Umsetzung prüfen RevisionID in IOHDACodecDevice ändern (PropertyInjector.kext)

Beitrag von „frankferrari“ vom 13. Dezember 2018, 19:27

Hallo!

Stand

Ich habe bereits hier mein System / Installation vorgestellt.

Ich hab soweit erstmal alles zum Laufen gebracht, alles bis auf den **Sound**. Des Problems Lösung beschreibt Brumbaer hier.

Um seinen Thread nicht weiter "voll zu müllen" und weil ich es bisher nicht geschafft habe es zu lösen, wollte ich hier um Hilfe bitten.

Wissensbasis

Ich hab mir seine Erklärungen durchgelesen zu Kext im Allg. und zum PropertyInjector selbst.

Problem & Lösung

Es soll also die Revision ID durch die eigentlich ersetzt werden. So wie ich das verstanden habe, wird hier das falsche Gerät von MacOS erkannt, oder zumindest die falsche Revision.

Sprich, es muss in der DSDT der richtige Wert der Revision meiner Soundkarte eingetragen/ersetzt werden. Dieser ist lt. IORegistryExplorer `0x100101`

Im Wiki von AppleALC wird zu meiner Karte (ALC1220) folgendes gelistet:

Realtek ALC1220 0x100003, layout 1, 2, 5, 7, 11, 13 15 (10.11)

Ich geh also davon aus das die korrekte RevisionID `0x100003` lautet.

Um diesen Wert zu ändern wollte ich den Lösungsansatz über den PropertyInjector(kext) gehen.

Also habe ich die kext entsprechend modifiziert, so wie ich es verstanden habe (siehe Bilder im Anhang).

Leider führte das dann dazu, dass MacOS nicht mehr booten wollte....

Was mach ich also falsch? Wo ist der Denkfehler?

Beitrag von „Brumbaer“ vom 13. Dezember 2018, 19:39

Es geht schon mal damit los, dass die Struktur falsch ist.

Wenn du einen Eintrag namens Audio erzeugst, musst du dessen Untereinträge auch "unter" bzw. "in" den Audio Eintrag legen.

Beitrag von „frankferrari“ vom 13. Dezember 2018, 20:48

Oh Gott. Traurig aber war.. es war schon sehr spät.

-

Ist mein Match die CodecVendorID oder die Device/VendorID?

device: 48a3

vendor: 8680

-> 0x48a38680

Jetzt wurde gebootet (verständlich bei dem Kraut davor). Allerdings hat sich die Revision nicht geändert.

Was mich wundert: Der ValueType ist "Number". Sollte das nicht "String" sein?

Beitrag von „Brumbaer“ vom 14. Dezember 2018, 00:14

Der PropertyInjector ändert das Property eines "Gerätes".

Bei welchem Gerät lässt du den propertyInjector die Eigenschaft ändern ?

Bei welchen Gerät soll die Eigenschaft geändert werden ?

Beitrag von „frankferrari“ vom 14. Dezember 2018, 18:37

So, ich denke laut:

Das Devices dessen Property ich ändern möchte ist der müsste **HDEF@1F,3** (in meinem Fall) sein? Also nicht der Codec (ist kein "Device" für mich, oder doch?).

Also:

vendor-id 8680

device-id 48a3

Somit: 0x48a38680

PS: Wobei ich nirgends direkt finden konnte was HDEF bedeutet...

Beitrag von „Harper Lewis“ vom 14. Dezember 2018, 18:53

0xa3488086 müsste es sein, wenn du Properties in das Device HDEF injizieren möchtest.

Beitrag von „apfelnico“ vom 14. Dezember 2018, 20:58

HDEF - High Definition (HD Audio)

Beitrag von „frankferrari“ vom 14. Dezember 2018, 22:58

> Zitat von Harper Lewis
>
> 0xa3488086 müsste es sein, wenn du Properties in das Device HDEF injizieren möchtest.

wo genau (in IORegistry?) hast du das rausgelesen?

Beitrag von „Brumbaer“ vom 15. Dezember 2018, 01:36

Gerät bedeutet nicht zwangsweise ein Gerät im Umgangssprachlichen Sinne.

Um genau zu sein hätte ich den Audruck Service verwenden sollen.

Bei welchem Service (Eintrag im IORegistry Baum im Anzeigemode Service) kommt das Feld, das du ändern willst, vor ?

