OpenCore
Referenzhandbuch (0.8.0)

[2022.04.16]

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

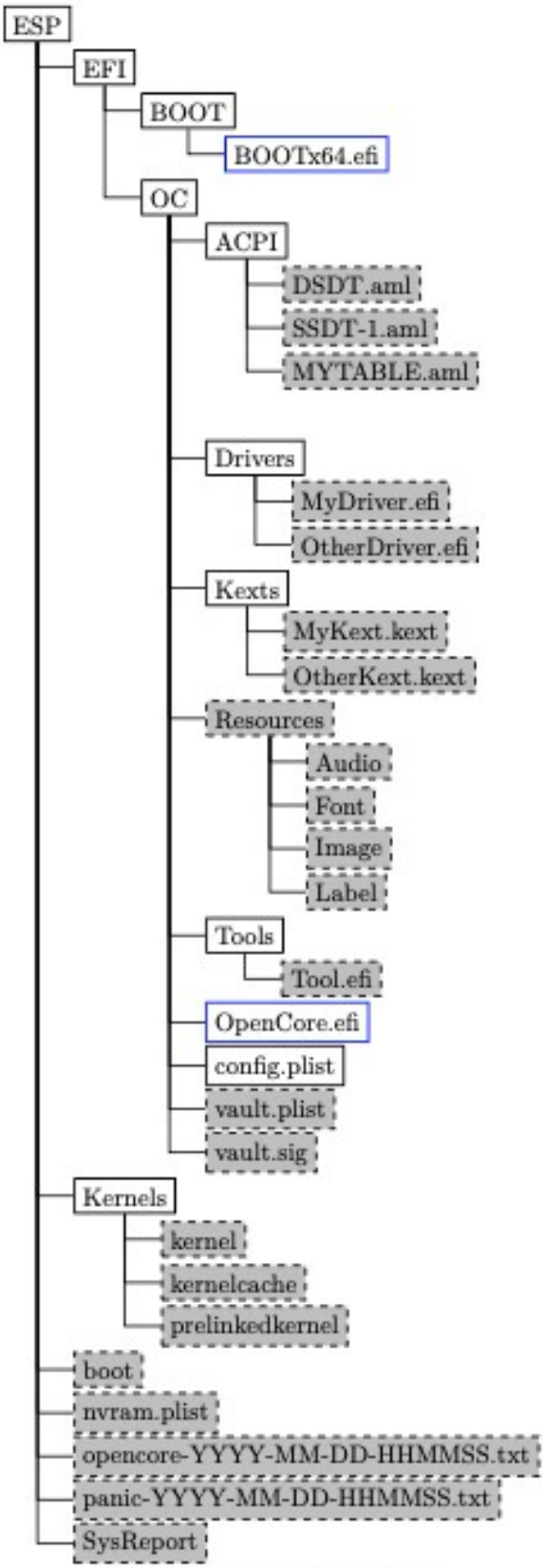

Figure 1. Directory Structure

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