Geräte Eigenschaften (Device Properties) ohne DSDT Patch ändern.

Beitrag von „Brumbaer“ vom 12. Dezember 2017, 14:28

Nichts währt ewig
Die Vega lief OOB von Anfang an. Sie brauchte keine Hilfs-Karte, Sound funktionierte und sogar Sleep ging ohne Patches.
Die Performance war allerdings schlechter als erwartet und die OpenGL Implementierung lies eine Menge Raum für Verbesserungen.
Seit der ersten 10.13.2 Beta waren Leistung und OpenGL Unterstützung zufrieden stellend, aber der Sleep Modus ging nicht mehr.
Jetzt mit der 10.13.3 Beta 1 geht der Sleep Modus wieder, wenn auch nur mit bestimmten Framebuffern.
Dies ist der Anlass eine alternative Methode zum Verändern von Geräteeigenschaften zu beschreiben.

Häh ?
Die Hardware des Hacks besteht aus vielen Geräten (Devices) CPU, PCIe Karten, USB Ports, Kartenleser usw. Manche Geräte haben wieder Untergeräte.
Alle Geräte, die das System kennt werden in der IORegistry zusammengefasst. Jedes Gerät hat Eigenschaften (Properties), manche tauchen bei verschiedenen Geräten auf und manche sind gerätespezifisch.

Aaah
Welcher und wieviele Framebuffer verwendet werden wird anhand solcher Geräteeigenschaften bestimmt. Diese spezielle Eigenschaft findet man natürlich nur bei Graphikgeräten.

Ja, wo kommst du denn her ?
Einige Geräteeigenschaften setzt der Geräte Treiber andere stammen aus der DSDT oder werden per Software gesetzt.
Clover z.B. kann für einige Geräte bestimmte Eigenschaften und für manche Geräte beliebige Eigenschaften setzen.
Man kann Eigenschaften auch über die DSDT oder eine SSDT setzen - vorausgesetzt das Gerät

taucht dort auf.
Die Eigenschaften setzt man in der _DSM Methode des Gerätes.
Für die Grafikkarte sähe das so aus:

Code

```
Device (PEGP)
{
Name (_ADR, Zero) // _ADR: Address
Device (LTRE)
{
Name (_ADR, Zero) // _ADR: Address
Device (GFX0)
{
Name (_ADR, Zero) // _ADR: Address
Method (_DSM, 4, NotSerialized) // _DSM: Device-Specific Method
{
If (LEqual (Arg2, Zero))
{
Return (Buffer (One)
{
0x03
})
}


Return (Package (0x10)
{
"AAPL,slot-name",
Buffer (0x07)
{
"Slot-1"
},


"@0,name",
Buffer (0x0D)
{
"ATY,Kamarang"
},
```

```
36.
37. "@1,name",
38. Buffer (0x0D)
39. {
40. "ATY,Kamarang"
41. },
42.
43.
44. "@2,name",
45. Buffer (0x0D)
46. {
47. "ATY,Kamarang"
48. },
49.
50.
51. "@3,name",
52. Buffer (0x0D)
53. {
54. "ATY,Kamarang"
55. },
56.
57.
58. "device_type",
59. Buffer (0x13)
60. {
61. "ATY,KamarangParent"
62. },
63.
64.
65. "model",
66. Buffer (0x13)
67. {
68. "Radeon Pro Vega 64"
69. },
70.
71.
72. "hda-gfx",
73. Buffer (0x0A)
74. {
75. "onboard-2"
76. }
77. })
78. }
```

```
79.  }
80.  }
81.  }
82.  }
```

Alles anzeigen

Hier werden 4 Ausgänge festgelegt, die jeweils den ATY, Kamarang Framebuffer verwenden.
Zusätzlich werden Slotname, Modelbezeichnung und Soundausgaabe festgelegt.

Immer *d*ie *s*elbe *d*röge *T*abelle
Ich ändere ungerne die DSDT.
Uns liegen die DSDT Quellen nicht vor, sondern wir verwenden ein Tool (MaciASL), dass aus dem vorliegenden Code die Quellen erzeugt.
Das funktioniert nicht perfekt und man verschwendet erst einmal einige Zeit damit die Umwandlungsfehler zu beseitigen.
Oft werden einfach Patches angewandt ob man sie braucht oder nicht und die Patches überschreiben oft was existiert ohne Teile, die man vielleicht braucht zu erhalten. Kürzlich kam ich zu einem System bei dem USB nicht funktionierte und es stellte sich heraus, dass die DSDT zwei verschiedene USB Chipsets unterstützt und die Auswahl des richtigen Chipsatzes aud der DSDT entfernt worden war.

Ohne vernünftige Dokumentation fängt man beim Wechsel des MoBos von vorne an - nachdem man erstmal die ganzen Patches zusammengesucht hat.
Es gibt aber auch Gründe die DSDT Patch Methode anderen vorzuziehen und wer glücklich damit ist soll auch dabei bleiben.