Wenn du den kennst, brauchst du seinen Klassennamen. Den zeigt dir der IORegistryexplorer auch an.

Beitrag von „frankferrari“ vom 15. Dezember 2018, 20:11

Naja, dann so gesehen, ist es für mich, wie eben ursprünglich gedacht, eigtl. der Audiocodec(Device), der "falsch" ist.

Der, dessen Revisions nicht die ist, die von AppleALC unterstützt wird.

a) richtig

b) falsch

c) vergiss es..

Beitrag von „Brumbaer“ vom 15. Dezember 2018, 20:25

Du hast einen IORegistry-Auszug gepostet. Indem kommt der Eintrag IOHDACodecRevisionID vor.

Das ist der der geändert werden muss.

In welchem Service kommt er vor und wie heisst die Klasse ?

Wo im Fenster des IORegistryExplorers steht der Service ?

Wo im Fenster des IORegistryExplorers steht die Klasse ? Du weisst wie die Klassen eines PCIDevices heisst, also könntest du ein PCI Device anwählen und schauen wo die Klasse im Fenster zu finden ist.

Beitrag von „frankferrari“ vom 15. Dezember 2018, 22:05

1. Ich habe einfach Probleme die Begriffe richtig einzuordnen..

2. Ich verstehe nicht wie ich mich auf den Codec beziehe, wenn ich das (PCI)Gerät angeben soll, da für mich der "IOPCIPrimayMatch" als einziger Punkt erscheint an dem ich den Ort des Geschehens festlegen kann.

Nach nochmaliger kurzer Studie deiner anderen Beiträge..

Richtig lesen, richtig umwandeln

Hier HDEF

Vendor-id 86 80 00 00 -> 0x8086

Device-id 48 a3 00 00 -> 0xa348

-> Match: 0xa3488086

endlich verstanden..

Richtig verstehen: “Service”

Service, Programm, Treiber...

IOProviderClass = Name eines Services als Startsignal

D.h. wird ein Service dieses Namens oder eine Subklasse davon geladen, dann startet die Überprüfung.

CFBundleIdentifier = Programmcode des Service

"zB. com.apple.driver.usb.AppleUSBXHCIPCI"

kann mehrere Services enthalten..

IOClass = “spezifiziert den Service im Bundle”

Richtig verstehen: “Klasse”

Gerät, Klasse, Art...

IOPCIClassMatch = “GeräteKLASSE”

-> GeräteKLASSE = Name d. (PCI) Gerätes

IOPCIPrimaryMatch = “GeräteID”

IORegistry, verstehen..

IOService (Plane): *Auflistung der installierten Services!* (Die an Geräte gebunden sind)

Jede Zeile auf der linken Seite ist also ein "Service"?

IOACPIPlane: Geräte..

So:

> Zitat
>
> Du hast einen IORegistry-Auszug gepostet. Indem kommt der Eintrag IOHDACodecRevisionID vor.
>
> Das ist der der geändert werden muss. Genau, verstanden
>
> In welchem Service kommt er vor (IOHDACodecDevice) und wie heisst die Klasse AppleHDAController ? IOHDACodecDevice?
>
> Wo im Fenster des IORegistryExplorers steht der Service ? Unter der Klasse (im Sinne von "Untereintrag")
>
> Wo im Fenster des IORegistryExplorers steht die Klasse ? Über der Klasse
>
> Du weisst wie die Klassen eines PCIDevices heisst (
>
> Geräte ID HDEF: 0xa3488086)
>
> , also könntest du ein PCI Device anwählen (IOPCIPrimaryMatch?) und schauen wo die Klasse im Fenster zu finden ist. ???
>
>  es tut mir Leid, dass ich so auf dem Schlauch stehe..
>
> Alles anzeigen

Beitrag von „Brumbaer“ vom 16. Dezember 2018, 00:21

Richtig verstehen: “Service”

Service, Programm, Treiber...

Dummerweise ist der Sprachgebrauch nicht eineindeutig. Wenn man von einem Service spricht kann es sein, dass man den Service im allgemeinen oder im Besonderen spricht.

Vergleichen wir ihn mit einem Programm, was ganz gut passt, denn letztendlich ist er eins.

Pages im allgemeinen ist ein Schreibprogramm mit folgenden Möglichkeiten ..... Es liegt in

meinem Programme Ordner.

