OpenCore
Referenzhandbuch (0.8.0)

[2022.04.16]

Contents

1 Einleitung

Dieses Dokument enthält Informationen über das Format der OpenCore-Benutzerkonfigurationsdatei, die verwendet wird, um das korrekte Funktionieren des macOS-Betriebssystems einzurichten. Es ist als offizielle Klarstellung des erwarteten OpenCore-Verhaltens zu verstehen. Alle Abweichungen, die in veröffentlichten OpenCore-Versionen gefunden werden, sind als Dokumentations- oder Implementierungsprobleme zu betrachten, die über den Acidanthera Bugtracker gemeldet werden sollten. Ein Errata Sheet ist im OpenCorePkg Repository verfügbar.

Dieses Dokument ist als Spezifikation strukturiert und soll keine Schritt-für-Schritt-Anleitung für die Konfiguration eines Board Support Packages (BSP) für den Endbenutzer darstellen. Die Zielgruppe des Dokuments sind Programmierer und Ingenieure mit einem grundlegenden Verständnis von macOS-Interna und UEFI-Funktionalität. Aus diesen Gründen ist dieses Dokument ausschließlich in englischer Sprache verfügbar, und alle anderen Quellen oder Übersetzungen dieses Dokuments sind inoffiziell und können Fehler enthalten.

Artikel, Hilfsprogramme, Bücher und Ähnliches von Drittanbietern können für ein breiteres Publikum nützlicher sein, da sie leitfadenähnliches Material bieten können. Sie unterliegen jedoch den Vorlieben ihrer Autoren, Fehlinterpretationen dieses Dokuments und unvermeidlicher Veralterung. Wenn Sie solche Quellen verwenden, wie z.B. den OpenCore Install Guide von Dortania und verwandtes Material, beziehen Sie sich bitte bei jeder Entscheidung auf dieses Dokument und bewerten Sie die möglichen Auswirkungen neu.

Bitte beachten Sie, dass unabhängig von den verwendeten Quellen die Benutzer verpflichtet sind, jede OpenCore-Konfigurationsoption und die dahinter stehenden Prinzipien vollständig zu verstehen, bevor sie Probleme in den Acidanthera Bugtracker eintragen.

Anmerkung: Die Erstellung dieses Dokuments wäre ohne die unschätzbaren Beiträge anderer Personen nicht möglich gewesen: Andrey1970, Goldfish64, dakanji, PMheart, und einige andere. Die vollständige Liste ist in der OpenCorePkg-Historie zu finden.

1.1 Allgemeine Begriffe

- plist - Untermenge des ASCII-Eigenschaftslistenformats, geschrieben in XML, auch bekannt als XML plist Format Version 1. Uniform Type Identifier (UTI): com.apple.property-list. Plists bestehen aus Plist-Objekten, die zu einer hierarchischen Struktur zusammengefasst sind. Da das plist-Format nicht genau definiert ist, können alle Definitionen dieses Dokuments erst angewendet werden, nachdem plist durch Ausführen von plutil -lint als gültig eingestuft wurde. Externe Referenzen: https://www.apple.com/DTDs/PropertyList-1.0.dtd, man plutil.
- plist type - plist Sammlungen (plist array, plist dictionary, plist key) und Primitive (plist string, plist data, plist date, plist boolean, plist integer, plist real).
- plist object - endgültige Realisierung des plist-Typs, der als Wert interpretiert werden kann.
- plist array - Array-ähnliche Sammlung, entspricht einem Array. Besteht aus null oder mehr plist-Objekten.
- plist dictionary - map-ähnliche (assoziatives Array) Sammlung, entspricht dict. Besteht aus null oder mehr plist Schlüsseln.
- plist key - enthält ein plist-Objekt mit dem Namen plist key, konform zu key. Besteht aus druckbaren 7-Bit-ASCII-Zeichen.
- plist string - druckbare 7-Bit-ASCII-Zeichenfolge, konform zu string.
- plist data - base64-kodierter Blob, entspricht data.
- plist date - ISO-8601-Datum, entspricht date, wird nicht unterstützt.
- plist boolean - logisches Zustandsobjekt, das entweder wahr (1) oder falsch (0) ist, konform zu true und false.
- plist integer - möglicherweise vorzeichenbehaftete Ganzzahl zur Basis 10, konform mit integer. Passt in eine 64-Bit-Ganzzahl ohne Vorzeichen in Zweierkomplement-Darstellung, es sei denn, ein kleinerer vorzeichenbehafteter oder vorzeichenloser ganzzahliger Typ wird in der spezifischen plist-Objektbeschreibung ausdrücklich erwähnt.
- plist real - Fließkommazahl, konform zu real, nicht unterstützt.

- plist multidata - Wert, der von der Implementierung in Daten umgewandelt wird. Erlaubt die Übergabe von plist string, in diesem Fall wird das Ergebnis durch eine null-terminierte Folge von Bytes (C-String) dargestellt, plist integer, in diesem Fall wird das Ergebnis durch eine 32-Bit Little-Endian-Folge von Bytes in Zweierkomplement-Darstellung dargestellt, plist boolean, in diesem Fall ist der Wert ein Byte: 01 für true und 00 für false, und plist data selbst. Alle anderen Typen oder größere Ganzzahlen führen zu undefiniertem Verhalten.

2 Konfiguration
2.1 Begriffe der Konfiguration
- OC config - OpenCore Konfigurationsdatei im plist-Format mit dem Namen config.plist. Sie bietet eine erweiterbare Möglichkeit, OpenCore zu konfigurieren und ist so strukturiert, dass sie in mehrere benannte Abschnitte aufgeteilt ist, die sich unter dem Root-Plist-Dictionary befinden. Diese Abschnitte können plist array oder plist dictionary Typen haben und werden in den entsprechenden Abschnitten dieses Dokuments beschrieben.
- gültiger Schlüssel - plist-Schlüsselobjekt der OC-Konfiguration, das in diesem Dokument oder seinen zukünftigen Überarbeitungen beschrieben wird. Neben den explizit beschriebenen gültigen Schlüsseln werden auch Schlüssel, die mit dem #-Symbol beginnen (z.B. #Hello), als gültige Schlüssel betrachtet, und obwohl sie sich wie Kommentare verhalten, d.h. ihre Werte effektiv verwerfen, müssen sie dennoch gültige Plist-Objekte sein. Alle anderen Plist-Schlüssel sind nicht gültig, und ihr Vorhandensein führt zu undefiniertem Verhalten.
- valid value - gültiges Plist-Objekt der in diesem Dokument beschriebenen OC-Config, das alle zusätzlichen Anforderungen in den spezifischen Plist-Objekt-Beschreibungen erfüllt, falls vorhanden.
- ungültiger Wert - gültiges Plist-Objekt der in diesem Dokument beschriebenen OC-Config, das von einem anderen Plist-Typ ist, nicht mit den zusätzlichen Anforderungen in den spezifischen Plist-Objekt-Beschreibungen übereinstimmt (z.B. Wertebereich) oder in der entsprechenden Collection fehlt. Ungültige Werte werden mit oder ohne Fehlermeldung als jeder mögliche Wert dieses Plist-Objekts auf unbestimmte Weise gelesen (d.h. die Werte sind möglicherweise nicht bei allen Neustarts gleich). Während das Lesen eines ungültigen Wertes dem Lesen bestimmter definierter gültiger Werte entspricht, kann die Anwendung inkompatibler Werte auf das Host-System zu undefiniertem Verhalten führen.
- optionaler Wert - gültiger Wert der in diesem Dokument beschriebenen OC-Konfiguration, der auf eine bestimmte, in der Beschreibung des spezifischen Plist-Objekts angegebene Weise gelesen wird (anstelle eines ungültigen Werts), wenn er in der OC-Konfiguration nicht vorhanden ist. Alle anderen Fälle von ungültigem Wert gelten weiterhin. Sofern nicht ausdrücklich als optionaler Wert gekennzeichnet, muss jeder andere Wert vorhanden sein und wird bei Fehlen als ungültiger Wert gelesen.
- fatal behaviour - Verhalten, das zum Abbruch des Bootvorgangs führt. Implementierungen müssen verhindern, dass der Boot-Prozess fortgesetzt wird, bis das Host-System neu gestartet wird. Es ist zulässig, aber nicht erforderlich, in solchen Fällen kalte Neustarts durchzuführen oder Warnmeldungen anzuzeigen.
- undefiniertes Verhalten - Verhalten, das in diesem Dokument nicht vorgeschrieben ist. Implementierungen können beliebige Maßnahmen ergreifen, einschließlich, aber nicht beschränkt auf Maßnahmen, die mit fatalem Verhalten, der Annahme eines Zustands oder Werts oder der Nichtberücksichtigung zugehöriger Zustände oder Werte verbunden sind. Dies gilt jedoch nur, wenn diese Maßnahmen die Systemintegrität nicht beeinträchtigen.
2.2 Verarbeitung der Konfiguration

Es wird garantiert, dass die OC-Konfigurationsdatei mindestens einmal verarbeitet wird, wenn sie gefunden wird. Vorbehaltlich des OpenCore-Bootstrapping-Mechanismus kann das Vorhandensein mehrerer OC-Config-Dateien zum Lesen einer beliebigen dieser Dateien führen. Es ist zulässig, dass keine OC-Config-Datei auf der Platte vorhanden ist. In solchen Fällen müssen, wenn die Implementierung den Bootvorgang nicht abbricht, alle Werte den Regeln für ungültige Werte und optionale Werte folgen.
Für die OC-Config-Datei gelten Beschränkungen hinsichtlich der Größe, der Verschachtelungsebenen und der Anzahl der Schlüssel:
- Die Größe der OC-Config-Datei darf 32 MB nicht überschreiten.
- Die OC-Config-Datei darf nicht mehr als 32 Verschachtelungsebenen haben.
- Die OC-Config-Datei darf bis zu 32.768 XML-Knoten in jedem plist-Objekt enthalten.
- Ein Plist-Dictionary-Element wird als ein Knotenpaar gezählt.
Das Lesen missgestalteter OC-Config-Dateien führt zu undefiniertem Verhalten. Beispiele für fehlerhafte OC-Konfigurationsdateien sind
die folgenden:
- OC-Konfigurationsdateien, die nicht der DTD PLIST 1.0 entsprechen.
- OC-Konfigurationsdateien mit nicht unterstützten oder nicht konformen Plist-Objekten, die in diesem Dokument gefunden wurden. - OC-Konfigurationsdateien, die gegen Beschränkungen hinsichtlich Größe, Verschachtelungsebenen und Anzahl der Schlüssel verstoßen.
Es wird empfohlen, aber nicht erforderlich, das Laden von fehlerhaften OC-Config-Dateien abzubrechen und so fortzufahren, als ob keine OC-Config-Datei vorhanden wäre. Aus Gründen der Vorwärtskompatibilität ist es empfehlenswert, aber nicht erforderlich, dass die Implementierung vor der Verwendung ungültiger Werte warnt. Übersetzt mit www.DeepL.com/Translator (kostenlose Version)

S. 4

Es wird empfohlen, bei der Interpretation ungültiger Werte die folgende Konvention einzuhalten, sofern sie anwendbar ist:

Type	**Value**
`Plist string`	Empty string (`<string></string>`)
`Plist data`	Empty data (`<data></data>`)
`Plist integer`	**`0 (<integer>0</integer>)`**
`Plist boolean`	False (`<false/>`)
`Plist tristate`	False (`<false/>`)

2.3 Aufbau der Konfiguration
Die OC-Config-Datei ist in Unterabschnitte unterteilt, die in separaten Abschnitten dieses Dokuments beschrieben werden, und ist so konzipiert, dass sie versucht, nichts standardmäßig zu

aktivieren und Kill-Switches über eine Enable-Eigenschaft für plist-dict-Einträge bereitzustellen, die optionale Plugins und Ähnliches darstellen.
Die Datei ist so strukturiert, dass verwandte Elemente in Unterabschnitten wie folgt gruppiert sind:
- Add bietet Unterstützung für das Hinzufügen von Daten. Vorhandene Daten werden nicht überschrieben, sondern müssen bei Bedarf separat mit Delete behandelt werden.
- Delete bietet Unterstützung für das Entfernen von Daten.
- Patch bietet Unterstützung für die Änderung von Daten.
- Quirks bietet Unterstützung für spezielle Workarounds.
Die Root-Konfigurationseinträge bestehen aus den folgenden Elementen:

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

Eine grundlegende Validierung einer OC-Konfigurationsdatei ist mit dem Dienstprogramm ocvalidate möglich. Bitte beachten Sie, dass die verwendete Version von ocvalidate mit der OpenCore-Version übereinstimmen muss und dass ungeachtet dessen möglicherweise nicht alle in einer OC-Config-Datei vorhandenen Konfigurationsprobleme erkannt werden.
Hinweis: Um die Systemintegrität zu wahren, haben Eigenschaften typischerweise vordefinierte Werte, auch wenn solche vordefinierten Werte nicht in der OC-Konfigurationsdatei angegeben sind. Allerdings müssen alle Eigenschaften explizit in der OC-Config-Datei angegeben werden, und auf dieses Verhalten sollte man sich nicht verlassen.

3 Setup

3.1 Directory Structure

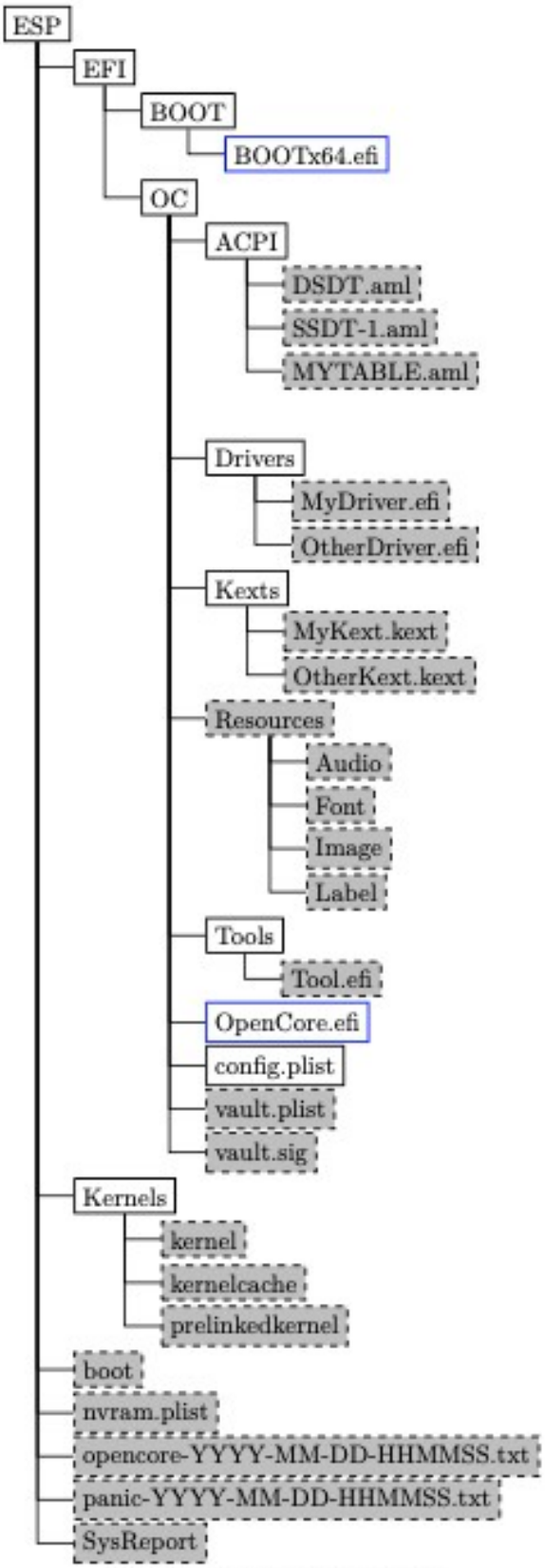

Figure 1. Directory Structure

Wenn das Verzeichnis-Boot verwendet wird, sollte die verwendete Verzeichnisstruktur den Beschreibungen im Abschnitt Verzeichnisstruktur entsprechen. S. 6

Verfügbare Einträge sind:
- BOOTx64.efi oder BOOTIa32.efi
Initiale Bootstrap-Lader, die OpenCore.efi laden. BOOTx64.efi wird von der Firmware standardmäßig in Übereinstimmung mit der UEFI-Spezifikation geladen. Sie kann jedoch auch umbenannt und an einem benutzerdefinierten Ort abgelegt werden, damit OpenCore neben Betriebssystemen wie Windows koexistieren kann, die BOOTx64.efi-Dateien als ihre Lader verwenden. Siehe die LauncherOption Eigenschaft für Details.
- Boot
Duet-Bootstrap-Loader, der die UEFI-Umgebung auf älterer BIOS-Firmware initialisiert und OpenCore.efi ähnlich wie andere Bootstrap-Loader lädt. Ein moderner Duet-Bootstrap-Loader wird standardmäßig OpenCore.efi auf derselben Partition laden, wenn vorhanden.
- ACPI
Verzeichnis zum Speichern zusätzlicher ACPI-Informationen für den ACPI-Abschnitt.
- Treiber
Verzeichnis zum Speichern zusätzlicher UEFI-Treiber für den UEFI-Bereich.
- Kexts
Verzeichnis zum Speichern zusätzlicher Kernel-Informationen für den Abschnitt Kernel.
- Ressourcen
Verzeichnis zum Speichern von Medienressourcen, wie z. B. Audiodateien für die Unterstützung von Bildschirmlesern. Einzelheiten finden Sie im Abschnitt UEFI-Audioeigenschaften. Dieses Verzeichnis enthält auch Bilddateien für die grafische Benutzeroberfläche. Weitere Informationen finden Sie im Abschnitt OpenCanopy.
- Werkzeuge
Verzeichnis zum Speichern zusätzlicher Tools.
- OpenCore.efi
Haupt-Boot-Anwendung, die für das Laden des Betriebssystems verantwortlich ist. Das Verzeichnis, in dem sich OpenCore.efi befindet, wird als Stammverzeichnis bezeichnet, das standardmäßig auf EFI\OC eingestellt ist. Beim direkten Start von OpenCore.efi oder über einen benutzerdefinierten Launcher werden jedoch auch andere Verzeichnisse mit OpenCore.efi-Dateien unterstützt.
- config.plist OC Config.
- vault.plist
Hashes für alle Dateien, die potentiell von OC Config geladen werden können.
- vault.sig
Signatur für vault.plist.
- SysReport
Verzeichnis mit Systemberichten, die mit der Option SysReport erstellt werden.
- nvram.plist
OpenCore-Variablen-Importdatei.
- opencore-YYYY-MM-DD-HHMMSS.txt OpenCore-Protokolldatei.
- panic-YYYY-MM-DD-HHMMSS.txt Kernel-Panik-Protokolldatei.
Hinweis: Es ist nicht garantiert, dass Pfade, die länger als OC_STORAGE_SAFE_PATH_MAX (128 Zeichen einschließlich des 0-Terminators) sind, in OpenCore zugänglich sind.
3.2 Installation und Upgrade
Um OpenCore zu installieren, replizieren Sie die im vorherigen Abschnitt beschriebene Konfigurationsstruktur in das EFI-Volume einer GPT-Partition. Während entsprechende Abschnitte dieses Dokuments einige Informationen zu externen Ressourcen wie ACPI-Tabellen, UEFI-Treibern oder Kernel-Erweiterungen (kexts) bereitstellen, liegt die Vollständigkeit der Materie außerhalb des Rahmens dieses Dokuments. Informationen über Kernel-Erweiterungen finden Sie in einem separaten Kext-Listen-Dokument, das im OpenCore-Repository verfügbar ist. Informationen zum Vaulting sind im Abschnitt Sicherheitseigenschaften dieses Dokuments enthalten.

Die OC-Konfigurationsdatei kann, wie jede andere Eigenschaftslistendatei auch, mit einem beliebigen Texteditor wie nano oder vim bearbeitet werden. Spezialisierte Software kann jedoch eine bessere Erfahrung bieten. Unter macOS ist die bevorzugte GUI-Anwendung Xcode. Der ProperTree-Editor ist eine leichtgewichtige, plattformübergreifende und Open-Source-Alternative. Es wird dringend empfohlen, Tools zur Konfigurationserstellung zu vermeiden, die die interne Konfigurationsstruktur kennen, da dies zu ungültigen Konfigurationen führen kann (da die Struktur ständig aktualisiert wird). Wenn solche Werkzeuge trotz dieser Warnung verwendet werden sollen, stellen Sie sicher, dass nur stabile Versionen von OpenCore verwendet werden, die von solchen Werkzeugen ausdrücklich unterstützt werden. In solchen

Fällen wird die Verwendung von Open-Source-Implementierungen mit transparenter Binärgenerierung (wie OCAT) empfohlen, da andere Tools Malware enthalten können. Darüber hinaus sollten Konfigurationen, die für eine bestimmte Hardwarekonfiguration erstellt wurden, niemals auf anderen Hardwarekonfigurationen verwendet werden.
Für das BIOS-Booten ist ein UEFI-Umgebungsanbieter eines Drittanbieters erforderlich, und OpenDuetPkg ist ein solcher UEFI-Umgebungsanbieter für ältere Systeme. Um OpenCore auf einem solchen Altsystem auszuführen, kann OpenDuetPkg mit einem speziellen Tool installiert werden - BootInstall (im Lieferumfang von OpenCore enthalten). Auf anderen Systemen als macOS können auch Dienstprogramme von Drittanbietern verwendet werden, um dies durchzuführen.
Für Upgrade-Zwecke wird auf das Dokument Differences.pdf verwiesen, das Informationen über Änderungen an der Konfiguration (im Vergleich zur vorherigen Version) enthält, sowie auf das Dokument Changelog.md (das eine Liste der Änderungen aller veröffentlichten Updates enthält).

3.3 Beitrag

OpenCore kann als Standard-EDK-II-Paket kompiliert werden und benötigt das EDK-II-Stable-Paket. Die derzeit unterstützte EDK II Version wird in acidanthera/audk gehostet. Erforderliche Patches für dieses Paket finden Sie im Verzeichnis Patches.
Wenn Sie die LaTeX-Dokumentation (z.B. Configuration.tex) aktualisieren, erstellen Sie bitte die PDF-Dateien nicht neu, bevor das Zusammenführen nach Master erfolgt. Dies vermeidet unnötige Merge-Konflikte:
- Externe Mitwirkende, die den Pull-Request-Ansatz verwenden, sollten die Betreuer in der Pull-Request-Nachricht bitten, die PDF-Neuerstellung zu übernehmen.
- Interne Mitwirkende sollten die Dokumentation zum Zeitpunkt des Zusammenführens im selben oder in einem separaten Commit neu erstellen. Man kann einen anderen Betreuer in der Pull-Request-Nachricht bitten, die Dokumentation neu zu erstellen, wenn man nicht über die notwendigen Werkzeuge verfügt.
Die einzige offiziell unterstützte Toolchain ist XCODE5. Andere Toolchains können funktionieren, werden aber weder unterstützt noch empfohlen. Beiträge von sauberen Patches sind willkommen. Bitte folgen Sie dem EDK II C Codestyle.
Um mit XCODE5 zu kompilieren, sollten Benutzer neben Xcode auch NASM und MTOC installieren. Es wird empfohlen, trotz des Toolchain-Namens die neueste Xcode-Version zu verwenden. Eine Beispiel-Befehlssequenz lautet wie folgt:

```
git clone --depth=1 https://github.com/acidanthera/audk UDK
cd UDK
git submodule update --init --recommend-shallow
git clone --depth=1 https://github.com/acidanthera/OpenCorePkg . ./edksetup.sh

make -C BaseTools
build -a X64 -b RELEASE -t XCODE5 -p OpenCorePkg/OpenCorePkg.dsc
```

Listing 1: Compilation Commands

Für die Verwendung von IDEs sind Xcode-Projekte im Stammverzeichnis der Repositories verfügbar. Ein anderer Ansatz könnte die Verwendung von Language Server Protocols sein. Zum Beispiel, Sublime Text mit LSP für Sublime Text Plugin. Fügen Sie die Datei compile_flags.txt mit ähnlichem Inhalt in das UDK-Stammverzeichnis ein:

```
-I/UefiPackages/MdePkg
-I/UefiPackages/MdePkg/Include
-I/UefiPackages/MdePkg/Include/X64
-I/UefiPackages/MdeModulePkg
-I/UefiPackages/MdeModulePkg/Include
-I/UefiPackages/MdeModulePkg/Include/X64
-I/UefiPackages/OpenCorePkg/Include/AMI
-I/UefiPackages/OpenCorePkg/Include/Acidanthera
-I/UefiPackages/OpenCorePkg/Include/Apple
-I/UefiPackages/OpenCorePkg/Include/Apple/X64
-I/UefiPackages/OpenCorePkg/Include/Duet
-I/UefiPackages/OpenCorePkg/Include/Generic
-I/UefiPackages/OpenCorePkg/Include/Intel
-I/UefiPackages/OpenCorePkg/Include/Microsoft

-I/UefiPackages/OpenCorePkg/Include/Nvidia

-I/UefiPackages/OpenCorePkg/Include/VMware
-I/UefiPackages/OvmfPkg/Include
-I/UefiPackages/ShellPkg/Include
-I/UefiPackages/UefiCpuPkg/Include
-IInclude

-include
/UefiPackages/MdePkg/Include/Uefi.h
-fshort-wchar
-Wall
-Wextra
-Wno-unused-parameter
-Wno-missing-braces
-Wno-missing-field-initializers
-Wno-tautological-compare
-Wno-sign-compare
-Wno-varargs
-Wno-unused-const-variable
-DOC_TARGET_NOOPT=1
-DNO_MSABI_VA_FUNCS=1
```

Listing 2: ECC Configuration

`Hinweis: /UefiPackages in der Beispieldatei bezeichnet einen absoluten Pfad.`

Warnung: Werkzeugentwickler, die die Datei config.plist oder andere OpenCore-Dateien ändern, müssen sicherstellen, dass ihre Werkzeuge die NVRAM-Variable opencore-version überprüfen (siehe Abschnitt Debug-Eigenschaften weiter unten) und die Benutzer warnen, wenn die aufgeführte Version nicht unterstützt wird oder eine Vorabversion ist. Die OpenCore-Konfiguration kann sich über verschiedene Versionen hinweg ändern, und solche Werkzeuge müssen sicherstellen, dass sie dieses Dokument sorgfältig befolgen.

Andernfalls können solche Tools als Malware eingestuft und mit allen Mitteln blockiert werden.

3.4 Kodierungskonventionen

Wie bei jedem anderen Projekt gibt es auch bei uns Konventionen, die wir bei der Entwicklung einhalten. Allen Drittanbietern wird empfohlen, sich an die unten aufgeführten Konventionen zu halten, bevor sie Patches einreichen. Um Arbeitsabbrüche und die potentielle Ablehnung von Beiträgen zu minimieren, sollten Dritte zunächst Probleme im Acidanthera Bugtracker ansprechen, bevor sie Patches einreichen.

Organisation. Die Codebasis ist im OpenCorePkg Repository enthalten, welches das primäre EDK II Paket ist.

- Wenn Änderungen in mehreren Repositories erforderlich sind, sollten separate Pull Requests an jedes Repository gesendet werden.

- Das Commit der Änderungen sollte zuerst in die abhängigen Repositories und dann in die primären Repositories erfolgen, um

automatische Build-Fehler zu vermeiden.

- Jeder einzelne Commit sollte mit XCODE5 und vorzugsweise mit anderen Toolchains kompilieren. In der Mehrzahl der

Fällen kann dies durch Zugriff auf die CI-Schnittstelle überprüft werden. Es sollte sichergestellt werden, dass die statische Analyse keine Warnungen findet.

- Externe Pull Requests und getaggte Commits müssen validiert werden. Das heißt, dass Commits in Master zwar gebaut werden können, aber

nicht unbedingt funktionieren.

- Interne Zweige sollten wie folgt benannt werden: Autor-Name-Datum, z. B. vit9696-ballooning-20191026.

- Commit-Nachrichten sollten das primäre Modul (z. B. Bibliothek oder Code-Modul) vorangestellt werden, in dem die Änderungen vorgenommen wurden.

vorgenommen wurden. Zum Beispiel: OcGuardLib: OC_ALIGNED Makro hinzufügen. Für nicht-bibliothekarische Änderungen werden die Präfixe Docs oder Build verwendet.

Entwurf. Die Codebasis ist in einer Untermenge von freistehendem C11 (C17) geschrieben, die von den meisten modernen Toolchains des EDK II unterstützt wird. Es wird empfohlen, gängige Softwareentwicklungspraktiken anzuwenden oder eine Klärung anzufordern, wenn ein bestimmter Fall im Folgenden nicht behandelt wird.

- Verlassen Sie sich niemals auf undefiniertes Verhalten und versuchen Sie, implementierungsdefiniertes Verhalten zu vermeiden, es sei denn, es wird im Folgenden explizit behandelt (Sie können gerne ein Problem erstellen, wenn ein relevanter Fall nicht vorhanden ist).

- Verwenden Sie die OcGuardLib, um sichere Integralarithmetik zu gewährleisten und Überläufe zu vermeiden. Vorzeichenloses Wraparound sollte mit Vorsicht eingesetzt und auf das notwendige Maß reduziert werden.

S. 9

- Prüfen Sie Zeiger auf korrekte Ausrichtung mit OcGuardLib und verlassen Sie sich nicht darauf, dass die Architektur in der Lage ist, nicht ausgerichtete Zeiger zu derefenzieren.

- Verwenden Sie bei Bedarf flexible Array-Mitglieder anstelle von Arrays der Länge Null oder einer Länge.

- Verwenden Sie statische Assertions (STATIC_ASSERT) für Typ- und Wertannahmen und Laufzeit-Assertions (ASSERT) für die Überprüfung von Vorbedingungen und Invarianten. Verwenden Sie keine Laufzeit-Assertions, um auf Fehler zu prüfen, da sie niemals

Kontrollfluss verändern und möglicherweise ausgeschlossen werden.

- Gehen Sie davon aus, dass UINT32/INT32 int-size sind und verwenden Sie %u, %d und %x, um sie zu drucken.

- Nehmen Sie an, dass UINTN/INTN eine unbestimmte Größe haben, und wandeln Sie sie in UINT64/INT64 um, damit sie mit %Lu, %Ld usw.

wie üblich.

- Verlassen Sie sich nicht auf Integer-Promotionen für numerische Literale. Verwenden Sie explizite Casts, wenn der Typ von der Implementierung

abhängig ist oder Suffixe, wenn die Typgröße bekannt ist. Nehmen Sie U für UINT32 und ULL für UINT64 an.

- Achten Sie auf vorzeichenlose Arithmetik, insbesondere in der bitweisen Mathematik, insbesondere bei Verschiebungen.

- Der sizeof-Operator sollte Variablen anstelle von Typen nehmen, wo es möglich ist, um fehleranfällig zu sein. Verwenden Sie ARRAY_SIZE, um

Array-Größe in Elementen zu erhalten. Verwenden Sie die Makros L_STR_LEN und L_STR_SIZE aus der OcStringLib, um die Größe von String-Literalen

Größen zu erhalten, um eine Compiler-Optimierung zu gewährleisten.

- Verwenden Sie nicht das Schlüsselwort goto. Bevorzugen Sie early return, break oder continue, nachdem Sie die Fehlerprüfung nicht bestanden haben, anstatt

Verschachtelung von Konditionalen.

- Verwenden Sie EFIAPI, erzwingen Sie die UEFI-Aufrufkonvention, nur in Protokollen, externen Rückrufen zwischen Modulen und Funktionen

mit variablen Argumenten.

- Stellen Sie für jede hinzugefügte Funktion eine Inline-Dokumentation bereit, die zumindest ihre Eingaben, Ausgaben, Vorbedingung und Nachbedingung beschreibt,

Nachbedingung, und eine kurze Beschreibung.

- Verwenden Sie RETURN_STATUS nicht. Gehen Sie davon aus, dass EFI_STATUS eine passende Obermenge ist, die immer dann verwendet werden soll, wenn

BOOLEAN nicht ausreichend ist.

- Sicherheitsverletzungen sollten das System anhalten oder einen erzwungenen Neustart auslösen.

Codestil. Die Codebasis folgt dem EDK II Codestyle mit ein paar Änderungen und Klarstellungen.

- Schreiben Sie Inline-Dokumentation für Funktionen und Variablen nur einmal: im Header, wenn ein Header-Prototyp verfügbar ist, und inline für statische Variablen und Funktionen.

- Verwenden Sie Zeilenlängen von 120 Zeichen oder weniger, vorzugsweise 100 Zeichen.

- Verwenden Sie Leerzeichen nach Casts, z.B. (VOID *)(UINTN) Variable.

- Verwenden Sie zwei Leerzeichen zum Einrücken von Funktionsargumenten, wenn Sie Zeilen aufteilen.

- Öffentliche Funktionen sind entweder mit Oc oder einem anderen eindeutigen Namen zu kennzeichnen.

- Stellen Sie privaten statischen Funktionen kein Präfix voran, sondern geben Sie privaten nicht-statischen Funktionen das Präfix Internal.

- Verwenden Sie SPDX-Lizenz-Header wie in acidanthera/bugtracker#483 gezeigt.

3.5 Fehlersuche

Die Codebasis enthält EDK II Debugging und einige benutzerdefinierte Funktionen, um die Erfahrung zu verbessern.

- Verwenden Sie Modul-Präfixe, 2-5 Buchstaben gefolgt von einem Doppelpunkt (:), für Debug-Meldungen. Für OpenCorePkg verwenden Sie OC:, für Bibliotheken und Treiber verwenden Sie ihre eigenen eindeutigen Präfixe.

- Verwenden Sie keine Punkte (.) am Ende von Debug-Meldungen und trennen Sie EFI_STATUS, gedruckt durch %r, mit einem Bindestrich (z. B. OCRAM: Allocation of %u bytes failed - %r\n).

- Verwenden Sie die Konstruktionen DEBUG_CODE_BEGIN () und DEBUG_CODE_END (), um Debug-Prüfungen zu verhindern, die möglicherweise die Leistung von Release-Builds beeinträchtigen können und ansonsten unnötig sind.

- Verwenden Sie das DEBUG-Makro, um Debug-Meldungen während des normalen Betriebs zu drucken, und RUNTIME_DEBUG für das Debugging nach EXIT_BOOT_SERVICES.

- Verwenden Sie den Debug-Level DEBUG_VERBOSE, um Debug-Meldungen für das zukünftige Debugging des Codes zu hinterlassen, die derzeit nicht notwendig sind. Standardmäßig werden DEBUG_VERBOSE-Meldungen auch in DEBUG-Builds ignoriert.

- Verwenden Sie die Debug-Ebene DEBUG_INFO für alle unkritischen Meldungen (einschließlich Fehler) und DEBUG_BULK_INFO für umfangreiche Meldungen, die nicht im NVRAM-Protokoll erscheinen sollen, dessen Größe stark begrenzt ist. Diese Meldungen werden in RELEASE-Builds ignoriert.

- Verwenden Sie DEBUG_ERROR, um kritische, für den Menschen sichtbare Meldungen zu drucken, die möglicherweise den Boot-Prozess anhalten können, und DEBUG_WARN für alle anderen für den Menschen sichtbaren Fehler, einschließlich RELEASE-Builds.

Die git-bisect-Funktionalität kann nützlich sein, wenn Sie versuchen, problematische Änderungen zu finden. Inoffizielle Quellen für OpenCore-Binär-Builds pro Commit, wie z.B. Dortania, können ebenfalls nützlich sein.

S. 10

4 ACPI

4.1 Einführung

ACPI (Advanced Configuration and Power Interface) ist ein offener Standard zur Erkennung und Konfiguration von Computerhardware. Die ACPI-Spezifikation definiert Standardtabellen (z.B. DSDT, SSDT, FACS, DMAR) und verschiedene Methoden (z.B. _DSM, _PRW) zur Implementierung. Moderne Hardware benötigt nur wenige Änderungen, um die ACPI-Kompatibilität aufrechtzuerhalten, und einige Optionen für solche Änderungen werden als Teil von OpenCore bereitgestellt.

Um ACPI-Tabellen zu kompilieren und zu disassemblieren, kann der von ACPICA entwickelte iASL-Compiler verwendet werden. Ein GUI-Frontend zum iASL-Compiler kann von Acidanthera/MaciASL heruntergeladen werden.

ACPI-Änderungen gelten global (für jedes Betriebssystem) mit der folgenden effektiven Reihenfolge:

- Patch wird verarbeitet.

- Löschen wird verarbeitet. - Hinzufügen wird verarbeitet.

- Quirks werden verarbeitet.

Durch die globale Anwendung der Änderungen werden die Probleme der fehlerhaften Erkennung des Betriebssystems (in Übereinstimmung mit der ACPI-Spezifikation nicht möglich, bevor das Betriebssystem bootet), des Ladens der Betriebssystemkette und der schwierigen ACPI-Fehlersuche gelöst. Daher ist beim Schreiben von Änderungen an _OSI möglicherweise mehr Aufmerksamkeit erforderlich.

Die frühzeitige Anwendung der Patches ermöglicht es, so genannte "Proxy"-Patches zu schreiben, bei denen die ursprüngliche Methode in der ursprünglichen Tabelle gepatcht und in der gepatchten Tabelle implementiert wird.

Es gibt mehrere Quellen für ACPI-Tabellen und Workarounds. Häufig verwendete ACPI-Tabellen werden mit den Versionen OpenCore, VirtualSMC, VoodooPS2 und WhateverGreen bereitgestellt. Darüber hinaus können verschiedene Anleitungen von Drittanbietern in den Unterforen AppleLife Laboratory und DSDT gefunden werden (z.B. Battery register splitting guide). Eine etwas benutzerfreundlichere Erklärung einiger Tabellen, die in OpenCore enthalten sind, finden Sie auch in Dortania's Getting started with ACPI guide. Für exotischere Fälle gibt es mehrere Alternativen wie dalianskys ACPI-Beispielsammlung (englische Übersetzung von 5T33Z0 et al). Bitte beachten Sie jedoch, dass die von Dritten vorgeschlagenen Lösungen veraltet sein oder Fehler enthalten können.

4.2 Eigenschaften

1. Add

Typ: plist-Array

Ausfallsicher: Leer

Beschreibung: Lädt ausgewählte Tabellen aus dem OC/ACPI-Verzeichnis.

Soll mit plist dict-Werten gefüllt werden, die jeden Add-Eintrag beschreiben. Einzelheiten finden Sie im Abschnitt Add Properties weiter unten.

2. Delete

Typ: plist-Array

Ausfallsicher: Leer

Beschreibung: Entfernt ausgewählte Tabellen aus dem ACPI-Stack.

Soll mit plist dict-Werten gefüllt werden, die jeden Löscheintrag beschreiben. Einzelheiten finden Sie im Abschnitt Delete Properties weiter unten.

3. Patch

Typ: plist-Array

Ausfallsicher: Leer

Beschreibung: Führt binäre Patches in ACPI-Tabellen durch, bevor Tabellen hinzugefügt oder entfernt werden.

Muss mit plist-Wörterbuchwerten gefüllt werden, die jeden Patch-Eintrag beschreiben. Einzelheiten finden Sie im Abschnitt Patch-Eigenschaften weiter unten.

4. Quirks

Typ: plist dict

Beschreibung: Wendet einzelne ACPI-Quirks an, die im Abschnitt Eigenschaften von Quirks weiter unten beschrieben werden.

S 11

4.3 Eigenschaften hinzufügen

1. Kommentar

Typ: plist string

Ausfallsicher: Leer

Beschreibung: Beliebige ASCII-Zeichenkette, die verwendet wird, um eine für den Menschen lesbare Referenz für den Eintrag zu liefern. Ob dieser Wert verwendet wird, hängt von der Implementierung ab.

2. Aktiviert

Typ: plist boolean

Ausfallsicher: false

Beschreibung: Auf true gesetzt, um diese ACPI-Tabelle hinzuzufügen.

3. Pfad

Typ: plist-Zeichenkette

Ausfallsicher: Leer

Beschreibung: Dateipfade, die als ACPI-Tabellen geladen werden sollen. Beispielwerte sind DSDT.aml, SubDir/SSDT-8.aml, SSDT-USBX.aml, usw.

Die Reihenfolge, in der die ACPI-Tabellen geladen werden, entspricht der Reihenfolge der Elemente im Array. ACPI-Tabellen werden aus dem Verzeichnis OC/ACPI geladen.
Hinweis: Alle Tabellen außer den Tabellen mit einer DSDT-Tabellenkennung (die durch das Parsen der Daten und nicht durch den Dateinamen bestimmt wird)

fügen neue Tabellen in den ACPI-Stapel ein. DSDT-Tabellen führen stattdessen eine Ersetzung von DSDT-Tabellen durch.

4.4 Eigenschaften löschen

1. Alle

Typ: plist boolean

Failsafe: false (Löscht nur die erste übereinstimmende Tabelle)

Beschreibung: Auf true setzen, um alle ACPI-Tabellen zu löschen, die der Bedingung entsprechen.

2. Kommentar

Typ: plist Zeichenfolge

Ausfallsicher: Leer

Beschreibung: Beliebige ASCII-Zeichenkette, die verwendet wird, um eine für den Menschen lesbare Referenz für den Eintrag zu liefern. Ob dieser Wert verwendet wird, hängt von der Implementierung ab.