Alternative Liste
Die hier vorgestellte Alternative ist ein Kext, dass die Properties aus einer Liste kopiert und einträgt.
[Edit]
Es sei nicht verschwiegen, dass es Fälle gibt in denen die Methode im Gegensatz zum Patchen der DSDT nicht funktioniert. Ein Beispiel sind FakeIDs[/Edit]

Man editiert die Info.plist des Kext und kopiert es in den EFI/CLOVER/kexts/Other Ordner der EFI Partition.

Zum Editieren der Info.plist sollte man XCode verwenden, es macht das Leben einfacher.

Nicht schon wieder Kexte

Aufbau und prinzipielle Funktionsweise eines Kextes sind in Kext as Kext Can beschrieben.

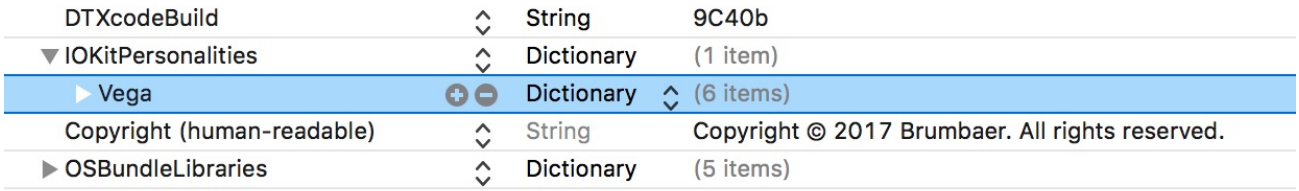
Das Kext heißt PropertyInjector.kext. Es enthält eigenen Programmcode.
Wenn man dessen Info.plist in XCode öffnet sieht man:

Key	Type	Value
▼ Information Property List	Dictionary	(20 items)
BuildMachineOSBuild	String	17C88
Localization native development re...	String	en
Executable file	String	PropertyInjector
Bundle identifier	String	de.Brumbaer.PropertyInjector
InfoDictionary version	String	6.0
Bundle name	String	PropertyInjector
Bundle OS Type code	String	KEXT
Bundle versions string, short	String	1.0
▶ CFBundleSupportedPlatforms	Array	(1 item)
Bundle version	String	1
DTCompiler	String	com.apple.compilers.llvm.clang.1_0
DTPlatformBuild	String	9C40b
DTPlatformVersion	String	GM
DTSDKBuild	String	17C76
DTSDKName	String	macosx10.13
DTXcode	String	0920
DTXcodeBuild	String	9C40b
▶ IOKitPersonalities	Dictionary	(3 items)
Copyright (human-readable)	String	Copyright © 2017 Brumbaer. All rights reserved.
▶ OSBundleLibraries	Dictionary	(5 items)

Bis auf IOKitPersonalities braucht uns das Alles nicht zu kümmern.
Im Folgenden wird deshalb nur noch der Ausschnitt mit den IOKitPersonalities gezeigt.

Man legt für jedes Gerät für das man Properties ändern oder hinzufügen will eine Personality an.

Ich will die Framebuffer meiner Vega ändern also nenne ich die Personality ... Vega, jeder andere Name hätte es auch getan, aber ich will ja ohne viel nachzudenken wissen worum es geht.



Jede Personality muss wissen welches Programmstück ausgeführt werden soll deshalb bekommt sie Einträge für IOClass und CFBundleIdentifier.

Die Werte müssen die dort angegebenen sein.

Als nächstes müssen wir das Gerät identifizieren, dessen Properties ergänzt oder geändert werden sollen.