Sobald ich Pages starte wird es geladen und ausgeführt und dieser laufende Code ist eine Instanz von Pages. Mit etwas Getrickse kann ich das Programm mehrmals starten und die Kopien parallel laufen lassen. Dann habe ich mehrere Instanzen von Pages gleichzeitig laufen.

So ist es auch mit Services. Es gibt den Service im Allgemeinen und dann gibt es Instanzen davon, also Code der ausgeführt wird. Es kann mehrere Instanzen des selben Service gleichzeitig geben.

Services werden programmtechnisch als Klassen realisiert. Entsprechend ist eine Instanz eines Service eine Instanz seiner Klasse.

Eine Instanz eines Service kann unter einem anderen Namen laufen, als dem Klassennamen.

IOProviderClass = Name eines Services als Startsignal

D.h. wird ein Service dieses Namens oder eine Subklasse davon geladen, dann startet die Überprüfung.

Ja, aber. Genau genommen ist es der Name der Klasse die den Service realisiert. Es ist denkbar, dass der Service und Klassenname sich unterscheiden. Und Instanzen eines Service müssen auch nicht den gleichen Namen haben wie der Service. Also muss man darauf bestehen, dass dies der Name der Klasse ist.

CFBundleIdentifier = Programmcode des Service

"zB. com.apple.driver.usb.AppleUSBXHCIPCI"

kann mehrere Services enthalten..

Richtig

IOClass = “spezifiziert den Service im Bundle”

Wieder das Namen-Problem. Es handelt sich um den Namen der Klasse, die den Service realisiert.

Richtig verstehen: “Klasse”

Gerät, Klasse, Art...

Ergibt sich aus dem Zusammenhang. Eine PCI Klasse ist etwas anderes als eine Klasse (Konzept in der Object-orientierten-Programmierung) die einen Service realisiert.

IOPCIClassMatch = “GeräteKLASSE”

-> GeräteKLASSE = Name d. (PCI) Gerätes

Nein. Das ist eine PCI-Geräte spezifische Klasseneinteilung. Sie dient dazu PCI Geräte einer Gruppe zuzuordnen. Grafikkarten, Kommunikationskarten etc.. Gibt es für eine Klasse von Geräten verbindliche Vorschriften für die Funktion und wie sie bereit gestellt wird, so kann man Treiber schreiben, die dann an Hand der Geräte Klasse geladen werden ohne dass man einen Treiber für jedes Gerät schreiben muss. man muss nicht einmal die Geräte-Id jedes einzelnen Gerätes kennen und eintragen.

IOPCIPrimaryMatch = “GeräteID”

Produkt und Hersteller ID eines PCI Gerätes. Wieder etwas was nur für PCI Geräte bzw. Services die PCI Geräten zugeordnet sind Bedeutung hat. Das gleiche Konzept gibt es u.a. bei USB Geräten.

IORegistry, verstehen..

IOService (Plane): *Auflistung der installierten Services!* (Die an Geräte gebunden sind)

Jein. Für gewöhnlich schon, aber ein Service kann auch einfach nur ein Programmstück sein, das nicht an ein Gerät gebunden ist.

Jede Zeile auf der linken Seite ist also ein "Service"?

Ja

IOACPIPlane: Geräte..

Die Einträge in der linken Spalte, sind Einträge, die macos in der DSDT gefunden hat. Das können auch Einträge sein, für die später keine Services geladen werden.

AppleALC verwendet die Eigenschaft IOHDACodecRevisionID um Patches zu identifizieren.

Deren Wert wollen wir anpassen.

Wir kennen den Namen der Eigenschaft.

Nun müssen wir rausfinden in welcher Instanz die Eigenschaft vorkommt.

Dann legen wir einen PropertyInjector Eintrag, der gestartet wird sobald diese Instanz erzeugt wird, an.

Dazu müssen wir wissen, welche Instanz die Eigenschaft enthält und von welcher Klasse es eine Instanz ist - denn wir verwenden IOProviderClass um das Erzeugen der Instanz zu erkennen und IOProviderClass will den Klassennamen haben.

Beitrag von „frankferrari“ vom 16. Dezember 2018, 14:49

Update:

Spoiler anzeigen

Klasse ist dank dir + Wikipedia etwas verständlicher geworden. Für mich sind das aktuell "Gruppennamen" für Services...

Ich frag mich trotzdem, an welcher Stelle wird die Klasse definiert?