3. Aktiviert

Typ: plist boolean

Ausfallsicher: false

Beschreibung: Auf true gesetzt, um diese ACPI-Tabelle zu entfernen.

4. OemTableId

Typ: Plist-Daten, 8 Bytes

Ausfallsicher: All zero (Match any table OEM ID) Beschreibung: Übereinstimmende Tabellen-OEM-ID gleich diesem Wert.

5. TabelleLänge

Typ: plist Ganzzahl

Failsafe: 0 (Übereinstimmung mit jeder Tabellengröße)

Beschreibung: Entspricht die Tabellengröße diesem Wert.

6. TabelleSignatur

Typ: plist-Daten, 4 Bytes

Failsafe: All zero (Match any table signature) Beschreibung: Entspricht einer Tabellensignatur, die diesem Wert entspricht.

Hinweis: Verwenden Sie keine Tabellensignaturen, wenn die Sequenz an mehreren Stellen ersetzt werden muss. Dies ist besonders wichtig, wenn verschiedene Arten von Umbenennungen durchgeführt werden.

4.5 Patch-Eigenschaften

1. Basis

Typ: plist-Zeichenkette

Ausfallsicher: Leer (ignoriert)

Beschreibung: Wählt die ACPI-Pfadbasis für die Patch-Suche (oder den sofortigen Ersatz) aus, indem der Offset zum angegebenen Pfad ermittelt wird.

Es werden nur vollqualifizierte absolute Pfade unterstützt (z.B. _SB.PCI0.LPCB.HPET). Derzeit unterstützte Objekttypen sind: Gerät, Feld, Methode.

Hinweis: Seien Sie vorsichtig, nicht alle OEM-Tabellen können geparst werden. Verwenden Sie das ACPIe-Dienstprogramm zum Debuggen. ACPIe, kompiliert mit dem make-Befehl DEBUG=1, erzeugt ein hilfreiches ACPI-Lookup-Tracing.

2. BaseSkip

Typ: plist Ganzzahl

Failsafe: 0 (keine Vorkommen überspringen)

Beschreibung: Anzahl der gefundenen Base-Vorkommen, die übersprungen werden sollen, bevor Suchen und Ersetzen angewendet werden.

3. Kommentar

Typ: plist-String

Ausfallsicher: Leer

Beschreibung: Beliebige ASCII-Zeichenkette, die verwendet wird, um eine für den Menschen lesbare Referenz für den Eintrag zu liefern. Ob dieser Wert verwendet wird, hängt von der Implementierung ab.

4. Anzahl

Typ: plist Ganzzahl

Failsafe: 0 (Patch auf alle gefundenen Vorkommen anwenden) Beschreibung: Anzahl der zu patchenden Vorkommen.

5. Aktiviert

Typ: plist boolescher Wert

Failsafe: false

Beschreibung: Wird auf true gesetzt, um diesen ACPI-Patch anzuwenden.

6. Finden

Typ: plist-Daten

Ausfallsicher: Leer

Beschreibung: Zu suchende Daten. Muss gleich der Größe von Replace sein, wenn gesetzt.

Hinweis: Kann leer sein, wenn Base angegeben ist; in diesem Fall erfolgt die Ersetzung sofort nach der Base-Suche.

7. Begrenzung

Typ: plist ganzzahlig

Failsafe: 0 (Suche in der gesamten ACPI-Tabelle)

Beschreibung: Maximale Anzahl von Bytes, nach denen gesucht werden soll.

8. Maske

Typ: plist-Daten

Failsafe: Leer (Ignoriert)

Beschreibung: Bitweise Datenmaske, die während des Suchvergleichs verwendet wird. Ermöglicht unscharfe Suche durch Ignorieren nicht maskierter (auf Null gesetzter) Bits. Muss gleich der Größe von Replace sein, wenn gesetzt.

9. OemTableId

Typ: Plist-Daten, 8 Bytes

Failsafe: Alle Nullen (Übereinstimmung mit jeder Tabellen-OEM-ID) Beschreibung: Entspricht die OEM-ID der Tabelle diesem Wert.

10. Ersetzen

Typ: plist-Daten

Failsafe: Leer

Beschreibung: Ersetzungsdaten von einem oder mehreren Bytes.

11. ErsetzenMaske

Typ: plist-Daten

Failsafe: Empty (Ignoriert)

Beschreibung: Bitweise Datenmaske, die bei der Ersetzung verwendet wird. Ermöglicht unscharfe Ersetzung durch Aktualisierung der maskierten (auf Nicht-Null gesetzten) Bits. Muss gleich der Größe von Replace sein, wenn gesetzt.

12. Überspringen

Typ: plist Ganzzahl

Failsafe: 0 (kein Vorkommen überspringen)

Beschreibung: Anzahl der gefundenen Vorkommen, die übersprungen werden sollen, bevor eine Ersetzung erfolgt.

13. TabelleLänge

Typ: plist Ganzzahl

Failsafe: 0 (passt zu jeder Tabellengröße)

Beschreibung: Entspricht die Tabellengröße diesem Wert.

14. TabelleSignatur

Typ: plist-Daten, 4 Bytes

Failsafe: All zero (Match any table signature) Beschreibung: Entspricht einer Tabellensignatur, die diesem Wert entspricht.

In den meisten Fällen sind ACPI-Patches nicht nützlich und schädlich:

- Vermeiden Sie die Umbenennung von Geräten mit ACPI-Patches. Dies kann fehlschlagen oder eine falsche Umbenennung von nicht verwandten Geräten (z. B. EC und EC0) durchführen, unnötig sein oder sogar Geräte in bestimmten Tabellen nicht umbenennen. Aus Gründen der ACPI-Konsistenz ist es

viel sicherer, Geräte auf der Ebene der I/O-Registry umzubenennen, wie es WhateverGreen macht.

- Vermeiden Sie, wann immer möglich, _OSI zu patchen, um einen höheren Funktionsumfang zu unterstützen. Dies ermöglicht zwar eine Reihe von Workarounds auf APTIO-Firmware, führt aber in der Regel dazu, dass zusätzliche Patches benötigt werden. Diese sind bei moderner Firmware in der Regel nicht erforderlich, und kleinere Patches funktionieren gut bei Firmware, die dies tut. Laptop-Hersteller verlassen sich jedoch oft auf diese Methode, um die Verfügbarkeit von Funktionen wie moderne I2C-

Eingangsunterstützung, thermische Anpassung und benutzerdefinierte Funktionserweiterungen zu bestimmen.

- Vermeiden Sie das Patchen des Embedded-Controller-Ereignisses _Qxx, nur um die Helligkeitstasten zu aktivieren. Der herkömmliche Prozess zum Auffinden dieser Tasten erfordert in der Regel erhebliche Änderungen an DSDT- und SSDT-Dateien, und außerdem ist der Debug-Kext auf neueren Systemen nicht stabil. Verwenden Sie stattdessen die eingebaute Helligkeitstastenerkennung in BrightnessKeys.

- Vermeiden Sie nach Möglichkeit Ad-hoc-Änderungen wie die Umbenennung von _PRW oder _DSM. Einige Fälle, in denen das Patchen tatsächlich nützlich ist, sind:

- Aktualisieren des Headers der HPET-Methode (oder eines anderen Geräts), um Kompatibilitätsprüfungen durch _OSI auf veralteter Hardware zu vermeiden. _STA-Methode mit if ((OSFL () == Null)) { If (HPTE) ... Rückgabe (Null) Inhalt kann gezwungen werden, immer0xFzurückzugeben, indemA0 10 93 4F 53 46 4C 00durchA4 0A 0F A3 A3 A3 A3 A3 ersetzt wird.

- Um eine benutzerdefinierte Methodenimplementierung innerhalb eines SSDT bereitzustellen, um z.B. Shutdown-Fixes auf bestimmten Computern zu injizieren, kann die ursprüngliche Methode durch einen Dummy-Namen ersetzt werden, indem _PTS mit ZPTS gepatcht wird und ein Callback zur ursprünglichen Methode hinzugefügt wird.

Die Tianocore AcpiAml.h Quelldatei kann helfen, die ACPI Opcodes besser zu verstehen.

Hinweis: Patches mit unterschiedlichen Find- und Replace-Längen werden nicht unterstützt, da sie ACPI-Tabellen beschädigen und das System aufgrund von Bereichsverschiebungen instabil machen können. Wenn solche Änderungen erforderlich sind, könnte die Verwendung von "Proxy"-Patches oder das Auffüllen von NOPs auf den verbleibenden Bereich in Betracht gezogen werden.

4.6

1.

Quirks Eigenschaften

FadtEnableReset

Typ: plist boolean

Ausfallsicher: false

Beschreibung: Bietet ein Reset-Register und ein Flag in der FADT-Tabelle, um einen Neustart und ein Herunterfahren zu ermöglichen.

Hauptsächlich auf älterer Hardware und einigen neueren Laptops erforderlich. Kann auch Tastenkombinationen für den Netzschalter reparieren. Nicht empfohlen, wenn nicht erforderlich.

2. NormalizeHeaders

Typ: plist boolean

Ausfallsicher: false

Beschreibung: Bereinigt ACPI-Header-Felder, um Fehler in der macOS ACPI-Implementierung zu umgehen, die zu Boot-Abstürzen führen. Referenz: Debugging AppleACPIPlatform on 10.13 von Alex James (auch bekannt als theracermaster). Das Problem wurde in macOS Mojave (10.14) behoben.

3. RebaseRegions

Typ: plist boolean

Ausfallsicher: false

Beschreibung: Versucht, ACPI-Speicherbereiche heuristisch zu verschieben. Nicht empfohlen.

ACPI-Tabellen werden oft dynamisch von der zugrunde liegenden Firmware-Implementierung erzeugt. Neben dem positionsunabhängigen Code können ACPI-Tabellen die physikalischen Adressen von MMIO-Bereichen enthalten, die für die Gerätekonfiguration verwendet werden, typischerweise gruppiert nach Region (z. B. OperationRegion). Änderungen der Firmware-Einstellungen oder der Hardware-Konfiguration, Upgrades oder Patches der Firmware führen unweigerlich zu Änderungen im dynamisch erzeugten ACPI-Code, was manchmal zu einer Verschiebung der Adressen in den oben genannten OperationRegion-Konstruktionen führt.

Aus diesem Grund ist die Anwendung von Änderungen an ACPI-Tabellen äußerst riskant. Am besten ist es, so wenig wie möglich an den ACPI-Tabellen zu ändern und das Ersetzen von Tabellen, insbesondere DSDT-Tabellen, zu vermeiden. Wenn sich dies nicht vermeiden lässt, stellen Sie sicher, dass alle benutzerdefinierten DSDT-Tabellen auf den neuesten DSDT-Tabellen basieren, oder versuchen Sie, Lese- und Schreibzugriffe für die betroffenen Bereiche zu entfernen.

Wenn nichts anderes hilft, kann diese Option versucht werden, um einen Stillstand in der PCI Configuration Begin-Phase beim Booten von macOS zu vermeiden, indem versucht wird, die ACPI-Adressen zu korrigieren. Sie ist jedoch kein Allheilmittel und funktioniert nur in den typischsten Fällen. Verwenden Sie sie nicht, wenn es nicht unbedingt notwendig ist, da sie auf bestimmten Plattformen das Gegenteil bewirken und zu Bootfehlern führen kann.

4. ResetHwSig

Typ: plist boolescher Wert

Ausfallsicher: false

Beschreibung: Setzt den Wert der FACS-Tabelle HardwareSignature auf 0 zurück.

Damit kann Firmware umgangen werden, die die Hardwaresignatur bei Neustarts nicht beibehält und Probleme beim Aufwachen aus dem Ruhezustand verursacht.

5. ResetLogoStatus

Typ: plist boolescher Wert

Ausfallsicher: false

Beschreibung: Setzt das Statusfeld der BGRT-Tabelle Displayed auf false zurück.

Damit kann Firmware umgangen werden, die eine BGRT-Tabelle bereitstellt, aber danach keine Bildschirmaktualisierungen mehr vornimmt.

6. SyncTableIds

Typ: plist boolescher Wert

Ausfallsicher: false

Beschreibung: Synchronisiert Tabellenbezeichner mit der SLIC-Tabelle.

Damit wird verhindert, dass gepatchte Tabellen mit der SLIC-Tabelle inkompatibel werden, was in älteren Windows-Betriebssystemen zu Lizenzierungsproblemen führt. Übersetzt mit www.DeepL.com/Translator (kostenlose Version)

5 Booter

5.1 Einführung

Dieser Abschnitt ermöglicht die Anwendung verschiedener Arten von UEFI-Modifikationen auf Betriebssystem-Bootloader, in erster Linie den Apple-Bootloader (boot.efi). Die Modifikationen bieten derzeit verschiedene Patches und Umgebungsänderungen für verschiedene Firmware-Typen. Einige dieser Funktionen wurden ursprünglich als Teil von AptioMemoryFix.efi implementiert, das nicht mehr gepflegt wird. Eine Anleitung zur Migration finden Sie im Abschnitt Tipps und Tricks.

Wenn dies zum ersten Mal auf angepasster Firmware verwendet wird, sollten die folgenden Voraussetzungen erfüllt sein, bevor Sie beginnen:

- Die aktuellste UEFI-Firmware (überprüfen Sie die Website des Motherboard-Herstellers).

- Fast Boot und Hardware Fast Boot in den Firmware-Einstellungen deaktiviert, falls vorhanden.

- Above 4G Decoding oder ähnliches in den Firmware-Einstellungen aktiviert, falls vorhanden. Beachten Sie, dass auf einigen Motherboards, insbesondere

dem ASUS WS-X299-PRO, diese Option zu negativen Auswirkungen führt und deaktiviert werden muss. Obwohl keine anderen Motherboards

mit demselben Problem bekannt sind, sollte diese Option immer dann zuerst überprüft werden, wenn erratische Boot-Fehler auftreten.

- DisableIoMapper-Quirk aktiviert, oder VT-d in den Firmware-Einstellungen deaktiviert, falls vorhanden, oder ACPI DMAR-Tabelle gelöscht.

- Kein "slide"-Boot-Argument im NVRAM oder an anderer Stelle vorhanden. Es ist nicht notwendig, es sei denn, das System lässt sich

überhaupt nicht gebootet werden kann oder Keine Slide-Werte verwendbar sind! Die Meldung 'Use custom slide!' ist im Protokoll zu sehen.

- CFG Lock (MSR 0xE2 Schreibschutz) in den Firmware-Einstellungen deaktiviert, falls vorhanden. Siehe die Hinweise zu ControlMsrE2

für Einzelheiten.

- CSM (Compatibility Support Module) in den Firmware-Einstellungen deaktiviert, falls vorhanden. Auf NVIDIA 6xx/AMD 2xx oder älter,

GOP ROM muss möglicherweise zuerst geflasht werden. Verwenden Sie GopUpdate (siehe den zweiten Beitrag) oder AMD UEFI GOP MAKER, falls

Fall einer möglichen Verwirrung.

- EHCI/XHCI Hand-off in den Firmware-Einstellungen nur aktiviert, wenn der Bootvorgang abbricht, wenn die USB-Geräte nicht getrennt werden.

- VT-x, Hyper Threading, Execute Disable Bit in den Firmware-Einstellungen aktiviert, falls vorhanden.

- Auch wenn es nicht unbedingt erforderlich ist, müssen manchmal Thunderbolt-Unterstützung, Intel SGX und Intel Platform Trust

in den Firmware-Einstellungen deaktiviert werden müssen.

Beim Debuggen von Problemen im Ruhezustand können Power Nap und automatisches Ausschalten (die auf älteren Plattformen manchmal Probleme beim Aufwachen mit schwarzem Bildschirm oder Bootschleifen verursachen) vorübergehend deaktiviert werden. Die spezifischen Probleme können variieren, aber in der Regel sollten zuerst die ACPI-Tabellen untersucht werden.

Hier ist ein Beispiel für einen Defekt, der auf einigen Z68-Motherboards gefunden wurde. Um Power Nap und die anderen zu deaktivieren, führen Sie die folgenden Befehle im Terminal aus:

_____

sudo pmset autopoweroff 0

sudo pmset powernap 0

sudo pmset standby 0

_____

Hinweis: Diese Einstellungen können durch Hardwareänderungen und unter bestimmten anderen Umständen zurückgesetzt werden. Um den aktuellen Status anzuzeigen, verwenden Sie den Befehl pmset -g im Terminal.

5.2 Eigenschaften

1. MmioWhitelist

Typ: plist-Array

Ausfallsicher: Leer

Beschreibung: Wird mit plist dict-Werten gefüllt, die Adressen beschreiben, die für die Funktion einer bestimmten Firmware kritisch sind, wenn DevirtualiseMmio quirk verwendet wird. Siehe den Abschnitt MmioWhitelist-Eigenschaften weiter unten für Details.

2. Patch

Typ: plist-Array

Ausfallsicher: Leer

Beschreibung: Führt binäre Patches im Booter durch.

Muss mit Plist-Wörterbuchwerten gefüllt werden, die jeden Patch beschreiben. Einzelheiten finden Sie im Abschnitt Patch Properties unten für Details.

3. Quirks

Typ: plist dict

Beschreibung: Wendet einzelne Booter-Quirks an, die im Abschnitt Eigenschaften von Quirks weiter unten beschrieben werden.

5.3 MmioWhitelist-Eigenschaften

1. Adresse

Typ: plist Ganzzahl

Failsafe: 0

Beschreibung: Außergewöhnliche MMIO-Adresse, deren Speicherdeskriptor von DevirtualiseMmio virtualisiert (unverändert) gelassen werden soll. Das bedeutet, dass die Firmware in der Lage sein wird, während des Betriebs des Betriebssystems direkt mit dieser Speicherregion zu kommunizieren, da der Region, in der sich dieser Wert befindet, eine virtuelle Adresse zugewiesen wird.

Die hier geschriebenen Adressen müssen Teil der Memory Map sein, den Typ EfiMemoryMappedIO haben und das Attribut EFI_MEMORY_RUNTIME (höchstes Bit) muss gesetzt sein. Das Debug-Log kann verwendet werden, um die Liste der Kandidaten zu finden.

2. Kommentar

Typ: plist string

Ausfallsicher: Leer

Beschreibung: Beliebige ASCII-Zeichenkette, die verwendet wird, um eine für den Menschen lesbare Referenz für den Eintrag zu liefern. Ob dieser Wert verwendet wird, hängt von der Implementierung ab.

3. Aktiviert

Typ: plist boolean

Ausfallsicher: false

Beschreibung: MMIO-Adresse von der Devirtualisierungsprozedur ausschließen.

5.4 Patch-Eigenschaften

1. Arch

Typ: plist-Zeichenfolge

Failsafe: Any (Anwenden auf jede unterstützte Architektur) Beschreibung: Booter-Patch-Architektur (i386, x86_64).

2. Kommentar

Typ: plist-Zeichenkette

Ausfallsicher: Leer

Beschreibung: Beliebige ASCII-Zeichenkette, die verwendet wird, um eine für den Menschen lesbare Referenz für den Eintrag zu liefern. Ob dieser Wert verwendet wird, hängt von der Implementierung ab.

3. Anzahl

Typ: plist Ganzzahl

Failsafe: 0 (auf alle gefundenen Vorkommen anwenden) Beschreibung: Anzahl der zu übertragenden Patch-Vorkommen.

4. Aktiviert

Typ: plist boolean

Failsafe: false

Beschreibung: Auf true gesetzt, um diesen Booter-Patch zu aktivieren.

5. Finden

Typ: plist-Daten

Ausfallsicher: Leer

Beschreibung: Zu suchende Daten. Muss gleich der Größe von Replace sein, wenn gesetzt.

6. Bezeichner

Typ: plist-String

Failsafe: Any (passt zu jedem Booter)

Beschreibung: Apple für macOS Booter (typischerweise boot.efi); oder ein Name mit einem Suffix, wie bootmgfw.efi, für einen bestimmten Booter.

7. Begrenzung

Typ: plist Ganzzahl

Failsafe: 0 (Durchsuchen des gesamten Booters)

Beschreibung: Maximale Anzahl von Bytes, nach denen gesucht werden soll.

8. Maske

Typ: plist-Daten

Ausfallsicher: Leer (Ignoriert)

Beschreibung: Bitweise Datenmaske, die während des Suchvergleichs verwendet wird. Ermöglicht unscharfe Suche durch Ignorieren nicht maskierter (auf Null gesetzter) Bits. Muss gleich der Größe von Find sein, wenn gesetzt.

9. Ersetzen

Typ: plist Daten

Failsafe: Leer

Beschreibung: Ersetzungsdaten von einem oder mehreren Bytes.

10. ErsetzenMaske

Typ: plist-Daten

Failsafe: Leer (Ignoriert)

Beschreibung: Bitweise Datenmaske, die bei der Ersetzung verwendet wird. Ermöglicht unscharfe Ersetzung durch Aktualisierung der maskierten (auf Nicht-Null gesetzten) Bits. Muss gleich der Größe von Replace sein, wenn gesetzt.

11. Überspringen

Typ: plist ganzzahlig

Failsafe: 0 (kein Vorkommen überspringen)

Beschreibung: Anzahl der gefundenen Vorkommen, die übersprungen werden sollen, bevor Ersetzungen angewendet werden.

5.5 Quirks Eigenschaften

1. AllowRelocationBlock

Typ: plist boolescher Wert

Failsafe: false

Beschreibung: Erlaubt das Booten von macOS über einen Relocation-Block.

Der Relocation-Block ist ein Scratch-Puffer, der in den unteren 4 GB für das Laden des Kernels und zugehöriger Strukturen von EfiBoot auf Firmware verwendet wird, wenn der untere Speicherbereich ansonsten durch (angenommene) Nicht-Laufzeitdaten belegt ist.

Unmittelbar vor dem Start des Kernels wird der Relocation-Block auf niedrigere Adressen zurückkopiert. In ähnlicher Weise werden auch alle anderen Adressen, die auf den Relocation-Block zeigen, sorgfältig angepasst. Der Verschiebungsblock kann verwendet werden, wenn:

- es keinen besseren Slide gibt (der gesamte Speicher ist belegt)

- slide=0 erzwungen wird (durch ein Argument oder den sicheren Modus)

- KASLR (slide) wird nicht unterstützt (dies ist macOS 10.7 oder älter)

Diese Eigenart erfordert die Aktivierung von ProvideCustomSlide und normalerweise auch die Aktivierung von AvoidRuntimeDefrag, um korrekt zu funktionieren. Der Ruhezustand wird nicht unterstützt, wenn mit einem Verschiebungsblock gebootet wird, der nur bei Bedarf verwendet wird, wenn die Eigenart aktiviert ist.

Hinweis: Während diese Eigenart erforderlich ist, um ältere macOS-Versionen auf Plattformen mit geringerem Arbeitsspeicher auszuführen, ist sie mit einiger Hardware und macOS 11 nicht kompatibel. In solchen Fällen sollten Sie stattdessen EnableSafeModeSlide verwenden.

2. AvoidRuntimeDefrag

Typ: plist boolescher Wert

Ausfallsicher: false

Beschreibung: Schützt vor boot.efi-Laufzeitspeicherdefragmentierung.

Diese Option behebt die Unterstützung von UEFI-Laufzeitdiensten (Datum, Uhrzeit, NVRAM, Energiesteuerung usw.) auf Firmware, die SMM-Unterstützung für bestimmte Dienste wie Variablenspeicher verwendet. SMM kann versuchen, über physische Adressen in Nicht-SMM-Bereichen auf den Speicher zuzugreifen, aber dieser kann manchmal von boot.efi verschoben worden sein. Diese Option verhindert, dass boot.efi solche Daten verschiebt.

Hinweis: Die meisten Arten von Firmware, außer Apple und VMware, benötigen diese Eigenart.

3. DevirtualiseMmio

Typ: plist boolescher Wert

Ausfallsicherheit: false

Beschreibung: Entfernt das Laufzeitattribut aus bestimmten MMIO-Regionen.

Diese Eigenart reduziert den gestohlenen Speicherplatz in der Speichermap, indem das Laufzeitbit für bekannte Speicherregionen entfernt wird. Diese Eigenart kann zu einer Erhöhung der verfügbaren KASLR-Folien führen, jedoch ohne zusätzliche Maßnahmen,

es ist nicht unbedingt mit der Zielplatine kompatibel. Diese Eigenart gibt in der Regel zwischen 64 und 256 Megabyte Speicher frei, der im Debug-Protokoll vorhanden ist, und ist auf einigen Plattformen die einzige Möglichkeit, macOS zu booten, was ansonsten mit Zuweisungsfehlern in der Bootloader-Phase fehlschlägt.

Diese Option ist bei allen Firmware-Typen nützlich, außer bei einigen sehr alten wie Sandy Bridge. Bei bestimmter Firmware kann eine Liste von Adressen erforderlich sein, die virtuelle Adressen für die korrekte NVRAM- und Hibernation-Funktionalität benötigen. Verwenden Sie dazu den Abschnitt MmioWhitelist.

4. DisableSingleUser

Typ: plist boolscher Wert

Failsafe: false

Beschreibung: Deaktiviert den Einzelbenutzermodus.

Dies ist eine Sicherheitsoption, die die Aktivierung des Einzelbenutzermodus einschränkt, indem der Hotkey CMD+S und das Boot-Argument -s ignoriert werden. Das Verhalten bei aktivierter Option soll dem Verhalten des T2-basierten Modells entsprechen. Lesen Sie diesen archivierten Artikel, um zu verstehen, wie Sie den Single-User-Modus mit dieser Eigenheit verwenden können.

5. DisableVariableWrite

Typ: plist boolescher Wert

Failsafe: false

Beschreibung: Schützt vor macOS NVRAM-Schreibzugriff.

Dies ist eine Sicherheitsoption, die den NVRAM-Zugriff unter macOS einschränkt. Diese Eigenart erfordert das OC_FIRMWARE_RUNTIME-Protokoll, das in OpenRuntime.efi implementiert ist.

Hinweis: Diese Eigenart kann auch als Ad-hoc-Workaround für defekte UEFI-Laufzeitdienstimplementierungen verwendet werden, die nicht in der Lage sind, Variablen in den NVRAM zu schreiben, was zu Betriebssystemausfällen führt.

6. DiscardHibernateMap

Typ: plist boolescher Wert

Ausfallsicher: false

Beschreibung: Ursprüngliche Hibernate-Map wiederverwenden.

Diese Option zwingt den XNU-Kernel, eine neu bereitgestellte Memory Map zu ignorieren und davon auszugehen, dass sie nach dem Aufwachen aus dem Ruhezustand nicht verändert wurde. Dieses Verhalten wird von Windows vorausgesetzt, damit es funktioniert.

Windows verlangt die Beibehaltung der Größe und des Ortes des Laufzeitspeichers nach dem Aufwachen aus S4.

Hinweis: Dies kann verwendet werden, um fehlerhafte Speicherzuordnungsimplementierungen auf älterer, seltener Legacy-Hardware zu umgehen. Beispiele für solche Hardware sind Ivy Bridge-Laptops mit Insyde-Firmware wie der Acer V3-571G. Verwenden Sie diese Option nicht, ohne sich über die Auswirkungen im Klaren zu sein.

7. EnableSafeModeSlide

Typ: plist boolescher Wert

Ausfallsicher: false

Beschreibung: Passt den Bootloader so an, dass KASLR im abgesicherten Modus aktiviert wird.

Diese Option ist für Benutzer relevant, die Probleme beim Booten in den abgesicherten Modus haben (z.B. durch Halten der Umschalttaste oder mit dem Argument -x boot). Standardmäßig erzwingt der abgesicherte Modus 0 slide, als ob das System mit dem Boot-Argument slide=0 gestartet worden wäre.

- Mit dieser Eigenart wird versucht, die Datei boot.efi zu patchen, um diese Beschränkung aufzuheben und die Verwendung anderer Werte (von 1 bis einschließlich 255) zu ermöglichen.

- Diese Eigenart erfordert die Aktivierung von ProvideCustomSlide.

Hinweis: Die Notwendigkeit dieser Option ist abhängig von der Verfügbarkeit des abgesicherten Modus. Sie kann aktiviert werden, wenn das Booten in den

abgesicherten Modus fehlschlägt.

8. EnableWriteUnprotector

Typ: plist boolean

Failsafe: false

Beschreibung: Erlaubt Schreibzugriff auf den Code der UEFI-Laufzeitdienste.

Diese Option umgeht WˆX-Berechtigungen in Codeseiten von UEFI-Laufzeitdiensten, indem das Schreibschutz-Bit (WP) aus dem CR0-Register während ihrer Ausführung entfernt wird. Diese Eigenart erfordert das OC_FIRMWARE_RUNTIME-Protokoll, das in OpenRuntime.efi implementiert ist.

Hinweis: Diese Eigenart kann möglicherweise die Sicherheit der Firmware schwächen. Bitte verwenden Sie RebuildAppleMemoryMap, wenn die Firmware die Memory Attribute Table (MAT) unterstützt. Schauen Sie im OCABC: MAT support is 1/0 log entry um festzustellen, ob MAT unterstützt wird.

9. ForceBooterSignature

Typ: plist boolescher Wert

Failsafe: false

Beschreibung: Setzt die macOS Boot-Signatur auf den OpenCore Launcher.

Die Bootersignatur, im Wesentlichen ein SHA-1-Hash des geladenen Images, wird von Mac EFI verwendet, um die Authentizität des Bootloaders beim Aufwachen aus dem Ruhezustand zu überprüfen. Diese Option zwingt macOS, den SHA-1-Hash des OpenCore-Launchers als Bootersignatur zu verwenden, damit OpenCore den Ruhezustand auf Mac EFI-Firmware aufwecken kann.

Hinweis: Der Pfad zum OpenCore-Launcher wird über die Eigenschaft LauncherPath ermittelt.

10. ForceExitBootServices

Typ: plist boolescher Wert

Ausfallsicherheit: false

Beschreibung: Wiederholung des ExitBootServices-Aufrufs mit neuer Memory Map bei Fehlschlag.

Versucht sicherzustellen, dass der ExitBootServices-Aufruf erfolgreich ist. Falls erforderlich, kann ein veraltetes MemoryMap-Schlüsselargument verwendet werden, indem die aktuelle MemoryMap abgerufen und der ExitBootServices-Aufruf erneut versucht wird.

Hinweis: Die Notwendigkeit dieser Eigenart wird durch frühe Boot-Abstürze der Firmware bestimmt. Verwenden Sie diese Option nicht, ohne sich über die Auswirkungen im Klaren zu sein.

11. ProtectMemoryRegions

Typ: plist boolescher Wert

Ausfallsicherheit: false

Beschreibung: Schützt Speicherbereiche vor falschem Zugriff.

Einige Firmware-Typen weisen bestimmte Speicherbereiche falsch zu:

- Die CSM-Region kann als Boot-Services-Code oder als Daten markiert sein, wodurch sie als freier Speicher für den XNU-Kernel übrig bleibt.

- MMIO-Regionen können als reservierter Speicher markiert werden und bleiben ungemappt. Es kann jedoch erforderlich sein, dass sie zur Laufzeit für die NVRAM-Unterstützung zugänglich sind.

Mit dieser Eigenart wird versucht, die Typen dieser Regionen festzulegen, z. B. ACPI NVS für CSM oder MMIO für MMIO.

Hinweis: Der Bedarf für diese Eigenart wird durch Artefakte, Sleep-Wake-Probleme und Boot-Fehler bestimmt. Diese Eigenart wird normalerweise nur von sehr alter Firmware benötigt.

12. ProtectSecureBoot

Typ: plist boolescher Wert

Ausfallsicherheit: false

Beschreibung: Schützt UEFI Secure Boot-Variablen vor dem Schreiben.

Meldet Sicherheitsverletzungen bei Versuchen, vom Betriebssystem aus in die Variablen db, dbx, PK und KEK zu schreiben.

Hinweis: Diese Eigenart versucht, Probleme mit NVRAM-Implementierungen mit Fragmentierungsproblemen zu vermeiden, wie z.B. auf dem MacPro5,1 sowie auf bestimmter Insyde-Firmware ohne Garbage Collection oder mit defekter Garbage Collection.

13. ProtectUefiServices

Typ: plist boolescher Wert

Ausfallsicher: false

Beschreibung: Schützt UEFI-Dienste davor, von der Firmware außer Kraft gesetzt zu werden.

Einige moderne Firmware, auch auf virtuellen Maschinen wie VMware, kann Zeiger auf UEFI-Dienste während des Ladens von Treibern und ähnlichen Aktionen aktualisieren. Folglich behindert dies direkt andere Macken, die sich auf die Speicherverwaltung auswirken, wie DevirtualiseMmio, ProtectMemoryRegions oder RebuildAppleMemoryMap, und kann auch andere Macken behindern, je nach Umfang dieser Macken.

GRUB shim nimmt ähnliche fliegende Änderungen an verschiedenen UEFI-Image-Diensten vor, die ebenfalls durch diese Eigenart geschützt sind.

Hinweis 1: Unter VMware kann die Notwendigkeit dieser Eigenart durch das Erscheinen der Meldung "Ihr Mac OS-Gast läuft möglicherweise unzuverlässig mit mehr als einem virtuellen Kern" bestimmt werden.

Hinweis 2: Diese Eigenart ist für den korrekten Betrieb erforderlich, wenn OpenCore von GRUB mit aktiviertem BIOS Secure Boot in der Kette geladen wird.

14. Benutzerdefinierte Folie bereitstellen

Typ: plist boolescher Wert

Ausfallsicherheit: false

Beschreibung: Benutzerdefinierte KASLR-Folien bei geringem Speicherplatz zur Verfügung stellen.

Diese Option führt eine Memory-Map-Analyse der Firmware durch und prüft, ob alle Slides (von 1 bis 255) verwendet werden können. Da boot.efi diesen Wert zufällig mit rdrand oder pseudo-zufällig mit rdtsc generiert, besteht die Möglichkeit, dass der Bootvorgang fehlschlägt, wenn ein konfliktbehafteter Slide ausgewählt wird. In Fällen, in denen potenzielle Konflikte bestehen, zwingt diese Option macOS dazu, einen Pseudozufallswert aus den verfügbaren Werten auszuwählen. Dies stellt auch sicher, dass das Argument slide= niemals an das Betriebssystem weitergegeben wird (aus Sicherheitsgründen).

Hinweis: Die Notwendigkeit dieser Eigenart wird durch den OCABC bestimmt: Only N/256 slide values are usable! im Debug-Protokoll.

15. ProvideMaxSlide

Typ: plist Ganzzahl

Failsafe: 0

Beschreibung: Stellt die maximale KASLR-Folie bereit, wenn höhere Folien nicht verfügbar sind.

Diese Option überschreibt das maximale Dia von 255 durch einen benutzerdefinierten Wert zwischen 1 und 254 (einschließlich), wenn ProvideCustomSlide aktiviert ist. Es wird davon ausgegangen, dass moderne Firmware den Pool-Speicher von oben nach unten alloziert, was effektiv zu freiem Speicher führt, wenn der Slide-Scan später als temporärer Speicher während des Kernel-Ladens verwendet wird. Wenn ein solcher Speicher nicht verfügbar ist, stoppt diese Option die Auswertung höherer Dias.

Hinweis: Die Notwendigkeit dieser Eigenart ergibt sich aus zufälligen Boot-Fehlern, wenn ProvideCustomSlide aktiviert ist und die zufällig ausgewählte Folie in den nicht verfügbaren Bereich fällt. Wenn AppleDebug aktiviert ist, enthält das Debug-Protokoll normalerweise Meldungen wie AAPL: [EB|'LD:LKC] } Err(0x9). Um den optimalen Wert zu finden, hängen Sie slide=X an die Boot-Args an, wobei X der Slide-Wert ist, und wählen Sie den größten Wert, der nicht zu Boot-Fehlern führt.

16. RebuildAppleMemoryMap

Typ: plist boolescher Wert

Ausfallsicher: false

Beschreibung: Erzeugt eine macOS-kompatible Memory Map.

Der Apple-Kernel hat einige Einschränkungen beim Parsen der UEFI-Speicherabbildung:

- Die Größe der Memory Map darf 4096 Bytes nicht überschreiten, da der Apple-Kernel sie als eine einzige 4K-Seite abbildet. Da einige Firmware-Typen sehr große Memory Maps haben können, möglicherweise über 100 Einträge, wird der Apple-Kernel beim Booten abstürzen.

- Die Tabelle der Speicherattribute wird ignoriert. EfiRuntimeServicesCode-Speicher erhält statisch RX-Berechtigungen, während alle anderen Speichertypen RW-Berechtigungen erhalten. Da einige Firmware-Treiber zur Laufzeit in globale Variablen schreiben können, stürzt der Apple-Kernel beim Aufruf von UEFI-Laufzeitdiensten ab, wenn der Abschnitt .data des Treibers nicht den Typ EfiRuntimeServicesData hat.

Um diese Einschränkungen zu umgehen, wendet diese Eigenart Speicherattribut-Tabellenberechtigungen auf die an den Apple-Kernel übergebene Speicherabbildung an und versucht optional, zusammenhängende Slots ähnlichen Typs zu vereinheitlichen, wenn die resultierende Speicherabbildung 4 KB überschreitet.

Hinweis 1: Da einige Firmware-Typen mit falschen Speicherschutztabellen ausgeliefert werden, wird diese Eigenart oft zusammen mit SyncRuntimePermissions verwendet.

Hinweis 2: Die Notwendigkeit dieser Eigenart wird durch frühe Boot-Fehler bestimmt. Diese Eigenart ersetzt EnableWriteUnprotector auf Firmware, die Memory Attribute Tables (MAT) unterstützt. Diese Eigenart ist normalerweise unnötig, wenn OpenDuetPkg verwendet wird, kann aber aus noch unklaren Gründen erforderlich sein, um macOS 10.6 und früher zu booten.

17. ResizeAppleGpuBars Typ: plist ganzzahlig

Failsafe: -1

Beschreibung: Reduziert die GPU PCI BAR-Größen für die Kompatibilität mit macOS.

Diese Eigenart reduziert die GPU PCI BAR-Größen für Apple macOS bis zum angegebenen Wert oder niedriger, wenn dieser nicht unterstützt wird. Der angegebene Wert folgt der PCI Resizable BAR Spezifikation. Während Apple macOS ein theoretisches Maximum von 1 GB unterstützt, kann es in der Praxis vorkommen, dass alle nicht standardmäßigen Werte nicht korrekt funktionieren. Aus diesem Grund ist der einzige unterstützte Wert für diese Eigenart die minimal unterstützte BAR-Größe, d.h. 0. Verwenden Sie -1, um diese Eigenart zu deaktivieren.

Für Entwicklungszwecke kann man Risiken eingehen und andere Werte ausprobieren. Nehmen wir eine GPU mit 2 BARs: - BAR0 unterstützt Größen von 256 MB bis 8 GB. Sein Wert ist 4 GB.

- BAR1 unterstützt Größen von 2 MB bis 256 MB. Sein Wert ist 256 MB.

Beispiel 1: Die Einstellung von ResizeAppleGpuBars auf 1 GB ändert BAR0 auf 1 GB und lässt BAR1 unverändert. Beispiel 2: Die Einstellung von ResizeAppleGpuBars auf 1 MB ändert BAR0 auf 256 MB und BAR0 auf 2 MB. Beispiel 3: Wenn Sie ResizeAppleGpuBars auf 16 GB setzen, werden keine Änderungen vorgenommen.

Hinweis: Siehe ResizeGpuBars quirk für die allgemeine Konfiguration der GPU PCI BAR-Größe und weitere Details über die Technologie.

18. SetupVirtualMap

Typ: plist boolean

Failsafe: false

Beschreibung: Richtet den virtuellen Speicher bei SetVirtualAddresses ein.

Einige Arten von Firmware greifen nach einem SetVirtualAddresses-Aufruf über virtuelle Adressen auf den Speicher zu, was zu frühen Boot-Abstürzen führt. Diese Eigenart umgeht das Problem, indem sie eine frühe Boot-Identitätszuordnung der zugewiesenen virtuellen Adressen zum physischen Speicher vornimmt.

Hinweis: Die Notwendigkeit dieser Eigenart wird durch frühe Boot-Fehler bestimmt.

19. SignalAppleOS

Typ: plist boolescher Wert

Ausfallsicherheit: false

Beschreibung: Meldet das Laden von macOS durch OS Info für jedes Betriebssystem.

Diese Eigenart ist bei Mac-Firmware nützlich, die verschiedene Betriebssysteme mit unterschiedlichen Hardwarekonfigurationen lädt. Zum Beispiel soll es die Intel GPU in Windows und Linux in einigen Dual-GPU MacBook Modellen aktivieren.

20. SyncRuntimePermissions

Typ: plist boolesch

Ausfallsicher: false

Beschreibung: Aktualisiert die Speicherberechtigungen für die Laufzeitumgebung.

Einige Firmware-Typen können Laufzeitberechtigungen nicht richtig handhaben:

- Sie markieren OpenRuntime fälschlicherweise als nicht ausführbar in der Memory Map.

- Sie markieren OpenRuntime fälschlicherweise als nicht ausführbar in der Speicherattribut-Tabelle. - Sie verlieren Einträge in der Speicherattribut-Tabelle, nachdem OpenRuntime geladen wurde.

- Sie markieren Einträge in der Speicherattribut-Tabelle als lesend-schreibend-ausführend.