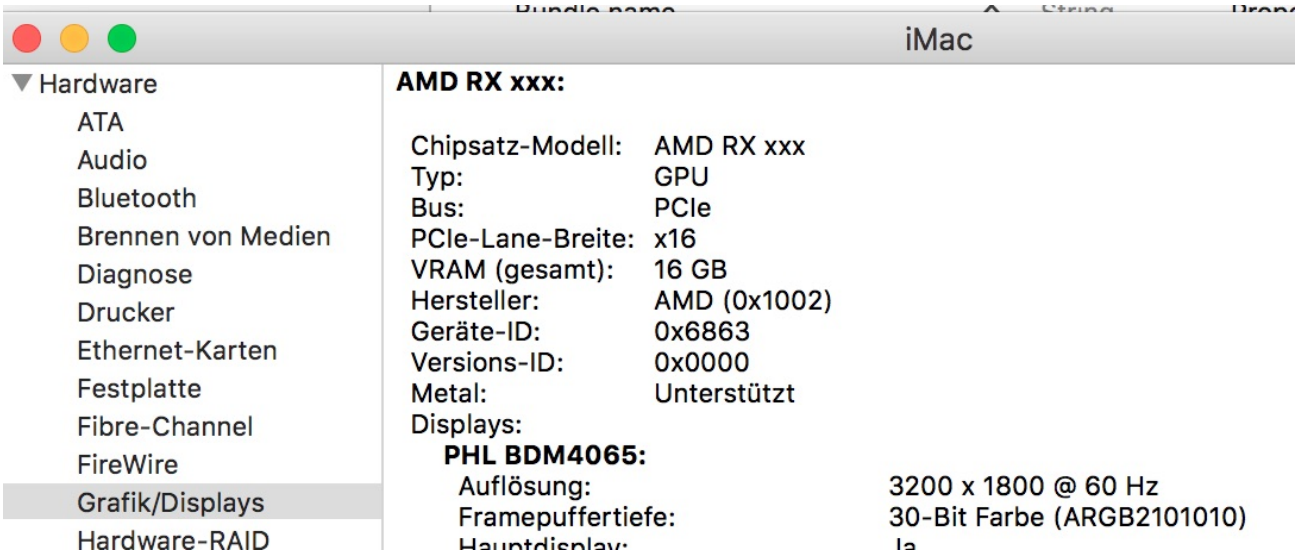
Nichts ist unmöglich

Es gibt eine Reihe von Parametern die man dazu verwenden kann und hier liegt ein Vorteil dieser Methode Properties zu editieren, man kann sie nicht nur auf Geräte, die in der DSDT vorkommen anwenden, sondern alle Geräte, die in der IORegistry auftauchen, wie z.B. USB Geräte.
Die Optionen zur Gerätewahl wären einen eigenen Artikel Wert - ein andermal.
Wir beschränken uns jetzt hier auf die Auswahl eines bestimmten PCIe Gerätes.

▼ IOKitPersonalities	Dictionary	(1 item)
▼ Vega	Dictionary	(4 items)
CFBundleIdentifier	String	de.Brumbaer.PropertyInjector
IOClass	String	PropertyInjector
IOProviderClass	String	IOPCIDevice
IOPCIPrimaryMatch	String	0x68631002
Copyright (human-readable)	String	Copyright © 2017 Brumbaer. All rights reserved.
▶ OSBundleLibraries	Dictionary	(5 items)

In diesem Fall wollen die Werte für ein PCI Gerät ändern, deshalb die IOProviderClass IOPCIDevice.
IOPCIPrimaryMatch sagt uns mittels Device und Vendor Id welches PCI Gerät gepatched werden soll..

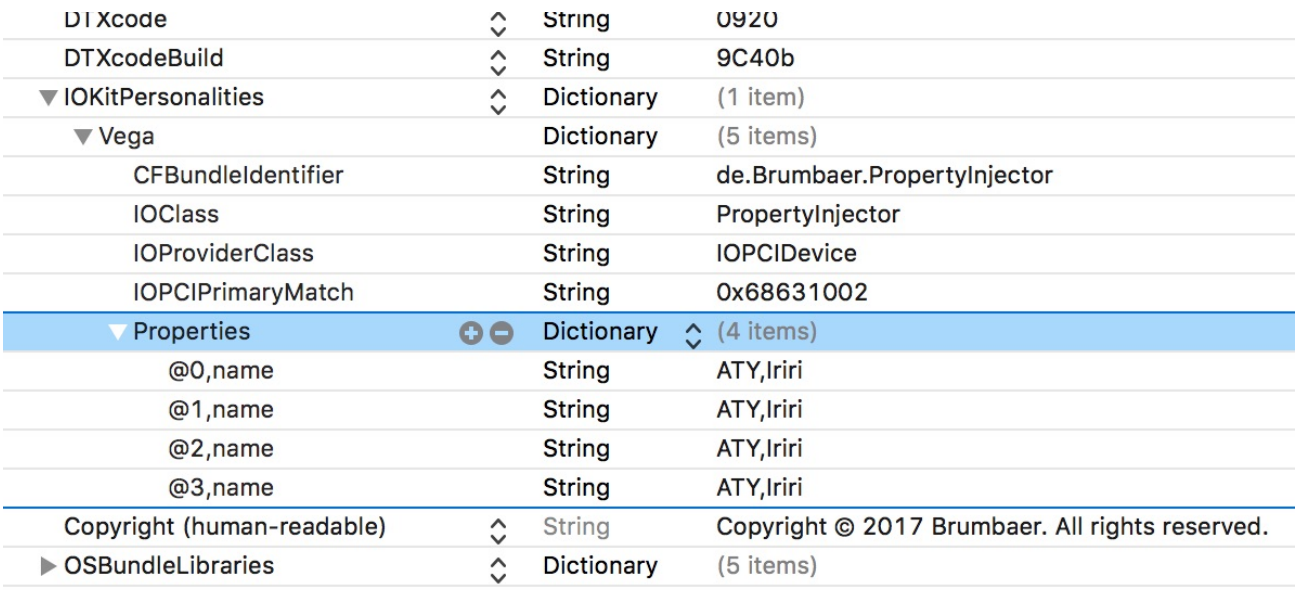
Es gibt verschiedene Möglichkeiten diese herauszufinden.
Im Systembericht unter Grafik/Displays zum Beispiel



GeräteId 0x6863 und Hersteller 0x1002 zusammengefasst zu einem Wert 0x68631002.