Jeder IOClass Eintrag? Somit gibt es dann auch Unterklassen?

Also:

AppleALC verwendet die Eigenschaft „IOHDACodecRevisionID“: um den richtigen Patch zu

identifizieren.

Startbedingung festlegen mit: IOProviderClass

Und wie wäre das Matchkriterium/syntax, wenn ich mit der Annahme richtig bin, dass ich mich NICHT direkt auf das PCI Gerät beziehe?

Gedanke a):

Name der Eigenschaft: IOHDACodecRevisionID

Instanz mit der Eigenschaft: IOHDACodecDevice

Klassenname der Instanz: AppleHDAController

Gedanke b):

Name der Eigenschaft: RevisionID

Instanz mit der Eigenschaft: AppleHDAController (/CodecList/0/RevisionID)

Klassenname der Instanz: AppleHDAController

PS: IORegistry Frage

Spoiler anzeigen

Beitrag von „Brumbaer“ vom 17. Dezember 2018, 04:36

Wie gesagt eine Klasse kann alles mögliche sein z.B. ein Gruppenname, aber die Klasse die einen Service realisiert ist eine Klasse im Sinner der Object Orientierten Programmierung.

Ich habe über Stunden hinweg ohne auf Programmierkenntnisse zurückzugreifen versucht zu beschreiben was eine Klasse ist und ich bin gescheitert.

Letztendlich ist eine Klasse ein Programmstück, das Instanzen von sich erzeugen kann und definiert was die Instanzen können (Interface) und wie sie es tun (Implementation).

Da Klassen Programmstücke sind, werden sie von einem Programmierer erstellt, mit dem Ziel eine bestimmte Aufgabe zu erfüllen.

Andere Klassen oder Programmstücke können auf Instanzen einer Klasse zugreifen und mit ihnen arbeiten, denn wie das geht sagt ihnen das Klassen Interface.

Weiss man welcher Klasse eine Instanz angehört, weiss man wie man mit ihr arbeiten kann ohne zu wissen zu müssen wie sie es genau tut.

Es gibt Mechanismen in macos, die Treiber laden. Die Treiber sind Instanzen von Treiberklassen.

Die Instanzen werden erzeugt und melden dann dem Betriebssystem, dass sie da sind.

Hat man nun ein Gerät, dass eine Instanz einer Klasse lädt, die macos nicht kannte als es erstellt wurde, sieht man alt aus, denn macos kennt ja das Klasseninterface nicht, da es die Klasse noch nicht gab als es erstellt wurde.

Da kommt die Vererbung ins Spiel. Man erzeugt Klassen, die aufeinander aufbauen. Die erste Klasse ist zum Beispiel vom Typ Treiber und beschreibt in ihrem Interface alles was ein beliebiger Treiber kennen muss.

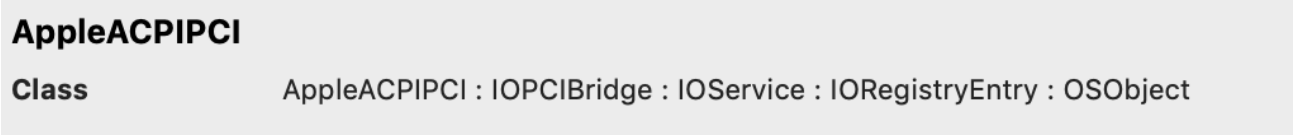
Nun legt man für bestimmte Gerätegruppen Unterklassen an. Die Unterklasse (Subclass) erbt das Interface ihrer Superklasse(Superclass).

Die Subclass erbt auch die Implementation (den Programmcode) ihrer Superklasse, kann die aber durch eigens auf sie angepassten Code ersetzen.

Bei Treibern wären das z.B. Klassen für USB, Netzwerk, Grafik. Jeder dieser Treiber hätte wieder Unterklassen, bei den USB Klassen z.B. für verschieden Chipsätze unterschiedliche Treiber.

Hat man nun einen Computer mit einem macos unbekannten USB Chipsatz und einem passenden Kext dazu, so wird über den vorhin erwähnten Mechanismus eine Instanz der Klasse erzeugt, die macos nicht kennt. Aber da es sich um einen USB Chipsatz Treiber handelt, ist er eine Unterklasse der USB Klasse und davon kennt macos das Interface und kann also damit arbeiten.

Im IORegistryExplorer kann man die Superklassen einer Klasse sehen.