Dieser Quirk versucht, die Memory Map und die Speicherattribut-Tabelle zu aktualisieren, um dies zu korrigieren.

Anmerkung: Die Notwendigkeit dieser Eigenart wird durch frühe Boot-Fehler angezeigt (Anmerkung: beinhaltet sowohl das Anhalten bei einem schwarzen Bildschirm als auch einen offensichtlicheren Absturz). Es ist besonders wahrscheinlich, dass das frühe Booten von Windows oder Linux (aber nicht immer beides) auf betroffenen Systemen betroffen ist. In der Regel ist nur Firmware betroffen, die nach 2017 veröffentlicht wurde.

6 DeviceProperties 6.1 Einführung

Die Gerätekonfiguration wird macOS mit einem speziellen Puffer namens EfiDevicePathPropertyDatabase zur Verfügung gestellt. Dieser Puffer ist eine serialisierte Zuordnung von DevicePaths zu einer Zuordnung von Eigenschaftsnamen und deren Werten.

Eigenschaftsdaten können mit gfxutil debuggt werden. Um aktuelle Eigenschaftsdaten zu erhalten, verwenden Sie unter macOS den folgenden Befehl:

---

ioreg -lw0 -p IODeviceTree -n efi -r -x | **grep** device-properties | **sed** 's/.*<//;s/>.*//' > /tmp/device-properties.hex &&
gfxutil /tmp/device-properties.hex /tmp/device-properties.plist && **cat** /tmp/device-properties.plist

---

Geräteeigenschaften sind Teil der IODeviceTree-Ebene (gIODT) der macOS I/O Registry. Diese Ebene hat mehrere Konstruktionsphasen, die für die Initialisierung der Plattform relevant sind. Während die frühe Konstruktionsphase vom XNU-Kernel in der IODeviceTreeAlloc-Methode durchgeführt wird, wird der Großteil der Konstruktion vom Plattformexperten durchgeführt, der in AppleACPIPlatformExpert.kext implementiert ist.

AppleACPIPlatformExpert beinhaltet zwei Stufen der IODeviceTree-Konstruktion, die durch den Aufruf von AppleACPIPlatformExpert::mergeDeviceProperties implementiert werden:

1. Während der Initialisierung der ACPI-Tabelle durch das rekursive Scannen des ACPI-Namensraums durch die Aufrufe von AppleACPIPlatformExpert::createDTNubs.

2. Während der IOService-Registrierung (IOServices::registerService) Callbacks, die als Teil der AppleACPIPlatformExpert::platformAdjustService-Funktion und ihrer privaten Worker-Methode AppleACPIPlatformExpert::platformAdjustPCIDevice spezifisch für die PCI-Geräte implementiert sind.

Die Anwendung der Stufen hängt vom Vorhandensein des Geräts in den ACPI-Tabellen ab. Die erste Stufe gilt sehr früh, aber ausschließlich für die in den ACPI-Tabellen vorhandenen Geräte. Die zweite Stufe gilt für alle Geräte viel später nach der PCI-Konfiguration und kann die erste Stufe wiederholen, wenn das Gerät nicht in ACPI vorhanden war.

Für alle Kernel-Erweiterungen, die die IODeviceTree-Ebene ohne Sondierung inspizieren können, wie Lilu und seine Plugins (z. B. WhateverGreen), ist es besonders wichtig, das Vorhandensein von Geräten in den ACPI-Tabellen sicherzustellen. Andernfalls kann es zu fehlerhaftem Verhalten kommen, weil die injizierten Geräteeigenschaften ignoriert werden, da sie nicht in der ersten Phase erstellt wurden. Siehe SSDT-IMEI.dsl und SSDT-BRG0.dsl für ein Beispiel.

6.2 Properties

1. Add

Typ: plist dict

Beschreibung: Setzt Geräteeigenschaften von einer Karte (plist dict) mit Gerätepfaden auf eine Karte (plist dict) mit Variablennamen und ihren Werten im plist multidata-Format.

Hinweis 1: Gerätepfade müssen im kanonischen String-Format angegeben werden (z. B. PciRoot(0x0)/Pci(0x1,0x0)/Pci(0x0,0x0)). Hinweis 2: Vorhandene Eigenschaften werden nicht geändert, es sei denn, sie werden im Abschnitt DeviceProperties Delete gelöscht.

2. Delete

Typ: plist dict

Beschreibung: Entfernt Geräteeigenschaften aus einer Zuordnung (plist dict) von Gerätepfaden zu einem Array (plist array) von Variablennamen im plist-String-Format.

Hinweis: Derzeit können bestehende Eigenschaften nur auf Firmware mit DeviceProperties-Treibern (z.B. Apple) vorhanden sein. Daher gibt es normalerweise keinen Grund, Variablen zu löschen, es sei denn, es wurde ein neuer Treiber installiert.

6.3 Common Properties

Einige bekannte Eigenschaften sind:

- device-id

Benutzerspezifischer Gerätebezeichner, der für den Abgleich von E/A-Kits verwendet wird. Hat einen 4-Byte-Datentyp.

S. 23

- vendor-id

Benutzerspezifische Hersteller-Kennung, die für den Abgleich von E/A-Kits verwendet wird. Hat 4 Byte Datentyp.

- AAPL,ig-platform-id

Intel GPU-Framebuffer-Kennung, die für die Framebuffer-Auswahl bei Ivy Bridge und neueren Modellen verwendet wird. Hat einen 4-Byte-Datentyp.

- AAPL,snb-platform-id

Intel GPU-Framebuffer-Kennung, die für die Framebuffer-Auswahl unter Sandy Bridge verwendet wird. Hat einen Datentyp von 4 Byte.

- layout-id

Audio-Layout, das für die Auswahl des AppleHDA-Layouts verwendet wird. Hat einen Datentyp von 4 Byte.

S. 24

7 Kernel

7.1 Einführung

Dieser Abschnitt ermöglicht die Anwendung verschiedener Arten von Kernelspace-Modifikationen am Apple Kernel (XNU). Die Modifikationen ermöglichen derzeit die Injektion von Treibern (kext), das Patchen von Kernel und Treibern sowie das Blockieren von Treibern.

7.2 Properties

1. Add

Typ: plist-Array

Failsafe: Empty

Beschreibung: Lädt ausgewählte Kernel-Erweiterungen (kexts) aus dem Verzeichnis OC/Kexts.

Muss mit plist dict-Werten gefüllt werden, die die einzelnen Kext beschreiben. Siehe den Abschnitt Add Properties section weiter unten für Details. Hinweis 1: Die Ladereihenfolge

basiert auf der Reihenfolge, in der die Kexts im Array erscheinen. Folglich müssen Abhängigkeiten

vor den Kexten erscheinen, die von ihnen abhängen.

Hinweis 2: Um die Reihenfolge der Abhängigkeiten zu verfolgen, prüfen Sie den OSBundleLibraries key in der Datei Info.plist des hinzuzufügenden Kexts. Jedes unter diesem Schlüssel enthaltene Kext ist eine Abhängigkeit, die vor dem hinzuzufügenden Kext erscheinen muss.

Hinweis 3: Kexts können innere Kexts (Plugins) enthalten, die im Bundle enthalten sind. Solche Plugins müssen separat hinzugefügt werden und folgen den gleichen globalen Ordnungsregeln wie andere Kexts.

2. Block

Typ: plist-Array

Failsafe: Empty

Beschreibung: Entfernt ausgewählte Kernel-Erweiterungen (kexts) aus dem vorverlinkten Kernel.

Wird mit plist-Wörterbuchwerten gefüllt, die jeden blockierten Kext beschreiben. Siehe die Block Properties section weiter unten für Details.

3. Emulate

Typ: plist dict

Beschreibung: Emulation bestimmter Hardware im Kernelspace über Parameter, die im Abschnitt Emulationseigenschaften weiter unten beschrieben werden.

4. Force

Typ: plist array

Failsafe: Empty

Beschreibung: Lädt Kernel-Erweiterungen (kexts) vom System-Volume, wenn sie nicht zwischengespeichert sind.

Soll mit plist dict-Werten gefüllt werden, die jeden kext beschreiben. Siehe den Abschnitt Force Properties weiter unten für Details. Dieser Abschnitt behebt das Problem des Einfügens von Kexts, die von anderen Kexts abhängen, die ansonsten nicht zwischengespeichert werden. Das Problem betrifft typischerweise ältere Betriebssysteme, bei denen verschiedene abhängige Kexts, wie IOAudioFamily oder IONetworkingFamily, nicht standardmäßig im Kernel-Cache vorhanden sind.

Anmerkung 1: Die Ladereihenfolge basiert auf der Reihenfolge, in der die Kexts im Array erscheinen. Daher müssen die Abhängigkeiten vor den Kexts erscheinen, die von ihnen abhängen.

Hinweis 2: Force geschieht vor Add.

Hinweis 3: Die Signatur des "erzwungenen" Kext wird in keiner Weise überprüft. Das macht die Verwendung dieser Funktion extrem gefährlich und für sicheres Booten unerwünscht.

Hinweis 4: Diese Funktion funktioniert möglicherweise nicht auf verschlüsselten Partitionen in neueren Betriebssystemen.

5. Patches

Typ: plist-Array

Failsafe: Empty

Beschreibung: Führt binäre Patches im Kernel und in den Treibern durch, bevor der Treiber hinzugefügt oder entfernt wird.

Muss mit Plist-Wörterbuchwerten gefüllt werden, die jeden Patch beschreiben. Einzelheiten finden Sie im Patch Properties section weiter unten.

6. Quirks

Typ: plist dict

Beschreibung: Wendet einzelne Kernel- und Treiber-Quirks an, die im Abschnitt Quirks-Eigenschaften weiter unten beschrieben werden.

7. Scheme

Typ: plist dict

Beschreibung: Definieren Sie den kernelspace operation mode über Parameter, die im Abschnitt Schemaeigenschaften weiter unten beschrieben werden.

7.3 Eigenschaften hinzufügen

1. Arch

Typ: plist-Zeichenkette

Failsafe: Any: (gilt für jede unterstützte Architektur) Beschreibung: Kext-Architektur (i386, x86_64).

2. BundlePfad

Typ: plist string

Failsafe: Empty

Beschreibung: Pfad des Kext-Bundles (z. B. Lilu.kext oder MyKext.kext/Contents/PlugIns/MySubKext.kext).

3. Kommentar

Typ: plist string

Failsafe: Empty

Beschreibung: Beliebige ASCII-Zeichenkette, die verwendet wird, um eine für den Menschen lesbare Referenz für den Eintrag zu liefern. Ob dieser Wert verwendet wird, hängt von der Implementierung ab.

4. Enabled

Typ: plist boolean

Failsafe: false

Beschreibung: Auf true gesetzt, um diese Kernel-Erweiterung hinzuzufügen.

5. AusführbarerPfad

Typ: plist string

Ausfallsicher: Leer

Beschreibung: Pfad der ausführbaren Datei von Kext relativ zum Bundle (z. B. Contents/MacOS/Lilu).

6. MaxKernel

Typ: plist-Zeichenkette

Failsafe: Empty

Beschreibung: Fügt die Kernel-Erweiterung auf der angegebenen macOS-Version oder älter hinzu.

Die Kernelversion kann mit dem Befehl uname -r ermittelt werden und sollte wie 3 durch Punkte getrennte Zahlen aussehen, z.B. 18.7.0 ist die Kernelversion für 10.14.6. Die Interpretation der Kernelversion ist wie folgt implementiert:

$$
\begin{aligned}
ParseDarwinVersion(\kappa, \lambda, \mu) = \kappa \cdot 10000 & \quad \text{Where } \kappa \in (0, 99) \text{ is kernel version major} \\
+ \lambda \cdot 100 & \quad \text{Where } \lambda \in (0, 99) \text{ is kernel version minor} \\
+ \mu & \quad \text{Where } \mu \in (0, 99) \text{ is kernel version patch}
\end{aligned}
$$

Kernel version comparison is implemented as follows:

$$
\alpha = \begin{cases} ParseDarwinVersion(\texttt{MinKernel}), & \text{If } \texttt{MinKernel} \text{ is valid} \\ 0 & Otherwise \end{cases}
$$

$$
\beta = \begin{cases} ParseDarwinVersion(\texttt{MaxKernel}), & \text{If } \texttt{MaxKernel} \text{ is valid} \\ \infty & Otherwise \end{cases}
$$

$$
\gamma = \begin{cases} ParseDarwinVersion(FindDarwinVersion()), & \text{If valid } \texttt{"Darwin Kernel Version"} \text{ is found} \\ \infty & Otherwise \end{cases}
$$

$$
f(\alpha, \beta, \gamma) = \alpha \leq \gamma \leq \beta
$$

Hier wird angenommen, dass das Argument ParseDarwinVersion 3 ganze Zahlen sind, die durch Aufteilung der Darwin-Kernelversionszeichenkette von links nach rechts durch das Symbol . erhalten werden. Die Funktion FindDarwinVersion sucht die Darwin-Kernelversion, indem sie die Zeichenfolge "Darwin Kernel Version κ.λ.μ" im Kernel-Image sucht.

7. MinKernel

Typ: plist string

Failsafe: Empty

Beschreibung: Fügt die Kernel-Erweiterung auf der angegebenen macOS-Version oder einer neueren Version hinzu.

Hinweis: Siehe die Beschreibung von Add MaxKernel für die passende Logik.

8. PlistPath

Typ: plist string

Failsafe: Empty

Beschreibung: Kext Info.plist-Pfad relativ zum Bundle (z. B. Contents/Info.plist).

7.4 Blockeigenschaften

1. Arch

Typ: plist string

Failsafe: Any (gilt für jede unterstützte Architektur) Beschreibung: Kext-Block-Architektur (i386, x86_64).

2. Kommentar

Typ: plist string

Failsafe: Empty

Beschreibung: Beliebige ASCII-Zeichenkette, die verwendet wird, um eine für den Menschen lesbare Referenz für den Eintrag zu liefern. Ob dieser Wert verwendet wird, hängt von der Implementierung ab.

3. Aktiviert

Typ: plist boolean

Failsafe: false

Beschreibung: Auf true gesetzt, um diese Kernel-Erweiterung zu blockieren.

4. Identifier:

Typ: plist string

Failsafe: Empty

Beschreibung: Identifier des Kext-Bündels (z. B. com.apple.driver.AppleTyMCEDriver).

5. MaxKernel

Typ: plist string

Failsafe: Empty

Beschreibung: Blockiert die Kernel-Erweiterung auf der angegebenen macOS-Version oder älter.

Hinweis: Siehe die Beschreibung von Add MaxKernel für die passende Logik.

6. MinKernel

Typ: plist string

Failsafe: Empty

Beschreibung: Blockiert die Kernel-Erweiterung auf der angegebenen macOS-Version oder neuer.

Hinweis: Siehe die Beschreibung von Add MaxKernel für die passende Logik.

7. Strategie

Typ: plist string

Failsafe: Disable (Zwingt den Kernel-Treiber kmod startup code dazu, einen Fehler zurückzugeben) Beschreibung: Bestimmt das Verhalten des Kernel-Treibers beim Blockieren.

Gültige Werte:

- Deaktivieren - Erzwingt, dass der kmod-Startcode des Kernel-Treibers einen Fehler zurückgibt.

- Exclude - Entfernt den Kernel-Treiber aus dem Kernel-Cache, indem der Plist-Eintrag gelöscht und mit Nullen aufgefüllt wird. Hinweis: Es ist riskant, einen Kext auszuschließen, der eine Abhängigkeit von anderen ist.

S. 27

Hinweis 2: Im Moment wird Exclude nur auf prelinkedkernel und neuere Mechanismen angewendet. Hinweis 3: In den meisten Fällen erfordert die Strategie Exclude, dass der neue Kext als Ersatz injiziert wird.

7.5 Emulieren von Eigenschaften

1. Cpuid1Data

Typ: plist data, 16 bytes

Failsafe: All zero

Beschreibung: Folge von EAX-, EBX-, ECX- und EDX-Werten, die den Aufruf CPUID (1) im XNU-Kernel ersetzen.

Diese Eigenschaft erfüllt in erster Linie drei Anforderungen:

- Ermöglichung der Unterstützung für ein nicht unterstütztes CPU-Modell (z. B. Intel Pentium).

- Ermöglichung der Unterstützung für ein CPU-Modell, das noch nicht von einer bestimmten Version von macOS unterstützt wird (typischerweise alte Versionen). - Aktivieren der XCPM-Unterstützung für eine nicht unterstützte CPU-Variante.

Hinweis 1: Es kann auch der Fall eintreten, dass das CPU-Modell zwar unterstützt wird, aber keine Energieverwaltung vorhanden ist (z. B. bei virtuellen Maschinen). In diesem Fall können MinKernel und MaxKernel gesetzt werden, um die CPU-Virtualisierung und

Dummy-Energiemanagement-Patches auf die jeweilige macOS-Kernelversion zu beschränken.

Hinweis 2: Normalerweise muss nur der Wert von EAX, der die vollständige CPUID darstellt, berücksichtigt werden; die übrigen Bytes sollten als Nullen belassen werden. Die Byte-Reihenfolge ist Little Endian. Zum Beispiel steht C3 06 03 00 für die CPUID 0x0306C3 (Haswell).

Hinweis 3: Für die XCPM-Unterstützung wird empfohlen, die folgenden Kombinationen zu verwenden. Bitte beachten Sie, dass Sie die richtigen Frequenzvektoren für die installierte CPU einstellen müssen.

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
- 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
- Rocket Lake (0x0A0670) to Comet Lake (0x0A0655):
  Cpuid1Data: 55 06 0A 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
- Alder Lake (0x090672) to Comet Lake (0x0A0655):
  Cpuid1Data: 55 06 0A 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

Hinweis 4: Beachten Sie, dass die folgenden Konfigurationen von XCPM nicht unterstützt werden (zumindest im Auslieferungszustand):

- Consumer Ivy Bridge (0x0306A9), da Apple XCPM für Ivy Bridge deaktiviert hat und für diese CPUs das Legacy Power Management empfiehlt. _xcpm_bootstrap sollte manuell gepatcht werden, um XCPM auf diesen CPUs anstelle dieser Option zu erzwingen.

- Low-End-CPUs (z. B. Haswell+ Pentium), da sie von macOS nicht richtig unterstützt werden. Legacy-Workarounds für ältere Modelle können im Abschnitt Spezielle Hinweise von acidanthera/bugtracker#365 gefunden werden.

2. Cpuid1Mask

Typ: plist data, 16 bytes

Failsafe: All zero

Beschreibung: Bitmaske der aktiven Bits in Cpuid1Data.

Wenn jedes Cpuid1Mask-Bit auf 0 gesetzt ist, wird das ursprüngliche CPU-Bit verwendet, andernfalls nehmen die gesetzten Bits den Wert von Cpuid1Data an.

3. DummyPowerManagement

Typ: plist boolean

Failsafe: false

Bedingung: 10.4

Beschreibung: Deaktiviert AppleIntelCpuPowerManagement.

S. 28

Hinweis 1: Diese Option ist eine bevorzugte Alternative zu NullCpuPowerManagement.kext für CPUs ohne nativen Energieverwaltungstreiber in macOS.

Hinweis 2: Während diese Option typischerweise benötigt wird, um AppleIntelCpuPowerManagement auf nicht unterstützten Plattformen zu deaktivieren, kann sie auch verwendet werden, um diesen Kext in anderen Situationen zu deaktivieren (z.B. wenn Cpuid1Data leer bleibt).

4. MaxKernel

Typ: plist string

Failsafe: Empty

Beschreibung: Emuliert CPUID und wendet DummyPowerManagement auf die angegebene macOS-Version oder älter an.

Hinweis: Siehe die Beschreibung von Add MaxKernel für die passende Logik.

5. MinKernel

Typ: plist string

Failsafe: Empty

Beschreibung: Emuliert CPUID und wendet DummyPowerManagement auf die angegebene macOS-Version oder neuer an.

Hinweis: Siehe die Beschreibung von Add MaxKernel für die passende Logik.

7.6 Force Properties

1. Arch

Typ: plist string

Failsafe: Any (auf jede unterstützte Architektur anwenden) Beschreibung: Kext-Architektur (i386, x86_64).

2. BundlePfad

Typ: plist string

Failsafe: Empty

Beschreibung: Kext-Bundle-Pfad (z. B. System\Library \Extensions \ IONetworkingFamily.kext).

3. Kommentar

Typ: plist string

Failsafe: Empty

Beschreibung: Beliebige ASCII-Zeichenkette, die verwendet wird, um eine für den Menschen lesbare Referenz für den Eintrag zu liefern. Ob dieser Wert verwendet wird, hängt von der Implementierung ab.

4. Enabled

Type: plist boolean

Failsafe: false

Beschreibung: Auf true gesetzt, um diese Kernel-Erweiterung vom Systemvolume zu laden, wenn sie nicht im Kernel-Cache vorhanden ist.

5. AusführbarerPfad

Typ: plist string

Failsafe: Empty

Beschreibung: Pfad der ausführbaren Datei von Kext relativ zum Bundle (z. B. Contents/MacOS/IONetworkingFamily).

6. Identifier

Typ: plist string

Failsafe: Empty

Beschreibung: Identifier des Kextreibers, der vor dem Hinzufügen auf Vorhandensein geprüft wird (z. B. com.apple.iokit.IONetworkingFamily). Nur Treiber, deren Bezeichner nicht im Cache gefunden werden, werden hinzugefügt.

7. MaxKernel

Typ: plist string

Failsafe: Empty

Beschreibung: Fügt die Kernel-Erweiterung auf der angegebenen macOS-Version oder älter hinzu.

Hinweis: Siehe die Beschreibung von Add MaxKernel für die passende Logik.

8. MinKernel

Typ: plist string

S. 29

Failsafe: Empty

Beschreibung: Fügt die Kernel-Erweiterung auf der angegebenen macOS-Version oder neuer hinzu.

Hinweis: Siehe die Beschreibung von Add MaxKernel für die passende Logik.

9. PlistPath

Typ: plist string

Failsafe: Empty

Beschreibung: Kext Info.plist-Pfad relativ zum Bundle (z. B. Contents/Info.plist).

7.7 Patch Properties

1. Arch

Typ: plist string

Failsafe: Any (Anwenden auf jede unterstützte Architektur) Beschreibung: Architektur des Kext-Patches (i386, x86_64).

2. Basis

Typ: plist string

Failsafe: Empty (Ignored)

Beschreibung: Wählt die zum Symbol passende Basis für die Patch-Suche (oder den sofortigen Ersatz) aus, indem die Adresse des angegebenen Symbolnamens ermittelt wird.

3. Kommentar

Typ: plist string

Failsafe: Empty

Beschreibung: Beliebige ASCII-Zeichenkette, die verwendet wird, um eine für den Menschen lesbare Referenz für den Eintrag zu liefern. Ob dieser Wert verwendet wird, hängt von der Implementierung ab.

4. Count

Typ: plist integer

Failsafe: 0

Beschreibung: Anzahl der zu übertragenden Patch-Vorkommen. 0 wendet den Patch auf alle gefundenen Vorkommen an.

5. Enabled

Type: plist boolean

Failsafe: false

Beschreibung: Dieser Kernel-Patch wird nicht verwendet, wenn er nicht auf true gesetzt ist.

6. Find

Typ: plist-Daten

Failsafe: Empty (Sofortige Ersetzung an der Basis)

Beschreibung: Zu suchende Daten. Muss gleich der Größe von Ersetzen sein, wenn gesetzt.

7. Identifier

Typ: plist string

Failsafe: Empty

Beschreibung: Kext-Bundle-Kennung (z.B. com.apple.driver.AppleHDA) oder Kernel für Kernel-Patch.

8. Limit

Typ: plist Limit

Failsafe: 0 (Suche im gesamten Kext oder Kernel) Beschreibung: Maximale Anzahl von Bytes, nach denen gesucht werden soll.

9. Mask

Typ: plist data

Failsafe: Empty (Ignoriert)

Beschreibung: Bitweise Datenmaske, die während des Suchvergleichs verwendet wird. Ermöglicht unscharfe Suche durch Ignorieren nicht maskierter (auf Null gesetzter) Bits. Muss gleich der Größe von Replace sein, wenn gesetzt.

10. MaxKernel

Typ: plist string

Failsafe: Empty

Beschreibung: Patches für Daten auf der angegebenen macOS-Version oder älter.

Hinweis: Siehe die Beschreibung von Add MaxKernel für die passende Logik.

11. MinKernel

Typ: plist string

Failsafe: Empty

Beschreibung: Passt die Daten auf die angegebene macOS-Version oder eine neuere an.

Hinweis: Siehe die Beschreibung von Add MaxKernel für die passende Logik.

12. Replace

Typ: plist-Daten

Failsafe: Empty

Beschreibung: Ersetzungsdaten von einem oder mehreren Bytes.

13. ReplaceMask

Typ: plist data

Failsafe: Empty (Ignoriert)

Beschreibung: Bitweise Datenmaske, die bei der Ersetzung verwendet wird. Ermöglicht unscharfe Ersetzung durch Aktualisierung der maskierten (auf Nicht-Null gesetzten) Bits. Muss gleich der Größe von Replace sein, wenn gesetzt.

14. Skip

Typ: plist integer

Failsafe: 0 (kein Vorkommen überspringen)

Beschreibung: Anzahl der gefundenen Vorkommen, die übersprungen werden sollen, bevor Ersetzungen angewendet werden.

7.8 Quirks Properties

1. AppleCpuPmCfgLock

Typ: plist plist boolean

Failsafe: false

Erfordernis: 10.4

Beschreibung: Deaktiviert die Änderung des MSR-Registers PKG_CST_CONFIG_CONTROL (0xE2) in der Datei AppleIntelCPUPowerManagement.kext, die häufig eine frühe Kernel-Panik verursacht, wenn sie für das Schreiben gesperrt ist.

Einige Firmware-Typen sperren das MSR-Register PKG_CST_CONFIG_CONTROL, und das mitgelieferte Tool ControlMsrE2 kann verwendet werden, um seinen Zustand zu überprüfen. Beachten Sie, dass bei einigen Firmware-Typen dieses Register nur auf einigen Kernen gesperrt ist. Da moderne Firmware eine CFG-Lock-Einstellung bietet, mit der die PKG_CST_CONFIG_CONTROL MSR-Registersperre konfiguriert werden kann, sollte diese Option nach Möglichkeit vermieden werden.

Bei APTIO-Firmware, die keine CFG-Lock-Einstellung in der GUI bietet, ist es möglich, direkt auf die Option zuzugreifen:

(a) Laden Sie UEFITool und IFR-Extractor herunter.

(b) Öffnen Sie das Firmware-Image in UEFITool und suchen Sie den Unicode-String CFG Lock. Wenn dieser nicht vorhanden ist, verfügt die Firmwaremöglicherweise nicht über diese Option und der Vorgang sollte daher abgebrochen werden.

(c) Extrahieren Sie die Setup.bin PE32 Image Section (die UEFITool gefunden hat) über die Menüoption Extract Body.

(d) Führen Sie IFR-Extractor für die extrahierte Datei aus (z.B. ./ifrextract Setup.bin Setup.txt).

(e) Suchen Sie CFG Lock, VarStoreInfo (VarOffset/VarName): in Setup.txt und merken Sie sich den Offset direkt nach (z.B. 0x123).

(f) Laden Sie die von brainsucker kompilierte modifizierte GRUB-Shell herunter und führen Sie sie aus oder verwenden Sie eine neuere Version von datasone.

(g) Geben Sie den Befehl setup_var 0x123 0x00 ein, wobei 0x123 durch den tatsächlichen Offset ersetzt werden sollte, und starten Sie neu.

Achtung! Variablen-Offsets sind nicht nur für jedes Motherboard einzigartig, sondern auch für dessen Firmware-Version. Versuchen Sie niemals, einen Offset zu verwenden, ohne dies zu überprüfen.

Auf ausgewählten Plattformen kann das Tool ControlMsrE2 auch solche versteckten Optionen ändern. Übergeben Sie das gewünschte Argument: lock, unlock für CFG Lock. Oder übergeben Sie interaktiv, um andere versteckte Optionen zu finden und zu ändern. Als letzten Ausweg können Sie das BIOS patchen (nur für fortgeschrittene Benutzer).

S. 31

2. AppleXcpmCfgLock

Typ: plist boolean

Failsafe: false

Erfordert: 10.8 (für ältere Versionen nicht erforderlich)

Beschreibung: Deaktiviert die PKG_CST_CONFIG_CONTROL (0xE2) MSR-Änderung im XNU-Kernel, die häufig eine frühe Kernel-Panik verursacht, wenn sie für das Schreiben gesperrt ist (XCPM-Energieverwaltung).

Hinweis: Diese Option sollte, wann immer möglich, vermieden werden. Siehe die Beschreibung von AppleCpuPmCfgLock für weitere Details.

3. AppleXcpmExtraMsrs

Typ: plist boolean

Failsafe: false

Erforderlich: 10.8 (für ältere Versionen nicht erforderlich)

Beschreibung: Deaktiviert den kritischen Mehrfachzugriff auf MSR für bestimmte CPUs, die keine native XCPM-Unterstützung haben.

Dies wird in der Regel in Verbindung mit dem Abschnitt Emulation auf Haswell-E, Broadwell-E, Skylake-SP und ähnlichen CPUs verwendet. Weitere Details zu den XCPM-Patches sind in acidanthera/bugtracker#365 beschrieben.

Hinweis: Für Ivy Bridge- oder Pentium-CPUs werden zusätzliche, nicht bereitgestellte Patches benötigt. Es wird empfohlen, AppleIntelCpuPowerManagement.kext für erstere zu verwenden.

4. AppleXcpmForceBoost

Typ: plist boolean

Failsafe: false

Erforderlich: 10.8 (für ältere Versionen nicht erforderlich)

Beschreibung: Erzwingt maximale Leistung im XCPM-Modus.

Dieser Patch schreibt 0xFF00 in MSR_IA32_PERF_CONTROL (0x199) und setzt damit den maximalen Multiplikator für die gesamte Zeit.

Hinweis: Obwohl dies die Leistung erhöhen kann, wird von diesem Patch auf allen Systemen mit Ausnahme derjenigen, die explizit für wissenschaftliche oder mediale Berechnungen vorgesehen sind, dringend abgeraten. Nur bestimmte Xeon-Modelle profitieren typischerweise von diesem Patch.

5. CustomPciSerialDevice

Typ: plist boolean

Failsafe: false

Bedingung: 10.7

Beschreibung: Ändert die Basisadresse der PMIO-Register auf einem angepassten seriellen PCI-Gerät.

Der Patch ändert die Basisadresse des PMIO-Registers, das der XNU-Kernel für die serielle Ein- und Ausgabe verwendet, von der standardmäßig eingebauten seriellen Schnittstelle COM1 0x3F8 auf die Basisadresse, die in der ersten IO BAR eines bestimmten PCI-Geräts gespeichert ist, oder auf eine spezifische Basisadresse (z. B. 0x2F8 für COM2).

Hinweis: Standardmäßig ist die serielle Protokollierung deaktiviert. Das Boot-Argument serial=3, das die serielle Ein- und Ausgabe aktiviert, sollte verwendet werden, damit XNU Protokolle auf der seriellen Schnittstelle ausgibt.

Hinweis 2: Zusätzlich zu diesem Patch sollte das Anhängen von kext Apple16X50PCI0 verhindert werden, damit die kprintf-Methode richtig funktioniert. Dies kann erreicht werden, indem die Eigenschaft class-code des PCI-Geräts für die serielle Schnittstelle im Abschnitt DeviceProperties auf FFFFFFFF gesetzt wird (d. h. erst löschen, dann hinzufügen). Als alternative Lösung kann auch ein codefreier Kext PCIeSerialDisable.kext verwendet werden, der im Spoiler PCIeSerialDisable.kext/Contents/Info.plist unter acidanthera/bugtracker#1954 zu finden ist.

Hinweis 3: Damit dieser Patch korrekt angewendet werden kann, muss Override aktiviert sein und alle Tasten müssen in Custom unter Misc->Serial richtig eingestellt sein.

Hinweis 4: Dieser Patch ist für die PMIO-Unterstützung gedacht und wird daher nicht angewendet, wenn UseMmio im Abschnitt Misc->Serial->Custom auf false gesetzt ist. Für

MMIO gibt es die Boot-Argumente pcie_mmio_uart=ADDRESS und mmio_uart=ADDRESS, die es dem Kernel erlauben, MMIO für den Zugriff auf die serielle Schnittstelle zu verwenden.

Hinweis 5: Die serielle Baudrate muss sowohl in BaudRate unter Misc->Serial->Custom als auch über das Boot-Argument serialbaud=VALUE korrekt eingestellt werden, wobei beide übereinstimmen sollten. Die Standard-Baudrate ist 115200.

6. CustomSMBIOSGuid Typ: plist boolean

Failsafe: false

S. 32

Bedingung: 10.4

Beschreibung: Führt GUID-Patching für UpdateSMBIOSMode Custom mode durch. Normalerweise relevant für Dell-Laptops.

7. DisableIoMapper

Typ: plist boolean

Failsafe: false

Erforderlich: 10.8 (für ältere Versionen nicht erforderlich)

Beschreibung: Deaktiviert die IOMapper-Unterstützung in XNU (VT-d), was zu Konflikten mit der Firmware-Implementierung führen kann.

Hinweis 1: Diese Option ist eine bevorzugte Alternative zum Löschen der DMAR-ACPI-Tabelle und dem Deaktivieren von VT-d in den Firmware-Einstellungen, was die VT-d-Unterstützung in anderen Systemen nicht behindert, falls diese diese benötigen.

Hinweis 2: Eine falsch konfigurierte IOMMU in der Firmware kann zu fehlerhaften Geräten wie Ethernet- oder Wi-Fi-Adaptern führen. Zum Beispiel kann ein Ethernet-Adapter ununterbrochen im Link-Up-Link-Down-Zustand zirkulieren und ein Wi-Fi-Adapter kann keine Netzwerke mehr erkennen. Gigabyte ist einer der häufigsten OEMs mit diesen Problemen.

8. DisableLinkeditJettison

Typ: plist boolean

Failsafe: false

Erfordernis: 11

Beschreibung: __LINKEDITJettison Deaktiviert den Code __LINKEDIT jettison.

Mit dieser Option können Lilu.kext und möglicherweise andere Kexts in macOS Big Sur mit ihrer besten Leistung arbeiten, ohne dass das Boot-Argument keepsyms=1 erforderlich ist.

9. DisableRtcChecksum

Typ: plist boolean

Failsafe: false

Erfordernis: 10.4

Beschreibung: Deaktiviert das Schreiben der primären Prüfsumme (0x58-0x59) in AppleRTC.

Hinweis 1: Diese Option schützt andere Bereiche nicht vor dem Überschreiben, siehe RTCMemoryFixup Kernel-Erweiterung, wenn dies gewünscht ist.

Hinweis 2: Diese Option schützt keine Bereiche vor dem Überschreiben in der Firmware-Phase (z.B. macOS-Bootloader), siehe AppleRtcRam-Protokollbeschreibung, wenn dies gewünscht ist.

10. ExtendBTFeatureFlags

Typ: plist boolescher Wert

failsafe: false

Erfordernis: 10.8-11

Beschreibung: FeatureFlags auf 0x0F setzen, um die volle Funktionalität von Bluetooth, einschließlich Continuity, zu erhalten.

Hinweis: Diese Option ist ein Ersatz für BT4LEContinuityFixup.kext, das aufgrund des späten Patching-Fortschritts nicht richtig funktioniert.

11. ExternalDiskIcons

Typ: plist boolean

failsafe: false

Erfordernis: 10.4

Beschreibung: Wendet Patches für den Symboltyp auf AppleAHCIPort.kext an, um interne Festplattensymbole für alle AHCI-Festplatten zu erzwingen.

Hinweis: Diese Option sollte nach Möglichkeit vermieden werden. Moderne Firmwares haben in der Regel kompatible AHCI-Controller.

12. ForceAquantiaEthernet

Typ: plist boolean

Failsafe: false

Erfordernis: 10.15.4

Beschreibung: Aktiviert die Unterstützung von Aquantia AQtion-basierten 10GbE-Netzwerkkarten.

Diese Option aktiviert die Unterstützung für Aquantia AQtion-basierte 10GbE-Netzwerkkarten, die vor macOS 10.15.4 nativ funktioniert haben.

Hinweis: Damit die Aquantia-Karten ordnungsgemäß funktionieren, muss DisableIoMapper deaktiviert sein, die DMAR ACPI-Tabelle darf nicht gelöscht werden und VT-d muss im BIOS aktiviert sein.

Hinweis 2: Dieser Patch sollte die Ethernet-Unterstützung für alle Aquantia AQtion-Serien ermöglichen, wurde jedoch nur auf AQC-107s-basierten 10GbE-Netzwerkkarten getestet.

13. ForceSecureBootScheme

Typ: plist boolean

Failsafe: false

Erfordernis: 11

Beschreibung: Erzwingt das x86-Schema für die IMG4-Überprüfung.

Hinweis: Diese Option ist auf virtuellen Maschinen erforderlich, wenn ein anderes SecureBootModel als x86legacy verwendet wird.

14. IncreasePciBarSize

Typ: plist boolean

Failsafe: false

Erforderlich: 10.10

Beschreibung: Erlaubt IOPCIFamily das Booten mit 2 GB PCI BARs.

Normalerweise beschränkt macOS die PCI BARs auf 1 GB. Das Aktivieren dieser Option erlaubt es macOS (noch) nicht, PCI-Geräte mit größeren BARs zu verwenden.

Hinweis: Diese Option sollte wann immer möglich vermieden werden. Ein Bedarf für diese Option deutet auf eine falsch konfigurierte oder defekte Firmware hin.

15. LapicKernelPanic

Typ: plist boolean

Failsafe: false

Erfordernis: 10.6 (64-bit)

Beschreibung: Deaktiviert Kernel-Panik bei LAPIC-Interrupts.

16. LegacyCommpage

Typ: plist boolean

Failsafe: false

Erfordernis: 10.4 - 10.6

Beschreibung: Ersetzt die standardmäßige 64-Bit-Bcopy-Implementierung von commpage durch eine, die kein SSSE3 benötigt, nützlich für ältere Plattformen. Dies verhindert eine commpage no match for last panic, da keine 64-Bit bcopy-Funktionen verfügbar sind, die kein SSSE3 erfordern.

17. PanicNoKextDump

Typ: plist boolean

Failsafe: false

Erforderlich: 10.13 (für ältere Versionen nicht erforderlich)

Beschreibung: Verhindert, dass der Kernel einen Kext-Dump in das Panic-Log ausgibt und damit die Beobachtung von Panic-Details verhindert. Betrifft 10.13 und höher.

18. PowerTimeoutKernelPanic

Typ: plist boolean

Failsafe: false

Erfordert: 10.15 (für ältere Versionen nicht erforderlich)

Beschreibung: Deaktiviert Kernel-Panik bei setPowerState-Zeitüberschreitung.

Eine zusätzliche Sicherheitsmaßnahme wurde zu macOS Catalina (10.15) hinzugefügt, die eine Kernel-Panik bei einer Zeitüberschreitung beim Stromwechsel für Apple-Treiber verursacht. Manchmal kann dies zu Problemen bei falsch konfigurierter Hardware führen, insbesondere bei digitalem Audio, das manchmal nicht aufwacht. Für Debug-Kernel sollte

das Boot-Argument setpowerstate_panic=0 verwendet werden, was ansonsten dieser Eigenart entspricht.

19. ProvideCurrentCpuInfo

Typ: plist boolean

Failsafe: false

Vorraussetzung: 10.8 (10.14)

Beschreibung: Stellt dem Kernel aktuelle CPU-Informationen zur Verfügung.

Diese Eigenart funktioniert je nach CPU unterschiedlich:

- Für Microsoft Hyper-V werden dem Kernel die korrekten TSC- und FSB-Werte bereitgestellt und die CPU-Topologieüberprüfung deaktiviert (10.8+).

- Für KVM und andere Hypervisoren stellt es vorberechnete MSR 35h-Werte bereit, die eine Kernel-Panik mit -cpu host lösen.

- Für Intel-CPUs fügt es Unterstützung für asymmetrische SMP-Systeme (z. B. Intel Alder Lake) hinzu, indem es die Anzahl der Kerne in die Anzahl der Threads umwandelt, zusammen mit den ergänzenden erforderlichen Änderungen (10.14+).

20. SetApfsTrimTimeout

Typ: plist integer

Failsafe: -1

Erforderlich: 10.14 (für ältere Versionen nicht erforderlich)

Beschreibung: Setzt den Trimm-Timeout in Mikrosekunden für APFS-Dateisysteme auf SSDs.

Das APFS-Dateisystem ist so konzipiert, dass der über die spaceman-Struktur kontrollierte Speicherplatz entweder belegt oder frei ist. Dies kann bei anderen Dateisystemen anders sein, bei denen die Bereiche als benutzt, frei und unmapped markiert werden können. Der gesamte freie Speicherplatz wird beim Start von macOS getrimmt (nicht zugeordnet/dealloziert). Der Trimmvorgang für NVMe-Laufwerke erfolgt in LBA-Bereichen aufgrund der Art des DSM-Befehls mit bis zu 256 Bereichen pro Befehl. Je stärker der Speicher auf dem Laufwerk fragmentiert ist, desto mehr Befehle sind notwendig, um den gesamten freien Speicherplatz zu trimmen.