Eigenes Schaffen

Jetzt fehlen nur noch die Properties.
Die Properties werden in ein Dictionary mit Namen Properties geschrieben.

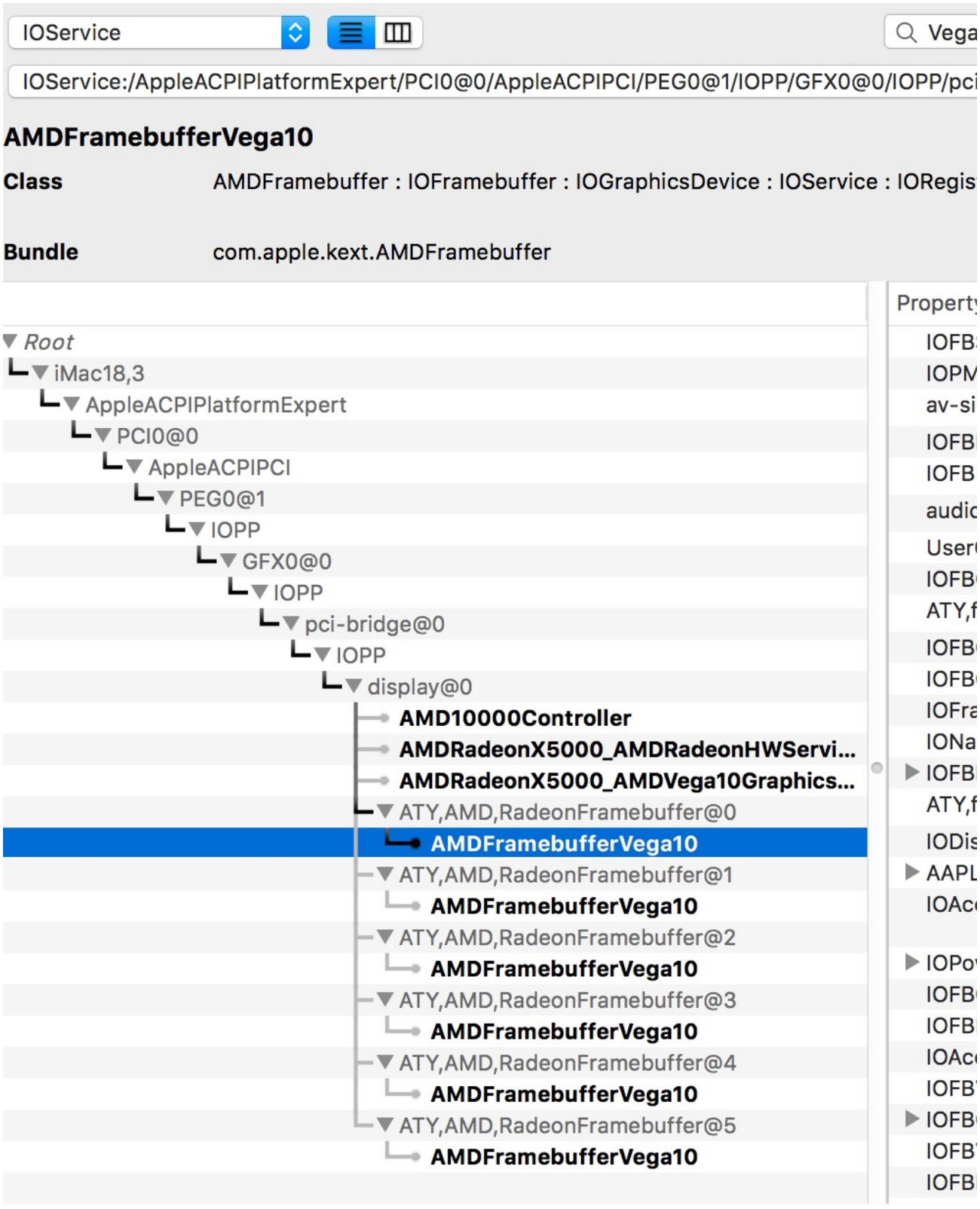


DTXcode	String	0920
DTXcodeBuild	String	9C40b
▼ IOKitPersonalities	Dictionary	(1 item)
▼ Vega	Dictionary	(5 items)
CFBundleIdentifier	String	de.Brumbaer.PropertyInjector
IOClass	String	PropertyInjector
IOProviderClass	String	IOPCIDevice
IOPCIPrimaryMatch	String	0x68631002
▼ Properties	Dictionary	(4 items)
@0,name	String	ATY,Iriri
@1,name	String	ATY,Iriri
@2,name	String	ATY,Iriri
@3,name	String	ATY,Iriri
Copyright (human-readable)	String	Copyright © 2017 Brumbaer. All rights reserved.
▶ OSBundleLibraries	Dictionary	(5 items)