Die Klasse AppleACPIPCI basiert auf IOPCIBridge das auf IOService basiert usw.

Alle Treiber sind Unterklassen von IOService, das eine Unterklasse von IORegistryEntry, das eine Unterklasse von OSObject ist.

OSObject stellt Standardfunktionalität für Objekte im Kernel zur Verfügung. Das betrifft hauptsächlich die Speicherverwaltung, also wie man Objekte anlegt und löscht.

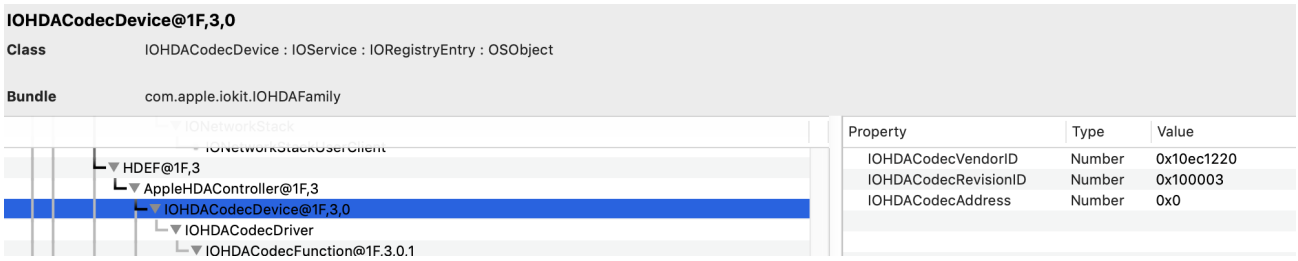
IORegistryEntry erweitert OSObject um Funktionen die ein Verwalten der Objekte in der IORegistry erlauben. Anlegen, verschieben und Löschen von Einträgen uä.

IOService fügt dann alles hinzu was jeder Service können muss. das ist ein A-voll Zeug wens interessiert, der kann's googeln.

Wir wissen das die Eigenschaft die wir ändern wollen IOHDACodecRevisionID heißt.

Diese kommt in IOHDACodecDevice vor.

Es besteht die Möglichkeit, dass die Eigenschaft von einem anderen Service übergeben wird. Aber das interessiert und erst, wenn das Ändern an der Stelle nicht funktioniert.



So an der Stelle haben wir die Instanz und die Klasse.

Die Klasse lautet ..... ?

Verwendet man IOProviderClass für das matching so wird man über jeden Service informiert, der der Klasse oder eine Unterklasse davon entspricht.

Gibt es mehrere Instanzen einer Klasse z.B. IOService, kann man die "Matches" weiter einschränken. Das ist dann der Schritt nachdem wir die Klasse bestimmt haben haben.

Ich verstehe die IORegistry Frage nicht.

HDEF@1F,3 und AppleHDAController@1F,3 sind unterschiedlicher Services.

Was meinst du mit unterschiedlichen Ansichten ? Service vs. ACPI ?

Beitrag von „frankferrari“ vom 17. Dezember 2018, 20:49

> Zitat
>
> Ich habe über Stunden hinweg ohne auf Programmierkenntnisse zurückzugreifen versucht zu beschreiben was eine Klasse ist und ich bin gescheitert.

😯 ...

Ich denke ich es jetzt wirklich gerafft. Hab mir auch noch mal nen Vortrag von einem studierten

IT Freund geben lassen. Sorry, das war mühsam.

Dank deines Screenshots ist mir die Beschreibung "Class Inheritance" und "Bundle Identifier" im oberen Teil der IORegistry jetzt erst aufgefallen, und da wird noch mal einiges klarer. Ich tendiere manchmal dazu zu oberflächlich die Dinge zu betrachten und übersehe dann wertvolle Details..

Zudem hab ich Erklärung / Bestätigung der Sub- und Superklassen gebraucht! Danke auch hierfür!

Allerdings zeigt mir IORegistry in seinem "Baum" auf der linken Seite nicht die tatsächliche Klassenhierarchie an. Sprich AppleHDAController ist gar keine Superklasse vom IOHDACodecDevice.

So. Weiter. Die Klasse lautet somit glasklar:

IOHDACodecDevice

Allerdings, wie erkenne ich ob es mehrere Instanzen gibt, wenn (wie oben beschrieben) der "Baum" nicht aussagekräftig ist. Doch nur wenn ich nach der Klasse in der Suchleiste suche, richtig?