Abhängig vom SSD-Controller und dem Grad der Fragmentierung des Laufwerks kann die Trim-Prozedur eine beträchtliche Zeit in Anspruch nehmen, was zu einer spürbaren Verlangsamung des Bootvorgangs führt. Der APFS-Treiber ignoriert explizit zuvor nicht

gemappte Bereiche und trimmt sie beim Booten wiederholt. Um solche Boot-Verzögerungen abzumildern, hat der macOS-Treiber eine Zeitüberschreitung (9,999999 Sekunden) eingeführt, die den Trim-Vorgang stoppt, wenn er nicht rechtzeitig beendet wird.

Auf einigen Controllern, wie z. B. Samsung, bei denen der Deallokationsprozess relativ langsam ist, kann diese Zeitüberschreitung sehr schnell erreicht werden. Im Wesentlichen bedeutet dies, dass der Fragmentierungsgrad hoch ist, so dass macOS versuchen wird, die gleichen unteren Blöcke zu trimmen, die zuvor freigegeben wurden, aber nie genug Zeit haben, um höhere Blöcke freizugeben. Das Ergebnis ist, dass das Trimmen auf solchen SSDs bald nach der Installation nicht mehr funktioniert, was zu einer zusätzlichen Abnutzung des Flashs führt.

Eine Möglichkeit, das Problem zu umgehen, besteht darin, die Zeitüberschreitung auf einen extrem hohen Wert zu erhöhen, was auf Kosten von langsamen Boot-Zeiten (zusätzliche Minuten) sicherstellt, dass alle Blöcke getrimmt werden. Wenn Sie diese Option auf einen hohen Wert setzen, z. B. 4294967295, wird sichergestellt, dass alle Blöcke abgeschnitten werden. Alternativ können Sie Over-Provisioning verwenden, sofern dies unterstützt wird, oder eine spezielle, nicht zugeordnete Partition erstellen, in der die Reserveblöcke vom Controller gefunden werden können. Umgekehrt kann die Trim-Operation größtenteils deaktiviert werden, indem ein sehr niedriger Timeout-Wert eingestellt wird, während 0 sie vollständig deaktiviert. Einzelheiten hierzu finden Sie in diesem Artikel.

Hinweis: Der Failsafe-Wert -1 zeigt an, dass dieser Patch nicht angewendet wird, so dass apfs.kext unangetastet bleibt.

Hinweis 2: Unter macOS 12.0 und höher ist es nicht mehr möglich, eine Zeitüberschreitung für das Trimmen anzugeben. Es kann jedoch durch Setzen von 0 deaktiviert werden.

Hinweis 3: Trimmoperationen sind nur in der Boot-Phase betroffen, wenn das Startvolume gemountet ist. Die Angabe von timeout oder die vollständige Deaktivierung von trim mit 0 hat keinen Einfluss auf den normalen Betrieb von macOS.

21. ThirdPartyDrives

Typ: plist boolean

Failsafe: false

Erfordernis: 10.6 (für ältere Versionen nicht erforderlich)

Beschreibung: Wendet Hersteller-Patches auf IOAHCIBlockStorage.kext an, um native Funktionen für Laufwerke von Drittanbietern zu aktivieren, z. B. TRIM auf SSDs oder Unterstützung für den Ruhezustand auf 10.15 und neuer.

Hinweis: Diese Option kann je nach Benutzerpräferenz vermieden werden. NVMe-SSDs sind auch ohne diese Änderung kompatibel. Für AHCI-SSDs auf modernen macOS-Versionen gibt es ein spezielles eingebautes Dienstprogramm namens trimforce. Ab 10.15 erstellt dieses Dienstprogramm die Variable EnableTRIM im Namensraum APPLE_BOOT_VARIABLE_GUID mit dem Wert 01 00 00 00.

22. XhciPortLimit

Typ: plist boolean

Failsafe: false

Erforderlich: 10.11 (für ältere nicht erforderlich)

S. 35

Beschreibung: Patch verschiedener Kexts (AppleUSBXHCI.kext, AppleUSBXHCIPCI.kext, IOUSBHostFamily.kext), um die Begrenzung der USB-Port-Anzahl auf 15 Ports aufzuheben.

Hinweis: Diese Option sollte nach Möglichkeit vermieden werden. Die Begrenzung der USB-Anschlüsse wird durch die Anzahl der verwendeten Bits im locationID-Format bestimmt, und es gibt keine Möglichkeit, dies ohne umfangreiche Änderungen am Betriebssystem zu umgehen. Die einzige gültige Lösung besteht darin, die Anzahl der verwendeten Ports auf 15 zu begrenzen (und einige zu verwerfen). Weitere Details können auf AppleLife.ru gefunden werden.

Scheme Properties

Diese Eigenschaften sind besonders für ältere macOS-Betriebssysteme relevant. Einzelheiten zur Installation und Fehlerbehebung bei solchen macOS-Installationen finden Sie im Abschnitt "Ältere Apple-Betriebssysteme".

1. CustomKernel

Typ: plist boolean

Failsafe: false

Beschreibung: Verwendet den angepassten Kernel-Cache aus dem Kernel-Verzeichnis, das sich im Stammverzeichnis der ESP-Partition befindet.

Nicht unterstützte Plattformen, einschließlich Atom und AMD, benötigen modifizierte Versionen des XNU-Kernels, um booten zu können. Diese Option bietet die Möglichkeit, einen angepassten Kernel-Cache zu verwenden, der solche Änderungen von der ESP-Partition enthält.

2. FuzzyMatch

Typ: plist boolean

Failsafe: false

Beschreibung: Verwendet Kernelcache mit unterschiedlichen Prüfsummen, wenn verfügbar.

Unter macOS 10.6 und früher hat der Dateiname von kernelcache eine Prüfsumme, die im Wesentlichen adler32 aus dem SMBIOS-Produktnamen und dem EfiBoot-device path ist.

Auf bestimmter Firmware unterscheidet sich der EfiBoot-Gerätepfad zwischen UEFI und macOS aufgrund von ACPI oder Hardwarespezifika, wodurch die Kernelcache-Prüfsumme immer unterschiedlich ist.

Diese Einstellung ermöglicht es, den neuesten Kernelcache mit einer passenden Architektur abzugleichen, wenn der Kernelcache ohne Suffix nicht verfügbar ist, was die Startleistung von macOS 10.6 auf verschiedenen Plattformen verbessert.

3. KernelArch

Typ: plist-Zeichenfolge

Failsafe: Auto (Wählt die bevorzugte Architektur automatisch aus)

Beschreibung: Bevorzugt die angegebene Kernel-Architektur (i386, i386-user32, x86_64), wenn verfügbar.

Unter macOS 10.7 und früher kann der XNU-Kernel mit anderen Architekturen als der üblichen x86_64 booten. Mit dieser Einstellung wird die angegebene Architektur zum Booten von macOS verwendet, wenn sie von macOS und der Konfiguration unterstützt wird:

- i386 - i386 (32-Bit)-Kernel verwenden, wenn verfügbar.

- i386-user32 - Verwende i386 (32-Bit) Kernel, wenn verfügbar und erzwinge die Verwendung von 32-Bit Userspace auf 64-Bit fähigen Prozessoren, falls vom Betriebssystem unterstützt.

- Unter macOS wird davon ausgegangen, dass 64-Bit-fähige Prozessoren SSSE3 unterstützen. Dies ist nicht der Fall bei älteren 64-Bit fähigen Pentium-Prozessoren, die unter macOS 10.6 einige Anwendungen zum Absturz bringen. Dieses Verhalten entspricht dem Argument -legacy kernel boot.

- Diese Option ist unter macOS 10.4 und 10.5 nicht verfügbar, wenn es auf 64-Bit-Firmware läuft, da ein nicht initialisiertes 64-Bit-Segment im XNU-Kernel, das AppleEFIRuntime veranlasst, 64-Bit-Code fälschlicherweise als 16-Bit-Code ausführt.

- x86_64 - Verwenden Sie x86_64 (64-Bit) Kernel, wenn verfügbar.

Der Algorithmus zur Bestimmung der bevorzugten Kernel-Architektur wird im Folgenden beschrieben.

(a) Das arch-Argument in den Image-Argumenten (z. B. beim Start über die UEFI-Shell) oder in der boot-args-Variable setzt alle Kompatibilitätsprüfungen außer Kraft und erzwingt die angegebene Architektur, was diesen Algorithmus vervollständigt.

(b) Die OpenCore-Build-Architektur schränkt die Fähigkeiten auf den i386- und i386-user32-Modus für die 32-Bit-Firmware-Variante ein.

(c) Die ermittelte EfiBoot-Version schränkt die Wahl der Architektur ein:

- 10.4-10.5 - i386 oder i386-user32 (nur bei 32-Bit-Firmware) - 10.6 - i386, i386-user32, oder x86_64

- 10.7 - i386 oder x86_64

- 10.8 oder neuere Versionen - x86_64

S. 36

(d) Wenn KernelArch auf Auto eingestellt ist und SSSE3 von der CPU nicht unterstützt wird, sind die Fähigkeiten auf i386-user32 beschränkt, sofern von EfiBoot unterstützt.

(e) Board Identifier (aus SMBIOS) basierend auf der EfiBoot-Version deaktiviert x86_64-Unterstützung auf einem nicht unterstützten Modell, wenn eine i386-Variante unterstützt wird. Auto wird hier nicht herangezogen, da die Liste in EfiBoot nicht überschreibbar ist.

(f) KernelArch schränkt die Unterstützung auf die explizit angegebene Architektur ein (wenn nicht auf Auto gesetzt), wenn die Architektur in den Fähigkeiten vorhanden bleibt.

(g) Die am besten unterstützte Architektur wird in dieser Reihenfolge ausgewählt: x86_64, i386, i386-user32.

Im Gegensatz zu macOS 10.7 (wo bestimmte Board-Identifikatoren nur als i386-Maschinen behandelt werden) und macOS 10.5 oder früher (wo x86_64 vom macOS-Kernel nicht unterstützt wird), ist macOS 10.6 sehr speziell. Die Wahl der Architektur unter macOS 10.6 hängt von vielen Faktoren ab, darunter nicht nur die Board-Kennung, sondern auch der macOS-Produkttyp (Client vs. Server), die macOS-Punktversion und die Menge an RAM. Die Erkennung all dieser Faktoren ist kompliziert und unpraktisch, da einige Versionen Implementierungsfehler aufwiesen, die dazu führten, dass die Servererkennung nicht richtig ausgeführt werden konnte. Aus diesem Grund greift OpenCore unter macOS 10.6 auf die x86_64-Architektur zurück, wann immer diese von der Karte unterstützt wird, wie es bei macOS 10.7 der Fall ist.

Eine Kompatibilitätsmatrix für 64-Bit-Mac-Modelle, die dem tatsächlichen Verhalten von EfiBoot unter macOS 10.6.8 und 10.7.5 entspricht, ist unten dargestellt.

Model	10.6 (minimal)	10.6 (client)	10.6 (server)	10.7 (any)
Macmini	4,x (Mid 2010)	5,x (Mid 2011)	4,x (Mid 2010)	3,x (Early 2009)
MacBook	Unsupported	Unsupported	Unsupported	5,x (2009/09)
MacBookAir	Unsupported	Unsupported	Unsupported	2,x (Late 2008)
MacBookPro	4,x (Early 2008)	8,x (Early 2011)	8,x (Early 2011)	3,x (Mid 2007)
iMac	8,x (Early 2008)	12,x (Mid 2011)	12,x (Mid 2011)	7,x (Mid 2007)
MacPro	3,x (Early 2008)	5,x (Mid 2010)	3,x (Early 2008)	3,x (Early 2008)
Xserve	2,x (Early 2008)	2,x (Early 2008)	2,x (Early 2008)	2,x (Early 2008)

Hinweis: 3+2 und 6+4 Hotkeys zur Auswahl der bevorzugten Architektur werden nicht unterstützt, da sie von EfiBoot gehandhabt werden und daher schwer zu erkennen sind.

4. KernelCache

Typ: plist string

Failsafe: Auto

Beschreibung: Bevorzugt den angegebenen Kernel-Cache-Typ (Auto, Cacheless, Mkext, Prelinked), wenn verfügbar.

Verschiedene Varianten von macOS unterstützen verschiedene Kernel-Caching-Varianten, um die Boot-Leistung zu verbessern. Diese Einstellung verhindert die Verwendung von schnelleren Kernel-Caching-Varianten, wenn langsamere Varianten aus Gründen der Fehlersuche und Stabilität verfügbar sind. Das heißt, dass durch die Angabe von Mkext, Prelinked für z.B. 10.6 deaktiviert wird, aber nicht für 10.7.

Die Liste der verfügbaren Kernel-Caching-Typen und ihre aktuelle Unterstützung in OpenCore ist unten aufgeführt.

macOS	i386 NC	i386 MK	i386 PK	x86_64 NC	x86_64 MK	x86_64 PK	x86_64 KC
10.4	YES	YES (V1)	NO (V1)	—	—	—	—
10.5	YES	YES (V1)	NO (V1)	—	—	—	—
10.6	YES	YES (V2)	YES (V2)	YES	YES (V2)	YES (V2)	—
10.7	YES	—	YES (V3)	YES	—	YES (V3)	—
10.8-10.9	—	—	—	YES	—	YES (V3)	—
10.10-10.15	—	—	—	—	—	YES (V3)	—
11+	—	—	—	—	—	YES (V3)	YES

Hinweis: Die erste Version (V1) des 32-Bit-Prelinkedkernels wird aufgrund der Beschädigung von Kext-Symboltabellen durch die Tools nicht unterstützt. Bei dieser Version wird die Einstellung Auto das Booten des Prelinkedkernels blockieren. Dies führt auch dazu, dass das Boot-Argument keepsyms=1 für kext-Frames auf diesen Systemen nicht funktioniert.

S. 37

8 Misc

8.1 Einleitung

Dieser Abschnitt enthält verschiedene Konfigurationsoptionen, die das Ladeverhalten des OpenCore-Betriebssystems beeinflussen, sowie weitere Optionen, die nicht ohne weiteres in andere Abschnitte passen.

OpenCore folgt im Großen und Ganzen dem "bless"-Modell, das auch als "Apple Boot Policy" bekannt ist. Der Hauptzweck des "bless"-Modells besteht darin, die Einbettung von Boot-Optionen in das Dateisystem zu ermöglichen (und über einen spezialisierten Treiber zugänglich zu sein) sowie eine breitere Palette von vordefinierten Boot-Pfaden zu unterstützen als die Liste der Wechselmedien, die in der UEFI-Spezifikation.

Partitionen können von OpenCore nur gebootet werden, wenn sie die Anforderungen einer vordefinierten Scan-Richtlinie erfüllen. Diese Richtlinie legt fest, welche spezifischen Dateisysteme eine Partition haben muss und auf welchen spezifischen Gerätetypen sich

eine Partition befinden muss, damit sie von OpenCore als Boot-Option zur Verfügung gestellt wird. Einzelheiten finden Sie in der Eigenschaft ScanPolicy.

Der Scan-Vorgang beginnt mit der Aufzählung aller verfügbaren Partitionen, gefiltert auf der Grundlage der Scan-Richtlinie. Jede Partition kann mehrere primäre und alternative Optionen erzeugen. Primäre Optionen stehen für die auf dem Datenträger installierten Betriebssysteme, während alternative Optionen Wiederherstellungsoptionen für die Betriebssysteme auf dem Datenträger darstellen.

- Alternative Optionen können auch ohne primäre Optionen existieren und umgekehrt.

- Optionen müssen nicht unbedingt Betriebssysteme auf derselben Partition darstellen. - Jede primäre und alternative Option kann eine Hilfsoption sein oder nicht.

- Einzelheiten hierzu finden Sie im Abschnitt HideAuxiliary. Der Algorithmus zur Bestimmung der Boot-Optionen verhält sich wie folgt:

1. Abrufen aller verfügbaren Partitions-Handles, gefiltert auf der Grundlage der Scan-Richtlinie (und der Treiberverfügbarkeit).

2. Abrufen aller verfügbaren Boot-Optionen aus der UEFI-Variablen BootOrder.

3. Für jede gefundene Boot-Option:

- Ermitteln Sie den Gerätepfad der Bootoption.

- Führen Sie Korrekturen (z. B. Korrektur des NVMe-Subtyps) und Erweiterungen (z. B. für Boot Camp) des Gerätepfads durch.

- Bei einem Fehler, wenn es sich um einen OpenCore-Gerätepfad mit benutzerdefiniertem Eintrag handelt, wird der entsprechende benutzerdefinierte Eintrag vorkonstruiert und erfolgreich.

- Ermitteln Sie das Gerätehandle, indem Sie den Gerätepfad des resultierenden Gerätepfads suchen (bei Fehlschlag ignorieren).

- Suchen Sie das Gerätehandle in der Liste der Partitionshandles (ignorieren Sie es, wenn es fehlt).

- Für Festplatten-Gerätepfade (ohne Angabe eines Bootloaders), führen Sie "bless" aus (kann > 1 Eintrag zurückgeben).

- Bei Dateigerätepfaden prüfen Sie direkt auf das Vorhandensein im Dateisystem.

- Auf der OpenCore-Bootpartition schließen Sie alle OpenCore-Bootstrap-Dateien durch Datei-Header-Prüfungen aus.

- Markieren Sie das Gerätehandle als verwendet in der Liste der Partitionshandles, falls vorhanden.

- Registrieren Sie die resultierenden Einträge als primäre Optionen und bestimmen Sie deren Typen.

Die Option wird für einige Typen (z.B. Apple HFS Recovery) zu einer Hilfsoption.

4. Für jedes Partitionshandle:

- Wenn das Partitions-Handle als unbenutzt markiert ist, führen Sie die Abfrage der Liste der primären Optionen "bless" aus. Falls eine BlessOverride-Liste gesetzt ist, werden sowohl Standard- als auch benutzerdefinierte "bless"-Pfade gefunden.

- Schließen Sie auf der OpenCore-Bootpartition OpenCore-Bootstrap-Dateien mithilfe von Header-Checks aus.

- Registrieren Sie die resultierenden Einträge als primäre Optionen und bestimmen Sie deren Typen, falls gefunden.

Für einige Typen (z.B. Apple HFS Recovery) wird die Option zu einer Hilfsoption.

- Wenn eine Partition bereits über primäre Optionen des Typs "Apple Recovery" verfügt, fahren Sie mit dem nächsten Handle fort.

- Suche nach alternativen Einträgen durch "bless"-Wiederherstellungsoptionslistenabruf und vordefinierte Pfade.

- Registrieren Sie die resultierenden Einträge als alternative Hilfsoptionen und bestimmen Sie deren Typen, falls gefunden.

5. Benutzerdefinierte Einträge und Werkzeuge, mit Ausnahme solcher, die bereits zuvor erstellt wurden, werden als primäre Optionen ohne jegliche Prüfung in Bezug auf Hilfsoptionen hinzugefügt.

6. Systemeinträge, wie z. B. Reset NVRAM, werden als primäre Hilfsoptionen hinzugefügt.

Die Anzeigereihenfolge der Boot-Optionen in der OpenCore-Auswahl und der Boot-Prozess werden unabhängig vom Scan-Algorithmus bestimmt.

Die Anzeigereihenfolge ist wie folgt:

- Die alternativen Optionen folgen den entsprechenden primären Optionen. Das heißt, die Apple-Wiederherstellungsoptionen folgen, wann immer möglich, der entsprechenden macOS-Option.

- Die Optionen werden in der Reihenfolge der Handle-Firmware des Dateisystems aufgelistet, damit die Reihenfolge bei jedem Neustart gleich bleibt, unabhängig davon, welches Betriebssystem zum Laden gewählt wurde.

- Benutzerdefinierte Einträge, Werkzeuge und Systemeinträge werden nach allen anderen Optionen hinzugefügt.

- Zusatzoptionen werden nur angezeigt, wenn Sie den "Erweiterten Modus" in der OpenCore-Auswahl aufrufen (normalerweise durch Drücken der Leertaste). Der Bootvorgang läuft wie folgt ab:

- Suche nach der ersten gültigen primären Option in der BootNext UEFI-Variable.

- Im Fehlerfall suchen Sie die erste gültige primäre Option in der UEFI-Variablen BootOrder. - Markieren Sie die Option als Standardoption zum Booten.

- Booten Sie die Option über den Picker oder ohne ihn, abhängig von der Option ShowPicker. - Zeige den Picker bei Fehlschlag an.

Hinweis 1: Dieser Prozess funktioniert nur dann zuverlässig, wenn die Option RequestBootVarRouting aktiviert ist oder die Firmware keine UEFI-Boot-Optionen steuert (OpenDuetPkg oder benutzerdefiniertes BDS). Wenn LauncherOption nicht aktiviert ist, können andere Betriebssysteme die OpenCore-Einstellungen überschreiben. Daher sollte diese Eigenschaft aktiviert werden, wenn Sie planen, andere Betriebssysteme zu verwenden.

Hinweis 2: Die Boot-Argumente der variablen UEFI-Boot-Optionen werden entfernt, wenn sie vorhanden sind, da sie Argumente enthalten können, die das Betriebssystem kompromittieren können, was unerwünscht ist, wenn sicheres Booten aktiviert ist.

Hinweis 3: Einige Betriebssysteme, wie z.B. Windows, erstellen eine Boot-Option und markieren diese als oberste Option beim ersten Start oder nach einem NVRAM-Reset innerhalb von OpenCore. In diesem Fall bleibt der Standard-Boot-Eintrag bis zur nächsten manuellen Neukonfiguration geändert.

8.2 Properties

1. Boot

Typ: plist dict

Beschreibung: Anwenden der Boot-Konfiguration, die im Abschnitt Booteigenschaften unten beschrieben wird.

2. BlessOverride

Typ: plist array

Failsafe: Empty

Beschreibung: Hinzufügen von benutzerdefinierten Scan-Pfaden durch das Bless-Modell.

Wird mit plist-String-Einträgen gefüllt, die absolute UEFI-Pfade zu angepassten Bootloadern enthalten, wie z.B. \EFI\debian\grubx64.efi für den Debian-Bootloader. Dadurch können nicht standardisierte Bootpfade automatisch von der OpenCore-Auswahl erkannt werden. Vom Design her sind sie äquivalent zu vordefinierten blessed paths, wie \System\Library\CoreServices\boot.efi oder \EFI\Microsoft\Boot\bootmgfw.efi, aber im Gegensatz zu vordefinierten blessed paths haben sie die höchste Priorität.

3. Debuggen

Typ: plist dict

Beschreibung: Wendet die Debug-Konfiguration an, die im Abschnitt Debug-Eigenschaften weiter unten beschrieben wird.

4. Entries

Typ: plist array

Failsafe: Empty

Beschreibung: Fügt dem OpenCore Picker Boot-Einträge hinzu.

Soll mit plist dict-Werten gefüllt werden, die jeden Ladeeintrag beschreiben. Siehe den Abschnitt Entry Properties section unten für Details.

5. Security

Typ: plist dict

Beschreibung: Wendet die Sicherheitskonfiguration an, die im Abschnitt Sicherheitseigenschaften unten beschrieben ist.

6. Serial

Typ: plist dict

Beschreibung: Führt die Initialisierung der seriellen Schnittstelle durch und konfiguriert die PCD-Werte, die von BaseSerialPortLib16550 benötigt werden, damit die seriellen Schnittstellen ordnungsgemäß funktionieren. Die Werte werden in den Abschnitten Serielle Eigenschaften und Benutzerdefinierte serielle Eigenschaften unten aufgeführt und beschrieben.

S. 39

Durch die Aktivierung von Init stellt dieser Abschnitt sicher, dass die serielle Schnittstelle initialisiert wird, wenn dies nicht durch die Firmware geschieht. Damit OpenCore Protokolle über die serielle Schnittstelle ausgeben kann, muss Bit 3 (d.h. serielles Logging) für Target im Abschnitt Misc->Debug gesetzt werden.

Beim Debuggen mit seriellen Schnittstellen erkennt BaseSerialPortLib16550 standardmäßig nur die vom Motherboard bereitgestellten internen Schnittstellen. Wenn die Option Override aktiviert ist, setzt dieser Abschnitt die in BaseSerialPortLib16550.inf aufgeführten PCD-Werte außer Kraft, so dass auch externe serielle Schnittstellen (z. B. von einer PCI-Karte) ordnungsgemäß funktionieren. Insbesondere bei der Fehlersuche unter macOS ist es zusätzlich zum Überschreiben dieser PCD-Werte notwendig, die CustomPciSerialDevice-Kernel-Eigenart zu aktivieren, damit die XNU solche externen seriellen Schnittstellen verwenden kann.

Siehe MdeModulePkg.dec für die Erklärungen zu den einzelnen Schlüsseln.

7. Tools

Typ: plist-Array

Failsafe: Empty

Beschreibung: Hinzufügen von tool entries in die OpenCore-picker.

Zu füllen mit plist dict-Werten, die jeden load entry beschreiben. Einzelheiten finden Sie im Abschnitt Entry Properties section weiter unten.

Hinweis: Bestimmte UEFI-Tools, wie z. B. UEFI Shell, können sehr gefährlich sein und dürfen NICHT in Produktionskonfigurationen erscheinen, insbesondere nicht in Vaulted-Konfigurationen und solchen, die durch Secure Boot geschützt sind, da solche Tools zur Umgehung der Secure Boot-Kette verwendet werden können. Beispiele für UEFI-Tools finden Sie im Abschnitt UEFI.

8.3 Boot-Properties

1. KonsoleAttribute

Typ: plist integer

Failsafe: 0

Beschreibung: Legt spezifische Attribute für die Konsole fest.

Der Text-Renderer unterstützt Farbargumente als Summe von Vorder- und Hintergrundfarben basierend auf der UEFI-Spezifikation. Der Wert für schwarzen Hintergrund und für schwarzen Vordergrund, 0, ist reserviert.

Liste der Farbwerte und -namen:

• 0x00 — EFI_BLACK
• 0x01 — EFI_BLUE
• 0x02 — EFI_GREEN
• 0x03 — EFI_CYAN
• 0x04 — EFI_RED
• 0x05 — EFI_MAGENTA
• 0x06 — EFI_BROWN
• 0x07 — EFI_LIGHTGRAY
• 0x08 — EFI_DARKGRAY
• 0x09 — EFI_LIGHTBLUE
• 0x0A — EFI_LIGHTGREEN • 0x0B — EFI_LIGHTCYAN
• 0x0C — EFI_LIGHTRED
• 0x0D — EFI_LIGHTMAGENTA • 0x0E — EFI_YELLOW

• 0x0F — EFI_WHITE
• 0x00 — EFI_BACKGROUND_BLACK
• 0x10 — EFI_BACKGROUND_BLUE

• 0x20 — EFI_BACKGROUND_GREEN
• 0x30 — EFI_BACKGROUND_CYAN
• 0x40 — EFI_BACKGROUND_RED
• 0x50 — EFI_BACKGROUND_MAGENTA • 0x60 — EFI_BACKGROUND_BROWN
• 0x70 — EFI_BACKGROUND_LIGHTGRAY

Hinweis: Diese Option funktioniert möglicherweise nicht gut mit dem Systemtext-Renderer. Die Einstellung eines anderen Hintergrunds als Schwarz könnte beim Testen der GOP-Funktionalität hilfreich sein.

2. HibernateMode

Typ: plist string

Failsafe: None

Beschreibung: Modus für die Erkennung des Ruhezustands. Die folgenden Modi werden unterstützt:

- Keine - Ruhezustand ignorieren.

- Auto - RTC- und NVRAM-Erkennung verwenden. - RTC - RTC-Erkennung verwenden.

- NVRAM - NVRAM-Erkennung verwenden.

Hinweis: Wenn die Firmware den Ruhezustand selbst handhaben kann (gilt für Mac EFI-Firmware), dann sollte None angegeben werden, um den Ruhezustand unverändert an OpenCore zu übergeben.

3. HideAuxiliary

Typ: plist boolean

Failsafe: false

Beschreibung: Wird auf true gesetzt, um Hilfseinträge aus dem Auswahlmenü auszublenden.

Ein Eintrag wird als Hilfseintrag betrachtet, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Der Eintrag ist macOS Recovery.

- Der Eintrag ist macOS Time Machine.

- Der Eintrag ist explizit als Hilfseintrag markiert. - Eintrag ist System (z. B. NVRAM zurücksetzen).

Um alle Einträge anzuzeigen, kann das Auswahlmenü durch Drücken der Leertaste in den "Erweiterten Modus" umgeschaltet werden. Das Ausblenden von Hilfseinträgen kann die Boot-Performance auf Systemen mit mehreren Festplatten erhöhen.

4. LauncherOption

Typ: plist string

Failsafe: Disabled

Beschreibung: Registriert die Launcher-Option in den Firmware-Einstellungen für die Persistenz.

Gültige Werte:

- Deaktiviert - nichts tun.

- Full - erstellt oder aktualisiert die Boot-Option mit der höchsten Priorität im UEFI-Variablenspeicher beim Start des Bootloaders.

- Damit diese Option funktioniert, muss RequestBootVarRouting aktiviert sein.

- Short - erstellt eine kurze Boot-Option anstelle einer vollständigen Option.

- Diese Variante ist nützlich für einige ältere Firmware-Typen, typischerweise von Insyde, die nicht in der Lage sind, vollständige Gerätepfade zu verwalten.

- System - erstellt keine Bootoption, sondern nimmt an, dass die angegebene benutzerdefinierte Option blessed ist.

- Diese Variante ist nützlich, wenn man sich auf die ForceBooterSignature-Eigenart verlässt und die Pfadverwaltung des OpenCore-Startprogramms die Verwaltung des Pfades durch Bless-Utilities ohne Beteiligung von OpenCore erfolgt.

Diese Option ermöglicht die Integration mit Betriebssysteminstallationen und Upgrades von Drittanbietern (die die Datei \EFI\BOOT\BOOTx64.efi überschreiben können). Die Datei BOOTx64.efi wird nicht mehr für das Bootstrapping von OpenCore verwendet, wenn eine benutzerdefinierte Option erstellt wird. Der für das Bootstrapping verwendete benutzerdefinierte Pfad kann mit der Option LauncherPath angegeben werden.

Hinweis 1: Einige Firmware-Typen können Fehler in der NVRAM-Implementierung, keine Unterstützung für Bootoptionen oder andere Inkompatibilitäten aufweisen. Auch wenn es unwahrscheinlich ist, kann die Verwendung dieser Option zu Bootfehlern führen und sollte nur auf bekanntermaßen kompatiblen Boards verwendet werden. (Siehe acidanthera/bugtracker#1222 für einige bekannte Probleme, die Haswell und andere Boards betreffen).

Hinweis 2: Während NVRAM-Resets, die von OpenCore aus ausgeführt werden, die in Bootstrap erstellte Bootoption normalerweise nicht löschen, wird die Bootoption durch die

Ausführung von NVRAM-Resets vor dem Laden von OpenCore gelöscht. Daher sollte ein NVRAM-Reset bei signifikanten Implementierungs-Updates, wie es bei OpenCore 0.6.4 der Fall war, bei deaktiviertem Bootstrap ausgeführt werden, um ihn anschließend wieder zu aktivieren.

S. 41

Hinweis 3: Einige Versionen von Intel Visual BIOS (z.B. auf Intel NUC) haben einen unglücklichen Fehler, der dazu führt, dass, wenn eine Boot-Option hinzugefügt wird, die sich auf einen Pfad auf einem USB-Laufwerk bezieht, dies die einzige Boot-Option ist, die angezeigt wird, wenn ein USB-Laufwerk angeschlossen wird. Wenn OpenCore von einem USB-Laufwerk mit dieser Firmware gestartet wird und die LauncherOption auf Full oder Short gesetzt ist, gilt dies und nur der OpenCore-Boot-Eintrag wird danach angezeigt, wenn ein anderes USB-Laufwerk eingesteckt wird (dieses höchst untypische BIOS-Verhalten betrifft auch andere Software). Der beste Weg, dies zu vermeiden, ist, LauncherOption auf Disabled oder System auf jeder Version von OpenCore zu belassen, die von einem USB-Laufwerk auf dieser Firmware gestartet wird. Wenn das Problem bereits aufgetreten ist, ist die schnellste und zuverlässigste Lösung:

- Aktivieren Sie die UEFI-Shell des Systems im Intel Visual BIOS

- Stecken Sie bei ausgeschaltetem Gerät einen OpenCore-USB ein.

- Schalten Sie ein und wählen Sie die System-UEFI-Shell

- Da die System-Shell kein bcfg enthält, verwenden Sie die System-Shell, um die OpenCore-OpenShell zu starten (z.B. durch den Befehl FS2:\EFI\OC\Tools\OpenShell.efi eingeben, aber Sie müssen jedoch herausfinden, welches Laufwerk, das richtige für OpenCore ist und die Laufwerksnummer FS#: entsprechend ändern)

- Verwenden Sie in der OpenShell bcfg boot dump, um die NVRAM-Bootoptionen anzuzeigen, und verwenden Sie dann bcfg boot rm # (wobei # die Nummer des OpenCore-Boot-Eintrags ist), um den OpenCore-Eintrag zu entfernen. Es ist alternativ möglich, die OpenShell direkt aus dem OpenCore-Bootmenü zu starten, wenn Sie einen funktionierenden OpenCore für das System konfiguriert haben. In diesem Fall und wenn OpenCore RequestBootVarRouting aktiviert hat, ist es notwendig, den Befehl \EFI\OC\Tools\OpenControl.efi disable auszuführen, bevor Sie bcfg verwenden. (Nach der Deaktivierung von OpenControl ist es notwendig, entweder einen Neustart durchzuführen oder OpenControl wiederherzustellen, bevor ein Betriebssystem gebootet wird). Es ist auch möglich, efibootmgr innerhalb von Linux zu verwenden, um den fehlerhaften Eintrag zu entfernen, wenn Sie eine funktionierende Version von Linux auf dem Rechner haben. Linux muss entweder nicht über OpenCore oder über OpenCore mit deaktiviertem RequestBootVarRouting gestartet werden, damit dies funktioniert.

5. LauncherPath

Typ: plist string

Failsafe: Default

Beschreibung: Startpfad für die LauncherOption-property.

Der Standardwert verweist auf OpenCore.efi. Benutzerdefinierte Pfade, z.B. \EFI\ SomeLauncher.efi, können verwendet werden, um benutzerdefinierte Lader bereitzustellen, die OpenCore.efi selbst laden sollen.

6. PickerAttribute

Typ: plist integer

Failsafe: 0

Beschreibung: Setzt spezifische Attribute für den OpenCore-Picker.

Verschiedene OpenCore-Picker können über die Attributmaske konfiguriert werden, die OpenCore-reservierte (BIT0~BIT15) und OEM-spezifische (BIT16~BIT31) Werte enthält.

Die aktuellen OpenCore-Werte umfassen:

- 0x0001 - OC_ATTR_USE_VOLUME_ICON, stellt benutzerdefinierte Icons für Booteinträge bereit:

OpenCore versucht, ein Volume-Icon zu laden, indem es wie folgt sucht, und greift bei einem Fehlschlag auf das Standard-Icon zurück:

- .VolumeIcon.icnsDateiimPrebootvolumeinper-volumedirectory(/System/Volumes/ Preboot/{GUID}/, wenn am Standardspeicherort in macOS gemountet) für APFS (falls vorhanden).

- .VolumeIcon.icns-Datei im Stammverzeichnis des Preboot-Volumes (/System/Volumes/Preboot/, wenn am Standardspeicherort innerhalb von macOS gemountet) für APFS (ansonsten).

- .VolumeIcon.icns-Datei im Volume-Root für andere Dateisysteme.

Hinweis 1: Die Apple-Auswahl unterstützt teilweise das Ablegen einer Volume-Icon-Datei im Volume-Stammverzeichnis des Betriebssystems, /System/Volumes/Data/, wenn es am Standardort innerhalb von macOS gemountet ist. Dieser Ansatz ist fehlerhaft: Die Datei ist weder für OpenCanopy noch für die Apple-Auswahl zugänglich, wenn FileVault 2, das als Standardeinstellung vorgesehen ist, aktiviert ist. Daher versucht OpenCanopy nicht, Apples Ansatz zu unterstützen. Eine Volume-Symboldatei kann im Stammverzeichnis des Preboot-Volumes platziert werden, um sowohl mit OpenCanopy als auch mit der Apple-Auswahl kompatibel zu sein, oder Sie verwenden den Preboot-Speicherort pro Volume wie oben beschrieben mit OpenCanopy als bevorzugte Alternative zu Apples Ansatz.

Hinweis 2: Beachten Sie, dass die Verwendung eines Volume-Symbols auf einem beliebigen Laufwerk das normale Verhalten der OpenCore-Auswahl für dieses Laufwerk außer Kraft setzt, indem das entsprechende Symbol ausgewählt wird, je nachdem, ob es sich um ein internes oder externes Laufwerk handelt.

- 0x0002 - OC_ATTR_USE_DISK_LABEL_FILE, verwendet benutzerdefinierte vorberechnete Titel für Booteinträge aus der Datei .disk_label (.disk_label_2x) neben dem Bootloader für alle Dateisysteme. Diese Bezeichnungen können mit dem Dienstprogramm disklabel oder dem Befehl bless --folder {FOLDER_PATH} --label {LABEL_TEXT} erzeugt werden. Wenn vor gerenderte Labels deaktiviert sind oder fehlen, verwenden Sie den Labeltext in der Datei .contentDetails (oder .disk_label.contentDetails) neben dem Bootloader, falls vorhanden, ansonsten wird der Eintragsname selbst gerendert.

- 0x0004 - OC_ATTR_USE_GENERIC_LABEL_IMAGE, bietet vordefinierte Beschriftungsbilder für Booteinträge ohne benutzerdefinierte Einträge. Dies kann jedoch weniger Details für den eigentlichen Booteintrag liefern.

- 0x0008 - OC_ATTR_HIDE_THEMED_ICONS, bevorzugt eingebaute Icons für bestimmte Icon-Kategorien, um sie an den Thema-Stil anzupassen. Dies könnte zum Beispiel die Anzeige des eingebauten Time Machine-Symbols erzwingen. Erfordert OC_ATTR_USE_VOLUME_ICON.

- 0x0010 - OC_ATTR_USE_POINTER_CONTROL, aktiviert die Zeigersteuerung in der OpenCore-Auswahl, wenn verfügbar. Dies könnte zum Beispiel die Verwendung der Maus oder des Trackpads zur Steuerung von UI-Elementen ermöglichen.

- 0x0020 - OC_ATTR_SHOW_DEBUG_DISPLAY, aktiviert die Anzeige zusätzlicher Timing- und Debug-Informationen, im Eingebauter Picker nur in DEBUG- und NOOPT-Builds.

- 0x0040 - OC_ATTR_USE_MINIMAL_UI, minimale UI-Anzeige verwenden, keine Shutdown- oder Restart-Schaltflächen, betrifft OpenCanopy und den eingebauten Picker.

- 0x0080 - OC_ATTR_USE_FLAVOUR_ICON, bietet flexiblen Booteinstieg Inhalt, geeignet für die Auswahl die besten Medien aus verschiedenen Inhaltssätzen auszuwählen: Wenn diese Option aktiviert ist, werden das Eingangssymbol in OpenCanopy und der Audio-Assist-Eingangston in OpenCanopy und dem eingebauten Boot-Picker nach dem sogenannten Content Flavour ausgewählt. Zur Bestimmung des Content Flavour wird der folgende Algorithmus verwendet:

- Für ein Tool wird der Wert aus dem Feld Flavour gelesen.

- Für einen automatisch entdeckten Eintrag, einschließlich der Protokolleinträge für den Booteintrag, wie z.B. diejenigen, die vom OpenLinuxBoot-Treiber erzeugt werden, wird der Wert aus der Datei .contentFlavour neben dem Bootloader gelesen, falls vorhanden ist.

- Für einen benutzerdefinierten Eintrag, der im Abschnitt Einträge angegeben ist, wird er aus der .contentFlavour-Datei neben dem Bootloader gelesen, wenn Flavour Auto ist, andernfalls wird er über den Flavour-Wert selbst angegeben.

- Wenn der gelesene Flavour Auto ist oder es keinen .contentFlavour gibt, wird der Flavour des Eintrags basierend auf dem Eintragstyp ausgewählt

(z. B. erhält Windows automatisch den Flavour Windows).

Der Flavour-Wert ist eine Folge von durch : getrennten Namen, die auf 64 druckbare 7-Bit-ASCII-Zeichen begrenzt ist. Dies ist so ausgelegt, dass bis zu fünf Namen unterstützt werden. Jeder Name bezieht sich auf eine Geschmacksrichtung, wobei der erste Name die höchste Priorität und der letzte Name die niedrigste Priorität hat. Eine solche Struktur ermöglicht es, einen Eintrag spezifischer zu beschreiben, wobei die Icons je nach Unterstützung durch das audiovisuelle Paket flexibel ausgewählt werden können. Eine fehlende Audio- oder Symboldatei bedeutet, dass die nächste Geschmacksrichtung versucht werden sollte, und wenn alle fehlen, erfolgt die Auswahl auf der Grundlage der Art des Eintrags. Beispiel für Flavour-Werte: BigSur:Apple, Windows10:Windows. OpenShell:UEFIShell:Shell.

