	Enter Apple BootPicker menu (exclusive for Macs with compatible GPUs).
`ChipTune*`	Test BeepGen protocol and generate audio signals of different style and length.
`CleanNvram*`	Reset NVRAM alternative bundled as a standalone tool.
`GopStop*`	Test GraphicsOutput protocol with a simple scenario.
`KeyTester*`	Test keyboard input in `SimpleText` mode.
`MemTest86`	Memory testing utility.
`OpenControl*`	Unlock and lock back NVRAM protection for other tools to be able to get full NVRAM access when launching from OpenCore.
`OpenShell*`	OpenCore-configured `UEFI Shell` for compatibility with a broad range of firmware.
`PavpProvision`	Perform EPID provisioning (requires certificate data configuration).
`ResetSystem*`	Utility to perform system reset. Takes reset type as an argument: `coldreset`, `firmware`, `shutdown`, `warmreset`. Defaults to `coldreset`.
`RtcRw*`	Utility to read and write RTC (CMOS) memory.
`ControlMsrE2*`	Check `CFG Lock` (MSR `0xE2` write protection) consistency across all cores and change such hidden options on selected platforms.

11.4 OpenCanopy

OpenCanopy is a graphical OpenCore user interface that runs in `External PickerMode` and relies on OpenCorePkg `OcBootManagementLib` similar to the builtin text interface.

OpenCanopy requires graphical resources located in `Resources` directory to run. Sample resources (fonts and images) can be found in OcBinaryData repository. Customised icons can be found over the internet (e.g. here or there).

OpenCanopy provides full support for `PickerAttributes` and offers a configurable builtin icon set. The ~~default~~ chosen icon set ~~depends~~ may depend on the `DefaultBackgroundColor` variable value. ~~For Light Gray~~ Refer to ~~`Oldicon set will be used, for other colours -- the one without a prefix`~~`PickerVariant` for more details.

Predefined icons are saved in the `PickerVariant`-derived subdirectory of the `\EFI\OC\Resources\Image` directory. A full list of supported icons (in `.icns` format) is provided below. When optional icons are missing, the closest available icon will be used. External entries will use `Ext`-prefixed icon if available (e.g. `OldExtHardDrive.icns`).

Note: In the following all dimensions are normative for the 1x scaling level and shall be scaled accordingly for other levels.

- `Cursor` — Mouse cursor (mandatory, up to 144x144).
- `Selected` — Selected item (mandatory, 144x144).
- `Selector` — Selecting item (mandatory, up to 144x40).
- `Left` — Scrolling left (mandatory, 40x40).
- `Right` — Scrolling right (mandatory, 40x40).

The use of `System` protocols is more complicated. Typically, 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, such as the `MacPro5,1`, may have incompatible console output when using modern GPUs, and thus only `BuiltinGraphics` may work for them in such cases. NVIDIA GPUs may require additional firmware upgrades.

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

   Set to `Max` to attempt using the largest available console mode. This option is currently ignored as the `Builtin` text renderer only supports one console mode.

   Note: This field is best left empty on most types of firmware.

3. `Resolution`
   Type: `plist string`
   Failsafe: Empty (Maintain current screen resolution)
   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 `Max` to attempt using the 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 `Builtin` text renderer, FileVault 2 UEFI password interface, and boot screen logo. Refer to the Recommended Variables section for 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. `ForceResolution`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Forces `Resolution` to be set in cases where the desired resolution is not available by default, such as on legacy Intel GMA and first generation Intel HD Graphics (Ironlake/Arrandale). Setting `Resolution` to `Max` will try to pull the largest available resolution from the connected display's EDID.

   Note: This option depends on the `OC_FORCE_RESOLUTION_PROTOCOL` protocol being present. This protocol is currently only supported by `OpenDuetPkg`. The `OpenDuetPkg` implementation currently only supports Intel iGPUs.

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

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

6. `DirectGopRendering`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Use builtin graphics output protocol renderer for console.

   On certain firmware, such as on the `MacPro5,1`, this may provide better performance or fix rendering issues. However, this option is not recommended unless there is an obvious benefit as it may result in issues such as slower scrolling.

This renderer fully supports `AppleEg2Info` protocol and will provide screen rotation for all EFI applications. In order to provide seamless rotation compatibility with `EfiBoot`, builtin `AppleFramebufferInfo` should also be used, i.e. it may need to be overridden on Mac EFI.

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

   This option provides the GOP protocol via a UGA-based proxy for firmware that do not implement the protocol.

   Note: This option requires `ProvideConsoleGop` to be enabled.

8. `IgnoreTextInGraphics`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Some types of firmware output text onscreen in both graphics and text mode. This is typically unexpected as 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 a different mode from `Text`.

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

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

    Some types of firmware do not implement the legacy UGA protocol but this may be required for screen output by older EFI applications such as EfiBoot from 10.4.

11.12 ProtocolOverrides Properties

1. `AppleAudio`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Replaces Apple audio protocols with builtin versions.

    Apple audio protocols allow OpenCore and the macOS bootloader to play sounds and signals for screen reading or audible error reporting. Supported protocols are beep generation and VoiceOver. The VoiceOver protocol is specific to Gibraltar machines (T2) and is not supported before macOS High Sierra (10.13). Older macOS versions use the AppleHDA protocol (which is not currently implemented) instead.

    Only one set of audio protocols can be available at a time, so this setting should be enabled in order to enable audio playback in the OpenCore user interface on Mac systems implementing some of these protocols.

    Note: The backend audio driver needs to be configured in `UEFI Audio` section for these protocols to be able to stream audio.

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

    Note: This option is advisable on certain Macs, such as the `MacPro5,1`, that are APFS compatible but on which the Apple Boot Policy protocol has recovery detection issues.

3. `AppleDebugLog`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Replaces the Apple Debug Log protocol with a builtin version.

4. `AppleEg2Info`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Replaces the Apple EFI Graphics 2 protocol with a builtin version.

    Note: This protocol allows newer `EfiBoot` versions (at least 10.15) to expose screen rotation to macOS. Refer to `ForceDisplayRotationInEFI` variable description on how to set screen rotation angle.

5. `AppleFramebufferInfo`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Replaces the Apple Framebuffer Info protocol with a builtin version. This may be used to override framebuffer information on VMs and legacy Macs to improve compatibility with legacy EfiBoot such as the one in macOS 10.4.

    Note: The current implementation of this property results in it only being active when GOP is available (it is always equivalent to `false` otherwise).

6. `AppleImageConversion`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Replaces the Apple Image Conversion protocol with a builtin version.