So. Bei IOHDACodecDevice gibt es keine weiteren Instanzen. Somit macht ein matching wohl hier erstmal keinen Sinn.

Somit ist meine Eintrag in der Info.plist doch ziemlich abgespeckt, sofern ich es jetzt richtig verstanden habe.

Führt aber nicht zum Erfolg (Neustart-> immer noch die alte IOHDACodecRevisionID).

Kann es also sein, dass wie du oben bemerkt hast, der Wert von einem anderen Service übergeben wird?

(Und sorry, ignoriere die letzte Frage mit den "Ansichten". Mein unaufmerksamer (müder) Blick war verwirrt durch das aufgeblähte Feld "Type: Data". Anyway. Vergiss das einfach..)

Beitrag von „Brumbaer“ vom 17. Dezember 2018, 21:13

Alles gut.

Das Prinzip hast du erfolgreich umgesetzt.

Wenn du dir im IORegistryExplorer anschaust welchen Datentyp IOHDACodecRevisionID hat, wirst du feststellen, dass du einen anderen verwendest.

Ändere den Datentyp in den Properties entsprechend und poste dann was IORegistryExplorer für das IOHDACodecDevice anzeigt.

Beitrag von „frankferrari“ vom 17. Dezember 2018, 22:16

 juhu

das hatte ich ja schon in Post #3 angemerkt.

String -> Number

dann wird aus 0x100003 (HEX) -> 1048579 (INT)

klappt nicht. Es wurde eher eine neue "Instanz" erstellt. Aber auch nicht mit der Revision die

ich eingetragen hatte. Daher nicht verständlich...Siehe Anhang

So. Nachdem ich nun den Kext aber noch mal ganz rausgenommen habe, also ohne PropertyInjector lade, sehe ich, dass es **doch 2 Instanzen** dieser Klasse gibt. Das also nicht mal ein "Fehler" dieser Zahlenumwandlung war, sondern scheinbar immer so war.

Daher müsste ich also nun doch eingrenzen.

Dafür fehlt mir aber irgendwie der passende Syntax. IOHDAPrimaryMatch?

Es würde sich ja die IOHDACodecVendorID anbieten... oder?

Beitrag von „Brumbaer“ vom 17. Dezember 2018, 22:44

Das ist das falsche Device. Ich wollte das IOHDACodecDevice unter HDEF sehen.

Um deine Frage zu beantworten.

PropertyInjector versucht den Wert für alle Services auf die die Match Bedingungen zutreffen zu ändern. Doof wenn man mehrere hat, aber nur einen ändern will.

Die Einschränkung an Hand von IOHDACodecVendorID macht Sinn, damit man nur das eine IOHDACodecDevice ändert .

Da wie aber keine Subklasse von IOPCIDevice haben funktioniert IOPrimaryMatch nicht.

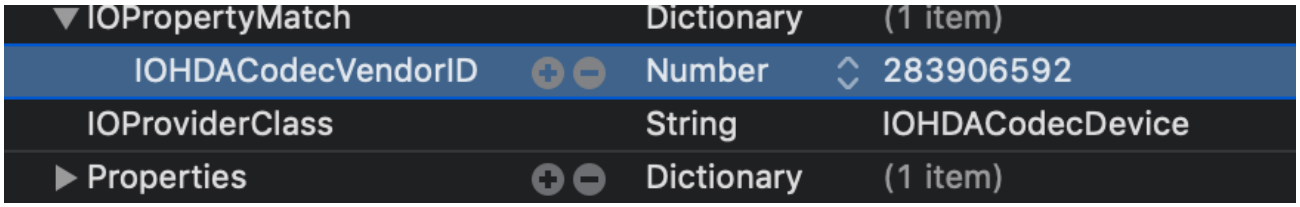
Es gibt aber die Möglichkeit auf einen beliebigen Eintrag zu matchen.

Man legt zusätzlich zu IOProviderClass einen Eintrag Namens IOPropertyMatch an. Dieser muss vom Typ Dictionary sein.

In diesem Dictionary listet man alle Eigenschaften mit dem Wert auf, den man erfüllt haben will.

In unserem Fall IOHDACodecVendorID und der Wert 0x10ec1220. Den Datentyp beachten.

Das sieht dann so aus:



XCode macht aus der Hexzahl eine Dezimalzahl, deshalb steht da 283906592.