Die Verwendung von Flavours bedeutet, dass Sie einfach zwischen Iconsets wechseln können, wobei das Flavour die besten verfügbaren Icons aus jedem Set auswählt. Wenn Sie z. B. den Icon-Flavour Debian:Linux angeben, wird das Icon Debian.icns verwendet, falls vorhanden, dann wird Linux.icns ausprobiert, und dann wird auf den Standard für ein Betriebssystem zurückgegriffen, nämlich HardDrive.icns.

Dinge, die zu beachten sind:

- Aus Sicherheitsgründen werden sowohl Ext<Flavour>.icnsand<Flavour>.icnsareboth unterstützt, und nur ext<Flavour>.icns wird verwendet, wenn sich der Eintrag auf einem externen Laufwerk befindet (gefolgt vom Standardfallback ExtHardDrive.icns).

- Wenn beides zutrifft, hat .VolumeIcon.icns Vorrang vor .contentFlavour.

- Damit Icons und Audio Assist für Tools (z.B. für die UEFI-Shell) korrekt funktionieren, müssen System

Standard-Boteintragssymbole (siehe Docs/Flavours.md), die in der Flavour-Einstellung für Tools oder Entries angegeben sind, auch bei deaktiviertem Flavour weiter verwendet. Nicht-System-Symbole werden in diesem Fall ignoriert. Darüber hinaus werden die Flavours UEFIShell und NVRAMReset speziell verarbeitet, wobei die jeweiligen Tools mit der korrekten Audiounterstützung, den standardmäßig eingebauten Bezeichnungen usw. versehen werden.

- Eine Liste der empfohlenen Flavours ist in Docs/Flavours.md enthalten.

7. PickerAudioAssist

Typ: plist boolean

Failsafe: false

Beschreibung: Aktiviert den Bildschirmleser standardmäßig im OpenCore-Picker.

Für den macOS-Bootloader wird die Bildschirmleser-Einstellung im preferences.efires-Archiv in der Datei isVOEnabled.int32 festgelegt und vom Betriebssystem gesteuert. Für die OpenCore-Bildschirmleserunterstützung ist diese Option ein unabhängiges

Äquivalent. Das Umschalten der Bildschirmleseunterstützung sowohl im OpenCore-Picker als auch im Anmeldefenster des macOS-Bootloaders FileVault 2 kann auch mit der Tastenkombination Befehl + F5 erfolgen.

Hinweis: Der Bildschirmleser erfordert eine funktionierende Audiounterstützung. Weitere Informationen finden Sie im Abschnitt UEFI-Audioeigenschaften.

8. PollAppleHotKeys

Typ: plist boolean

Failsafe: false

Beschreibung: Aktiviert die Behandlung von Modifier-Hotkeys in der OpenCore-Auswahl.

Zusätzlich zu den Aktions-Hotkeys, die teilweise im Abschnitt PickerMode beschrieben sind und typischerweise von Apple BDS verwaltet werden, gibt es auch Modifier Keys, die vom Bootloader des Betriebssystems (boot.efi) verwaltet werden. Diese Tasten ermöglichen es, das Verhalten des Betriebssystems zu ändern, indem verschiedene Bootmodi angeboten werden.

Bei bestimmter Firmware kann die Verwendung von Modifier Keys aufgrund von Treiberinkompatibilitäten problematisch sein. Um dieses Problem zu umgehen, ermöglicht diese Option die Registrierung bestimmter Hotkeys auf eine freizügigere Weise innerhalb der OpenCore-Auswahl. Zu diesen Erweiterungen gehören die Unterstützung von Tastenkombinationen vor der Auswahl des Boot-Elements und die zuverlässige Erkennung der Umschalttaste bei der Auswahl des Boot-Elements, um die Tatsache zu umgehen, dass Hotkeys, die während des Bootens ständig gehalten werden, auf vielen PS/2-Tastaturen nicht zuverlässig erkannt werden.

Die Liste der bekannten Modifier-Hotkeys umfasst:

- CMD+C+MINUS - Deaktivieren der Board-Kompatibilitätsprüfung.

- CMD+K - bootet den Release-Kernel, ähnlich wie bei kcsuffix=release.

- CMD+S - Einzelbenutzermodus.

- CMD+S+MINUS - KASLR-Dia deaktivieren, erfordert deaktiviertes SIP.

- CMD+V - ausführlicher Modus.

- Shift+Enter, Shift+Index - sicherer Modus, kann in Kombination mit CTRL+Enter, CTRL+Index verwendet werden.

9. ZeigePicker

Typ: plist boolean

Failsafe: false

Beschreibung: Zeigt einen einfachen Picker an, um die Auswahl des Booteintrags zu ermöglichen.

10. StartVerzögerung

Typ: plist integer, 32 bit

Failsafe: 0

Beschreibung: Verzögerung in Mikrosekunden, die ausgeführt wird, bevor die Start- und Aktions-Hotkeys des OpenCore-Pickers bedient werden.

Die Einführung einer Verzögerung kann zusätzliche Zeit geben, um die richtige Aktions-Hotkey-Sequenz zu halten, um z.B. in den Wiederherstellungsmodus zu booten. Auf einigen Plattformen kann es aufgrund der Beschaffenheit des Tastaturtreibers erforderlich sein, diese Option auf ein Minimum von 5000-10000 Mikrosekunden zu setzen, um auf Aktions-Hotkeys zuzugreifen.

11. Timeout

Type: plist integer, 32 bit

Failsafe: 0

Beschreibung: Zeitüberschreitung in Sekunden in der OpenCore-Auswahl vor dem automatischen Booten des Standard-Boot-Eintrags. Zum Deaktivieren auf 0 setzen.

12. PickerMode

Typ: plist string

Failsafe: Builtin

Beschreibung: Auswahl des für die Boot-Verwaltung verwendeten Picker.

PickerMode beschreibt die zugrunde liegende Boot-Verwaltung mit einer optionalen Benutzeroberfläche, die für die Handhabung von Boot-Optionen zuständig ist.

Die folgenden Werte werden unterstützt:

- Builtin - die Boot-Verwaltung wird von OpenCore gehandhabt, es wird eine einfache textbasierte Benutzeroberfläche verwendet.

- Extern - ein externes Boot-Management-Protokoll wird verwendet, falls verfügbar. Andernfalls wird der Modus Builtin

verwendet.

- Apple - die Apple-Boot-Verwaltung wird verwendet, wenn sie verfügbar ist. Andernfalls wird der Modus Eingebaut verwendet.

Bei Erfolg kann der externe Modus die gesamte Boot-Verwaltung in OpenCore mit Ausnahme der Richtliniendurchsetzung vollständig deaktivieren. Im Apple-Modus kann zusätzlich die Richtliniendurchsetzung umgangen werden. Ein Beispiel für eine angepasste Benutzeroberfläche finden Sie im OpenCanopy-Plugin.

Der in OpenCore integrierte Picker enthält eine Reihe von Aktionen, die während des Bootvorgangs ausgewählt werden. Die Liste der unterstützten Aktionen ähnelt der von Apple BDS und kann in der Regel durch Halten von Aktions-Hotkeys während des Bootvorgangs aufgerufen werden.

Die folgenden Aktionen werden derzeit berücksichtigt:

- Standard - dies ist die Standardoption und lässt den eingebauten OpenCore-Picker die Standard-Boot-Option laden, die im Einstellungsfenster Startdiskette angegeben ist.

- ZeigeAuswahl - diese Option erzwingt die Anzeige der OpenCore-Auswahl. Dies kann in der Regel erreicht werden, indem man die OPT-Taste während des Bootens gedrückt hält. Wenn Sie ShowPicker auf true setzen, wird ShowPicker zur Standardoption.

- ResetNvram - diese Option löscht bestimmte UEFI-Variablen und wird normalerweise durch Halten der Tastenkombination CMD+OPT+P+R während des Bootens ausgeführt. Eine andere Möglichkeit, UEFI-Variablen zu löschen, ist die Auswahl von Reset NVRAM in der OpenCore-Auswahl. Diese Option erfordert, dass AllowNvramReset auf true gesetzt ist.

- BootApple - mit dieser Option wird das erste gefundene Apple-Betriebssystem gebootet, es sei denn, das gewählte Standard-Betriebssystem ist eines von Apple. Halten Sie die X-Taste gedrückt, um diese Option zu wählen.

- BootAppleRecovery - mit dieser Option wird in die Recoverypartition des Apple-Betriebssystems gebootet. Dies ist entweder diejenige, die mit dem gewählten Standard-Betriebssystem verbunden ist, oder die erste, die gefunden wird, wenn das gewählte Standard-Betriebssystem nicht von Apple stammt oder keine Wiederherstellungspartition hat. Halten Sie die Tastenkombination CMD+R gedrückt, um diese Option zu wählen.

Hinweis 1: Auf Nicht-Apple-Firmware werden KeySupport, OpenUsbKbDxe oder ähnliche Treiber für die Tastenbedienung benötigt. Allerdings können nicht alle Funktionen der Tastenbedienung auf verschiedenen Firmware-Typen implementiert werden.

Hinweis 2: Zusätzlich zu OPT unterstützt OpenCore sowohl die Escape- als auch die Null-Taste, um den OpenCore-Picker aufzurufen, wenn ShowPicker deaktiviert ist. Escape existiert, um die Koexistenz mit dem Apple-Picker zu unterstützen (einschließlich des OpenCore-Apple-Picker-Modus) und um Firmware zu unterstützen, die die gehaltene OPT-Taste nicht meldet, wie bei einigen PS/2-Tastaturen. Darüber hinaus wird Zero zur Unterstützung von Systemen bereitgestellt, auf denen Escape bereits einer anderen Pre-Boot-Firmware-Funktion zugewiesen ist. Bei Systemen, die keine KeySupport-Unterstützung benötigen, sollte das Drücken und Halten einer dieser Tasten nach dem Einschalten bis zum Erscheinen des Picker immer erfolgreich sein. Dasselbe sollte auch für den KeySupport-Modus gelten, wenn er für das System richtig konfiguriert ist, d. h. mit einem ausreichend langen KeyForgetThreshold. Wenn das Drücken und Halten der Taste

nicht zum Erfolg führt, können Sie stattdessen mehrere wiederholte Tastendrücke versuchen.

Hinweis 3: Auf Macs mit problematischem GOP kann es schwierig sein, die Apple-Auswahl zu erreichen. Das Dienstprogramm BootKicker kann zur Umgehung dieses Problems eingesetzt werden, auch ohne OpenCore zu laden. Auf manchen Macs kann das BootKicker-Dienstprogramm jedoch nicht von OpenCore aus gestartet werden.

13. PickerVariant

Typ: plist string

Failsafe: Auto

Beschreibung: Auswahl eines bestimmten Symbolsets, das für die Boot-Verwaltung verwendet werden soll.

Ein Icon-Set ist ein Verzeichnispfad relativ zu Resources\Image, in dem sich die Icons und ein optionales Manifest befinden. Es wird empfohlen, dass die Künstler ihre Sets im Vendor\Set-Format bereitstellen, z. B. Acidanthera\GoldenGate.

Beispielressourcen, die als Teil des OcBinaryData-Repository bereitgestellt werden, bieten das folgende Icon-Set:

- Acidanthera\GoldenGate - Iconset im Stil von macOS 11. - Acidanthera\Syrah - macOS 10.10 gestalteter Icon-Satz.

- Acidanthera\Chardonnay - macOS 10.4 gestylter Icon-Satz.

Der Einfachheit halber gibt es auch vordefinierte Aliasnamen:

- Auto - Automatische Auswahl eines Icon-Sets basierend auf der DefaultBackground-Farbe: Acidanthera\GoldenGate für Syrah Black und Acidanthera\Chardonnay für Light Gray.

- Standard - Acidanthera\GoldenGate.

8.4 Debug-Properties

1. AppleDebug

Typ: plist boolean

Failsafe: false

Beschreibung: Aktiviert das Schreiben des boot.efi-Debug-Protokolls in das OpenCore-Protokoll.

Hinweis: Diese Option gilt nur für 10.15.4 und neuer.

2. ApplePanic

Typ: plist boolean

Failsafe: false

Beschreibung: Speichert die macOS-Kernel-Panikausgabe auf der OpenCore-Root-Partition.

Die Datei wird als panic-YYYY-MM-DD-HHMMSS.txt gespeichert. Es wird dringend empfohlen, das Boot-Argument keepsyms=1 zu setzen, um Debugsymbole im Panic-Log zu sehen. In Fällen, in denen dieses Argument nicht vorhanden ist, kann das Dienstprogramm kpdescribe.sh

(im Lieferumfang von OpenCore enthalten) verwendet werden, um den Stacktrace teilweise wiederherzustellen.

Entwicklungs- und Debug-Kernel erzeugen nützlichere Kernel-Panic-Protokolle. Ziehen Sie in Betracht, das KernelDebugKit von developer.apple.com herunterzuladen und zu installieren, wenn Sie ein Problem beheben wollen. Um einen Entwicklungskernel zu aktivieren, sollte das Boot-Argument kcsuffix=development hinzugefügt werden. Verwenden Sie den Befehl uname -a, um sicherzustellen, dass der aktuell geladene Kernel ein Entwicklungs- (oder ein Debug-) Kernel ist.

In Fällen, in denen der OpenCore-Kernel-Panik-Sicherungsmechanismus nicht verwendet wird, können Kernel-Panik-Protokolle immer noch im Verzeichnis /Library/Logs/DiagnosticReports gefunden werden.

Beginnend mit macOS Catalina werden Kernel-Panics im JSON-Format gespeichert und müssen daher vor der Übergabe an kpdescribe.sh vorverarbeitet werden:

---

cat Kernel.panic | grep macOSProcessedStackshotData |

python3 -c 'import json,sys;print(json.load(sys.stdin)["macOSPanicString"])'

---

3. DisableWatchDog

Typ: plist boolean

Failsafe: false

Beschreibung: Bei einigen Firmware-Typen gelingt es möglicherweise nicht, das Betriebssystem schnell zu booten, insbesondere im Debug-Modus. Dies führt dazu, dass der Watchdog-Timer den Prozess abbricht. Diese Option schaltet den Watchdog-Timer aus.

4. DisplayDelay

Typ: plist integer

Failsafe: 0

Beschreibung: Verzögerung in Mikrosekunden, die nach jeder gedruckten, auf dem Bildschirm (d.h. der Konsole) sichtbaren Zeile ausgeführt wird.

5. AnzeigeLevel

Typ: plist integer, 64 bit

Failsafe: 0

Beschreibung: EDK II Debug-Level-Bitmaske (Summe), die auf dem Bildschirm angezeigt wird. Sofern Target nicht den Konsolendruck (Onscreen) aktiviert, ist die Debug-Ausgabe auf dem Bildschirm nicht sichtbar.

Die folgenden Level werden unterstützt (mehr dazu in DebugLib.h):

• 0x00000002 (bit 1) — DEBUG_WARN in DEBUG, NOOPT, RELEASE.
• 0x00000040 (bit 6) — DEBUG_INFO in DEBUG, NOOPT.
• 0x00400000 (bit 22) — DEBUG_VERBOSE in custom builds.
• 0x80000000 (bit 31) — DEBUG_ERROR in DEBUG, NOOPT, RELEASE.

6. LogModule

Typ: plist-Zeichenkette

Ausfallsicher: *

Beschreibung: Protokolleinträge nach Modulen filtern.

Diese Option filtert die von bestimmten Modulen erzeugten Protokolleinträge, sowohl im Protokoll als auch auf dem Bildschirm. Es werden zwei Modi unterstützt:

• + — Positive Filterung: Nur ausgewählte Module werden angezeigt.

• - —Negativ-Filterung: Ausgewählte Module ausschließen.

Wenn mehrere Module ausgewählt werden, sollte ein Komma (,) als Trennzeichen verwendet werden. Zum Beispiel bedeutet +OCCPU,OCA,OCB, dass nur OCCPU, OCA, OCB gedruckt werden, während -OCCPU,OCA,OCB bedeutet, dass diese Module herausgefiltert (d.h. nicht protokolliert) werden. Wenn kein Symbol angegeben wird, wird eine positive Filterung (+) verwendet. * bedeutet, dass alle Module protokolliert werden.

Hinweis 1: Die Abkürzungen der Bibliotheken finden Sie im Abschnitt Bibliotheken weiter unten.

Hinweis 2: Meldungen, die vor der Konfiguration des Protokolls gedruckt wurden, können nicht gefiltert werden.

7. SysReport

Typ: plist boolean

Failsafe: false

Beschreibung: Systembericht im ESP-Ordner erstellen.

Mit dieser Option wird ein SysReport-Verzeichnis in der ESP-Partition erstellt, sofern es nicht bereits vorhanden ist. Das Verzeichnis enthält ACPI-, SMBIOS- und Audio-Codec-Dumps. Für Audiocodec-Dumps muss ein Audio-Backend-Treiber geladen werden.

Hinweis: Um die Systemintegrität zu wahren, ist die Option SysReport in RELEASE-Builds nicht verfügbar. Verwenden Sie einen DEBUG-Build, wenn diese Option erforderlich ist.

8. Target

Typ: plist integer

Failsafe: 0

Beschreibung: Eine Bitmaske (Summe) von aktivierten Protokollierungszielen. Die Logging-Ausgabe ist standardmäßig ausgeblendet und diese Option muss gesetzt werden, wenn eine solche Ausgabe erforderlich ist, z. B. beim Debuggen.

Die folgenden Protokollierungsziele werden unterstützt:

- 0x01 (bit 0) — Enable logging, otherwise all log is discarded.
- 0x02 (bit 1) — Enable basic console (onscreen) logging.
- 0x04 (bit 2) — Enable logging to Data Hub.
- 0x08 (bit 3) — Enable serial port logging.
- 0x10 (bit 4) — Enable UEFI variable logging.
- 0x20 (bit 5) — Enable non-volatile UEFI variable logging.
- 0x40 (bit 6) — Enable logging to file.

Die Konsolenprotokollierung gibt weniger aus als die anderen Varianten. Je nach Build-Typ (RELEASE, DEBUG oder NOOPT) werden unterschiedlich viele Protokolle gelesen (vom geringsten bis zum größten).

Um Data Hub-Protokolle zu erhalten, verwenden Sie den folgenden Befehl unter macOS (beachten Sie, dass Data Hub-Protokolle keine Kernel- und Kext-Patches protokollieren):

_____

ioreg -lw0 -p IODeviceTree | grep boot-log | sort | sed 's/.*<\(.*\)>.*/\1/' | xxd -r -p

_____

Das UEFI-Variablenprotokoll enthält einige Meldungen nicht und weist keine Leistungsdaten auf. Um die Systemintegrität zu wahren, ist die Protokollgröße auf 32 Kilobyte begrenzt. Einige Firmware-Typen können es viel früher abschneiden oder ganz abbrechen, wenn sie keinen Speicher haben. Wenn Sie das non-volatile flag verwenden, wird das Protokoll nach jeder gedruckten Zeile in den NVRAM-Flash geschrieben.

Um UEFI-Variablenprotokolle zu erhalten, verwenden Sie unter macOS den folgenden Befehl:

_____

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

_____

Warnung: Bestimmte Firmware scheint eine fehlerhafte NVRAM-Garbage-Collection zu haben. Infolgedessen sind sie möglicherweise nicht in der Lage, den Speicherplatz nach dem Löschen von Variablen immer freizugeben. Aktivieren Sie die non-volatile NVRAM-Protokollierung auf solchen Geräten nicht, es sei denn, dies ist ausdrücklich erforderlich.

Während das OpenCore-Bootprotokoll bereits grundlegende Versionsinformationen einschließlich Build-Typ und Datum enthält, können diese Informationen auch in der NVRAM-Variable opencore-version gefunden werden, selbst wenn die Boot-Protokollierung deaktiviert ist.

Die Dateiprotokollierung erstellt eine Datei mit dem Namen opencore-YYYY-MM-DD-HHMMSS.txt (in UTC) unter dem EFI-Volume-Root mit dem Protokollinhalt (die Großbuchstabenfolge wird durch Datum und Uhrzeit der Firmware ersetzt). Bitte seien Sie gewarnt, dass einige der in der Firmware vorhandenen Dateisystemtreiber nicht zuverlässig sind und Daten beim Schreiben von Dateien durch die Firmware beschädigen können.

UEFI. Das Schreiben von Protokollen wird auf die sicherste Weise versucht und ist daher sehr langsam. Stellen Sie sicher, dass DisableWatchDog auf true gesetzt ist, wenn ein langsames Laufwerk verwendet wird. Versuchen Sie, die häufige Verwendung dieser Option bei Flash-Laufwerken zu vermeiden, da große E/A-Mengen die Speicherabnutzung beschleunigen und das Flash-Laufwerk schneller unbrauchbar machen können.

Bei der Interpretation des Protokolls ist zu beachten, dass den Zeilen ein Tag vorangestellt ist, der die betreffende Stelle (Modul) der Protokollzeile beschreibt und eine bessere Zuordnung der Zeile zu der jeweiligen Funktionalität ermöglicht.

Die Liste der derzeit verwendeten Tags lautet wie folgt. Treiber und tools:

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

Libraries:

• AAPL — OcLogAggregatorLib, 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
• OCFV — OcFirmwareVolumeLib
• OCHS — OcHashServicesLib
• OCI4 — OcAppleImg4Lib
• OCIC — OcImageConversionLib
• OCII — OcInputLib
• OCJS — OcApfsLib
• OCKM — OcAppleKeyMapLib
• OCL — OcLogAggregatorLib
• OCM — OcMiscLib
• OCMCO — OcMachoLib
• OCME — OcHeciLib

• OCMM — OcMemoryLib
• OCPE — OcPeCoffLib, OcPeCoffExtLib
• OCPI — OcFileLib, partition info
• OCPNG — OcPngLib
• OCRAM — OcAppleRamDiskLib
• OCRTC — OcRtcLib

S. 48

• OCSB — OcAppleSecureBootLib • OCSMB — OcSmbiosLib
• OCSMC — OcSmcLib
• OCST — OcStorageLib
• OCS — OcSerializedLib
• OCTPL — OcTemplateLib
• OCUC — OcUnicodeCollationLib
• OCUT — OcAppleUserInterfaceThemeLib • OCXML — OcXmlLib

8.5 Security Properties

1. AllowNvramReset

Typ: plist boolean

Failsafe: false

Beschreibung: Erlaubt die CMD+OPT+P+R-Bedienung und aktiviert die Anzeige des Eintrags NVRAM Reset in der OpenCore-Auswahl.

Hinweis 1: Es ist bekannt, dass einige Lenovo-Laptops einen Firmware-Fehler haben, der dazu führt, dass sie nach einem NVRAM-Reset nicht mehr gestartet werden können. Siehe acidanthera/bugtracker#995 für Details.

Hinweis 2: Das Zurücksetzen des NVRAMs löscht auch alle Boot-Optionen, die nicht mit dem bless-Befehl gesichert wurden. Zum Beispiel Linux-Installationen an benutzerdefinierten Orten, die nicht in BlessOverride definiert sind.

2. AllowSetDefault

Typ: plist boolean

Failsafe: false

Beschreibung: Erlaubt die Verwendung von CTRL+Enter und CTRL+Index, um die Standard-Boot-Option in der OpenCore-Auswahl festzulegen.

Hinweis 1: Kann in Kombination mit Shift+Enter oder Shift+Index verwendet werden, wenn PollAppleHotKeys aktiviert ist.

Hinweis 2: Um Systeme mit nicht reagierenden Modifiern während des Preboots zu unterstützen (was den V1- und V2-KeySupport-Modus auf einiger Firmware einschließt), erlaubt OpenCore auch das Halten der =/+ Taste, um den 'set default'-Modus auszulösen.

3. AllowToggleSip

Typ: plist boolean

Failsafe: false

Beschreibung: Eintrag zum Aktivieren und Deaktivieren des Systemintegritätsschutzes in der OpenCore-Auswahl.

Damit wird die Apple NVRAM-Variable csr-active-config zwischen 0 für SIP Enabled und einem praktischen Standardwert für SIP Disabled umgeschaltet.

Hinweis 1: Es wird dringend empfohlen, es sich nicht zur Gewohnheit zu machen, macOS mit deaktiviertem SIP zu starten. Die Verwendung dieser Boot-Option kann es einfacher machen, den SIP-Schutz schnell zu deaktivieren, wenn er wirklich benötigt wird - er sollte danach wieder aktiviert werden.

Hinweis 2: OpenCore verwendet 0x27F, während csrutil disable unter macOS Big Sur und Monterey 0x7F setzt.

- CSR_ALLOW_UNAPPROVED_KEXTS (0x200) ist im Allgemeinen nützlich, wenn Sie SIP ohnehin deaktivieren müssen, da es die Installation von unsignierten Kexts ohne manuelle Genehmigung in den Systemeinstellungen erlaubt.

- CSR_ALLOW_UNAUTHENTICATED_ROOT (0x800) ist nicht enthalten, da es bei seiner Verwendung sehr einfach ist, versehentlich OS-Siegel zu brechen und inkrementelle OTA-Updates zu verhindern.

Hinweis3: Für jeden anderen Wert, den Sie benötigen, ist es möglich, CsrUtil.efi als TextMode Tools-Eintrag zu konfigurieren, um einen anderen Wert zu konfigurieren, z. B. mit dem Toggle 0x77 in Arguments, um den in macOS Catalina standardmäßig eingestellten Wert für SIP deaktiviert umzuschalten.

4. ApECID

Typ: plist integer,

64 bit Failsafe: 0

Beschreibung: Apple Enclave Bezeichner.

S. 49

Wenn Sie diesen Wert auf eine beliebige 64-Bit-Ganzzahl ungleich Null setzen, können Sie personalisierte Apple Secure Boot-Kennungen verwenden. Um diese Einstellung zu verwenden, generieren Sie eine 64-Bit-Zufallszahl mit einem kryptografisch sicheren Zufallszahlengenerator. Als Alternative können die ersten 8 Bytes von SystemUUID für ApECID verwendet werden, dies ist in macOS 11 für Macs ohne T2-Chip zu finden.

Wenn dieser Wert gesetzt und SecureBootModel gültig (und nicht deaktiviert) ist, ist es möglich, die volle Sicherheit von Apple Secure Boot zu erreichen.

Um den personalisierten Apple Secure Boot zu nutzen, muss das Betriebssystem neu installiert oder personalisiert werden. Solange das Betriebssystem nicht personalisiert ist,

kann nur macOS DMG Recovery geladen werden. In Fällen, in denen die DMG-Wiederherstellung fehlt, kann sie mit dem Dienstprogramm macrecovery heruntergeladen und in com.apple.recovery.boot gespeichert werden, wie im Abschnitt Tipps und Tricks beschrieben. Beachten Sie, dass das Laden von DMGs auf "Signed" (Unterzeichnet) gesetzt werden muss, um eine DMG mit Apple Secure Boot zu verwenden.

Um ein bestehendes Betriebssystem zu personalisieren, verwenden Sie den bless-Befehl nach dem Laden in die macOS DMG Recovery. Mounten Sie die Partition des Systemvolumes, sofern sie nicht bereits gemountet ist, und führen Sie den folgenden Befehl aus:

```
bless --folder "/Volumes/Macintosh HD/System/Library/CoreServices" \
    --bootefi --personalize
```

Unter macOS 11 und neuer verwendet das dedizierte x86-Legacy-Modell immer ApECID. Wenn diese Konfigurationseinstellung auf 0 belassen wird, werden stattdessen die ersten 8 Bytes der system-id-Variable verwendet.

Bei macOS-Versionen vor macOS 11, das ein dediziertes x86-Legacy-Modell für Modelle ohne T2-Chip einführte, funktioniert der personalisierte Apple Secure Boot möglicherweise nicht wie erwartet. Bei der Neuinstallation des Betriebssystems geht dem macOS-Installationsprogramm von macOS 10.15 und älter oft der freie Speicher auf der /var/tmp-Partition aus, wenn es versucht, macOS mit dem personalisierten Apple Secure Boot zu installieren. Kurz nach dem Herunterladen des macOS-Installer-Images erscheint die Fehlermeldung Unable to verify macOS.

Um dieses Problem zu umgehen, weisen Sie eine dedizierte RAM-Disk von 2 MB für die macOS-Personalisierung zu, indem Sie die folgenden Befehle in das macOS-Wiederherstellungsterminal eingeben, bevor Sie die Installation starten:

```
disk=$(hdiutil attach -nomount ram://4096)
diskutil erasevolume HFS+ SecureBoot $disk
diskutil unmount $disk
mkdir /var/tmp/OSPersonalizationTemp
diskutil mount -mountpoint /var/tmp/OSPersonalizationTemp $disk
```

5. AuthRestart

Typ: plist boolean

Failsafe: false

Beschreibung: Aktiviert den VirtualSMC-kompatiblen authentifizierten Neustart.

Authentifizierter Neustart ist eine Möglichkeit, FileVault 2-fähiges macOS ohne Eingabe des Passworts neu zu starten. Ein spezieller Terminalbefehl kann verwendet werden, um

authentifizierte Neustarts durchzuführen: sudo fdesetup authrestart. Er wird auch bei der Installation von Betriebssystem-Updates verwendet.

VirtualSMC führt authentifizierte Neustarts durch, indem es Festplattenverschlüsselungsschlüssel zwischen NVRAM und RTC aufteilt und speichert, die zwar entfernt werden, sobald OpenCore startet, aber als Sicherheitsrisiko angesehen werden können und daher optional sind.

6. BlacklistAppleUpdate

Typ: plist boolean

Failsafe: false

Beschreibung: Ignoriert Boot-Optionen, die versuchen, Apple-Peripherie-Firmware zu aktualisieren (z. B. MultiUpdater.efi).

Hinweis: Bestimmte Betriebssysteme, wie z.B. macOS Big Sur, sind nicht in der Lage, Firmware-Updates durch Verwendung der run-efi-updater NVRAM-Variable zu deaktivieren.

7. DmgLoading

Typ: plist string

Failsafe: Signed

Beschreibung: Definiert die für die macOS-Wiederherstellung verwendete Richtlinie zum Laden von Disk-Images (DMG).

Gültige Werte:

- Deaktiviert - das Laden von DMG-Images wird fehlschlagen. Die Richtlinie Deaktiviert lässt in den meisten Fällen das Laden der macOS-Wiederherstellung zu, da in der Regel boot.efi-Dateien vorhanden sind, die mit Apple Secure Boot kompatibel sind. Manuell heruntergeladene DMG-Images, die in com.apple.recovery.boot-Verzeichnissen gespeichert sind, werden jedoch nicht geladen.

- Signiert - nur von Apple signierte DMG-Images werden geladen. Aufgrund des Designs von Apple Secure Boot lässt die Richtlinie Signiert jedes von Apple signierte macOS Recovery unabhängig vom Apple Secure Boot-Status laden, was nicht immer erwünscht ist. Während die Verwendung von signierten DMG-Images wünschenswerter ist, kann die Überprüfung der Image-Signatur die Boot-Zeit leicht verlangsamen (um bis zu 1 Sekunde).

- Any - alle DMG-Images werden als normale Dateisysteme gemountet. Von der Option Any wird dringend abgeraten, da sie zu Bootfehlern führt, wenn Apple Secure Boot aktiv ist.

8. EnablePassword

Typ: plist boolean

Failsafe: false

Beschreibung: Aktivieren Sie den Passwortschutz, um sensible Vorgänge zu erleichtern.

Der Passwortschutz stellt sicher, dass sensible Vorgänge wie das Booten eines nicht standardmäßigen Betriebssystems (z. B. macOS Recovery oder ein Tool), das Zurücksetzen des NVRAM-Speichers oder der Versuch, in einen nicht standardmäßigen Modus zu booten (z. B. verbose mode oder safe mode), ohne explizite Benutzerauthentifizierung durch ein benutzerdefiniertes Passwort nicht erlaubt sind. Derzeit werden Passwort und Salt mit 5000000 Iterationen von SHA-512 gehasht.

Hinweis: Diese Funktionalität befindet sich noch in der Entwicklung und ist nicht für Produktionsumgebungen geeignet.

9. ExposeSensitiveData

Typ: plist integer

Failsafe: 0x6

Beschreibung: Bitmaske (Summe) zur Freigabe sensibler Daten an das Betriebssystem.

• 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.

Der offengelegte Booter-Pfad verweist auf OpenCore.efi oder dessen Booter, je nach Ladereihenfolge. Um den Booter-Pfad zu erhalten, verwenden Sie unter macOS den folgenden Befehl:

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

Um einen Booter-Pfad zu verwenden, um ein Booter-Volume zu mounten, verwenden Sie den folgenden Befehl in macOS:

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

Um die aktuelle OpenCore-Version zu erhalten, verwenden Sie unter macOS den folgenden Befehl:

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

Wenn die OpenCore-Version nicht bekannt ist, enthält die Variable die Sequenz UNK-000-0000-00-00. Um OEM-Informationen zu erhalten, verwenden Sie unter macOS die folgenden Befehle:

---

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*

---

10. HaltLevel

Typ: plist integer, 64 bit

Failsafe: 0x80000000 (DEBUG_ERROR)

Beschreibung: EDK II-Debug-Level-Bitmaske (Summe), die die CPU zum Anhalten (Stoppen der Ausführung) veranlasst, nachdem eine Meldung von HaltLevel erhalten wurde. Mögliche Werte entsprechen den DisplayLevel-Werten.

11. PasswordHash

Type: plist data 64 bytes

Failsafe: all zero

Beschreibung: Passwort-Hash, der verwendet wird, wenn EnabledPassword gesetzt ist.

12. PasswordSalt

Type: plist data

Failsafe: empty

Beschreibung: PasswordSalt, das verwendet wird, wenn EnabledPassword gesetzt ist.

13. Vault

Type: plist string

Failsafe: Secure

Beschreibung: Aktiviert den OpenCore Vaulting-Mechanismus.

Gültige Werte:

- Optional — require nothing, no vault is enforced, insecure.
- Basic — require vault.plist file present in OC directory. This provides basic filesystem integrity verification
- and may protect from unintentional filesystem corruption.
- Secure — require vault.sig signature file for vault.plist in OC directory. This includes Basic integrity checking but also attempts to build a trusted bootchain.

Die Datei vault.plist sollte SHA-256-Hashes für alle von OpenCore verwendeten Dateien enthalten. Das Vorhandensein dieser Datei wird dringend empfohlen, um sicherzustellen, dass unbeabsichtigte Dateiänderungen (einschließlich Dateisystembeschädigungen) nicht unbemerkt bleiben. Um diese Datei automatisch zu erstellen, verwenden Sie das Skript create_vault.sh. Unabhängig vom zugrunde liegenden Dateisystem müssen die Pfadnamen und Groß- und Kleinschreibung zwischen config.plist und vault.plist übereinstimmen.

Die Datei vault.sig sollte eine 256-Byte-RSA-2048-Rohsignatur aus einem SHA-256-Hash von vault.plist enthalten. Die Signatur wird anhand des in OpenCore.efi eingebetteten öffentlichen Schlüssels überprüft.

Um den öffentlichen Schlüssel einzubetten, sollte einer der folgenden Schritte durchgeführt werden:

- Provide public key during the OpenCore.efi compilation in OpenCoreVault.c file.
- Binary patch OpenCore.efi replacing zeroes with the public key between =BEGIN OC VAULT= and ==END OC VAULT== ASCII markers.

Die Beschreibung des 520-Byte-Formats des öffentlichen RSA-Schlüssels ist in der Chromium OS-Dokumentation zu finden. Zum Konvertieren des öffentlichen Schlüssel aus einem X.509-Zertifikat oder aus einer PEM-Datei zu konvertieren, verwenden Sie RsaTool. Der vollständige Satz von Befehlen zu:

• Create vault.plist.
• Create a new RSA key (always do this to avoid loading old configuration). • Embed RSA key into OpenCore.efi.
• Create vault.sig.

Das kann wie folgt aussehen:

```
cd /Volumes/EFI/EFI/OC
/path/to/create_vault.sh .
/path/to/RsaTool -sign vault.plist vault.sig vault.pub
off=$(($(strings -a -t d OpenCore.efi | grep "=BEGIN OC VAULT=" | cut -f1 -d' ')+16)) dd
of=OpenCore.efi if=vault.pub bs=1 seek=$off count=528 conv=notrunc
rm vault.pub
```

Hinweis 1: Auch wenn es offensichtlich erscheint, ist eine externe Methode erforderlich, um OpenCore.efi und BOOTx64.efi für den sicheren Boot-Pfad zu verifizieren. Dazu wird empfohlen, UEFI SecureBoot mit einem eigenen Zertifikat zu aktivieren und OpenCore.efi und BOOTx64.efi mit einem eigenen Schlüssel zu signieren. Weitere Details zum Anpassen von SecureBoot auf moderner Firmware finden Sie im Dokument Taming UEFI SecureBoot (auf Russisch).

Hinweis 2: Unabhängig von dieser Option wird vault.plist immer verwendet, wenn sie vorhanden ist, und sowohl vault.plist als auch vault.sig werden verwendet und sind erforderlich, wenn ein öffentlicher Schlüssel in OpenCore.efi eingebettet ist, und Fehler führen zum Abbruch des

Boot-Prozess in beiden Fällen. Durch das Setzen dieser Option kann OpenCore den Benutzer warnen, wenn die Konfiguration nicht den Anforderungen entspricht, um eine erwartete höhere Sicherheitsstufe zu erreichen.

14. ScanPolicy

Typ: plist integer, 32 bit

Failsafe: 0x10F0103

Beschreibung: Definieren Sie eine Richtlinie für die Erkennung von Betriebssystemen.

Dieser Wert ermöglicht das Verhindern des Scannens (und Bootens) von nicht vertrauenswürdigen Quellen auf der Grundlage einer Bitmaske (Summe) aus einer Reihe von Flags. Da es nicht möglich ist, jedes Dateisystem oder jeden Gerätetyp zuverlässig zu erkennen, kann man sich in offenen Umgebungen nicht vollständig auf diese Funktion verlassen, sondern muss zusätzliche Maßnahmen ergreifen.

Treiber von Drittanbietern können zusätzliche Sicherheits- (und Leistungs-) Maßnahmen im Anschluss an die bereitgestellte Scan-Richtlinie einführen. Die aktive Scan-Richtlinie wird in der scan-policy-Variable 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102 GUID nur für UEFI-Boot-Dienste angezeigt.

- 0x00000001 (bit 0) — OC_SCAN_FILE_SYSTEM_LOCK, restricts scanning to only known file systems defined as a part of this policy. File system drivers may not be aware of this policy. Hence, to avoid mounting of undesired file systems, drivers for such file systems should not be loaded. This bit does not affect DMG mounting, which may have any file system. Known file systems are prefixed with OC_SCAN_ALLOW_FS_.
- 0x00000002 (bit 1) — OC_SCAN_DEVICE_LOCK, restricts scanning to only known device types defined as a part of this policy. It is not always possible to detect protocol tunneling, so be aware that on some systems, it may be possible for e.g. USB HDDs to be recognised as SATA instead. Cases like this must be reported. Known device types are prefixed with OC_SCAN_ALLOW_DEVICE_.
- 0x00000100 (bit 8) — OC_SCAN_ALLOW_FS_APFS, allows scanning of APFS file system.
- 0x00000200 (bit 9) — OC_SCAN_ALLOW_FS_HFS, allows scanning of HFS file system.
- 0x00000400 (bit 10) — OC_SCAN_ALLOW_FS_ESP, allows scanning of EFI System Partition file system.
- 0x00000800 (bit 11) — OC_SCAN_ALLOW_FS_NTFS, allows scanning of NTFS (Msft Basic Data) file system.
- 0x00001000 (bit 12) — OC_SCAN_ALLOW_FS_LINUX_ROOT, allows scanning of Linux Root file systems.
- 0x00002000 (bit 13) — OC_SCAN_ALLOW_FS_LINUX_DATA, allows scanning of Linux Data file systems.
- 0x00004000 (bit 14) — OC_SCAN_ALLOW_FS_XBOOTLDR, allows scanning the Extended Boot Loader Partition
- as defined by the Boot Loader Specification.
- 0x00010000 (bit 16) — OC_SCAN_ALLOW_DEVICE_SATA, allow scanning SATA devices.
- 0x00020000 (bit 17) — OC_SCAN_ALLOW_DEVICE_SASEX, allow scanning SAS and Mac NVMe devices.
- 0x00040000 (bit 18) — OC_SCAN_ALLOW_DEVICE_SCSI, allow scanning SCSI devices.
- 0x00080000 (bit 19) — OC_SCAN_ALLOW_DEVICE_NVME, allow scanning NVMe devices.
- 0x00100000 (bit 20) — OC_SCAN_ALLOW_DEVICE_ATAPI, allow scanning CD/DVD devices and old SATA.
- 0x00200000 (bit 21) — OC_SCAN_ALLOW_DEVICE_USB, allow scanning USB devices.

- 0x00400000 (bit 22) — OC_SCAN_ALLOW_DEVICE_FIREWIRE, allow scanning FireWire devices.
- 0x00800000 (bit 23) — OC_SCAN_ALLOW_DEVICE_SDCARD, allow scanning card reader devices.
- 0x01000000 (bit 24) — OC_SCAN_ALLOW_DEVICE_PCI, allow scanning devices directly connected to PCI bus (e.g. VIRTIO).