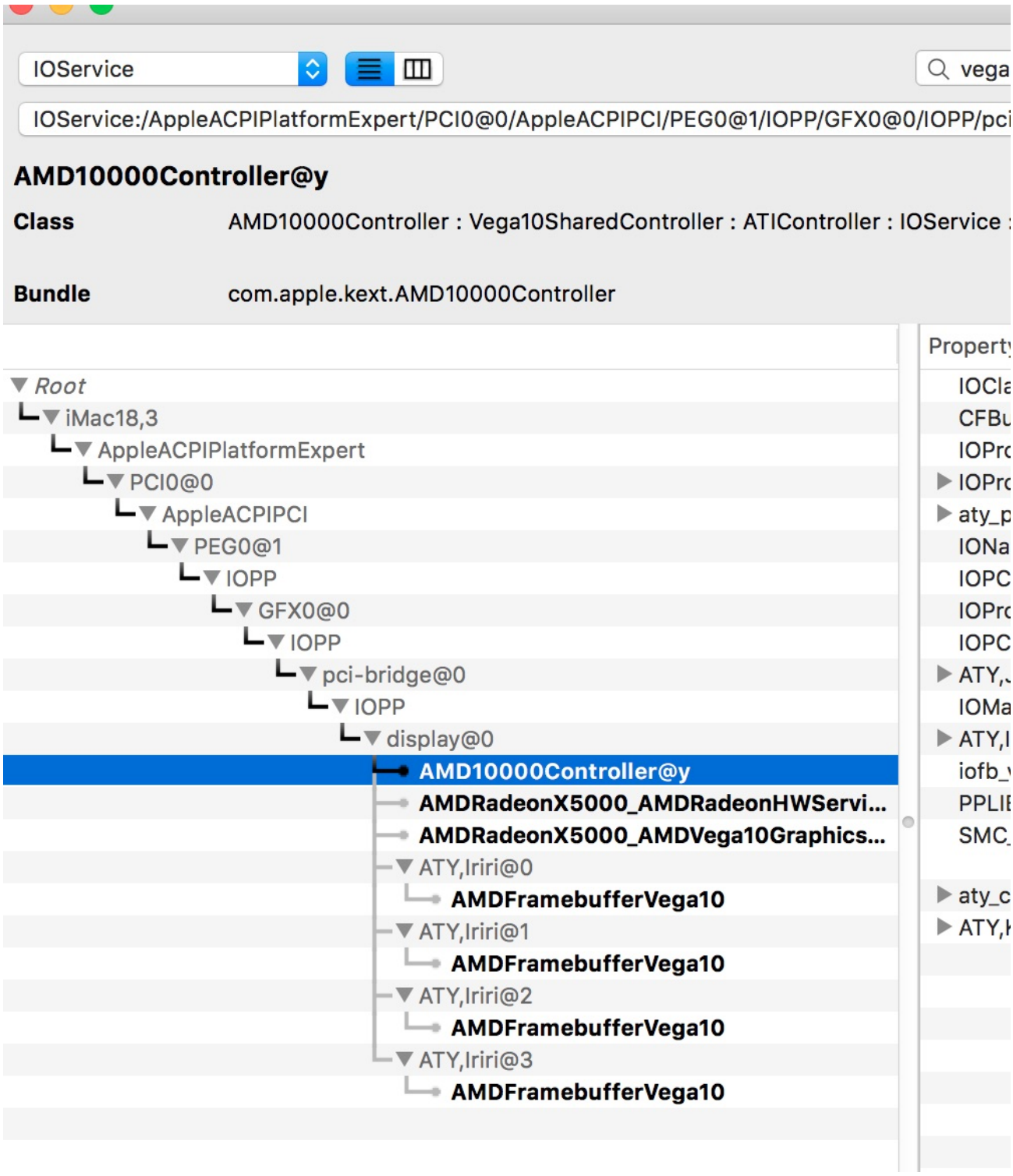
Dass die Properties @0,name bis @3,name heißen haben wir oben festgestellt. Der Framebuffer heißt ATY,Iriri und ist vom Typ String.

Neustart und fertig.

