OpenCore

Reference Manual (0.8.~~5~~.6)

[2022.10.06]

In addition to installing emulated NVRAM, this driver additionally installs an OpenCore compatible protocol enabling the following:

- NVRAM values are loaded from `NVRAM/nvram.plist` (or from `NVRAM/nvram.fallback` if it is present and `NVRAM/nvram.plist` is missing) on boot
- The Reset NVRAM option installed by the `ResetNvramEntry` driver removes the above files instead of affecting underlying NVRAM
- `CTRL+Enter` in the OpenCore bootpicker updates or creates `NVRAM/nvram.plist`

Recommended configuration settings for this driver:

- `OpenVariableRuntimeDxe.efi` loaded using `LoadEarly=true`~~(driver not required with OpenDuet)~~. OpenDuet users should not load this driver, as it is included in OpenDuet.
- `OpenRuntime.efi` specified after `OpenVariableRuntimeDxe.efi` (when applicable), also loaded using `LoadEarly=true` for correct operation of `RequestBootVarRouting`.
  - `RequestBootVarRouting` is never strictly needed while using emulated NVRAM, but it can be convenient to leave it set on a system which needs to switch between real and emulated NVRAM.
  - `RequestBootVarRouting` is never required on an OpenDuet system, since there are no BIOS-managed boot entries to protect, therefore on OpenDuet recommended settings are `LoadEarly=false` for `OpenRuntime.efi` and `RequestBootVarRouting=false`.
- `LegacySchema` populated.
  - For simpler testing (allows arbitrary test variables), and future-proofing against changes in the variables required by macOS updates, use `<string>*</string>` settings, as described in notes below.
  - For increased security, populate sections with known required keys only, as shown in OpenCore's sample `.plist` files.
- `ExposeSensitiveData` with at least bit `0x1` set to make `boot-path` variable containing the OpenCore EFI partition UUID available to the `Launchd.command` script.

Variable loading happens prior to the NVRAM `Delete` (and `Add`) phases. Unless `LegacyOverwrite` is enabled, it will not overwrite any existing variable. Variables allowed for loading and for saving with `CTRL+Enter` must be specified in `LegacySchema`.

In order to allow changes to NVRAM within macOS to be captured and saved, an additional script must be installed. An example of such script can be found in `Utilities/LogoutHook/Launchd.command`.

Note 1: This driver requires working FAT write support in firmware, and sufficient free space on the OpenCore EFI partition for up to three saved NVRAM files.

Note 2: The `nvram.plist` (and `nvram.fallback` if present) files must have a root `plist dictionary` type and contain two fields:

- `Version` — `plist integer`, file version, must be set to 1.
- `Add` — `plist dictionary`, equivalent to `Add` from `config.plist`.

Note 3: When setting up legacy NVRAM, it can be convenient to set `<string>*</string>` as the value for the following three GUID keys in `LegacySchema`:

- 36C28AB5-6566-4C50-9EBD-CBB920F83843
- 7C436110-AB2A-4BBB-A880-FE41995C9F82
- 8BE4DF61-93CA-11D2-AA0D-00E098032B8C

This enables all variables saved by `Launchd.command` to be saved to `nvram.plist`, therefore it allows all arbitrary user test variables (e.g. as set by `sudo nvram foo=bar`) to be saved. Using this permissive policy is also future-proof against any changes in the variables which need to be passed from macOS update setup to the `macOS Installer` stage, in order for it to succeed. Nevertheless, once emulated NVRAM is set up, only allowing known strictly required variables (as shown in OpenCore's sample `.plist` files) is considerably more secure. See also the following warning about the overall security of loading NVRAM variables from a non-vaulted file.

Warning: The ability to load NVRAM from a file on disk can be dangerous, as it passes unprotected data to firmware variable services. Only use when no hardware NVRAM implementation is provided by the firmware or when the NVRAM implementation available in firmware is incompatible or dangerously fragile (e.g. in a state where excessive use may cause bricked hardware).