Hinweis: Angesichts der obigen Beschreibung wird bei einem Wert von 0xF0103 Folgendes erwartet:

• Permit scanning SATA, SAS, SCSI, and NVMe devices with APFS file systems.
• Prevent scanning any devices with HFS or FAT32 file systems.
• Prevent scanning APFS file systems on USB, CD, and FireWire drives.

Die Kombination lautet wie folgt:

- `OC_SCAN_FILE_SYSTEM_LOCK`
- `OC_SCAN_DEVICE_LOCK`
- `OC_SCAN_ALLOW_FS_APFS`
- `OC_SCAN_ALLOW_DEVICE_SATA`
- `OC_SCAN_ALLOW_DEVICE_SASEX`
- `OC_SCAN_ALLOW_DEVICE_SCSI`
- `OC_SCAN_ALLOW_DEVICE_NVME`

15. SecureBootModel Typ: plist string

S. 53

15. SecureBootModel Typ: plist string

Failsafe: Default

Beschreibung: Apple Secure Boot Hardware-Modell.

Legt das Apple Secure Boot-Hardwaremodell und die Richtlinie fest. Durch die Angabe dieses Wertes wird festgelegt, welche Betriebssysteme gebootet werden können. Betriebssysteme, die vor der Veröffentlichung des angegebenen Modells ausgeliefert wurden, können nicht gebootet werden.

Gültige Werte:

• Default — Matching model for current SMBIOS.
• Disabled — No model, Secure Boot will be disabled.

• j137 — iMacPro1,1 (December 2017).
• j680 — MacBookPro15,1 (July 2018).
• j132 — MacBookPro15,2 (July 2018).
• j174 — Macmini8,1 (October 2018).
• j140k — MacBookAir8,1 (October 2018). Minimum macOS 10.14.1 (18B2084)

• j780 — MacBookPro15,3 (May 2019).
• j213 — MacBookPro15,4 (July 2019).
• j140a — MacBookAir8,2 (July 2019).
• j152f — MacBookPro16,1 (November 2019). Minimum macOS 10.15.1 (19B2093)

Minimum macOS 10.13.2 (17C2111)
Minimum macOS 10.13.6 (17G2112)
Minimum macOS 10.13.6 (17G2112)

Minimum macOS 10.14 (18A2063)

Minimum macOS 10.14.5 (18F132)
Minimum macOS 10.14.5 (18F2058)
Minimum macOS 10.14.5 (18F2058)

• j160 — MacPro7,1 (December 2019). Minimum macOS 10.15.1 (19B88)
• j230k — MacBookAir9,1 (March 2020). Mindestens macOS 10.15.3 (19D2064)
• j214k — MacBookPro16,2 (May 2020). Mindestens macOS 10.15.4 (19E2269)
• j223 — MacBookPro16,3 (May 2020). Mindestens macOS 10.15.4 (19E2265)
• j215 — MacBookPro16,4 (June 2020). Mindestens macOS 10.15.5 (19F96)
• j185 — iMac20,1 (August 2020). Minimum macOS 10.15.6 (19G2005)
• j185f — iMac20,2 (August 2020). Minimum macOS 10.15.6 (19G2005)
• x86legacy — Macs without T2 chip and VMs. Minimum macOS 11.0.1 (20B29)

Achtung! Nicht alle Apple Secure Boot Modelle werden auf allen Hardwarekonfigurationen unterstützt.

Apple Secure Boot erschien in macOS 10.13 auf Modellen mit T2-Chips. Vor macOS 12 waren PlatformInfo und SecureBootModel unabhängig, so dass Apple Secure Boot mit jedem SMBIOS mit und ohne T2 verwendet werden kann. Beginnend mit macOS 12 muss SecureBootModel mit dem SMBIOS-Mac-Modell übereinstimmen. Das Standardmodell leitet das Modell basierend auf dem SMBIOS-Board-Identifikator ab, der entweder automatisch über den Abschnitt Generic oder manuell über den Abschnitt SMBIOS gesetzt wird. Wenn es keine Board-Identifikation gibt, wird das Modell heuristisch vom OEM SMBIOS abgeleitet.

Die Einstellung SecureBootModel auf einen beliebigen gültigen Wert außer Disabled entspricht der mittleren Sicherheit von Apple Secure Boot. Der ApECID-Wert muss ebenfalls angegeben werden, um volle Sicherheit zu erreichen. Aktivieren Sie ForceSecureBootScheme, wenn Sie Apple Secure Boot auf einer virtuellen Maschine verwenden.

Beachten Sie, dass die Aktivierung von Apple Secure Boot bei ungültigen Konfigurationen, fehlerhaften macOS-Installationen und bei nicht unterstützten Setups sehr anspruchsvoll ist.

Einige Dinge sind zu beachten:

(a) Wie bei T2 Macs können alle unsignierten Kernel-Erweiterungen sowie mehrere signierte Kernel-Erweiterungen, einschließlich NVIDIA Web Drivers, nicht installiert werden.

(b) Die Liste der zwischengespeicherten Kernel-Erweiterungen kann unterschiedlich sein, was dazu führt, dass die Liste der hinzugefügten oder erzwungenen Kernel-Erweiterungen geändert werden muss. In diesem Fall kann zum Beispiel IO80211Family nicht injiziert werden.

(c) Änderungen an Systemvolumes auf Betriebssystemen mit Versiegelung, wie z. B. macOS 11, können dazu führen, dass das Betriebssystem nicht mehr bootfähig ist. Versuchen Sie nicht, die Verschlüsselung des Systemvolumens zu deaktivieren, wenn Apple Secure Boot nicht deaktiviert ist.

(d) Boot-Fehler können auftreten, wenn die Plattform bestimmte Einstellungen erfordert, diese aber nicht aktiviert wurden, weil die damit verbundenen Probleme nicht früher entdeckt wurden. Seien Sie besonders vorsichtig mit IgnoreInvalidFlexRatio oder HashServices.

(e) Betriebssysteme, die vor der Veröffentlichung von Apple Secure Boot veröffentlicht wurden (z. B. macOS 10.12 oder früher), booten weiterhin, bis UEFI Secure Boot aktiviert ist. Dies liegt daran, dass Apple Secure Boot diese als inkompatibel behandelt und sie dann von der Firmware gehandhabt werden (wie Microsoft Windows).

(f) Auf älteren CPUs (z.B. vor Sandy Bridge) kann die Aktivierung von Apple Secure Boot zu einem etwas langsameren Laden führen (bis zu 1 Sekunde).

(g) Da der Standardwert mit der Zeit ansteigt, um die neuesten Betriebssysteme zu unterstützen, wird nicht empfohlen, die ApECID- und die Standardeinstellungen zusammen zu verwenden.

(h) Die Installation von macOS mit aktiviertem Apple Secure Boot ist bei Verwendung von HFS+-Zielvolumes nicht möglich. Dies kann HFS+-formatierte Laufwerke einschließen, wenn kein APFS-Ersatzlaufwerk verfügbar ist.

Das installierte Betriebssystem kann manchmal veraltete Apple Secure Boot-Manifeste auf der Preboot-Partition haben, was zu Startfehlern führt. Dies ist wahrscheinlich der Fall, wenn eine "OCB: Apple Secure Boot prohibits this boot entry, enforcing!”-Meldung protokolliert wird.

Wenn dies der Fall ist, installieren Sie entweder das Betriebssystem neu oder kopieren Sie die Manifeste (Dateien mit der Erweiterung .im4m, wie boot.efi.j137.im4m) von /usr/standalone/i386 nach /Volumes/Preboot/<UUID>/System/Library/CoreServices. Dabei

ist <UUID> der System-Volume-Bezeichner. Bei HFS+-Installationen sollten die Manifeste nach /System/Library/CoreServices auf dem Systemvolume kopiert werden.

Weitere Einzelheiten zur Konfiguration von Apple Secure Boot mit UEFI Secure Boot finden Sie im Abschnitt UEFI Secure Boot

8.6 Serial Properties

1. Custom

Type: plist dict

Beschreibung: Aktualisierung der Eigenschaften der seriellen Schnittstelle in BaseSerialPortLib16550.

Dieser Abschnitt listet die PCD-Werte auf, die von der BaseSerialPortLib16550 verwendet werden. Wenn die Option Override auf false gesetzt ist, ist dieses dictionary optional.

2. Init

Typ: plist boolean

Failsafe: false

Beschreibung: Initialisierung der seriellen Schnittstelle durchführen.

Diese Option führt die Initialisierung der seriellen Schnittstelle innerhalb von OpenCore durch, bevor die Debug-Protokollierung (überhaupt) aktiviert wird.

Siehe den Abschnitt Debugging für Details.

3. Override

Type: plist boolean

Failsafe: false

Beschreibung: Überschreibt die Eigenschaften der seriellen Schnittstelle. Wenn diese Option auf false gesetzt ist, werden keine Schlüssel von Custom überschrieben.

Diese Option überschreibt die Eigenschaften der seriellen Schnittstelle, die im Abschnitt Benutzerdefinierte serielle Eigenschaften unten aufgeführt sind.

8.6.1 Benutzerdefinierte serielle Eigenschaften

1. BaudRate

Typ: plist integer

Failsafe: 115200

Beschreibung: Legt die Baudrate für die serielle Schnittstelle fest.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialBaudRate, definiert in Mde-ModulePkg.dec.

2. ClockRate

Typ: plist integer

Failsafe: 1843200

Beschreibung: Legt die Taktrate für die serielle Schnittstelle fest.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialClockRate, definiert in Mde-ModulePkg.dec.

3. ExtendedTxFifoSize Typ: plist Ganzzahl

S. 55

3. ExtendedTxFifoSize

Typ: plist integer

Failsafe: 64

Beschreibung: Legt die erweiterte Sende-FIFO-Größe für die serielle Schnittstelle fest.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialExtendedTxFifoSize, der in MdeModulePkg.dec festgelegt ist.

4. FifoControl

Typ: plist integer

Failsafe: 0x07

Beschreibung: Konfiguriert die Einstellungen für die FIFO-Steuerung der seriellen Schnittstelle.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialFifoControl, definiert in MdeModulePkg.dec.

5. LineControl

Type: plist integer

Failsafe: 0x07

Beschreibung: Konfiguriert die Einstellungen für die Leitungssteuerung der seriellen Schnittstelle.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialLineControl, definiert in MdeModulePkg.dec.

6. PciDeviceInfo

Typ: plist data

Failsafe: 0xFF

Beschreibung: Legt die Informationen zum seriellen PCI-Gerät fest.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialPciDeviceInfo, definiert in MdeModulePkg.dec.

Hinweis: Die maximal zulässige Größe dieser Option beträgt 41 Bytes. Siehe acidanthera/bugtracker#1954 für weitere Details.

Hinweis 2: Diese Option kann durch Ausführen des FindSerialPort-Tools gesetzt werden.

7. RegisterAccessWidth

Type: plist integer

Failsafe: 8

Beschreibung: Legt die Zugriffsbreite auf die Register der seriellen Schnittstelle fest.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialRegisterAccessWidth, definiert in MdeModulePkg.dec.

8. RegisterBase

Typ: plist integer

Failsafe: 0x03F8

Beschreibung: Legt die Basisadresse der Register der seriellen Schnittstelle fest.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialRegisterBase, definiert in MdeModulePkg.dec.

9. RegisterStride

Typ: plist integer

Failsafe: 1

Beschreibung: Legt die Länge der Register der seriellen Schnittstelle in Bytes fest.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialRegisterStride, definiert in MdeModulePkg.dec.

10. UseHardwareFlowControl

Typ: plist boolean

Failsafe: false

Beschreibung: Aktiviert die Hardware-Flusskontrolle der seriellen Schnittstelle.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialUseHardwareFlowControl, definiert in MdeModulePkg.dec.

S. 56

11. UseMmio

Typ: plist boolean

Failsafe: false

Beschreibung: Gibt an, ob die Register der seriellen Schnittstelle im MMIO-Raum liegen.

Diese Option überschreibt den Wert von gEfiMdeModulePkgTokenSpaceGuid.PcdSerialUseMmio, definiert in Mde-ModulePkg.dec.

8.7 Entry Properties

1. Arguments

Type: plist string

Failsafe: Empty

Beschreibung: Beliebige ASCII-Zeichenkette, die als Boot-Argumente (Ladeoptionen) des angegebenen Eintrags verwendet wird.

2. Auxiliary

Typ: plist boolean

Failsafe: false

Beschreibung: Auf true gesetzt, um diesen Eintrag auszublenden, wenn HideAuxiliary ebenfalls auf true gesetzt ist. Drücken Sie die Leertaste, um in den "erweiterten Modus" zu gelangen und den Eintrag anzuzeigen, wenn er ausgeblendet ist.

3. Comment

Type: plist string

Failsafe: Empty

Beschreibung: Beliebige ASCII-Zeichenkette, die verwendet wird, um eine von Menschen lesbare Referenz für den Eintrag zu liefern. Ob dieser Wert verwendet wird, hängt von der Implementierung ab.

4. Enabled

Type: plist boolean

Failsafe: false

Beschreibung: Auf true gesetzt, wird dieser Eintrag aktiviert.

5. Flavour

Type: plist string

Failsafe: Auto

Beschreibung: Geben Sie die Geschmacksrichtung für diesen Eintrag an. Siehe OC_ATTR_USE_FLAVOUR_ICON Flag für die Dokumentation.

6. Name:

Typ: plist string

Failsafe: Empty

Beschreibung: Von Menschen lesbarer Eintragsname, der in der OpenCore-Auswahl angezeigt wird.

7. Path

Type: plist string

Failsafe: Empty

Beschreibung: Speicherort des Eintrags je nach Eintragstyp.

- Einträge geben externe Boot-Optionen an und nehmen daher Gerätepfade im Schlüssel Path an. Es ist Vorsicht geboten, da diese Werte nicht überprüft werden. Beispiel: PciRoot(0x0)/Pci(0x1,0x1)/.../\EFI\COOL.EFI - Tools geben interne Boot-Optionen an, die Teil des Bootloader-Datenspeichers sind, und nehmen daher Dateipfade

relativ zum OC/Tools-Verzeichnis. Beispiel: OpenShell.efi.

8. RealPath

Typ: Type: plist boolean

Failsafe: false

Beschreibung: Übergibt beim Starten den vollständigen Pfad zum Werkzeug.

Dies sollte normalerweise deaktiviert werden, da die Übergabe des Werkzeugverzeichnisses bei Werkzeugen, die versehentlich versuchen, auf Dateien zuzugreifen, ohne deren Integrität zu prüfen, unsicher sein kann. Ein Grund, diese Eigenschaft zu aktivieren, sind Fälle, in denen Werkzeuge nicht ohne externe Dateien arbeiten können oder diese für erweiterte Funktionen benötigen, wie z. B. memtest86 (für Protokollierung und Konfiguration) oder Shell (für automatische Skriptausführung).

Hinweis: Diese Eigenschaft ist nur für Tools gültig und kann nicht für Einträge angegeben werden (ist immer true).

9. TextMode

Type: plist boolean

Failsafe: false

Beschreibung: Führt den Eintrag im Textmodus anstelle des Grafikmodus aus.

Diese Einstellung kann für einige ältere Werkzeuge, die eine Textausgabe erfordern, von Vorteil sein, da alle Werkzeuge standardmäßig im Grafikmodus gestartet werden. Informationen zu den Textmodi finden Sie im Abschnitt Ausgabeeigenschaften weiter unten.

S. 58

9 NVRAM 9.1 Einleitung

Dieser Abschnitt ermöglicht das Setzen von nichtflüchtigen UEFI-Variablen, die allgemein als NVRAM-Variablen bezeichnet werden. Siehe „man nvram“ für Details. Das macOS-Betriebssystem verwendet in großem Umfang NVRAM-Variablen für die Kommunikation zwischen Betriebssystem, Bootloader und Firmware. Daher ist die Versorgung mit mehreren NVRAM-Variablen für das ordnungsgemäße Funktionieren von macOS erforderlich.

Jede NVRAM-Variable besteht aus ihrem Namen, ihrem Wert, ihren Attributen (siehe UEFI-Spezifikation) und ihrer GUID, die angibt, zu welcher "Sektion" die NVRAM-Variable gehört. Das macOS-Betriebssystem verwendet mehrere GUIDs, einschließlich, aber nicht beschränkt auf:

- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14 (APPLE_VENDOR_VARIABLE_GUID)
- 7C436110-AB2A-4BBB-A880-FE41995C9F82 (APPLE_BOOT_VARIABLE_GUID)
- 5EDDA193-A070-416A-85EB-2A1181F45B18 (Apple Hardware-Konfigurationsspeicher für MacPro7,1)
- 8BE4DF61-93CA-11D2-AA0D-00E098032B8C (EFI_GLOBAL_VARIABLE_GUID)
- 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102 (OC_VENDOR_VARIABLE_GUID)

Hinweis: Einige der Variablen können durch die Unterabschnitte PlatformNVRAM oder Generic des Abschnitts PlatformInfo hinzugefügt werden. Bitte stellen Sie sicher, dass die in diesem Abschnitt gesetzten Variablen nicht mit Elementen in diesen Unterabschnitten kollidieren, da das Implementierungsverhalten ansonsten undefiniert ist.

Die OC_FIRMWARE_RUNTIME-Protokollimplementierung, die derzeit als Teil des OpenRuntime-Treibers angeboten wird, ist häufig erforderlich, damit macOS ordnungsgemäß funktioniert. Obwohl dies viele Vorteile mit sich bringt, gibt es einige Einschränkungen, die für bestimmte Anwendungsfälle berücksichtigt werden sollten.

1. Nicht alle Werkzeuge kennen geschützte Namespaces.

Wenn RequestBootVarRouting verwendet wird, ist der Zugriff auf die Variablen mit Boot-Präfix in einem separaten Namespace eingeschränkt und geschützt. Um auf die ursprünglichen Variablen zugreifen zu können, müssen die Werkzeuge die OC_FIRMWARE_RUNTIME-Logik kennen.

9.2 Properties

1. Add

Type: plist dict

Beschreibung: Setzt NVRAM-Variablen aus einer Map (plist dict) von GUIDs in eine Map (plist dict) von Variablennamen und deren Werten im plist multidata-Format. GUIDs müssen im kanonischen String-Format in Groß- oder Kleinbuchstaben angegeben werden (z.B. 8BE4DF61-93CA-11D2-AA0D-00E098032B8C).

Die Attribute EFI_VARIABLE_BOOTSERVICE_ACCESS und EFI_VARIABLE_RUNTIME_ACCESS der erstellten Variablen werden gesetzt. Variablen werden nur gesetzt, wenn sie nicht vorhanden oder gelöscht sind. Das heißt, um einen vorhandenen Variablenwert zu überschreiben, fügen Sie den Variablennamen in den Abschnitt Löschen ein. Dieser Ansatz ermöglicht die Bereitstellung von Standardwerten, bis das Betriebssystem die Führung übernimmt.

Hinweis: Das Verhalten der Implementierung ist undefiniert, wenn der plist-Schlüssel nicht dem GUID-Format entspricht.

2. Delete

Typ: plist dict

Beschreibung: Entfernt NVRAM-Variablen aus einer Map (plist dict) von GUIDs in ein Array (plist array) von Variablennamen im plist-String-Format.

3. LegacyEnable

Type: plist boolean

Failsafe: false

Beschreibung: Ermöglicht das Laden einer NVRAM-Variablendatei namens nvram.plist vom EFI-Volumenstamm.

Diese Datei muss einen Root-Plist-Dictionary-Typ haben und zwei Felder enthalten: - Version - plist Integer, Dateiversion, muss auf 1 gesetzt werden.

- Add - plist-dictionary, äquivalent zu Add aus config.plist.

Das Laden der Variablen erfolgt vor den Phasen Delete (und Add). Sofern LegacyOverwrite nicht aktiviert ist, werden keine

werden keine bestehenden Variablen überschrieben.

Variablen, die gesetzt werden dürfen, müssen in LegacySchema angegeben werden.

Zum Erstellen der Datei nvram.plist können Skripte von Drittanbietern verwendet werden. Ein Beispiel für ein solches Skript finden Sie in Utilities/LogoutHook. Bei der Verwendung von Skripten von Drittanbietern muss ExposeSensitiveData möglicherweise auf 0x3 gesetzt werden, um die Boot-Pfad-Variable mit der OpenCore EFI-Partitions-UUID bereitzustellen.

Warnung: Diese Funktion kann gefährlich sein, da sie ungeschützte Daten an Firmware-Variablendienste weitergibt. Verwenden Sie sie nur, wenn die Firmware keine Hardware-NVRAM-Implementierung bereitstellt oder wenn die NVRAM-Implementierung nicht kompatibel ist.

4. LegacyOverwrite

Typ: plist boolean

Failsafe: false

Beschreibung: Erlaubt das Überschreiben von Firmware-Variablen aus der Datei nvram.plist.

Hinweis: Nur Variablen, die vom Betriebssystem aus zugänglich sind, werden überschrieben.

5. LegacySchema

Typ: plist dict

Beschreibung: Ermöglicht das Setzen bestimmter NVRAM-Variablen aus einer Map (plist dict) von GUIDs in ein Array (plist array) von Variablennamen im plist-String-Format.

Der *-Wert kann verwendet werden, um alle Variablen für eine bestimmte GUID zu akzeptieren.

WARNUNG: Wählen Sie die Variablen sorgfältig aus, da die Datei nvram.plist nicht gesichert ist. Fügen Sie zum Beispiel nicht boot-args oder csr-active-config hinzu, da diese zur Umgehung von SIP verwendet werden können.

6. WriteFlash

Typ: plist boolean

Failsafe: false

Beschreibung: Aktiviert das Schreiben in den Flash-Speicher für alle hinzugefügten Variablen.

Hinweis: Dieser Wert sollte bei den meisten Firmware-Typen aktiviert sein, ist aber konfigurierbar, um Firmware zu berücksichtigen, die Probleme mit der Garbage Collection von NVRAM-Variablenspeichern o.ä. haben könnte.

Der Befehl nvram kann verwendet werden, um NVRAM-Variablenwerte aus macOS zu lesen, indem die GUID und der Variablenname durch ein : Symbol getrennt aneinandergehängt werden. Zum Beispiel: nvram 7C436110-AB2A-4BBB-A880-FE41995C9F82:boot-args.

Eine laufend aktualisierte Variablenliste ist in einem entsprechenden Dokument zu finden: NVRAM-Variablen.

9.3 Obligatorische Variablen

Warnung: Diese Variablen können durch die PlatformNVRAM- oder Generic-Unterabschnitte des PlatformInfo-Abschnitts hinzugefügt werden. Die Verwendung von PlatformInfo ist der empfohlene Weg, um diese Variablen zu setzen.

Die folgenden Variablen sind für die Funktion von macOS zwingend erforderlich:

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeatures 32-Bit-FirmwareFeatures. Auf allen Macs vorhanden, um zusätzliches Parsen von SMBIOS-Tabellen zu vermeiden.

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeaturesMask 32-Bit-FirmwareFeaturesMask. Auf allen Macs vorhanden, um zusätzliches Parsen von SMBIOS-Tabellen zu vermeiden.

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:MLB BoardSerialNumber. Auf neueren Macs (mindestens 2013+) vorhanden, um zusätzliches Parsen von SMBIOS-Tabellen zu vermeiden, insbesondere in boot.efi.

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ROM MAC-Adresse des primären Netzwerkadapters oder Ersatzwert. Auf neueren Macs (mindestens 2013+) vorhanden, um den Zugriff auf einen speziellen Speicherbereich zu vermeiden, insbesondere in boot.efi.

9.4 Empfohlene Variablen

Die folgenden Variablen werden für einen schnelleren Start oder andere Verbesserungen empfohlen:

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:BridgeOSHardwareModel Variable für das Hardware-Modell des Bridge-Betriebssystems, die von EfiBoot für die Weitergabe an das IODT-Bridge-Modell verwendet wird. Gelesen von hw.target sysctl, verwendet von SoftwareUpdateCoreSupport.

- 7C436110-AB2A-4BBB-A880-FE41995C9F82:csr-active-config
  32-bit System Integrity Protection bitmask. Declared in XNU source code in csr.h.
- 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 ro- tation 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 Andere Variablen

Die folgenden Variablen können für bestimmte Konfigurationen oder zur Fehlersuche nützlich sein:

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:boot-args

Kernel-Argumente, die verwendet werden, um die Konfiguration an den Apple-Kernel und die Treiber zu übergeben. Es gibt viele Argumente, die durch die Suche nach der Verwendung der Funktion PE_parse_boot_argn im Kernel- oder Treibercode gefunden werden können. Einige der bekannten Boot-Argumente enthalten:

– 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)

S. 61

– nvda_drv=1 (legacy way to enable NVIDIA web driver, removed in 10.12) – npci=0x2000 (legacy, disables kIOPCIConfiguratorPFM64)

– lapic_dont_panic=1 (disable lapic spurious interrupt panic on AP cores)

– panic_on_display_hang=1 (trigger panic on display hang)

– panic_on_gpu_hang=1 (trigger panic on GPU hang)

– serial=VALUE (configure serial logging mode) — The following bits are used by XNU:

∗ 0x01 (SERIALMODE_OUTPUT, bit 0) — Enable serial output.

∗ 0x02 (SERIALMODE_INPUT, bit 1) — Enable serial input.

∗ 0x04 (SERIALMODE_SYNCDRAIN, bit 2) — Enable serial drain synchronisation.

∗ 0x08 (SERIALMODE_BASE_TTY, bit 3) — Load Base/Recovery/FVUnlock TTY.

∗ 0x10 (SERIALMODE_NO_IOLOG, bit 4) — Prevent IOLogs writing to serial.

– slide=VALUE (manually set KASLR slide)

– smcdebug=VALUE (AppleSMC debug mask)

– spin_wait_for_gpu=1 (reduces GPU timeout on high load)

– -amd_no_dgpu_accel (alternative to WhateverGreen’s -radvesa for new GPUs) – -nehalem_error_disable (disables the AppleTyMCEDriver)

– -no_compat_check (disable model checking on 10.7+)

– -s (single mode)

– -v (verbose mode)

– -x (safe mode)

Es gibt mehrere externe Stellen, die macOS-Argumentlisten zusammenfassen: Beispiel 1, Beispiel 2.

7C436110-AB2A-4BBB-A880-FE41995C9F82:bootercfg

Booter-Argumente, ähnlich wie boot-args, aber für boot.efi. Akzeptiert eine Reihe von Argumenten, die hexadezimale 64-Bit-Werte mit oder ohne 0x sind. In verschiedenen Stadien fordert boot.efi verschiedene Debugging- (Protokollierungs-) Modi an (z. B. wird nach ExitBootServices nur auf seriellem Weg gedruckt). Mehrere Booter-Argumente steuern, ob diese Anfragen erfolgreich sein werden. Die Liste der bekannten Anforderungen ist unten aufgeführt:

– 0x00 – INIT.

– 0x01 – VERBOSE (e.g. -v, force console logging).

– 0x02 – EXIT.

– 0x03 – RESET:OK.

– 0x04 – RESET:FAIL (e.g. unknown board-id, hibernate mismatch, panic loop, etc.).

– 0x05 – RESET:RECOVERY.

– 0x06 – RECOVERY.

– 0x07 – REAN:START.

– 0x08 – REAN:END.

– 0x09 – DT (can no longer log to DeviceTree).

– 0x0A – EXITBS:START (forced serial only).

– 0x0B – EXITBS:END (forced serial only).

– 0x0C – UNKNOWN.

In 10.15 war die Debugging-Unterstützung bis zum Release 10.15.4 aufgrund von Refactoring-Problemen sowie der Einführung eines neuen Debug-Protokolls defekt. Einige der unten aufgeführten Argumente und ihre Werte sind möglicherweise für Versionen vor 10.15.4 nicht gültig. Die Liste der bekannten Argumente ist unten aufgeführt:

– boot-save-log=VALUE — debug log save mode for normal boot.

∗0

∗1

∗ 2 — (default).

∗3

∗ 4 — (save to file).

– wake-save-log=VALUE — debug log save mode for hibernation wake.

∗ 0 — disabled.

∗1

∗ 2 — (default).

∗ 3 — (unavailable).

∗ 4 — (save to file, unavailable).

– breakpoint=VALUE — enables debug breaks (missing in production boot.efi).

∗ 0 — disables debug breaks on errors (default).

∗ 1 — enables debug breaks on errors.

S. 62

– console=VALUE — enables console logging.

∗ 0 — disables console logging.

∗ 1 — enables console logging when debug protocol is missing (default).

∗ 2 — enables console logging unconditionally (unavailable).

– embed-log-dt=VALUE — enables DeviceTree logging.

∗ 0 — disables DeviceTree logging (default).

∗ 1 — enables DeviceTree logging.

– kc-read-size=VALUE — Chunk size used for buffered I/O from network or disk for prelinkedkernel reading and related. Set to 1MB (0x100000) by default, can be tuned for faster booting.

– log-level=VALUE — log level bitmask.

∗ 0x01 — enables trace logging (default).

– serial=VALUE — enables serial logging.

∗ 0 — disables serial logging (default).

∗ 1 — enables serial logging for EXITBS:END onwards.

∗ 2 — enables serial logging for EXITBS:START onwards.

∗ 3 — enables serial logging when debug protocol is missing.

∗ 4 — enables serial logging unconditionally.

– timestamps=VALUE — enables timestamp logging.

∗ 0 — disables timestamp logging.

∗ 1 — enables timestamp logging (default).

– log=VALUE — deprecated starting from 10.15.

∗ 1 — AppleLoggingConOutOrErrSet/AppleLoggingConOutOrErrPrint (classical ConOut/StdErr)

∗ 2 — AppleLoggingStdErrSet/AppleLoggingStdErrPrint (StdErr or serial?)

∗ 4 — AppleLoggingFileSet/AppleLoggingFilePrint (BOOTER.LOG/BOOTER.OLD file on EFI partition)

– debug=VALUE — deprecated starting from 10.15.

∗ 1 — enables print something to BOOTER.LOG (stripped code implies there may be a crash)

∗ 2 — enables perf logging to /efi/debug-log in the device three

∗ 4 — enables timestamp printing for styled printf calls

– level=VALUE — deprecated starting from 10.15. Verbosity level of DEBUG output.

Hinweis: Aktivieren Sie die Option AppleDebug, um ausführliche Ausgaben von boot.efi auf modernen macOS-Versionen anzuzeigen. Dadurch wird das Protokoll in der allgemeinen OpenCore-Protokolldatei gespeichert. Für Versionen vor 10.15.4, setzen Sie bootercfg auf log=1. Dadurch wird eine ausführliche Ausgabe auf dem Bildschirm ausgegeben.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:bootercfg-once

Booter-Argumente werden nach dem ersten Start entfernt. Ansonsten äquivalent zu bootercfg.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:csr-data

Angabe der Quellen von Kexts, die unabhängig vom SIP CSR_ALLOW_UNAPPROVED_KEXTS-Wert zugelassen werden.

Beispielinhalte: <dict><key>kext-allowed-teams</key><array><string>{DEVELOPER-TEAM-ID}</string></array></dict>%00

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:efiboot-perf-record

Aktiviert das Speichern des Leistungsprotokolls in boot.efi. Das Leistungsprotokoll wird im physischen Speicher gespeichert und wird durch die Variablen efiboot-perf-record-data und efiboot-perf-record-size angezeigt. Ab 10.15.4 kann es auch im OpenCore-Protokoll gespeichert werden, indem die Option AppleDebug gesetzt wird.

- 7C436110-AB2A-4BBB-A880-FE41995C9F82:fmm-computer-name Aktuell gespeicherter Rechnername. ASCII-Zeichenfolge.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:nvda_drv

NVIDIA Web Driver Steuerungsvariable. Nimmt die ASCII-Ziffer 1 oder 0 an, um den installierten Treiber zu aktivieren oder zu deaktivieren.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:run-efi-updater

Überschreibt die EFI-Firmware-Aktualisierungsunterstützung in macOS (MultiUpdater, ThorUtil, usw.). Wenn Sie dies auf Nein oder einen anderen booleschen Wert setzen, werden Firmware-Aktualisierungen in macOS mindestens ab 10.10 verhindert.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:StartupMute

Stummschaltung des Startgongs in der Firmware-Audiounterstützung. 8-Bit-Ganzzahl. Der Wert 0x00 bedeutet nicht stummgeschaltet. Fehlende Variable oder ein anderer Wert bedeutet stummgeschaltet.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:SystemAudioVolume

Systemaudio-Lautstärkepegel für die Audiounterstützung der Firmware. 8-Bit-Ganzzahl ohne Vorzeichen. Das Bit von 0x80 bedeutet stummgeschaltet. Die restlichen Bits werden verwendet, um die rohe Verstärkungseinstellung zu kodieren, die auf den verwendeten Audioverstärker angewendet wird. Was dieser Wert genau bedeutet, hängt vom Codec (und möglicherweise vom spezifischen Verstärker innerhalb des Codecs) ab.

S.63

Dieser Wert wird von macOS auf den Wert des AppleHDA-Layouts MaximumBootBeepVolume begrenzt, um eine zu laute Audiowiedergabe in der Firmware Audiowiedergabe in der Firmware zu vermeiden.

• 7C436110-AB2A-4BBB-A880-FE41995C9F82:SystemAudioVolumeDB

Aktueller System-Audio-Lautstärkepegel in Dezibel (dB). 8-Bit-Ganzzahl mit Vorzeichen. Der Wert stellt den Audio-Offset (Verstärkung, wenn positiv, Dämpfung, wenn negativ) in dB relativ zum Verstärker-Referenzwert von 0 dB dar. Welcher Lautstärkepegel genau durch 0 dB dargestellt wird, hängt vom Codec (und möglicherweise vom spezifischen Verstärker innerhalb des Codecs) ab, liegt aber normalerweise bei oder nahe der maximal verfügbaren Verstärkerlautstärke. Typische Werte dieser Variablen reichen von etwa -60 (der genaue Wert hängt von der Audio-Hardware ab) bis genau 0. Bei untypischer Audio-Hardware kann der Wert über Null gehen.

Hinweis: Im Gegensatz zu SystemAudioVolume ist dieser Wert nicht begrenzt.

• 5EDDA193-A070-416A-85EB-2A1181F45B18:PEXConf

PCI-Erweiterungssteckplatzkonfiguration für MacPro7,1. 8-Byte-Sequenz, die die Standard-PCI-Steckplatzkonfiguration beschreibt. Jedes Byte bezieht sich auf eine Konfiguration für einen bestimmten PCI-Steckplatz.

- Steckplatz 1 befindet sich unter IOService:/AppleACPIPlatformExpert/PC01@0/AppleACPIPCI/BR1A@0 und sein Pfad ist fest codiert. Dieser Steckplatz befindet sich nicht hinter einem Muxer.

- Steckplatz 3 befindet sich unter IOService:/AppleACPIPlatformExpert/PC03@0/AppleACPIPCI/BR3A@0, und sein Pfad ist hartcodiert. Dieser Steckplatz befindet sich nicht hinter einem Muxer.

- Die Steckplätze 2, 4-8 sind dynamisch und werden auf der Grundlage der Eigenschaft AAPL,slot-name mit dem Wert Slot-N abgeglichen, wobei N die Steckplatznummer ist. Alle diese Steckplätze befinden sich hinter einem Muxer.

Weitere Einzelheiten zur Konfiguration der MacPro7,1-Steckplätze finden Sie auf der Support-Seite.

• 5EDDA193-A070-416A-85EB-2A1181F45B18:SlotUtilPEXConf

Benutzer-PCI-Erweiterungssteckplatzkonfiguration für MacPro7,1. 8-Byte-Sequenz zur Beschreibung der Benutzer-PCI-Steckplatzkonfiguration.

10 PlattformInfo

Die Plattforminformationen bestehen aus mehreren Identifikationsfeldern, die generiert oder manuell ausgefüllt werden, um mit den macOS-Diensten kompatibel zu sein. Der Basisteil der Konfiguration kann von AppleModels bezogen werden, das selbst eine Reihe von Schnittstellen auf der Grundlage einer Datenbank im YAML-Format generiert. Diese Felder werden an drei Ziele geschrieben:

- SMBIOS - Data Hub - NVRAM

Die meisten Felder spezifizieren die Überschreibungen in SMBIOS, und ihre Feldnamen entsprechen der EDK2 SmBios.h Header-Datei. Einige wichtige Felder befinden sich jedoch im Data Hub und im NVRAM. Einige der Werte sind in mehr als einem Feld und/oder Ziel zu finden, so dass es zwei Möglichkeiten gibt, ihren Aktualisierungsprozess zu steuern: manuell, wobei alle Werte angegeben werden (die Vorgabe), und halbautomatisch, wobei (automatisch) nur bestimmte Werte angegeben und später für die Systemkonfiguration verwendet werden.

Das Dienstprogramm dmidecode kann verwendet werden, um SMBIOS-Inhalte zu untersuchen. Eine Version mit macOS-spezifischen Erweiterungen kann von Acidanthera/dmidecode heruntergeladen werden.

10.1 Eigenschaften

1. Automatisch

Typ: plist boolean

Failsafe: false

Beschreibung: Erzeugt PlatformInfo basierend auf dem Generic-Abschnitt, anstatt Werte aus den DataHub-, NVRAM- und SMBIOS-Abschnitten zu verwenden.

Das Aktivieren dieser Option ist nützlich, wenn der Generic-Abschnitt flexibel genug ist:

- Wenn aktiviert, sind SMBIOS-, DataHub- und PlatformNVRAM-Daten unbenutzt.

- Wenn deaktiviert, ist der generische Abschnitt unbenutzt.

Warnung: Es wird dringend davon abgeraten, diese Option auf false zu setzen, wenn Sie beabsichtigen, Plattforminformationen zu aktualisieren. Die Einstellung false ist in der Regel nur für kleinere Korrekturen an SMBIOS-Werten auf älterer Apple-Hardware gültig. In allen anderen Fällen kann das Setzen von Automatic auf false zu schwer zu behebenden Fehlern führen, die aus inkonsistenten oder ungültigen Einstellungen resultieren.

2. CustomMemory

Type: plist boolean

Failsafe: false

Beschreibung: Verwendet die im Abschnitt Speicher definierte benutzerdefinierte Speicherkonfiguration. Dies ersetzt vollständig jede bestehende Speicherkonfiguration im SMBIOS und ist nur aktiv, wenn UpdateSMBIOS auf true gesetzt ist.

3. UpdateDataHub

Typ: plist boolean

Failsafe: false

Beschreibung: Aktualisiert DataHub-Felder. Diese Felder werden aus den Abschnitten Generic oder DataHub gelesen, je nach Einstellung der Eigenschaft Automatic.

Hinweis: Die Implementierung des Data-Hub-Protokolls in der EFI-Firmware auf praktisch allen Systemen, einschließlich Apple-Hardware, bedeutet, dass vorhandene Data-Hub-Einträge nicht überschrieben werden können. Stattdessen werden neue Einträge am Ende des Data Hub hinzugefügt, wobei macOS die alten Einträge ignoriert. Dies kann umgangen werden, indem das Data Hub-Protokoll mithilfe des Abschnitts ProtocolOverrides ersetzt wird. Einzelheiten finden Sie in der Beschreibung der protocol override description

4. UpdateNVRAM

Typ: plist boolean

Failsafe: false

Beschreibung: Aktualisiert die NVRAM-Felder, die sich auf die Plattforminformationen beziehen.

Diese Felder werden aus den Abschnitten Generic oder PlatformNVRAM gelesen, je nach Einstellung der Eigenschaft Automatic. Alle anderen Felder müssen mit dem NVRAM-Abschnitt angegeben werden.

Wenn UpdateNVRAM auf false gesetzt ist, können die oben genannten Variablen mit dem NVRAM-Abschnitt aktualisiert werden. Wenn UpdateNVRAM auf true gesetzt ist, ist das Verhalten undefiniert, wenn eines der Felder im NVRAM-Abschnitt vorhanden ist.

S. 65

5. UpdateSMBIOS

Typ: plist boolean

Failsafe: false

Beschreibung: Aktualisiert SMBIOS-Felder. Diese Felder werden aus den Abschnitten Generic oder SMBIOS gelesen, je nach Einstellung der Eigenschaft Automatic.

6. UpdateSMBIOSMode

Typ: plist string

Failsafe: Create

Beschreibung: SMBIOS-Felder aktualisieren Ansatz:

- TryOverwrite - Überschreiben, wenn die neue Größe <= als das spage-aligned original ist und es keine Probleme mit dem Entsperren von Legacy-Regionen gibt. Andernfalls erstellen. Hat Probleme bei einigen Firmware-Typen.

- Create - Ersetzt die Tabellen durch neu zugewiesene EfiReservedMemoryType bei AllocateMaxAddress ohne Fallbacks.

- Overwrite - Überschreibt vorhandene gEfiSmbiosTableGuid und gEfiSmbiosTable3Guid Daten, wenn sie zur neuen Größe passen. Ansonsten Abbruch mit nicht spezifiziertem Status.

- Custom-WriteSMBIOS tables (gEfiSmbios(3)TableGuid) to gOcCustomSmbios(3)TableGuidumgeht das Überschreiben des SMBIOS-Inhalts durch die Firmware bei ExitBootServices. Ansonsten gleichwertig mit Create. Erfordert einen Patch in AppleSmbios.kext und AppleACPIPlatform.kext, um von einer anderen GUID zu lesen: "EB9D2D31" – "EB9D2D35" (in ASCII), was automatisch durch die CustomSMBIOSGuid-quirk geschieht.