Alles roger ?
Mal sehen. Startet man den IORegistryExplorer und sucht nach Vega sieht man vor der Änderung 6 Framebuffer vom Typ ATY,AMD,RadeonFramebuffer:


IOService
Vega
IOService:/AppleACPIPlatformExpert/PCI0@0/AppleACPIPCI/PEG0@1/IOPP/GFX0@0/IOPP/pci
AMDFramebufferVega10
Class
AMDFramebuffer : IOFramebuffer : IOGraphicsDevice : IOService : IORegis
Bundle
com.apple.kext.AMDFramebuffer
Property
Root
iMac18,3
AppleACPIPlatformExpert
PCI0@0
AppleACPIPCI
PEG0@1
IOPP
GFX0@0
IOPP
pci-bridge@0
IOPP
display@0
AMD10000Controller
AMDRadeonX5000_AMDRadeonHWServi...
AMDRadeonX5000_AMDVega10Graphics...
ATY,AMD,RadeonFramebuffer@0
AMDFramebufferVega10
ATY,AMD,RadeonFramebuffer@1
AMDFramebufferVega10
ATY,AMD,RadeonFramebuffer@2
AMDFramebufferVega10
ATY,AMD,RadeonFramebuffer@3
AMDFramebufferVega10
ATY,AMD,RadeonFramebuffer@4
AMDFramebufferVega10
ATY,AMD,RadeonFramebuffer@5
AMDFramebufferVega10

Und nach der Änderung 4 vom Typ ATY,Iriri:



Das soll es schon gewesen sein ?
Weiter oben sieht man, dass die Karte als AMD RX xx erscheint.
Auch das kann man mit einem Property ändern.

DTXcodeBuild	String	9C40b
▼ IOKitPersonalities	Dictionary	(1 item)
▼ Vega	Dictionary	(5 items)
CFBundleIdentifier	String	de.Brumbaer.PropertyInjector
IOClass	String	PropertyInjector
IOProviderClass	String	IOPCIDevice
IOPCIPrimaryMatch	String	0x68631002
▼ Properties	Dictionary	(5 items)
model	Data	Radeon Vega Frontier
@0,name	String	ATY,Iriri
@1,name	String	ATY,Iriri
@2,name	String	ATY,Iriri
@3,name	String	ATY,Iriri
Copyright (human-readable)	String	Copyright © 2017 Brumbaer. All rights reserved.
▶ OSBundleLibraries	Dictionary	(5 items)

Dann erscheint die Grafikkarte als

War's das jetzt ?

Mmmmh, fast.

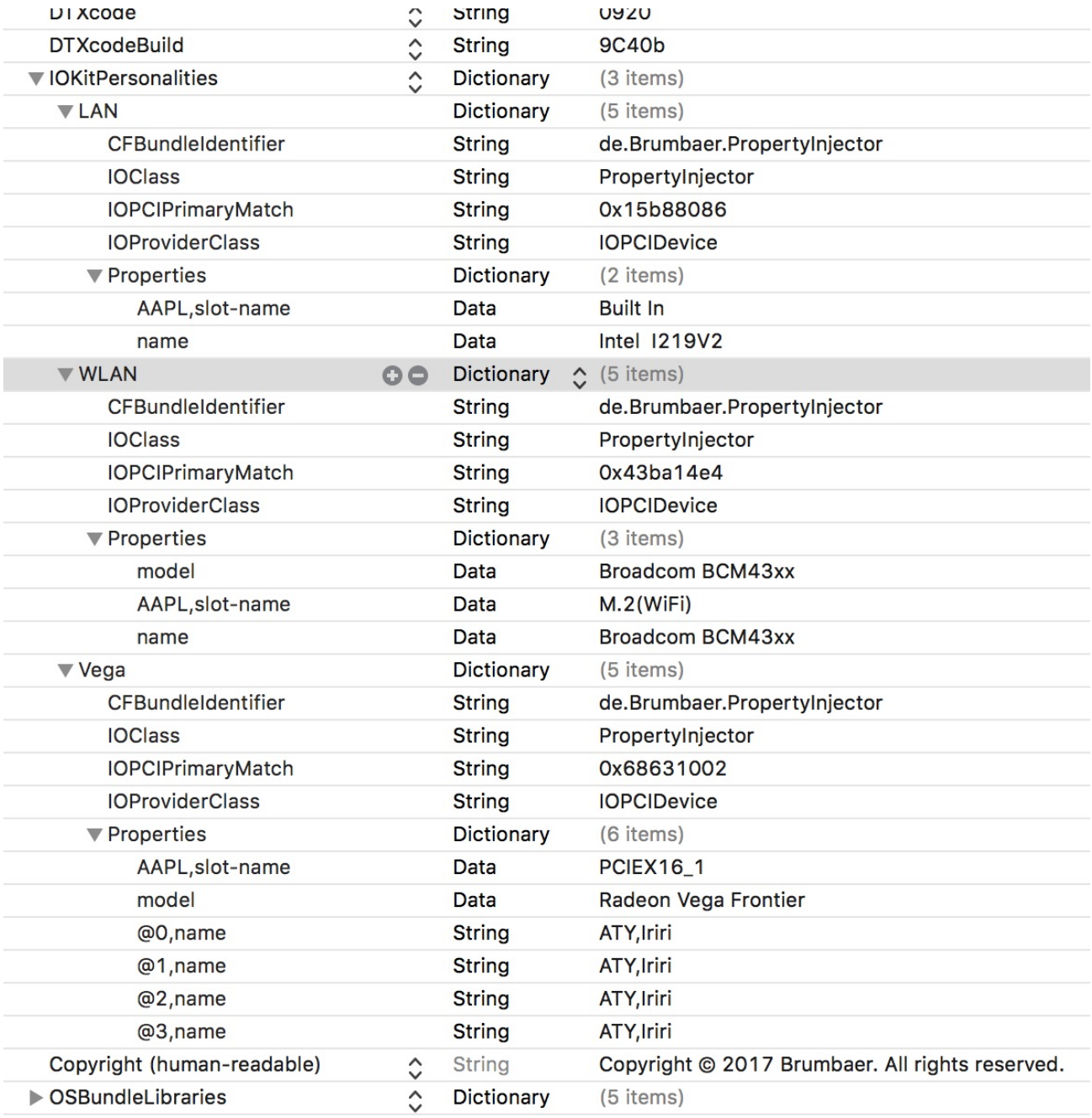
Im Systembericht erscheint unter PCI nur eine Fehlermeldung.

