OpenCore

Reference Manual (0.6.6.7)

[2021.02.04]

10 PlatformInfo

Platform information is comprised of several identification fields generated or filled manually to be compatible with macOS services. The base part of the configuration may be obtained from `AppleModels`, which itself generates a set of interfaces based on a database in YAML format. These fields are written to three select destinations:

- SMBIOS
- Data Hub
- NVRAM

Most of the fields specify the overrides in SMBIOS, and their field names conform to EDK2 SmBios.h header file. However, several important fields reside in Data Hub and NVRAM. Some of the values can be found in more than one field and/or destination, so there are two ways to control their update process: manual, where all the values are specified (the default), and semi-automatic, where (`Automatic`) only select values are specified, and later used for system configuration.

To inspect SMBIOS contents dmidecode utility can be used. Version with macOS specific enhancements can be downloaded from Acidanthera/dmidecode.

10.1 Properties

1. `Automatic`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Generate PlatformInfo based on `Generic` section instead of using values from `DataHub`, `NVRAM`, and `SMBIOS` sections.

   Enabling this option is useful when `Generic` section is flexible enough:

   - When enabled `SMBIOS`, `DataHub`, and `PlatformNVRAM` data is unused.
   - When disabled `Generic` section is unused.

   Warning: It is strongly discouraged set this option to `false` when intending to update platform information. The only reason to do that is when doing minor correction of the SMBIOS present and similar. In all other cases not using `Automatic` may lead to hard to debug errors.

2. `CustomMemory`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Use custom memory configuration defined in the `Memory` section. This completely replaces any existing memory configuration in SMBIOS, and is only active when `UpdateSMBIOS` is set to `true`.

3. `UpdateDataHub`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Update Data Hub fields. These fields are read from `Generic` or `DataHub` sections depending on `Automatic` value.

   Note: The implementation of the Data Hub protocol in EFI firmware on essentially all systems, including Apple hardware, means that existing Data Hub entries cannot be overridden, while new entries are added to the end with macOS ignoring them. You can work around this by reinstalling the Data Hub protocol using the `ProtocolOverrides` section. Refer to the `DataHub` protocol override description for details.

4. `UpdateNVRAM`
   Type: `plist boolean`
   Failsafe: `false`
   Description: Update NVRAM fields related to platform information.

   These fields are read from `Generic` or `PlatformNVRAM` sections depending on `Automatic` value. All the other fields are to be specified with `NVRAM` section.

   If `UpdateNVRAM` is set to `false` the aforementioned variables can be updated with `NVRAM` section. If `UpdateNVRAM` is set to `true` the behaviour is undefined when any of the fields are present in `NVRAM` section.

Failsafe: `false`
Description: Reinstalls Apple Debug Log protocol with a builtin version.

4. `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.

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

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

7. `AppleImg4Verification`
Type: `plist boolean`
Failsafe: `false`
Description: Reinstalls Apple IMG4 Verification protocol with a builtin version. This protocol is used to verify `im4m` manifest files used by Apple Secure Boot.

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

9. `AppleRtcRam`
Type: `plist boolean`
Failsafe: `false`
Description: Reinstalls Apple RTC RAM protocol with builtin version.

   Note: Builtin version of Apple RTC RAM protocol may filter out I/O attempts to select RTC memory addresses. The list of addresses can be specified in `4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:rtc-blacklist` variable as a data array.

10. `AppleSecureBoot`
Type: `plist boolean`
Failsafe: `false`
Description: Reinstalls Apple Secure Boot protocol with a builtin version.

11. `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.

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

13. `DataHub`
Type: `plist boolean`
Failsafe: `false`
Description: Reinstalls Data Hub protocol with a builtin version. ~~This will delete all previous properties~~

Note: This will discard all previous entries if the protocol was already installed, so all properties required for safe operation of the system must be specified in your configuration.

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

    Note: This will discard all previous entries if the protocol was already installed, so all properties required for safe operation of the system must be specified in your configuration.

15. `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.

    Note: Several virtual machines including VMware may have corrupted cursor image in HiDPI mode and thus may also require this setting to be enabled.

16. `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.

17. `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.

18. `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.

11.12 Quirks Properties

1. `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 you use UEFI Secure Boot.

2. `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 should be found in some future. Expect 3 to 5 seconds to be adequate when this quirk is needed.

3. `IgnoreInvalidFlexRatio`
    Type: `plist boolean`