OpenCore

Reference Manual (0.8.5.6)

[2022.10.27]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

   Note 1: A halt will only occur if bit `0` (i.e. enable logging) for `Target` under section `Misc->Debug` is set.

   Note 2: A halt will only occur after the configuration is loaded and logging is configured. If any log messages occur at the specified halt level in early log (i.e. before this), they will cause a halt when they are flushed to the log once it has been configured.

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

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

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

    Valid values:

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).

7. `PointerPollMax`
   Type: `plist integer`
   Failsafe: `0`
   Description: Configure maximum pointer polling period in ms.

   This is the maximum period the OpenCore builtin AppleEvent driver polls pointer devices (e.g. mice, trackpads) for motion events. The period is increased up to this value as long as the devices do not respond in time. The current implementation defaults to 80 ms. Setting `0` leaves this default unchanged.

   Certain trackpad drivers often found in Dell laptops can be very slow to respond when no physical movement happens. This can affect OpenCanopy and FileVault 2 user interface responsiveness and loading times. Increasing the polling periods can reduce the impact.

   Note: The OEM Apple implementation uses a polling rate of 2 ms.

8. `PointerPollMask`
   Type: `plist integer, 32 bit`
   Failsafe: `-1`
   Description: Configure indices of polled pointers.

   Selects pointer devices to poll for AppleEvent motion events. `-1` implies all devices. A bit sum is used to determine particular devices. E.g. to enable devices 0, 2, 3 the value will be `1+4+8` (corresponding powers of two). A total of 32 configurable devices is supported.

   Certain pointer devices can be present in the firmware even when no corresponding physical devices are available. These devices usually are placeholders, aggregate devices, or proxies. Gathering information from these devices may result in inaccurate motion activity in the user interfaces and even cause performance issues. Disabling such pointer devices is recommended for laptop setups having issues of this kind.

   The amount of pointer devices available in the system can be found in the log. Refer to `Found N pointer devices` message for more details.

   Note: Has no effect when using the OEM Apple implementation (see `AppleEvent` setting).

9. `PointerSpeedDiv`
   Type: `plist integer`
   Failsafe: `1`
   Description: Configure pointer speed divisor in the OpenCore re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see `AppleEvent` setting).

   Configures the divisor for pointer movements. The Apple OEM default value is `1`. `0` is an invalid value for this option.

   Note: The recommended value for this option is `1`. This value may optionally be modified in combination with `PointerSpeedMul`, according to user preference, to achieve customised mouse movement scaling.

10. `PointerSpeedMul`
    Type: `plist integer`
    Failsafe: `1`
    Description: Configure pointer speed multiplier in the OpenCore re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see `AppleEvent` setting).

    Configures the multiplier for pointer movements. The Apple OEM default value is `1`.

    Note: The recommended value for this option is `1`. This value may optionally be modified in combination with `PointerSpeedDiv`, according to user preference, to achieve customised mouse movement scaling.

11. `PointerDwellClickTimeout`
    Type: `plist integer`
    Failsafe: `0`
    Description: Configure pointer dwell-clicking single left click timeout in milliseconds in the OpenCore re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see `AppleEvent` setting).

When the timeout expires, a single left click is issued at the current position. `0` indicates the timeout is disabled.

12. `PointerDwellDoubleClickTimeout`
    Type: `plist integer`
    Failsafe: `0`
    Description: Configure pointer dwell-clicking single left double click timeout in milliseconds in the OpenCore re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see `AppleEvent` setting).

    When the timeout expires, a single left double click is issued at the current position. `0` indicates the timeout is disabled.

13. `PointerDwellRadius`
    Type: `plist integer`
    Failsafe: `0`
    Description: Configure pointer dwell-clicking tolerance radius in pixels in the OpenCore re-implementation of the Apple Event protocol. Has no effect when using the OEM Apple implementation (see `AppleEvent` setting).

    The radius is scaled by `UIScale`. When the pointer leaves this radius, the timeouts for `PointerDwellClickTimeout` and `PointerDwellDoubleClickTimeout` are reset and the new position is the centre for the new dwell-clicking tolerance radius.

11.13 Audio Properties

1. `AudioCodec`
   Type: `plist integer`
   Failsafe: `0`
   Description: Codec address on the specified audio controller for audio support.

   This typically contains the first audio codec address on the builtin analog audio controller (`HDEF`). Audio codec addresses, e.g. `2`, can be found in the debug log (marked in bold-italic):

   ```
   OCAU: 1/3 PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x1)/VenMsg(<redacted>,00000000) (4 outputs)
   OCAU: 2/3 PciRoot(0x0)/Pci(0x3,0x0)/VenMsg(<redacted>,00000000) (1 outputs)
   OCAU: 3/3 PciRoot(0x0)/Pci(0x1B,0x0)/VenMsg(<redacted>,02000000) (7 outputs)
   ```

   As an alternative, this value can be obtained from `IOHDACodecDevice` class in I/O Registry containing it in `IOHDACodecAddress` field.

2. `AudioDevice`
   Type: `plist string`
   Failsafe: Empty
   Description: Device path of the specified audio controller for audio support.

   This typically contains builtin analog audio controller (`HDEF`) device path, e.g. `PciRoot(0x0)/Pci(0x1b,0x0)`. The list of recognised audio controllers can be found in the debug log (marked in bold-italic):

   ```
   OCAU: 1/3 PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x1)/VenMsg(<redacted>,00000000) (4 outputs)
   OCAU: 2/3 PciRoot(0x0)/Pci(0x3,0x0)/VenMsg(<redacted>,00000000) (1 outputs)
   OCAU: 3/3 PciRoot(0x0)/Pci(0x1B,0x0)/VenMsg(<redacted>,02000000) (7 outputs)
   ```

   If using `AudioDxe`, the available controller device paths are also output on lines formatted like this:

   ```
   HDA: Connecting controller - PciRoot(0x0)/Pci(0x1B,0x0)
   ```

   Finally, `gfxutil -f HDEF` command can be used in macOS to obtain the device path.

   Specifying an empty device path results in the first available codec and audio controller being used. The value of `AudioCodec` is ignored in this case. This can be a convenient initial option to try to get UEFI audio working. Manual settings as above will be required when this default value does not work.

3. `AudioOutMask`
   Type: `plist integer`