Hinweis: Ein Nebeneffekt der Verwendung des benutzerdefinierten Ansatzes ist, dass SMBIOS-Updates exklusiv für macOS durchgeführt werden, wodurch eine Kollision mit bestehender Windows-Aktivierung und benutzerdefinierter OEM-Software vermieden wird, aber möglicherweise der Betrieb von Apple-spezifischen Tools behindert wird.

7. UseRawUuidEncoding

Typ: plist boolean

Failsafe: false

Beschreibung: Rohkodierung für SMBIOS-UUIDs verwenden.

Jede UUID AABBCCDD-EEFF-GGHH-IIJJ-KKLLMMNNOOPP ist im Wesentlichen eine hexadezimale 16-Byte-Zahl. Sie kann auf zwei Arten kodiert werden:

- Big Endian - indem man alle Bytes so schreibt, wie sie sind, ohne die Reihenfolge zu ändern ({AA BB CC DD EE FF GG HH II JJ KK LL MM NN OO PP}). Diese Methode ist auch als RFC 4122-Kodierung oder Raw-Kodierung bekannt.

- Little Endian - durch Interpretation der Bytes als Zahlen und Verwendung der Little Endian Byte-Darstellung ({DD CC BB AA FF EE HH GG II JJ KK LL MM NN OO PP}).

Die SMBIOS-Spezifikation spezifizierte nicht explizit das Kodierungsformat für die UUID bis SMBIOS 2.6, wo es hieß, dass die Little-Endian-Kodierung verwendet werden soll. Dies führte zu Verwirrung sowohl bei Firmware-Implementierungen als auch bei Systemsoftware, da die verschiedenen Hersteller vorher unterschiedliche Kodierungen verwendeten.

- Apple verwendet überall das Big Endian Format, ignoriert aber SMBIOS UUID innerhalb von macOS.

- dmidecode verwendet das Big Endian Format für SMBIOS 2.5.x oder niedriger und das Little Endian Format für 2.6 und neuere Versionen. Acidanthera dmidecode gibt alle drei Formate aus.

- Windows verwendet überall das Little Endian Format, aber das betrifft nur die visuelle Darstellung der Werte.

OpenCore setzt immer eine aktuelle SMBIOS-Version (derzeit 3.2), wenn es die modifizierten DMI-Tabellen erzeugt. Wenn UseRawUuidEncoding aktiviert ist, wird das Big-Endian-Format zum Speichern der SystemUUID-Daten verwendet. Andernfalls wird das Little-Endian-Format verwendet.

Hinweis: Diese Einstellung hat keinen Einfluss auf die in DataHub und NVRAM verwendeten UUIDs, da diese nicht standardisiert sind und von Apple hinzugefügt werden. Im Gegensatz zu SMBIOS werden sie immer im Big-Endian-Format gespeichert.

8. Generic

Type: plist dictionary

Beschreibung: Aktualisiert alle Felder im automatischen Modus.

Hinweis: Dieser Abschnitt wird ignoriert, darf aber nicht entfernt werden, wenn Automatisch auf falsch gesetzt ist.

9. DataHub

Typ: plist dictionary

9. DataHub

Typ: plist dictionary

Beschreibung: Aktualisierung von Data Hub-Feldern im nicht-automatischen Modus.

Hinweis: Dieser Abschnitt wird ignoriert und kann entfernt werden, wenn Automatic ist true.

10. Memory

Type: plist dictionary

Beschreibung: Definieren Sie die benutzerdefinierte Speicherkonfiguration.

Hinweis: Dieser Abschnitt wird ignoriert und kann entfernt werden, wenn CustomMemory false ist.

11. PlattformNVRAM

Typ: plist dictionary

Beschreibung: Aktualisieren der Plattform-NVRAM-Felder im nicht-automatischen Modus.

Hinweis: Dieser Abschnitt wird ignoriert und kann entfernt werden, wenn Automatic ist true.

12. SMBIOS

Typ: plist dictionary

Beschreibung: Aktualisiert SMBIOS-Felder im nicht-automatischen Modus.

Hinweis: Dieser Abschnitt wird ignoriert und kann entfernt werden, wenn Automatic ist true.

10.2 Generic Properties

1. SpoofVendor

Type: plist boolean

Failsafe: false

Beschreibung: Setzt SMBIOS-Herstellerfelder auf Acidanthera. Aus den in der SystemManufacturer-Beschreibung genannten Gründen kann es gefährlich sein, "Apple" in SMBIOS-Anbieterfeldern zu verwenden. Bestimmte Firmware kann jedoch andernfalls keine gültigen Werte liefern, was den Betrieb einiger Software behindern könnte.

2. AdviseFeatures

Typ: plist boolean

Failsafe: false

Beschreibung: Aktualisiert FirmwareFeatures mit unterstützten Bits.

Bits zu FirmwareFeatures hinzugefügt:

- FW_FEATURE_SUPPORTS_CSM_LEGACY_MODE (0x1) - Ohne dieses Bit ist es nicht möglich, Windows neu zu starten, das auf einem Laufwerk mit einer EFI-Partition installiert ist, die nicht die erste Partition auf der Festplatte ist.

- FW_FEATURE_SUPPORTS_UEFI_WINDOWS_BOOT (0x20000000) - Ohne dieses Bit ist es nicht möglich, Windows neu zu starten, das auf einem Laufwerk mit einer EFI-Partition installiert ist, die die erste Partition auf der Festplatte ist.

- FW_FEATURE_SUPPORTS_APFS (0x00080000) - Ohne dieses Bit ist es nicht möglich, macOS auf einer APFS-Festplatte zu installieren.

- FW_FEATURE_SUPPORTS_LARGE_BASESYSTEM (0x800000000) - Ohne dieses Bit ist es nicht möglich, macOS-Versionen mit großen BaseSystem-Images zu installieren, wie z. B. macOS 12.

Hinweis: Bei den meisten neueren Firmwares sind diese Bits bereits gesetzt, die Option kann notwendig sein, wenn man die Firmware mit neuen Funktionen "aufrüstet".

3. MaxBIOSVersion

Typ: plist boolean

Failsafe: false

Beschreibung: Setzt BIOSVersion auf 9999.999.999.999.999, empfohlen für ältere Macs bei Verwendung von Automatic PlatformInfo, um BIOS-Updates in nicht offiziell unterstützten macOS-Versionen zu vermeiden.

4. SystemMemoryStatus

Typ: plist string

Failsafe: Auto

Beschreibung: Gibt an, ob der Systemspeicher in PlatformFeature aufrüstbar ist. Dies steuert die Sichtbarkeit der Registerkarte Speicher in "Über diesen Mac".

Gültige Werte:

- Auto - verwendet den ursprünglichen PlatformFeature-Wert.

- Upgradable - explicitly unset

- PT_FEATURE_HAS_SOLDERED_SYSTEM_MEMORY (0x2) in PlatformFeature explizit zurücksetzen. - Gelötet - setzen Sie explizit PT_FEATURE_HAS_SOLDERED_SYSTEM_MEMORY (0x2) in PlatformFeature.

Hinweis: Auf bestimmten Mac-Modellen, wie dem MacBookPro10,x und jedem MacBookAir, ignoriert SPMemoryReporter.spreporter

PT_FEATURE_HAS_SOLDERED_SYSTEM_MEMORY und nimmt an, dass der Systemspeicher nicht aufrüstbar ist.

5. ProzessorTyp

Typ: plist integer

Failsafe: 0 (Automatic)

Beschreibung: Siehe SMBIOS ProcessorType.

6. SystemProduktname

Typ: plist string

Failsafe: Empty (OEM angegeben oder nicht installiert) Beschreibung: Siehe SMBIOS SystemProductName.

7. SystemSerialNumber

Typ: plist-String

Ausfallsicher: Empty (OEM angegeben oder nicht installiert) Beschreibung: Siehe SMBIOS SystemSerialNumber.

Geben Sie den speziellen String-Wert OEM an, um den aktuellen Wert aus dem NVRAM (SSN-Variable) oder dem SMBIOS zu extrahieren und ihn in den Abschnitten zu verwenden. Diese Funktion kann nur bei Mac-kompatibler Firmware verwendet werden.

8. SystemUUID

Typ: plist string, GUID

Failsafe: Empty (OEM angegeben oder nicht installiert) Beschreibung: Siehe SMBIOS SystemUUID.

Spezifizieren Sie den speziellen String-Wert OEM, um den aktuellen Wert aus dem NVRAM (system-id-Variable) oder SMBIOS zu extrahieren und in den Abschnitten zu verwenden. Da nicht jede Firmware-Implementierung über gültige (und eindeutige) Werte verfügt, ist diese Funktion auf einige Konfigurationen nicht anwendbar und kann unerwartete Ergebnisse liefern. Es wird dringend empfohlen, die UUID explizit anzugeben. Siehe UseRawUuidEncoding, um zu bestimmen, wie der SMBIOS-Wert geparst wird.

9. MLB

Typ: plist string

Failsafe: Empty (OEM angegeben oder nicht installiert) Beschreibung: Siehe SMBIOS BoardSerialNumber.

Geben Sie den speziellen String-Wert OEM an, um den aktuellen Wert aus dem NVRAM (MLB-Variable) oder dem SMBIOS zu extrahieren und ihn in den Abschnitten zu verwenden. Diese Funktion kann nur bei Mac-kompatibler Firmware verwendet werden.

10. ROM

Typ: plist multidata, 6 bytes

Failsafe: Empty (OEM angegeben oder nicht installiert)

Beschreibung: Siehe 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ROM.

Geben Sie den speziellen String-Wert OEM an, um den aktuellen Wert aus dem NVRAM (ROM-Variable) zu extrahieren und ihn in den Abschnitten zu verwenden. Diese Funktion kann nur bei Mac-kompatibler Firmware verwendet werden.

10.3 DataHub Eigenschaften

1. Plattformname

Typ: pplist string

Failsafe: Empty (Nicht installiert)

Beschreibung: Setzt den Namen in gEfiMiscSubClassGuid. Der auf Macs gefundene Wert ist platform in ASCII.

2. SystemProduktName

Typ: plist string

Failsafe: Empty (Nicht installiert)

S. 68

Beschreibung: Setzt Model in gEfiMiscSubClassGuid. Der auf Macs gefundene Wert ist gleich SMBIOS SystemProductName in Unicode.

3. SystemSerialNumber

Typ: plist string

Failsafe: Empty (Not installed)

Beschreibung: Setzt die SystemSerialNumber in gEfiMiscSubClassGuid. Der auf Macs gefundene Wert ist gleich der SMBIOS SystemSerialNumber in Unicode.

4. SystemUUID

Typ: pplist string, GUID

Failsafe: Empty (Not installed)

Beschreibung: Setzt die System-ID in gEfiMiscSubClassGuid. Der auf Macs gefundene Wert ist gleich der SMBIOS SystemUUID (mit vertauschter Byte-Reihenfolge).

5. BoardProdukt

Typ:plist string

Failsafe: Empty (Not installed)

Beschreibung: Setzt die Board-Id in gEfiMiscSubClassGuid. Der auf Macs gefundene Wert ist gleich SMBIOS BoardProduct in ASCII.

6. BoardRevision

Typ: plist data, 1 byte

Failsafe: 0

Beschreibung: Setzt board-rev in gEfiMiscSubClassGuid. Der auf Macs gefundene Wert scheint der internen Board-Revision zu entsprechen (z.B. 01).

7. StartupPowerEvents

Typ: plist integer, 64-bit

Failsafe: 0

Beschreibung: Setzt StartupPowerEvents in gEfiMiscSubClassGuid. Der auf Macs gefundene Wert ist die Bitmaske für den Energieverwaltungsstatus, normalerweise 0. Bekannte Bits werden von X86PlatformPlugin.kext gelesen:

- 0x00000001 - Abschaltursache war ein PWROK-Ereignis (entspricht GEN_PMCON_2 Bit 0)

- 0x00000002 - Abschaltursache war ein SYS_PWROK-Ereignis (wie GEN_PMCON_2 Bit 1)

- 0x00000004 - Ursache für das Herunterfahren war ein THRMTRIP#-Ereignis (wie GEN_PMCON_2 Bit 3)

- 0x00000008 - Neustart aufgrund eines SYS_RESET#-Ereignisses (wie GEN_PMCON_2 Bit 4)

- 0x00000010 - Stromausfall (wie GEN_PMCON_3 Bit 1 PWR_FLR)

- 0x00000020 - Verlust der RTC Well Power (entspricht GEN_PMCON_3 Bit 2 RTC_PWR_STS)

- 0x00000040 - Allgemeiner Reset-Status (wie GEN_PMCON_3 Bit 9 GEN_RST_STS)

- 0xffffff80 - SUS Well Power Loss (Wie GEN_PMCON_3 Bit 14)

- 0x00010000 - Wake-Ursache war ein ME-Wake-Ereignis (wie PRSTS Bit 0, ME_WAKE_STS)

- 0x00020000 - Cold Reboot war ein ME-induziertes Ereignis (wie PRSTS Bit 1 ME_HRST_COLD_STS) - 0x00040000 - Warm Reboot war ein ME-induziertes Ereignis (wie PRSTS Bit 2 ME_HRST_WARM_STS) - 0x00080000 - Shutdown war ein ME-induziertes Ereignis (wie PRSTS Bit 3 ME_HOST_PWRDN)

- 0x00100000 - Globales ME-Watchdog-Timer-Ereignis zurückgesetzt (wie PRSTS-Bit 6)

- 0x00200000 - Globales Zurücksetzen des PowerManagement-Watchdog-Timer-Ereignisses (wie PRSTS-Bit 15)

8. InitialTSC

Typ: plist integer, 64-bit

Failsafe: 0

Beschreibung: Setzt InitialTSC in gEfiProcessorSubClassGuid. Setzt den anfänglichen TSC-Wert, normalerweise 0.

9. FSBFrequenz

Typ: plist integer, 64-bit

Failsafe: 0 (Automatic)

Beschreibung: Legt die FSBFrequenz in gEfiProcessorSubClassGuid fest.

Legt die CPU-FSB-Frequenz fest. Dieser Wert ist gleich der CPU-Nominalfrequenz geteilt durch das maximale Busverhältnis der CPU und wird in Hz angegeben. Siehe MSR_NEHALEM_PLATFORM_INFO (CEh) MSR-Wert zur Bestimmung des maximalen Busverhältnisses bei modernen Intel-CPUs.

Hinweis: Dieser Wert wird bei Skylake und neueren Versionen nicht verwendet, ist aber dennoch vorhanden, um den Anforderungen zu entsprechen.

10. ARTFrequency

Typ: plist integer, 64-bit

Failsafe: 0 (Automatic)

Beschreibung: Setzt ARTFrequency in gEfiProcessorSubClassGuid.

Dieser Wert enthält die CPU-ART-Frequenz, auch bekannt als Quarztaktfrequenz. Er existiert nur bei der Skylake-Generation und neueren Versionen. Der Wert wird in Hz angegeben und beträgt normalerweise 24 MHz für das Client-Intel-Segment, 25 MHz für das Server-Intel-Segment und 19,2 MHz für Intel-Atom-CPUs. macOS bis einschließlich 10.15 geht standardmäßig von 24 MHz aus.

Hinweis: Auf Intel Skylake X kann die ART-Frequenz aufgrund einer speziellen EMI-Reduktionsschaltung, wie im Acidanthera Bugtracker beschrieben, etwas weniger (ca. 0,25%) als 24 oder 25 MHz betragen.

11. DevicePathsSupported

Typ: plist integer, 32-bit

Failsafe: 0 (Not installed)

Beschreibung: Setzt DevicePathsSupported in gEfiMiscSubClassGuid. Muss auf 1 gesetzt werden, damit AppleACPIPlat- form.kext SATA-Gerätepfade an die Variablen Boot#### und efi-boot-device-data anhängt. Auf allen modernen Macs auf 1 gesetzt.

12. SmcRevision

Typ: plist-Daten, 6 Bytes

Ausfallsicher: Empty (Nicht installiert)

Beschreibung: Setzt REV in gEfiMiscSubClassGuid. Benutzerdefinierte Eigenschaft, die von VirtualSMC oder FakeSMC gelesen wird, um den SMC REV-Schlüssel zu erzeugen.

13. SmcBranch

Typ: plist data, 6 bytes

Failsafe: Empty (Not installed)

Beschreibung: Setzt RBr in gEfiMiscSubClassGuid. Benutzerdefinierte Eigenschaft, die von VirtualSMC oder FakeSMC gelesen wird, um SMC RBr-Schlüssel zu erzeugen.

14. SmcPlatform

Typ: plist data, 8 bytes

Failsafe: Empty (Not installed)

Beschreibung: Setzt RPlt in gEfiMiscSubClassGuid. Benutzerdefinierte Eigenschaft, die von VirtualSMC oder FakeSMC gelesen wird, um den SMC RPlt-Schlüssel zu erzeugen.

10.4 Memory Properties

1. DataWidth

Type: plist integer, 16-bit

Failsafe: 0xFFFF (unknown)

SMBIOS: Speichergerät (Typ 17) - Datenbreite

Beschreibung: Gibt die Datenbreite des Speichers in Bits an. Eine DataWidth von 0 und eine TotalWidth von 8 zeigt an, dass das Gerät nur zur Bereitstellung von 8 Fehlerkorrekturbits verwendet wird.

2. Geräte

Typ: plist array

Failsafe: Empty

Beschreibung: Gibt die hinzuzufügenden benutzerdefinierten Speichergeräte an.

Wird mit plist-dictionary values gefüllt, die jedes Speichergerät beschreiben. Siehe den Abschnitt Eigenschaften von Speichergeräten weiter unten. Dies sollte alle Speichersteckplätze umfassen, auch wenn sie nicht gefüllt sind.

3. ErrorCorrection

Type: plist integer, 8-bit

Failsafe: 0x03

SMBIOS: Physikalisches Speicher-Array (Typ 16) - Speicher-Fehlerkorrektur

Beschreibung: Gibt die primäre Hardware-Fehlerkorrektur oder -Erkennungsmethode an, die vom Speicher unterstützt wird.

S. 70

• 0x01 — Other

• 0x02 — Unknown

• 0x03 — None

• 0x04 — Parity

• 0x05 — Single-bit ECC • 0x06 — Multi-bit ECC • 0x07 — CRC

4. FormFactor

Typ: plist integer, 8-bit

Failsafe: 0x02

SMBIOS: Memory Device (Typ 17) - Formfaktor

Beschreibung: Gibt den Formfaktor des Speichers an. Bei Macs sollte dies normalerweise DIMM oder SODIMM sein. Häufig verwendete Formfaktoren sind unten aufgeführt.

Wenn CustomMemory false ist, wird dieser Wert automatisch anhand des Mac-Produktnamens festgelegt.

Wenn Automatic true ist, wird der ursprüngliche Wert des entsprechenden Mac-Modells gesetzt, falls verfügbar. Andernfalls wird der Wert aus der OcMacInfoLib gesetzt. Wenn Automatisch false ist, wird ein benutzerspezifischer Wert gesetzt, falls verfügbar. Andernfalls wird der ursprüngliche Wert aus der Firmware gesetzt. Wenn kein Wert angegeben wird, wird der Failsafe-Wert gesetzt.

- 0x01 - Other

- 0x02 - Unbekannt

- 0x09 - DIMM

- 0x0D - SODIMM

- 0x0F - FB-DIMM

5. MaxCapacity

Typ: plist integer, 64-bit

Failsafe: 0

SMBIOS: Physikalisches Memory Array (Typ 16) - Maximale Kapazität

Beschreibung: Gibt die maximale Menge an Speicher in Bytes an, die vom System unterstützt wird.

6. TotalWidth

Typ: plist integer, 16-bit

Failsafe: 0xFFFF (unknown)

SMBIOS: Speichergerät (Typ 17) - Gesamtbreite

Beschreibung: Gibt die Gesamtbreite des Speichers in Bits an, einschließlich aller Prüf- oder Fehlerkorrekturbits. Wenn keine Fehlerkorrekturbits vorhanden sind, sollte dieser Wert gleich DataWidth sein.

7. Typ

Typ: plist integer, 8-bit

Failsafe: 0x02

SMBIOS: Memory Device (Typ 17) - Speichertyp

Beschreibung: Gibt den Speichertyp an. Häufig verwendete Typen sind unten aufgeführt.

- 0x01 - Andere

- 0x02 - Unbekannt

- 0x0F - SDRAM

- 0x12 - DDR

- 0x13 - DDR2

- 0x14 - DDR2 FB-DIMM - 0x18 - DDR3

- 0x1A - DDR4

- 0x1B - LPDDR

- 0x1C - LPDDR2

- 0x1D - LPDDR3

- 0x1E - LPDDR4

8. TypDetail

Typ: plist integer, 16-bit

8. TypDetail

Typ: plist integer, 16-bit

Failsafe: 0x4

SMBIOS: Memory Device (Typ 17) - Typ Detail

Beschreibung: Gibt zusätzliche Informationen zum Speichertyp an.

• Bit 0 — Reserved, set to 0

• Bit 1 — Other

• Bit 2 — Unknown

• Bit 7 — Synchronous

• Bit 13 — Registered (buffered)

• Bit 14 — Unbuffered (unregistered)

10.4.1 Memory Device Properties

1. AssetTag

Typ: plist string

Failsafe: Unknown

SMBIOS: Memory Device (Typ 17) - Asset-Tag Beschreibung: Gibt das Asset-Tag dieses Speichergeräts an.

2. BankLocator

Typ: plist string

Failsafe: Unknown

SMBIOS: Memory Device (Typ 17) - Bank Locator

Beschreibung: Gibt die physikalisch gekennzeichnete Bank an, in der sich das Memory Device befindet.

3. DeviceLocator

Typ: plist string

Failsafe: Unknown

SMBIOS: Memory Device (Typ 17) - Gerätelocator

Beschreibung: Gibt den physikalisch gekennzeichneten Sockel oder die Platinenposition an, an der sich das Memory Device befindet.

4. Manufacturer

Type: plist string

Failsafe: Unknown

SMBIOS: Memory Device (Typ 17) - Hersteller Beschreibung: Gibt den Hersteller dieses Memory Device an.

Für leere Steckplätze muss dies auf NO DIMM gesetzt werden, damit der macOS System Profiler die Speichersteckplätze auf bestimmten Mac-Modellen, z.B. MacPro7,1, korrekt anzeigt. MacPro7,1 stellt zusätzliche Anforderungen an das Speicherlayout:

- Die Anzahl der installierten Sticks muss eine der folgenden sein: 4, 6, 8, 10, 12. Die Verwendung eines anderen Wertes führt zu einem Fehler im System Profiler.

- Die Anzahl der Speichersteckplätze muss gleich 12 sein. Die Verwendung eines anderen Wertes führt zu einem Fehler im System Profiler.

- Speichersticks müssen in dedizierten Speichersteckplätzen installiert werden, wie auf der Support-Seite beschrieben. SMBIOS-Speichergeräte sind den folgenden Steckplätzen zugeordnet: 8, 7, 10, 9, 12, 11, 5, 6, 3, 4, 1, 2.

5. PartNumber

Type: plist string

Failsafe: Unknown

SMBIOS: Memory Device (Typ 17) - Teilenummer Beschreibung: Gibt die Teilenummer dieses Memory Device an.

6. SerienNummer

Typ: plist string

Failsafe: Unknown

SMBIOS: Memory Device (Typ 17) - Seriennummer Beschreibung: Gibt die Seriennummer dieses Memory Device an.

7. Größe

Typ: plist integer, 32-bit Failsafe: 0

S. 72

SMBIOS: Memory Device (Typ 17) - Größe

Beschreibung: Gibt die Größe des Memory Device in Megabyte an. 0 bedeutet, dass dieser Steckplatz nicht belegt ist.

8. Geschwindigkeit

Typ: Plist-Ganzzahl, 16-Bit

Ausfallsicher: 0

SMBIOS: Memory Device (Typ 17) - Geschwindigkeit

Beschreibung: Gibt die maximal mögliche Geschwindigkeit des Memory Device in Megatransfers pro Sekunde (MT/s) an. 0 bedeutet eine unbekannte Geschwindigkeit.

10.5 PlatformNVRAM-Eigenschaften

1. BID

Typ: plist string

Failsafe: Empty (Not installed)

Beschreibung: Gibt den Wert der NVRAM-Variablen 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_BID an.

2. ROM

Typ: plist data, 6 bytes

Failsafe: Empty (Not installed)

Beschreibung: Gibt die Werte der NVRAM-Variablen 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_ROM und 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ROM an.

3. MLB

Typ: plist string

Failsafe: Empty (Not installed)

Beschreibung: Gibt die Werte der NVRAM-Variablen 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_MLB und 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:MLB an.

4. FirmwareFeatures

Typ: plist data, 8 bytes

Failsafe: Empty (Not installed)

Beschreibung: Diese Variable kommt im Paar mit FirmwareFeaturesMask. Gibt die Werte der NVRAM-Variablen an:

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeatures

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeatures

5. FirmwareFeaturesMask

Typ: plist data, 8 bytes

Failsafe: Empty (Not installed)

Beschreibung: Diese Variable kommt im Paar mit FirmwareFeatures. Gibt die Werte der NVRAM-Variablen an:

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeaturesMask

• 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeaturesMask

6. SystemSerialNumber

Typ: plist string

Failsafe: Empty (Not installed)

Beschreibung: Gibt die Werte der NVRAM-Variablen 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:HW_SSN und 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:SSN an.

7. SystemUUID

Typ: plist string

Failsafe: Empty (Not installed)

Beschreibung: Gibt den Wert der NVRAM-Variable 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:system-id nur für Bootdienste an. Der auf Macs gefundene Wert entspricht der SMBIOS SystemUUID.

10.6 SMBIOS Properties

1. BIOSVendor

Type: plist string

Failsafe: Empty (OEM specified)

S. 73

SMBIOS: BIOS Information (Type 0) — Vendor

Beschreibung: BIOS-Hersteller. Es gelten alle Regeln des SystemManufacturer.

2. BIOSVersion

Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: BIOS-Informationen (Typ 0) - BIOS-Version

Beschreibung: Firmware-Version. Dieser Wert wird aktualisiert und nimmt an der Konfiguration der Update-Auslieferung und der Kompatibilität mit der macOS-Version teil. Dieser Wert könnte wie MM71.88Z.0234.B00.1809171422 in älterer Firmware aussehen und ist in BiosId.h beschrieben. In neuerer Firmware sollte er wie 236.0.0.0.0 oder 220.230.16.0.0 aussehen (iBridge: 16.16.2542.0.0,0). iBridge-Version wird aus der BridgeOSVersion-Variable gelesen und ist nur auf Macs mit T2 vorhanden.

Apple ROM-Version

BIOS-ID: MBP151.88Z.F000.B00.1811142212

Modell: MBP151

EFI-Version: 220.230.16.0.0

Gebaut von:

Datum:

Revision:

ROM-Version: F000_B00

Build-Typ:

Compiler:

UUID:

UUID:

Offizieller Build, RELEASE

Apple LLVM Version 10.0.0 (clang-1000.2.42)

E5D1475B-29FF-32BA-8552-682622BA42E1

151B0907-10F9-3271-87CD-4BF5DBECACF5

root@quinoa

Wed Nov 14 22:12:53 2018

220.230.16 (B&I)

3. BIOSReleaseDate

Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: BIOS-Informationen (Typ 0) - BIOS-Freigabedatum

Beschreibung: Datum der Veröffentlichung der Firmware. Ähnlich wie BIOSVersion. Kann wie 12/08/2017 aussehen.

4. SSystemManufacturer

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Systeminformationen (Typ 1) - Hersteller

Beschreibung: Manufacturer

Description: OEM manufacturer of the particular board. Verwenden Sie Failsafe, wenn dies nicht unbedingt erforderlich ist. Überschreiben Sie nicht, um Apple Inc. auf Nicht-Apple-Hardware zu enthalten, da dies zahlreiche Dienste im Betriebssystem durcheinanderbringt, wie z.B. Firmware-Updates, eficheck, sowie in Acidanthera entwickelte Kernel-Erweiterungen, wie Lilu und seine Plugins. Außerdem werden dadurch einige Betriebssysteme wie Linux unbootbar.

5. SystemProduktname

Typ: list string

Failsafe: Empty (OEM specified)

SMBIOS: Systeminformationen (Typ 1), Produktname

Beschreibung: Bevorzugtes Mac-Modell, das verwendet wird, um das Gerät als vom Betriebssystem unterstützt zu kennzeichnen. Dieser Wert muss bei jeder Konfiguration angegeben werden, damit später automatisch die entsprechenden Werte in dieser und anderen SMBIOS-Tabellen und damit verbundenen Konfigurationsparametern generiert werden. Wenn SystemProductName nicht mit dem Zielbetriebssystem kompatibel ist, kann das Boot-Argument -no_compat_check als Override verwendet werden.

Hinweis: Wenn SystemProductName nicht bekannt ist und die zugehörigen Felder nicht spezifiziert sind, sollte davon ausgegangen werden, dass die Standardwerte auf MacPro6,1-Daten gesetzt sind. Die Liste der bekannten Produkte kann in AppleModels gefunden werden.

6. SystemVersion

Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Systeminformationen (Typ 1) - Version

Beschreibung: Versionsnummer der Produkt-Iteration. Kann wie 1.1 aussehen.

7. SystemSerialNumber Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Systeminformationen (Typ 1) - Seriennummer

Beschreibung: Seriennummer des Produkts in einem bestimmten Format. Bekannte Formate sind in macserial beschrieben.

8. SystemUUID

Typ: plist string, GUID

Failsafe: Empty (OEM specified)

SMBIOS: Systeminformationen (Typ 1) - UUID

Beschreibung: Eine UUID ist ein Bezeichner, der so konzipiert ist, dass er sowohl zeitlich als auch räumlich eindeutig ist. Sie erfordert keinen zentralen Registrierungsprozess.

9. SystemSKUNumber

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Systeminformationen (Typ 1) - SKU-Nummer

Beschreibung: Mac-Platinen-ID (board-id). Kann bei älteren Modellen wie Mac-7BA5B2D9E42DDD94 oder Mac-F221BEC8 aussehen. Manchmal kann sie auch einfach leer sein.

10. SystemFamily

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Systeminformation (Typ 1) - Family Description: Name der Familie. Kann wie iMac Pro aussehen.

11. BoardManufacturer

Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Baseboard (oder Modul) Information (Typ 2) - Hersteller Beschreibung: Hersteller der Platine. Es gelten alle Regeln von SystemManufacturer.

12. BoardProdukt

Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Grundplatine (oder Modul) Informationen (Typ 2) - Produkt

Beschreibung: Mac-Platinen-ID (board-id). Kann wie Mac-7BA5B2D9E42DDD94 oder Mac-F221BEC8 in älteren Modellen aussehen.

13. BoardVersion

plist string

Failsafe: Empty (OEM specified)

SMBIOS: Grundplatinen- (oder Modul-) Informationen (Typ 2) - Version

Beschreibung: Versionsnummer der Platine. Variiert, kann mit SystemProductName oder SystemProductVersion übereinstimmen.

14. BoardSerialNumber

Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Grundplatinen- (oder Modul-) Informationen (Typ 2) - Seriennummer

Beschreibung: Seriennummer des Boards in einem bestimmten Format. Bekannte Formate sind in macserial beschrieben.

15. BoardAssetTag

Typ: pplist string

Failsafe: Empty (OEM specified)

SMBIOS: Grundplatinen- (oder Modul-) Informationen (Typ 2) - Asset-Tag

Beschreibung: Asset-Tag-Nummer. Variiert, kann leer oder Typ2 - Board Asset Tag sein.

16. BoardType

Type: plist integer

Failsafe: 0 (OEM specified)

SMBIOS: Grundplatinen- (oder Modul-) Informationen (Typ 2) - Platinentyp

Beschreibung: Entweder 0xA ((includes processor, memory, and I/O) oder 0xB (Processor/Memory Modulel). Einzelheiten finden Sie in Tabelle 15 - Baseboard: Board Type für Einzelheiten.

17. BoardLocationInChassis

Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Baseboard (oder Modul) Information (Typ 2) - Position im Gehäuse
Beschreibung: Variiert, kann leer oder Teilkomponente sein.

18. ChassisManufacturer

Type: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Enclosure oder Chassis (Typ 3) - Hersteller Beschreibung: Hersteller der Platine. Es gelten alle Regeln von SystemManufacturer.

19. ChassisType

Type: plist integer

Failsafe: 0 (OEM specified)

SMBIOS: Systemgehäuse oder Chassis (Typ 3) - Typ

Beschreibung: Gehäusetyp. Einzelheiten finden Sie in Tabelle 17 - Systemgehäuse- oder Chassistypen.

20. ChassisVersion

Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: System Enclosure oder Chassis (Typ 3) - Version Beschreibung: Sollte mit BoardProduct übereinstimmen.

21. ChassisSerialNumber

Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Systemgehäuse oder Chassis (Typ 3) - Version Beschreibung: Sollte mit der SystemSerialNumber übereinstimmen.

22. ChassisAssetTag

Typ: plist string

Failsafe: Empty (OEM specified)

SMBIOS: Systemgehäuse oder Chassis (Typ 3) - Asset-Tag-Nummer Beschreibung: Name des Gehäusetyps. Variiert, kann leer oder MacBook-Aluminium sein.

23. PlattformFeature

Typ: plist integer, 32-bit

Failsafe: 0xFFFFFFFF (OEM specified on Apple hardware, do not provide the table otherwise) SMBIOS: APPLE_SMBIOS_TABLE_TYPE133 - PlatformFeature

Beschreibung: Bitmaske für Plattformfunktionen (fehlt bei älteren Macs). Siehe AppleFeatures.h für Details.

24. SmcVersion

Typ: plist data, 16 bytes

Failsafe: All zero (OEM specified on Apple hardware, do not provide the table otherwise) SMBIOS: APPLE_SMBIOS_TABLE_TYPE134 - Version

Beschreibung: ASCII-String, der die SMC-Version in Großbuchstaben enthält. Fehlt bei T2-basierten Macs.

25. FirmwareFeatures

Typ: plist data, 8 bytes

Failsafe: 0 (OEM specified on Apple hardware, 0 otherwise)

SMBIOS: APPLE_SMBIOS_TABLE_TYPE128 - FirmwareFeatures und ExtendedFirmwareFeatures Beschreibung: Bitmaske für 64-Bit-Firmware-Funktionen. Siehe AppleFeatures.h für Details. Die unteren 32 Bits entsprechen FirmwareFeatures. Die oberen 64 Bits entsprechen ExtendedFirmwareFeatures.

26. FirmwareFeaturesMask

Typ: plist data, 8 bytes

Failsafe: 0 (OEM specified on Apple hardware, 0 otherwise)

SMBIOS: APPLE_SMBIOS_TABLE_TYPE128 - FirmwareFeaturesMask und ExtendedFirmwareFeaturesMask

Beschreibung: Unterstützte Bits der Bitmaske für erweiterte Firmware-Funktionen. Siehe AppleFeatures.h für Details. Die unteren 32 Bits entsprechen der FirmwareFeaturesMask. Die oberen 64 Bits entsprechen der ExtendedFirmwareFeaturesMask

S. 76

27. ProzessorTyp

Typ: plist integer, 16-bit

Failsafe: 0 (Automatic)

SMBIOS: APPLE_SMBIOS_TABLE_TYPE131 - ProcessorType

Beschreibung: Kombiniert die Prozessortypen Major und Minor.

Die automatische Wertgenerierung versucht, den genauesten Wert für die aktuell installierte CPU zu ermitteln. Wenn dies fehlschlägt, bitte einen Fehler melden und sysctl machdep.cpu und dmidecode-Ausgabe bereitstellen. Eine vollständige Liste der verfügbaren Werte und ihrer Einschränkungen (der Wert gilt nur, wenn die Anzahl der CPU-Kerne übereinstimmt) finden Sie hier im Header der Apple SMBIOS-Definitionen.

S.77

11 UEFI

11.1 Einführung

UEFI (Unified Extensible Firmware Interface) ist eine Spezifikation, die eine Software-Schnittstelle zwischen einem Betriebssystem und der Plattform-Firmware definiert. Dieser Abschnitt ermöglicht das Laden zusätzlicher UEFI-Module sowie die Durchführung von Änderungen an der Onboard-Firmware. Um den Inhalt der Firmware zu überprüfen, Änderungen vorzunehmen und Upgrades durchzuführen, können UEFITool und zusätzliche Dienstprogramme verwendet werden.

11.2 Treiber

Je nach Firmware kann ein anderer Satz von Treibern erforderlich sein. Das Laden eines inkompatiblen Treibers kann dazu führen, dass das System nicht mehr gestartet werden kann oder sogar die Firmware dauerhaft beschädigt wird. Einige der bekannten Treiber sind im Folgenden aufgeführt:

AudioDxe*

HDA-Audio-Unterstützungstreiber in der UEFI-Firmware für die meisten Intel- und einige andere analoge Audiocontroller. Staging-Treiber, siehe acidanthera/bugtracker#740 für bekannte Probleme in AudioDxe. Open-Source-BTRFS-Dateisystemtreiber, der für das Booten mit OpenLinuxBoot von einem Dateisystem benötigt wird, das nun recht häufig unter Linux verwendet wird.

btrfs_x64 BiosVideo*

CSM-Videotreiber, der ein Grafikausgabeprotokoll implementiert, das auf VESA- und Legacy-BIOS-Schnittstellen basiert. Wird für UEFI-Firmware mit fragiler GOP-Unterstützung (z.B. niedrige Auflösung) verwendet. Erfordert ReconnectGraphicsOnConnect. In OpenDuet von Haus aus enthalten.

CrScreenshotDxe*

Screenshot-Treiber, der beim Drücken von F10 Bilder im Wurzelverzeichnis der OpenCore-Partition (ESP) oder in jedem anderen verfügbaren beschreibbaren Dateisystem speichert. Dies ist eine modifizierte Version des CrScreenshotDxe-Treibers von Nikolaj Schlej

ExFatDxe

Proprietärer ExFAT-Dateisystemtreiber für Bootcamp-Unterstützung, der häufig in Apple-Firmware zu finden ist. Für Sandy Bridge und frühere CPUs sollte der ExFatDxeLegacy-Treiber verwendet werden, da er keine RDRAND-Befehle unterstützt.

ext4_x64

Open-Source-Treiber für das EXT4-Dateisystem, erforderlich für das Booten mit OpenLinuxBoot von dem unter Linux am häufigsten verwendeten Dateisystem.

HfsPlus

Empfohlen. Proprietärer HFS-Dateisystemtreiber mit Bless-Unterstützung, der häufig in Apple-Firmware zu finden ist. Für Sandy Bridge und frühere CPUs sollte der HfsPlusLegacy-Treiber verwendet werden, da er keine RDRAND-Befehle unterstützt.

HiiDatabase*

HII-Dienste unterstützen den Treiber von MdeModulePkg. Dieser Treiber ist in den meisten Firmware-Typen ab der Ivy Bridge-Generation enthalten. Einige Anwendungen mit GUI, wie z.B. UEFI Shell, benötigen diesen Treiber, um richtig zu funktionieren.

EnhancedFatDxe

FAT-Dateisystemtreiber von FatPkg. Dieser Treiber ist in alle UEFI-Firmware eingebettet und kann nicht von OpenCore verwendet werden. Einige Firmware-Typen haben eine fehlerhafte Implementierung der FAT-Unterstützung, die zu beschädigten Dateisystemen bei Schreibversuchen führen kann. Die Einbettung dieses Treibers in die Firmware kann erforderlich sein, wenn während des Bootvorgangs auf die EFI-Partition geschrieben werden muss.

NvmExpressDxe*

NVMe-Unterstützungstreiber von MdeModulePkg. Dieser Treiber ist in der meisten Firmware ab der Broadwell-Generation enthalten. Für Haswell und frühere Generationen kann es günstiger sein, ihn in die Firmware einzubetten, wenn ein NVMe-SSD-Laufwerk installiert ist.

OpenCanopy*

OpenCore-Plugin zur Implementierung der grafischen Schnittstelle.

OpenRuntime*

OpenCore-Plugin zur Implementierung des OC_FIRMWARE_RUNTIME-Protokolls.

OpenLinuxBoot*

OpenCore-Plugin, das OC_BOOT_ENTRY_PROTOCOL implementiert, um die direkte Erkennung und das Booten von Linux-Distributionen von OpenCore aus zu ermöglichen, ohne Kettenladen über GRUB.

OpenUsbKbDxe*

USB-Tastaturtreiber, der Unterstützung für AppleKeyMapAggregator-Protokolle zusätzlich zu einer eigenen USB-Tastaturtreiber-Implementierung bietet. Dies ist eine Alternative zum eingebauten KeySupport, der je nach Firmware besser oder schlechter funktionieren kann.

OpenPartitionDxe*

Partitionsverwaltungstreiber mit Unterstützung des Apple Partitioning Scheme. Dieser Treiber kann verwendet werden, um das Laden älterer DMG-Wiederherstellungen wie macOS 10.9 mit Apple Partitioning Scheme zu unterstützen. OpenDuet enthält diesen Treiber bereits.

Ps2KeyboardDxe*

PS/2-Tastaturtreiber aus MdeModulePkg. OpenDuetPkg und einige Arten von Firmware enthalten diesen Treiber möglicherweise nicht, aber er ist notwendig, damit die PS/2-Tastatur funktioniert. Beachten Sie, dass dieser Treiber im Gegensatz zu OpenUsbKbDxe keine AppleKeyMapAggregator-Unterstützung hat und daher KeySupport aktiviert sein muss.