Möchte man, dass Geräte erscheinen benötigen sie einen AAPL,slot-name, name und model Eintrag. Je nach Gerät ist der eine oder andere schon definiert.

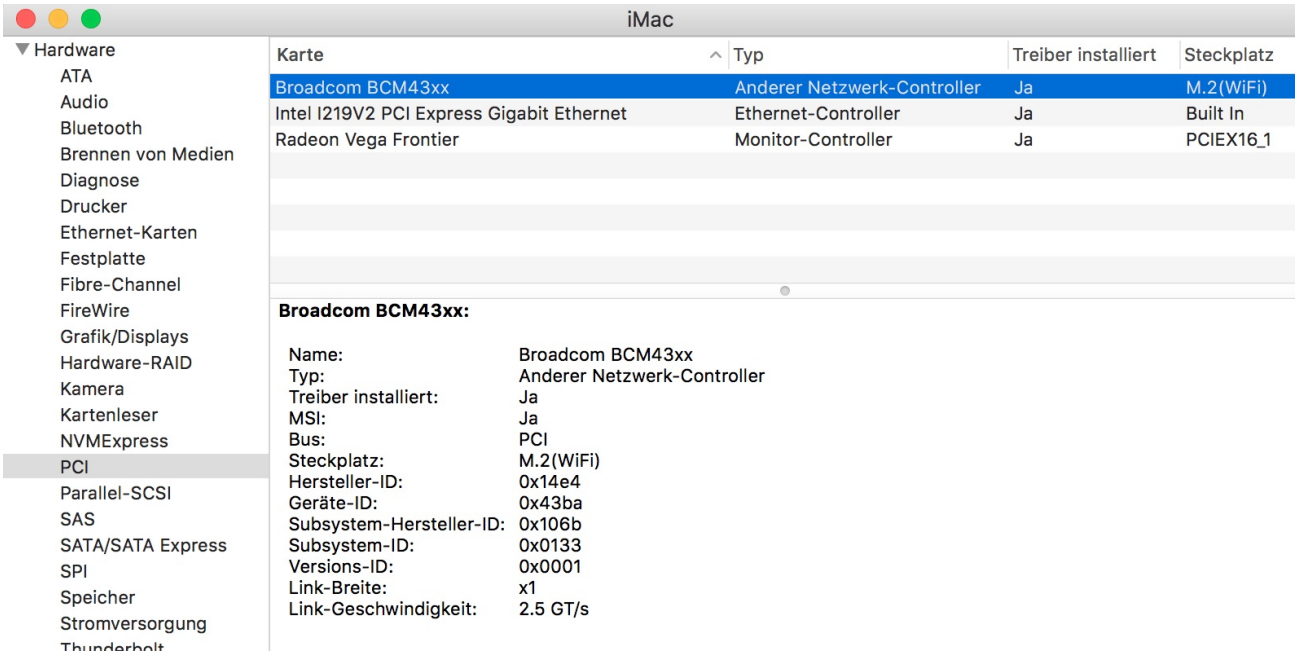
Wichtig ist, dass diese Einträge vom Typ Data sind.

Hier ist das Kext mit dem Framebuffer Patch, den Vega Frontier Patch und den "PCI Patches" für LAN, WLAN und Vega.

Da LAN und WLAN andere PCI Geräte sind benötigen sie eigene Einträge unter IOKitPersonalities.