Eine einfache Methode zu testen ob das Matching klappt ist es eine zusätzliche Eigenschaft zu definieren.

Z.B.

Name: Frank

Typ: String

Data: "was here"

Wenn das matching funktioniert wird der Eintrag im IOHDACodecDevice zu sehen sein.

Beitrag von „frankferrari“ vom 18. Dezember 2018, 00:35

PropertyMatch macht Sinn...

Gemacht.

Und wie ich sehe, sehe ich nix. Sprich, kein "Frank.." Eintrag. Und auch nicht die richtige RevisionID.

Bzw. würde ich nicht auch einfach sehen ob es geklappt hat, wenn die geänderte RevisionID eingetragen wäre

---

Beitrag von „Brumbaer“ vom 18. Dezember 2018, 01:02

Ja, aber es könnte theoretisch sein, dass es jemand wieder überschreibt. Und Frank ist sicher.

Das sollte jetzt funktionieren.

Zippe bitte das Kext und lade es hoch. Ich würde gerne überprüfen ob irgendetwas fehlt ?

Das Kext ist im Other Ordner des kexts Ordners des CLOVER Ordners des EFI Ordners der EFI Partition von der du startest ?

Du hast das Kext nicht über Clover disabled ?

Beitrag von „frankferrari“ vom 18. Dezember 2018, 08:41

So ist es.

Ich wüsste nicht, dass ich es disabled hätte. Bzw. wie das geht. Wollte ich ohne diesen Kext starten, hab ich ihn einfach aus "Other" entfernt.

Beitrag von „Brumbaer“ vom 18. Dezember 2018, 15:21

Ich habe eine gute und eine schlechte Nachricht.

Die gute ist, dass was wir bisher alles richtig gemacht haben und es mit einer Ergänzung läuft.

Die schlechte ist, ich weiß nicht warum.

Im Anhang ist dein Kext. Ich habe nur einen zweiten PropertyPatch(unter IOKitProperties) eingefügt. Dieser ist für die CPU, aber man kann auch einen anderen nehmen, aber auch nicht jeden. Ein Verdoppeln des Audio Eintrages funktioniert z.B. nicht.

Mit dem zusätzlichen Patch funktioniert auf meinem System dein Kext.

Mein Kext hat auf Anhieb funktioniert, weil in meinem Kext von Anfang an weitere Patches vorhanden waren.

Ich habe kenie Ahnung warum das so ist, vielleicht eine Racing Condition. Ich werde versuchen es rauszubekommen.

Aber zuerst möchte ich dich bitten zu testen ob das beiliegende Kext bei dir funktioniert.

Beitrag von „frankferrari“ vom 18. Dezember 2018, 17:42

Jetzt hat es geklappt... unglaublich schwere Geburt.

Jetzt heißt es nur noch die richtige LayoutID herausfinden..

(und ggf. warum man einen zweiten Patch benötigt)

Trotzdem, erstmal ein RIESEN DANKESCHÖN für die Mühe! Wirklich.

Und die ausführliche Erklärung. Dass du mir nicht den Fisch hingeworfen hast, sondern mir ein wenig das Angeln beigebracht hast.

Kann man eigtl in einer nicht all zu fernen Zukunft, in einer nicht all zu fernen Galaxy darauf hoffen, dass AppleALC den Treiber auf "unsere" Revision hin anpasst/ergänzt?

Beitrag von „Harper Lewis“ vom 18. Dezember 2018, 18:19

Sehr interessanter Thread 

~~Ich weiß nicht, ob der Autor von AppleALC auf github pull requests annimmt, aber~~ neue layout-ids werden z.B. häufig im Nachbarforum InsanelyMac gepostet (auch von MacPeet) und dann auch übernommen.

Beitrag von „frankferrari“ vom 18. Dezember 2018, 20:22

Exakt.. diesen hab ich jetzt auch installiert.

LayoutID16

funktioniert.

Auch ohne RevisionID /PropertyInject  ..

Beitrag von „Brumbaer“ vom 18. Dezember 2018, 20:58

Na ja, wenigsten haben wir uns von den Weihnachtsgeschenken abgelenkt.

Beitrag von „derHackfan“ vom 18. Dezember 2018, 21:29

Brumbaer ach komm schon, dein Einsatz in diesem Thread ist doch schon ein Weihnachtsgeschenk für die Community.