S. 78

Ps2MouseDxe*

PS/2-Maustreiber aus MdeModulePkg. Einige sehr alte Laptop-Firmware enthält diesen Treiber möglicherweise nicht, aber er ist notwendig, damit das Touchpad in grafischen UEFI-Schnittstellen wie OpenCanopy funktioniert.

OpenHfsPlus*

HFS-Dateisystemtreiber mit Bless-Unterstützung. Dieser Treiber ist eine Alternative zum Closed-Source-Treiber HfsPlus, der häufig in Apple-Firmware zu finden ist. Er bietet zwar alle Funktionen, ist aber etwa dreimal langsamer und muss noch einer Sicherheitsüberprüfung unterzogen werden.

UsbMouseDxe*

USB-Maustreiber von MdeModulePkg. Einige virtuelle Maschinen-Firmware wie OVMF enthält diesen Treiber möglicherweise nicht, aber er ist notwendig, damit die Maus in UEFI-Grafikschnittstellen wie OpenCanopy funktioniert.

XhciDxe*

XHCI-USB-Controller-Unterstützungstreiber von MdeModulePkg. Dieser Treiber ist in den meisten Firmware-Typen ab der Sandy Bridge-Generation enthalten. Für frühere Firmware oder Legacy-Systeme kann er verwendet werden, um externe USB 3.0 PCI-Karten zu unterstützen.

Die mit * gekennzeichneten Treiber werden mit OpenCore gebündelt. Um die Treiber aus UDK (EDK II) zu kompilieren, kann derselbe Befehl wie für die OpenCore-Kompilierung verwendet werden, allerdings muss ein entsprechendes Paket ausgewählt werden:

```
git clone https://github.com/acidanthera/audk UDK
cd UDK
source edksetup.sh
make -C BaseTools
build -a X64 -b RELEASE -t XCODE5 -p FatPkg/FatPkg.dsc
build -a X64 -b RELEASE -t XCODE5 -p MdeModulePkg/MdeModulePkg.dsc
```

11.3 Werkzeuge und Anwendungen

Eigenständige Tools können bei der Fehlersuche in Firmware und Hardware helfen. Einige der bekannten Tools sind unten aufgeführt. Während einige Tools aus OpenCore heraus gestartet werden können (siehe den Unterabschnitt Tools für weitere Details), sollten die meisten separaten entweder direkt oder aus der Shell heraus gestartet werden.

Um OpenShell oder ein anderes Tool direkt zu starten, speichern Sie OpenShell.efi unter dem Namen EFI\BOOT\BOOTX64.EFI auf einer FAT32-Partition. Es ist normalerweise unwichtig, ob das Partitionsschema GPT oder MBR ist.

Während der vorherige Ansatz sowohl auf Macs als auch auf anderen Computern funktioniert, gibt es einen alternativen Ansatz nur für Macs, um das Tool auf einem HFS+- oder APFS-Volume zu segnen:

```
sudo bless --verbose --file /Volumes/VOLNAME/DIR/OpenShell.efi \ --folder
/Volumes/VOLNAME/DIR/ --setBoot
```

Listing 3: Blessing-Werkzeug

Hinweis 1: /System/Library/CoreServices/BridgeVersion.bin sollte nach /Volumes/VOLNAME/DIR kopiert werden.

Hinweis 2: Um den bless-Befehl verwenden zu können, muss der Systemintegritätsschutz deaktiviert werden.

Hinweis 3: Um booten zu können, muss Secure Boot, falls vorhanden, deaktiviert werden.

Einige der bekannten Tools sind unten aufgeführt (eingebaute Tools sind mit * gekennzeichnet):

- BootKicker* Apple BootPicker Menü aufrufen (exklusiv für Macs mit kompatiblen GPUs).

- ChipTune* Testen Sie das BeepGen-Protokoll und erzeugen Sie Audiosignale unterschiedlicher Art und Läng

- CleanNvram* Zurücksetzen des NVRAM alternativ als eigenständiges Tool.

- CsrUtil* Einfache Implementierung der SIP-bezogenen Funktionen von Apple csrutil.

- GopStop* Test des GraphicsOutput-Protokolls mit einem einfachen Szenario.

- KeyTester* Testen der Tastatureingabe im SimpleText-Modus.

- MemTest86 Dienstprogramm zum Testen des Speichers.

- OpenControl* Freigeben und Sperren des NVRAM-Schutzes für andere Tools, um vollen NVRAM-Zugriff zu erhalten, wenn sie von OpenCore aus gestartet werden.

- OpenShell* OpenCore-konfigurierte UEFI-Shell für Kompatibilität mit einer breiten Palette von Firmware.

- PavpProvision EPID-Bereitstellung (erfordert die Konfiguration von Zertifikatsdaten).

- ResetSystem* Dienstprogramm zur Durchführung eines Systemresets. Nimmt den Reset-Typ als Argument: Coldreset, Firmware, Shutdown, Warmreset. Standardmäßig wird ein Kaltreset durchgeführt.

- RtcRw* Dienstprogramm zum Lesen und Schreiben des RTC-Speichers (CMOS).

- ControlMsrE2* Überprüft die Konsistenz von CFG Lock (MSR 0xE2 Schreibschutz) über alle Kerne und ändert solche versteckten Optionen auf ausgewählten Plattformen.

- TpmInfo* Überprüfen Sie die Intel PTT (Platform Trust Technology) Fähigkeit auf der Plattform, die die Verwendung von fTPM 2.0 erlaubt, wenn sie aktiviert ist. Das Tool prüft nicht, ob fTPM 2.0 tatsächlich aktiviert ist.

11.4 OpenCanopy

OpenCanopy ist eine grafische OpenCore-Benutzeroberfläche, die im External PickerMode läuft und ähnlich wie die eingebaute Textschnittstelle auf OpenCorePkg OcBootManagementLib aufbaut.

OpenCanopy benötigt zum Ausführen grafische Ressourcen, die sich im Resources-Verzeichnis befinden. Beispielressourcen (Schriftarten und Bilder) sind im OcBinaryData-Repository zu finden. Angepasste Icons können über das Internet gefunden werden (z.B. hier:(https://github.com/blackosx/OpenCanopyIcons) oder dort: (`https://applelife.ru/threads/kastomizacija-opencanopy.2945020/`).

OpenCanopy bietet volle Unterstützung für PickerAttributes und einen konfigurierbaren, integrierten Icon-Satz. Der gewählte Icon-Satz kann von dem Wert der Variablen DefaultBackgroundColor abhängen. Siehe PickerVariant für weitere Details.

Vordefinierte Symbole werden in dem von PickerVariant abgeleiteten Unterverzeichnis des Verzeichnisses \EFI\OC\Resources\Image gespeichert. Eine vollständige Liste der unterstützten Symbole (im .icns-Format) finden Sie weiter unten. Wenn optionale Symbole fehlen, wird das nächstgelegene verfügbare Symbol verwendet. Bei externen Einträgen wird das Symbol mit dem Präfix Ext verwendet, sofern verfügbar (z. B. OldExtHardDrive.icns).

Hinweis: Im Folgenden sind alle Dimensionen normativ für die Skalierungsstufe 1x und müssen für andere Stufen entsprechend skaliert werden.

- Cursor - Mauszeiger (obligatorisch, bis zu 144x144).

- Selected - Ausgewähltes Element (obligatorisch, 144x144).

- Selektor - Auswählendes Element (obligatorisch, bis zu 144x40).

- SetDefault - Auswahl der Standardeinstellung (obligatorisch, bis zu 144x40; muss dieselbe Breite wie der Selektor haben).

- Left - Bildlauf nach links (obligatorisch, 40x40).

- Right - Bildlauf nach rechts (obligatorisch, 40x40).

- HardDrive - Allgemeines Betriebssystem (obligatorisch, 128x128).

- Background - Zentriertes Hintergrundbild.

- Apple - Apple-Betriebssystem (128x128).

- AppleRecv - Apple Recovery OS (128x128).

- AppleTM - Apple Time Machine (128x128).

- Windows - Windows (128x128).

- Other - Benutzerdefinierter Eintrag (siehe Einträge, 128x128).

- ResetNVRAM - NVRAM zurücksetzen - Systemaktion oder Werkzeug (128x128).

- Shell - Eintrag mit UEFI-Shell-Namen, z. B. OpenShell (128x128). - Werkzeug - Jedes andere Werkzeug (128x128).

Vordefinierte Bezeichnungen werden im Verzeichnis \EFI\OC\Resources\Label gespeichert. Jede Beschriftung hat das Suffix .lbl oder .l2x, um die Skalierungsebene anzugeben. Eine vollständige Liste der Bezeichnungen finden Sie weiter unten. Alle Bezeichnungen sind obligatorisch.

- EFIBoot - Allgemeines Betriebssystem.

- Apple - Apple-Betriebssystem.

- AppleRecv - Apple Wiederherstellungsbetriebssystem.

- AppleTM - Apple Time Machine.

S.80

- Windows - Windows

- Other - Benutzerdefinierter Eintrag (siehe Einträge).

- ResetNVRAM - NVRAM-Systemaktion oder -Tool zurücksetzen.

- SIPDisabled - Toogle SIP-Tool mit deaktiviertem SIP.

- SIPEnabled - Toogle SIP-Tool mit aktiviertem SIP.

- Shell - Eintrag mit UEFI-Shell-Namen (z. B. OpenShell).

- Tool - Jedes andere Tool.

Hinweis: Alle Beschriftungen müssen eine Höhe von genau 12 px haben. Es gibt keine Begrenzung für ihre Breite.

Die Erstellung von Etiketten und Symbolen kann mit den gebündelten Dienstprogrammen disklabel und icnspack durchgeführt werden. Die Schriftart ist Helvetica 12 pt mal Skalierungsfaktor.

Das Schriftformat entspricht dem AngelCode-Binärformat BMF. Obwohl es viele Hilfsprogramme gibt, um Schriftdateien zu erzeugen, wird derzeit empfohlen, dpFontBaker zu verwenden, um eine Bitmap-Schrift zu erzeugen (mit CoreText werden die besten Ergebnisse erzielt) und fonverter, um sie in ein Binärformat zu exportieren.

11.5 OpenRuntime

OpenRuntime ist ein OpenCore-Plugin, das das OC_FIRMWARE_RUNTIME-Protokoll implementiert. Dieses Protokoll implementiert mehrere Funktionen, die für OpenCore erforderlich sind und die sonst nicht in OpenCore selbst implementiert werden können, da sie zur Laufzeit, d.h. während der Funktion des Betriebssystems, benötigt werden. Besondere Merkmale:

- NVRAM-namespaces, die es ermöglichen, Betriebssysteme vom Zugriff auf ausgewählte Variablen zu isolieren (z.B. RequestBootVarRouting oder ProtectSecureBoot).

- Read-only and write-only NVRAM variables, die die Sicherheit von OpenCore, Lilu und Lilu-Plugins wie VirtualSMC, das AuthRestart-Unterstützung implementiert, erhöhen.

- NVRAM isolation, die es ermöglicht, alle Variablen vor dem Schreiben durch ein nicht vertrauenswürdiges Betriebssystem zu schützen (z.B. DisableVariableWrite).

- UEFI Runtime Services Speicherschutzverwaltung zur Umgehung der Nur-Lese-Zuordnung (z. B. EnableWriteUnprotector).

11.6 OpenLinuxBoot

OpenLinuxBoot ist ein OpenCore-Plugin, das OC_BOOT_ENTRY_PROTOCOL implementiert. Es zielt darauf ab, die meisten Linux-Distributionen ohne zusätzliche Konfiguration automatisch zu erkennen und zu booten.

Die Verwendung ist wie folgt:

- Fügen Sie OpenLinuxBoot.efi und typischerweise (siehe unten) auch ext4_x64.efi zum Abschnitt Drivers der config.plist hinzu.

- Stellen Sie sicher, dass RequestBootVarRouting und LauncherOption in der config.plist aktiviert sind; es wird auch empfohlen, HideAuxiliary zu aktivieren, um ältere Linux-Kernel auszublenden, außer wenn sie benötigt werden (sie werden als auxiliary entries hinzugefügt und können dann durch Drücken der Leertaste im OpenCore-Bootmenü angezeigt werden).

- Installieren Sie Linux wie gewohnt, falls dies noch nicht geschehen ist - OpenLinuxBoot ist an diesem Schritt nicht beteiligt.

- Starten Sie OpenCore neu: die installierte Linux-Distribution sollte einfach erscheinen und direkt von OpenCore booten, wenn sie ausgewählt wird, was sie ohne Chainloading über GRUB tut.

Wenn OpenCore bereits manuell für das Booten von Linux eingerichtet wurde, z.B. über BlessOverride oder über Entries, dann können diese Einstellungen entfernt werden, so dass die Linux-Distribution nicht zweimal im Bootmenü angezeigt wird.

Es wird empfohlen, Linux mit seinem Standard-Bootloader zu installieren, auch wenn dieser beim Booten über OpenLinuxBoot nicht aktiv verwendet wird. Das liegt daran, dass OpenLinuxBoot die korrekten Kernel-Optionen erkennen muss, und das tut es, indem es in den vom Standard-Bootloader hinterlassenen Dateien sucht. Wenn kein Bootloader installiert wurde (oder diese Optionen nicht gefunden werden können), ist das Booten immer noch möglich, aber die richtigen Boot-Optionen müssen manuell angegeben werden, bevor OpenLinuxBoot versucht, die Distribution zu starten.

OpenLinuxBoot benötigt typischerweise Dateisystemtreiber, die nicht in der Firmware verfügbar sind, wie z.B. EXT4- und BTRFS-Treiber. Diese Treiber können von externen Quellen bezogen werden. Treiber, die in einfachen Szenarien getestet wurden, können von OcBinaryData heruntergeladen werden. Beachten Sie, dass diese Treiber weder auf Zuverlässigkeit in allen Szenarien noch auf Manipulationssicherheit getestet wurden und daher potenzielle Sicherheitsrisiken oder Datenverluste bergen können.

S- 81

Die meisten Linux-Distributionen benötigen den ext4_x64-Treiber, einige wenige benötigen den btrfs_x64-Treiber und einige wenige benötigen keinen zusätzlichen Dateisystemtreiber: dies hängt vom Dateisystem der Boot-Partition der installierten Distribution ab und davon, welche Dateisysteme bereits von der Firmware des Systems unterstützt werden. LVM wird derzeit nicht unterstützt - dies liegt daran, dass es derzeit keinen eigenständigen UEFI-LVM-Dateisystemtreiber gibt.

Achten Sie auf die Eigenart von SyncRuntimePermissions, die unter Umständen gesetzt werden muss, um ein frühzeitiges Booten des Linux-Kernels (typischerweise mit einem schwarzen Bildschirm) zu vermeiden, was auf einen Firmware-Fehler einiger nach 2017 veröffentlichter Firmware zurückzuführen ist. Wenn dieser vorhanden ist und nicht durch diese Eigenart gemildert wird, wirkt sich dies auf das Booten über OpenCore mit oder ohne OpenLinuxBoot aus.

Nach der Installation von OpenLinuxBoot wird empfohlen, die im OpenCore-Debugprotokoll angezeigten Optionen beim Booten (oder beim Versuch zu booten) einer bestimmten Distribution mit den Optionen zu vergleichen, die mit dem Shell-Befehl cat /proc/cmdline angezeigt werden, wenn dieselbe Distribution über ihren nativen Bootloader gebootet wurde. Im Allgemeinen sollten diese Optionen (aus Gründen der Sicherheit der laufenden Distro) übereinstimmen, und falls nicht, wird empfohlen, die unten aufgeführten Treiberargumente (insbesondere LINUX_BOOT_ADD_RO, LINUX_BOOT_ADD_RW, autoopts:{PARTUUID} und autoopts) zu verwenden, um die Optionen nach Bedarf zu ändern. Beachten Sie jedoch, dass die folgenden Unterschiede normal sind und nicht behoben werden müssen:

- Wenn der Standard-Bootloader GRUB ist, enthalten die von OpenLinuxBoot generierten Optionen keinen Wert für BOOT_IMAGE=..., wo die GRUB-Optionen dies tun, und sie enthalten einen Wert für initrd=..., wo die GRUB-Optionen dies nicht tun.

- OpenLinuxBoot verwendet PARTUUID und nicht die UUID des Dateisystems, um den Speicherort von initrd zu identifizieren. Dies ist so gewollt, da UEFI-Dateisystemtreiber keine Linux-Dateisystem-UUID-Werte zur Verfügung stellen.

- Weniger wichtige Grafikübergabeoptionen (wie im Ubuntu-Beispiel in autoopts unten beschrieben) werden nicht genau übereinstimmen, dies ist nicht wichtig, solange die Distribution erfolgreich bootet.

Wenn Sie OpenLinuxBoot mit Secure Boot verwenden, möchten Sie vielleicht das shim-to-cert.tool verwenden, das in den OpenCore-Utilities enthalten ist. Es kann verwendet werden, um den öffentlichen Schlüssel zu extrahieren, der benötigt wird, um den Kernel einer Distro direkt zu booten, so wie es bei der Verwendung von OpenCore mit OpenLinuxBoot gemacht wird, anstatt über GRUB shim. Für Nicht-GRUB-Distros muss der benötigte öffentliche Schlüssel durch Benutzerrecherche gefunden werden.

11.6.1 Konfiguration

Die Standard-Parameterwerte sollten in den meisten Fällen ohne Änderungen funktionieren, aber bei Bedarf können die folgenden Optionen für den Treiber in UEFI/Drivers/Arguments angegeben werden:

- flags - Standard: alle Flags sind gesetzt, außer den folgenden:

- LINUX_BOOT_ADD_RW,

- LINUX_BOOT_LOG_VERBOSE und

- LINUX_BOOT_ADD_DEBUG_INFO.

Verfügbare Flags sind:

- 0x00000001 (Bit 0) - LINUX_BOOT_SCAN_ESP, Ermöglicht die Suche nach Einträgen auf der EFI-Systempartition.

- 0x00000002 (Bit 1) - LINUX_BOOT_SCAN_XBOOTLDR, Erlaubt die Suche nach Einträgen auf der Extended Boot Loader Partition.

- 0x00000004 (Bit 2) - LINUX_BOOT_SCAN_LINUX_ROOT, Ermöglicht das Scannen nach Einträgen auf Linux Root-Dateisystemen. - 0x00000008 (Bit 3)

- LINUX_BOOT_SCAN_LINUX_DATA, Ermöglicht das Scannen nach Einträgen in Linux Data-Dateisystemen.

- 0x00000080 (Bit 7) - LINUX_BOOT_SCAN_OTHER, Ermöglicht das Scannen nach Einträgen auf Dateisystemen, die nicht durch die nicht von den obigen Optionen abgedeckt werden.

Die folgenden Hinweise gelten für alle der oben genannten Optionen:

Hinweis 1: Die Apple-Dateisysteme APFS und HFS werden nie gescannt.

Hinweis 2: Unabhängig von den obigen Flags muss ein Dateisystem zuerst von Misc/Security/ScanPolicy zugelassen werden, bevor es von OpenLinuxBoot oder einem anderen OC_BOOT_ENTRY_PROTOCOL-Treiber gesehen werden kann.

Hinweis 3: Es wird empfohlen, das Scannen von LINUX_ROOT und LINUX_DATA sowohl in den OpenLinuxBoot-Flags als auch in Misc/Security/ScanPolicy zu aktivieren, um sicher

zu gehen, dass alle gültigen Linux-Installationen erkannt werden, da Linux-Boot-Dateisysteme sehr oft als LINUX_DATA markiert sind.

- 0x00000100 (Bit 8) - LINUX_BOOT_ALLOW_AUTODETECT, wenn gesetzt, erlaubt es die automatische Erkennung und Verknüpfung von vmlinuz* und init* Ramdisk-Dateien, wenn Loader/Entries-Dateien nicht gefunden werden.

- 0x00000200 (Bit 9) - LINUX_BOOT_USE_LATEST, Wenn ein von OpenLinuxBoot erzeugter Linux-Eintrag als Standard-Boot-Eintrag in OpenCore ausgewählt ist, wird automatisch zum neuesten Kernel gewechselt, wenn eine neue Version installiert wird.

Wenn diese Option gesetzt ist, wird eine interne Menüeintrags-ID zwischen Kernel-Versionen derselben Linux-Installation geteilt. Linux-Boot-Optionen werden immer nach der höchsten Kernel-Version sortiert. Das bedeutet, dass die neueste Kernel-Version der gleichen Installation immer als Standard angezeigt wird, wenn diese Option gesetzt ist.

Hinweis: Diese Option wird für alle Systeme empfohlen.

- 0x00000400 (Bit 10) - LINUX_BOOT_ADD_RO, Diese Option gilt nur für automatisch erkannte Linux-Systeme (d.h. nicht für BLSpec- oder Fedora-ähnliche Distributionen, die /loader/entries/*.conf-Dateien haben). Einige Distributionen führen beim Laden eine Dateisystemprüfung durch, die erfordert, dass das Root-Dateisystem anfänglich über die Kerneloption ro schreibgeschützt gemountet wird, was erfordert, dass diese Option zu den autodetected-Optionen hinzugefügt wird. Setzen Sie dieses Bit, um diese Option bei automatisch erkannten Distros hinzuzufügen; es sollte harmlos sein, aber die Bootzeit bei Distros, die es nicht benötigen, sehr leicht verlangsamen (aufgrund der erforderlichen Neueinhängung als read-write). Wenn es mehrere Distros gibt und es erforderlich ist, diese Option nur für bestimmte Distros anzugeben, verwenden Sie autoopts:{PARTUUID} +=ro, um die Option manuell hinzuzufügen, wo sie benötigt wird, anstatt dieses Flag zu verwenden.

- 0x00000800 (Bit 11) - LINUX_BOOT_ADD_RW, Wie LINUX_BOOT_ADD_RO, gilt diese Option nur für automatisch erkannte Linux-Distributionen. Sie wird für die meisten Distributionen nicht benötigt (die normalerweise entweder ro oder nichts zu den erkannten Boot-Optionen hinzufügen), ist aber bei einigen von Arch abgeleiteten Distributionen erforderlich, z.B. EndeavourOS. Wenn es mehrere Distros gibt und es erforderlich ist, diese Option nur für bestimmte Distros anzugeben, verwenden Sie autoopts:{PARTUUID} +=rw, um die Option manuell hinzuzufügen, wo sie benötigt wird, anstatt dieses Flag zu verwenden. Wenn diese Option und LINUX_BOOT_ADD_RO beide angegeben sind, wird nur diese Option angewendet und LINUX_BOOT_ADD_RO wird ignoriert.

- 0x00002000 (Bit 13) - LINUX_BOOT_ALLOW_CONF_AUTO_ROOT, In einigen Fällen von BootLoaderSpecByDefault in Kombination mit ostree geben die /loader/entries/*.conf-Dateien keine erforderliche root=... Kerneloption an - sie wird von GRUB hinzugefügt. Wenn dieses Bit gesetzt ist und diese Situation erkannt wird, dann wird diese Option automatisch hinzugefügt. (Erforderlich zum Beispiel für Endless OS.)

- 0x00004000 (Bit 14) - LINUX_BOOT_LOG_VERBOSE, Fügt zusätzliche Debug-Log-Informationen über gefundene Dateien und Autodetect-Optionen hinzu, die beim Scannen nach Linux-Boot-Einträgen hinzugefügt wurden.

- 0x00008000 (Bit 15) - LINUX_BOOT_ADD_DEBUG_INFO, fügt jedem generierten Eintragsnamen einen menschenlesbaren Dateisystemtyp, gefolgt von den ersten acht Zeichen der eindeutigen Partitions-Uuid, hinzu. Kann bei der Fehlersuche in Bezug auf den Ursprung der vom Treiber erzeugten Einträge helfen, wenn mehrere Linux-Installationen auf einem System vorhanden sind.

Flag-Werte können hexadezimal beginnend mit 0x oder dezimal angegeben werden, z.B. flags=0x80 oder flags=128. Es ist auch möglich, Flags hinzuzufügen oder zu entfernen, indem man eine Syntax wie flags+=0xC000 verwendet, um alle Debugging-Optionen hinzuzufügen oder flags-=0x400, um die Option LINUX_BOOT_ADD_RO zu entfernen.

- autoopts:{PARTUUID}[+]="{options}" - Voreinstellung: nicht gesetzt.

Ermöglicht die manuelle Angabe von Kerneloptionen, die im Autodetect-Modus nur für eine bestimmte Partition verwendet werden sollen. Ersetzen Sie den Text {PARTUUID} durch die spezifische Partitions-UUID, auf der die Kernel gespeichert sind (im normalen Layout die Partition, die /boot enthält), z.B. autoopts:11223344-5566-7788-99aa-bbccddeeff00+="vt.handoff=7". Wenn mit += angegeben, werden diese Optionen zusätzlich zu den automatisch erkannten Optionen verwendet, wenn mit = angegeben, werden sie stattdessen verwendet. Wird nur für die automatische Erkennung von Linux verwendet - die hier angegebenen Werte werden niemals für Einträge verwendet, die aus /loader/entries/*.conf-Dateien erstellt wurden.

Hinweis: Der hier anzugebende PARTUUID-Wert ist in der Regel derselbe wie der PARTUUID-Wert, der in root=PARTUUID=... in den Boot-Optionen des Linux-Kernels steht (einsehbar mit cat /proc/cmdline). Alternativ und für fortgeschrittene Szenarien ist es möglich, zu untersuchen, wie die Partitionen der Distribution mit dem Linux-Befehl mount eingehängt sind, und dann die partuuid der relevanten eingehängten Partitionen herauszufinden, indem man die Ausgabe von ls -l /dev/disk/by-partuuid untersucht.

- autoopts[+]="{options}" - Voreinstellung: Keine angegeben.

Ermöglicht die manuelle Angabe von Kernel-Optionen, die im Autodetect-Modus verwendet werden sollen. Das alternative Format autoopts:{PARTUUID} ist besser geeignet, wenn es mehrere Distros gibt, aber autoopts ohne PARTUUID kann für nur eine Distro bequemer sein. Wenn mit += angegeben, werden diese zusätzlich zu den automatisch erkannten Optionen verwendet, wenn mit = angegeben, werden sie stattdessen verwendet. Wird nur für automatisch erkannte Linux-Distributionen verwendet - die hier angegebenen Werte werden niemals für Einträge verwendet, die aus /loader/entries/*.conf-Dateien erstellt wurden.

Als Beispiel für die Verwendung ist es möglich, das Format += zu verwenden, um eine vt.handoff-Option hinzuzufügen, wie z.B. autopts+="vt.handoff=7" oder autopts+="vt.handoff=3" (überprüfen Sie cat /proc/cmdline, wenn Sie über den Standard-Bootloader der Distro gebootet haben) auf Ubuntu und verwandten Distros, um die vt.handoff-Option zu den automatisch erkannten GRUB-Standardwerten hinzuzufügen und zu vermeiden, dass ein blinkender Text vor dem Splash-Screen der Distro angezeigt wird.

S. 83

11.6.2 Zusätzliche Informationen

OpenLinuxBoot kann die loader/entries/*.conf-Dateien erkennen, die gemäß der Bootloader-Spezifikation (https://systemd.io/BOOT_LOADER_SPECIFICATION/) oder der

eng verwandten systemd BootLoaderSpecByDefault (https://fedoraproject.org/wiki/Changes/BootLoaderSpecByDefault) erstellt wurden. Erstere ist spezifisch für systemd-boot und wird von Arch Linux verwendet, letztere gilt für die meisten Fedora-verwandten Distributionen, einschließlich Fedora selbst, RHEL und Varianten.

Wo die oben genannten Dateien nicht vorhanden sind, kann OpenLinuxBoot die {boot}/vmlinuz*-Kerneldateien automatisch erkennen und direkt booten. Es verknüpft diese automatisch - basierend auf der Kernelversion im Dateinamen - mit ihren zugehörigen {boot}/init*-Ramdisk-Dateien. Dies gilt für die meisten Debian-bezogenen Distros, einschließlich Debian selbst, Ubuntu und Varianten.

Bei der automatischen Erkennung in /boot als Teil des Root-Dateisystems sucht OpenLinuxBoot in /etc/default/grub nach Kernel-Bootoptionen und in /etc/os-release nach dem Distro-Namen. Bei der automatischen Erkennung in einer eigenständigen Bootpartition (d.h. wenn /boot einen eigenen Einhängepunkt hat) kann OpenLinuxBoot keine Kernelargumente automatisch erkennen und alle Kernelargumente außer initrd=... müssen vollständig von Hand mit autoopts=... oder autoopts:{partuuid}=... angegeben werden (+=-Varianten dieser Optionen werden nicht funktionieren, da diese nur zusätzliche Argumente hinzufügen).

BootLoaderSpecByDefault (aber nicht die reine Boot Loader Specification) kann GRUB-Variablen in den *.conf-Dateien erweitern - und dies wird in der Praxis in bestimmten Distros wie CentOS verwendet. Um dies korrekt zu handhaben, extrahiert OpenLinuxBoot, wenn diese Situation erkannt wird, alle Variablen aus {boot}/grub2/grubenv und auch alle bedingungslos gesetzten Variablen aus {boot}/grub2/grub.cfg, und erweitert diese dann, wo erforderlich, in *.conf-Dateieinträgen.

Die einzige derzeit unterstützte Methode, Linux-Kernel zu starten, beruht darauf, dass sie mit EFISTUB kompiliert wurden. Dies gilt für fast alle modernen Distros, insbesondere für diejenigen, die systemd verwenden. Beachten Sie, dass die meisten modernen Distros systemd als Systemmanager verwenden, auch wenn die meisten nicht systemd-boot als Bootloader verwenden.

systemd-boot-Benutzer (wahrscheinlich fast ausschließlich Arch Linux-Benutzer) sollten sich darüber im Klaren sein, dass OpenLinuxBoot das systemd-boot-spezifische Boot Loader Interface (`https://systemd.io/BOOT_LOADER_INTERFACE/`) nicht unterstützt; daher muss efibootmgr anstelle von bootctl für jegliche Low-Level-Linux-Kommandozeilen-Interaktion mit dem Boot-Menü verwendet werden.

11.7 AudioDxe

Treiber zur Unterstützung von High Definition Audio in der UEFI-Firmware für die meisten Intel- und einige andere analoge Audiocontroller. Hinweis: AudioDxe ist ein Staging-Treiber, siehe acidanthera/bugtracker#740 (https://github.com/acidanthera/bugtracker/issues/740) für bekannte Probleme.

11.7.1 Konfiguration

Die meisten UEFI-Audiokonfigurationen werden über den Abschnitt UEFI-Audioeigenschaften abgewickelt, aber bei Bedarf können die folgenden zusätzlichen Konfigurationsoptionen (die benötigt werden, um Sound auf der meisten Apple-Hardware

und möglicherweise einigen anderen zu erzeugen) in UEFI/Treiber/Argumente angegeben werden:

--gpio-setup - Der Standardwert ist 0 (GPIO-Setup deaktiviert), wenn das Argument nicht angegeben wird, oder 7 (alle GPIO-Setup-Stufen aktiviert), wenn das Argument mit keinem Wert angegeben wird.

Verfügbare Werte, die durch Addieren kombiniert werden können, sind:

- 0x00000001 (Bit 0) - GPIO_SETUP_STAGE_DATA, setzt GPIO-Pin-Daten an den angegebenen Pins auf High. Erforderlich z.B. bei MacBookPro10,2 und MacPro5,1.

- 0x00000002 (Bit 1) - GPIO_SETUP_STAGE_DIRECTION, setzt die Richtung der GPIO-Daten an den angegebenen Pins auf Ausgang. Erforderlich z.B. auf MacPro5,1.

- 0x00000004 (Bit 2) - GPIO_SETUP_STAGE_ENABLE, aktiviert die angegebenen GPIO-Pins. Erforderlich z.B. auf MacPro5,1.

Wenn Audio auf dem korrekten Codec zu 'spielen' scheint, z.B. basierend auf dem Debug-Log, aber kein Ton auf irgendeinem Kanal zu hören ist, wird empfohlen, --gpio-setup (ohne Wert) in den AudioDxe-Treiber-Argumenten zu verwenden. Wenn kein Wert angegeben wird, werden alle Stufen aktiviert (entspricht der Angabe von 7). Wenn dies zu einem guten Ergebnis führt, ist es möglich, weniger Bits zu verwenden, z.B. --gpio-setup=1, --gpio-setup=3, um herauszufinden, welche Stufen tatsächlich benötigt werden.

Hinweis: Der Wert 7 (alle Flags aktiviert) dieser Option - wie für den MacPro5,1 erforderlich - ist mit den meisten Systemen kompatibel, aber es ist bekannt, dass er auf einer kleinen Anzahl anderer Systeme Probleme mit dem Sound verursacht (vorherige Sounds dürfen nicht beendet werden, bevor neue Sounds beginnen), daher ist diese Option nicht standardmäßig aktiviert.

--gpio-pins - Default: 0, automatische Erkennung.

Gibt an, welche GPIO-Pins von --gpio-setup bedient werden sollen. Dies ist eine Bitmaske, mit möglichen Werten von 0x0 bis 0xFF. Das nutzbare Maximum hängt von der Anzahl der verfügbaren Pins an der Audio-Out-Funktionsgruppe des verwendeten Codecs ab, z. B. 0x3 (niedrigste zwei Bits), wenn zwei GPIO-Pins vorhanden sind, 0x7, wenn drei Pins vorhanden sind, usw.

S. 84

Wenn --gpio-setup aktiviert ist (d.h. nicht Null), dann ist 0 ein spezieller Wert für --gpio-pins, was bedeutet, dass die Pin-Maske automatisch generiert wird, basierend auf der gemeldeten Anzahl von GPIO-Pins des angegebenen Codecs (siehe AudioCodec), z.B. wenn die Audio-Out Funktionsgruppe des Codecs 4 GPIO-Pins meldet, wird eine Maske von 0xF verwendet. Der verwendete Wert kann im Debug-Protokoll in einer Zeile wie der folgenden gesehen werden:

    HDA: GPIO-Setup auf Pins 0x0F - Success

Werte für Treiberparameter können hexadezimal beginnend mit 0x oder dezimal angegeben werden, z.B. --gpio-pins=0x12

oder --gpio-pins=18.

--restore-nosnoop - Boolean flag, aktiviert wenn vorhanden.

AudioDxe löscht das Intel HDA No Snoop Enable (NSNPEN) Bit. Auf einigen Systemen muss diese Änderung beim Beenden wieder rückgängig gemacht werden, um zu vermeiden, dass der Sound in Windows unterbrochen wird. Wenn dies der Fall ist, sollte dieses Flag zu den AudioDxe-Treiberargumenten hinzugefügt werden. Standardmäßig nicht aktiviert, da das Wiederherstellen des Flags verhindern kann, dass der Ton unter macOS auf einigen anderen Systemen funktioniert.

11.8 Eigenschaften

1. APFS

Typ: plist dict

Failsafe: None

Beschreibung: Bietet APFS-Unterstützung wie im Abschnitt APFS-Eigenschaften unten konfiguriert.

2. Audio

Typ: plist dict

Failsafe: None

Beschreibung: Konfiguriert die Audio-Backend-Unterstützung wie im Abschnitt Audio-Eigenschaften unten beschrieben.

Sofern nicht anders dokumentiert (z.B. ResetTrafficClass), beziehen sich die Einstellungen in diesem Abschnitt nur auf die UEFI-Audiounterstützung (z.B. OpenCore-generierter Boot-Chime und Audio-Assist) und sind unabhängig von der Konfiguration, die für die OS Audio-Unterstützung (z.B. AppleALC).

Die UEFI-Audiounterstützung bietet einen Weg für Upstream-Protokolle, mit der ausgewählten Audio-Hardware und den Ressourcen zu interagieren. Alle Audioressourcen sollten sich im Verzeichnis \EFI\OC\Resources\Audio befinden. Derzeit werden die Audiodateiformate MP3 und WAVE PCM unterstützt. Es ist zwar treiberabhängig, welches Audio-Stream-Format unterstützt wird, aber die meisten gängigen Audiokarten unterstützen 16-Bit-Signed-Stereo-Audio mit 44100 oder 48000 Hz.

Der Pfad der Audiodatei wird durch den Audiotyp, die Audiolokalisierung und den Audiopfad bestimmt. Jeder Dateiname sieht wie folgt aus: [Audio-Typ]_[Audio-Lokalisierung]_[Audio-Pfad].[Audio ext]. Bei nicht lokalisierten Dateien enthält der Dateiname nicht den Sprachcode und sieht wie folgt aus: [audio type]_[audio path].[audio ext]. Die Audioerweiterung kann entweder mp3 oder wav sein.

- Der Audiotyp kann OCEFIAudio für OpenCore-Audiodateien oder AXEFIAudio für macOS-Bootloader-Audiodateien sein.

- Die Audiolokalisierung ist ein zweistelliger Sprachcode (z. B. en) mit einer Ausnahme für Chinesisch, Spanisch und Portugiesisch. Eine Liste aller unterstützten Lokalisierungen finden Sie in der Definition von APPLE_VOICE_OVER_LANGUAGE_CODE

(https://github.com/acidanthera/OpenCorePkg/blob/master/Include/Apple/Protocol/AppleVoiceOver.h).

- Der Audiopfad ist der Basisdateiname, der einem Dateibezeichner entspricht. Für macOS-Bootloader-Audiopfade siehe die APPLE_VOICE_OVER_AUDIO_FILE-Definition (https://github.com/acidanthera/OpenCorePkg/blob/master/Include/Apple/Protocol/AppleVoiceOver.h). Für OpenCore-Audiopfade siehe OC_VOICE_OVER_AUDIO_FILE Definition (https://github.com/acidanthera/OpenCorePkg/blob/master/Include/Acidanthera/Protocol/OcAudio.h). Die einzige Ausnahme ist die OpenCore-Boot-Chime-Datei, die OCEFIAudio_VoiceOver_Boot.mp3 heißt.

Die Audio-Lokalisierung wird für macOS-Bootloader und OpenCore getrennt festgelegt. Für den macOS-Bootloader wird sie im preferences.efires-Archiv in der Datei systemLanguage.utf8 festgelegt und durch das Betriebssystem gesteuert. Für OpenCore wird der Wert der Variable prev-lang:kbd verwendet. Wenn die native Audio-Lokalisierung einer bestimmten Datei fehlt, wird die englische Sprachlokalisierung (en) verwendet. Beispiel-Audiodateien sind im OcBinaryData-Repository (https://github.com/acidanthera/OcBinaryData) zu finden.

3. ConnectDrivers

Typ: plist boolean

Failsafe: false

Beschreibung: Führt die UEFI-Controller-Verbindung nach dem Laden des driver durch.

Diese Option ist nützlich für das Laden von driver, die dem UEFI-drivermodell folgen, da sie möglicherweise nicht von selbst starten. Beispiele für solche driver sind Dateisystem- oder Audiodriver. Obwohl diese Option effektiv ist, ist sie für driver, die eine automatische Verbindung herstellen, nicht unbedingt notwendig und kann den Bootvorgang leicht verlangsamen.

S. 85

Hinweis: Einige Firmware-Typen, insbesondere die von Apple, verbinden nur das Startlaufwerk, um den Startvorgang zu beschleunigen. Aktivieren Sie diese Option, um alle Boot-Optionen zu sehen, wenn Sie mehrere Laufwerke verwenden.

4. Drivers

Typ: plist array

Failsafe: Empty

Beschreibung: Lädt ausgewählte drivers aus dem Verzeichnis OC/Drivers.

Soll mit plist dict-Werten gefüllt werden, die jeden Treiber beschreiben. Siehe Abschnitt "TDrivers Properties" weiter unten.

5. Input

Type: plist dict

Failsafe: None

Beschreibung: Wenden Sie individuelle Einstellungen für die Eingabe (Tastatur und Maus) im Abschnitt Input Properties unten an.

6. Output

Type: plist dict

Failsafe: None

Beschreibung: Wenden Sie die individuellen Einstellungen für die Ausgabe (Text und Grafik) im Abschnitt Output Properties unten an.

7. ProtocolOverrides

Typ: plist dict

Failsafe: None

Beschreibung: Erzwingt eingebaute Versionen bestimmter Protokolle, die im Abschnitt Eigenschaften von ProtocolOverrides unten beschrieben sind.

Hinweis: Alle Protokollinstanzen werden vor dem Laden des Treibers installiert.

8. Quirks

Type: plist dict

Failsafe: None

Beschreibung: Wendet einzelne Firmware-Quirks an, die im Abschnitt Quirks Properties weiter unten beschrieben werden.

9. ReservedMemory

Typ: plist-Array

Failsafe: Empty

Beschreibung: Soll mit plist dict-Werten gefüllt werden, die Speicherbereiche beschreiben, die ausschließlich für bestimmte Firmware- und Hardwarefunktionen vorgesehen sind und vom Betriebssystem nicht verwendet werden sollten. Beispiele für solche Speicherbereiche könnten die zweiten 256 MB sein, die durch die Intel HD 3000 beschädigt wurden, oder ein Bereich mit fehlerhaftem RAM. Weitere Informationen finden Sie im Abschnitt ReservedMemory Properties weiter unten.