



OpenCore

Reference Manual (~~0.5.9~~0.6.0)

[2020.06.05]

1 Introduction

This document provides information on OpenCore user configuration file format used to setup the correct functioning of macOS operating system. It is to be read as the official clarification of expected OpenCore behaviour. All deviations, if found in published OpenCore releases, shall be considered documentation or implementation bugs, and are requested to be reported through Acidanthera Bugtracker. [Errata sheet is available in](#) OpenCorePkg repository.

This document is structured as a specification, and is not meant to provide a step by step algorithm for configuring end-user board support package (BSP). The intended audience of the document are programmers and engineers with basic understanding of macOS internals and UEFI functioning. For these reasons this document is available exclusively in English, and all other sources or translations of this document are unofficial and may contain errors.

Third-party articles, utilities, books, and alike may be more useful for a wider audience as they could provide guide-like material. However, they are prone to their authors' preferences, tastes, this document misinterpretation, and essential obsolescence. In case you use these sources, for example, Dortania's OpenCore Desktop Guide and related material, please ensure to follow this document for every made decision and judge its consequences.

Be warned that regardless of the sources used you are required to fully understand every dedicated OpenCore configuration option and concept prior to reporting any issues in Acidanthera Bugtracker.

1.1 Generic Terms

- **plist** — Subset of ASCII Property List format written in XML, also known as XML plist format version 1. Uniform Type Identifier (UTI): `com.apple.property-list`. Plists consist of **plist** objects, which are combined to form a hierarchical structure. Due to plist format not being well-defined, all the definitions of this document may only be applied after plist is considered valid by running `plutil -lint`. External references: <https://www.apple.com/DTDs/PropertyList-1.0.dtd>, `man plutil`.
- **plist type** — plist collections (**plist array**, **plist dictionary**, **plist key**) and primitives (**plist string**, **plist data**, **plist date**, **plist boolean**, **plist integer**, **plist real**).
- **plist object** — definite realisation of **plist type**, which may be interpreted as value.
- **plist array** — array-like collection, conforms to **array**. Consists of zero or more **plist** objects.
- **plist dictionary** — map-like (associative array) collection, conforms to **dict**. Consists of zero or more **plist** keys.
- **plist key** — contains one **plist** object going by the name of **plist key**, conforms to **key**. Consists of printable 7-bit ASCII characters.
- **plist string** — printable 7-bit ASCII string, conforms to **string**.
- **plist data** — base64-encoded blob, conforms to **data**.
- **plist date** — ISO-8601 date, conforms to **date**, unsupported.
- **plist boolean** — logical state object, which is either true (1) or false (0), conforms to **true** and **false**.
- **plist integer** — possibly signed integer number in base 10, conforms to **integer**. Fits in 64-bit unsigned integer in two's complement representation, unless a smaller signed or unsigned integral type is explicitly mentioned in specific **plist** object description.
- **plist real** — floating point number, conforms to **real**, unsupported.
- **plist metadata** — value cast to data by the implementation. Permits passing **plist string**, in which case the result is represented by a null-terminated sequence of bytes (aka C string), **plist integer**, in which case the result is represented by 32-bit little endian sequence of bytes in two's complement representation, **plist boolean**, in which case the value is one byte: 01 for **true** and 00 for **false**, and **plist data** itself. All other types or larger integers invoke undefined behaviour.

Type	Value
<code>plist integer</code>	0 (<integer>0</integer>)
<code>plist boolean</code>	False (<false/>)
<code>plist tristate</code>	False (<false/>)

2.3 Configuration Structure

OC `config` is separated into following sections, which are described in separate sections of this document. By default it is tried to not enable anything and optionally provide kill switches with `Enable` property for `plist dict` entries. In general the configuration is written idiomatically to group similar actions in subsections:

- `Add` provides support for data addition. Existing data will not be overridden, and needs to be handled separately with `Delete` if necessary.
- `Delete` provides support for data removal.
- `Patch` provides support for data modification.
- `Quirks` provides support for specific hacks.

Root configuration entries consist of the following:

- `ACPI`
- `Booter`
- `DeviceProperties`
- `Kernel`
- `Misc`
- `NVRAM`
- `PlatformInfo`
- `UEFI`

It is possible to perform basic validation of the configuration by using `ConfigValidityocvalidate` utility. Please note, that `ConfigValidityocvalidate` must match the used OpenCore release and may not be able to detect all configuration flaws present in the file.

Note: Currently most properties try to have defined values even if not specified in the configuration for safety reasons. This behaviour should not be relied upon, and all fields must be properly specified in the configuration.