Das sieht dann im Systembericht so aus:



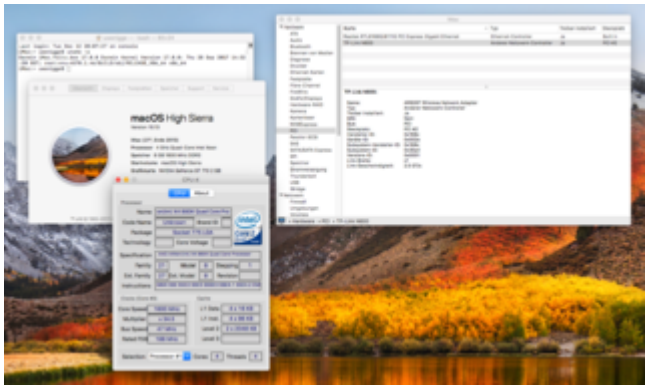
Is gut jetzt, lass es
Ok, das war's.

Falls Fragen bitte nur zur Methode, nicht welches Property wozu dient oder welches Property welches Feature beim Laptop xyz frei schaltet.
Solche Fragen gehören in den jeweiligen Problem Thread.

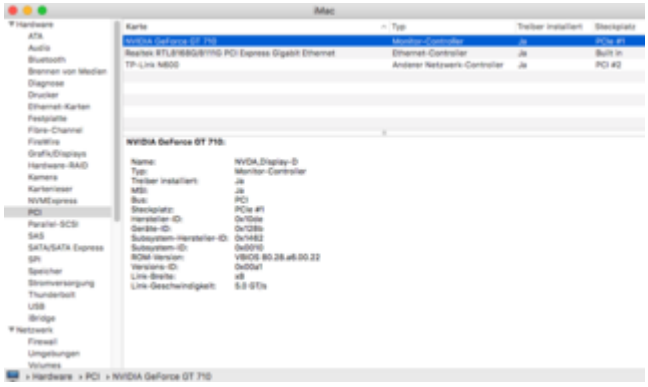
P.S.
Das Beispiel Kext
PropertyInjector.zip
Für eigene Projekte die IOKitPersonalities löschen, ändern oder ergänzen.

Beitrag von „derHackfan“ vom 12. Dezember 2017, 15:38

Richtig krasser Kram, vielen Dank für diese (deine) Methode, du verstehst es wirklich die Leute zu beeindrucken.



Edit: Das funzt sogar mit einem AMD System! 👍



Edit: Grafikkarte hinzugefügt.

Edit: Audio hinzugefügt.

Beitrag von „kuckkuck“ vom 12. Dezember 2017, 15:45

Wirklich eine super coole Sache!

Die Eingangsproblematik verstehe ich sehr gut, weshalb ich ebenfalls auf keine vorgefertigten DSDT Patches zurückgreife.

> Zitat von Brumbaer
>
> Immer die selbe dröge Tabelle
> Ich ändere ungerne die DSDT.
> Uns liegen die DSDT Quellen nicht vor, sondern wir verwenden ein Tool (MaciASL),dass aus dem vorliegenden Code die Quellen erzeugt.
> Das funktioniert nicht perfekt und man verschwendet erst einmal einige Zeit damit die Umwandlungsfehler zu beseitigen.
> Oft werden einfach Patches angewandt ob man sie braucht oder nicht und die Patches überschreiben oft was existiert ohne Teile, die man vielleicht braucht zu erhalten.

Also wenn DSDT edit, dann mit der DSDT, extrahiert aus dem BIOS und korrekt dekompiliert(zB mit der häufig benutzten refs.txt Methode) und dann von Hand, ohne Patches. Die Patchessind wirklich häufig sehr grob. Da mich das selber gestört hat, benutze ich an meinem System ausschließlich Renames und danach SSDTs, um unter anderem _DSM Properties hinzuzufügen. Also genau wie die Kext — oder auch nicht?!?
Mich würde interessieren, wie die Kext die Properties injected und zu welchem Zeitpunkt sie das tut. Ist das im Prinzip das gleiche wie _DSM SSDTs aber in grün? Oder unterscheidet sich die Funktionsweise vom injecten durch zusätzliche ACPI Tabellen? Wird hier direkt in das ACPI eingefügt oder in den Kernel oder wie soll man sich das vorstellen?
Falls das ganze auf ACPI Ebene passiert, wie sieht es mit anderem als Properties aus? Könnte man auch FakeDevices (EC, ALS0, PNLF, MCHC und diese Klassiker) einfügen oder auch _INI Methoden über die Kext injecten?

Generell ein sehr spannendes Thema und vielen Dank für diese tolle Idee und Ausführung!

Beitrag von „jboeren“ vom 12. Dezember 2017, 15:57

Pffff ich verstehe nur Bahnhof! Trotzdem sehr interessant zu lesen was es für möglichkeiten

gibt!

Beitrag von „anonymous_writer“ vom 12. Dezember 2017, 15:57

Sehr interessant und super geschrieben. Werde das mal testen.

Beitrag von „Sascha_77“ vom 12. Dezember 2017, 15:58

Krasse Sache. Kann man damit auch die IDs von rebrandeten WLan-Karten umbiegen?

Bzw. heisst das generell, das man im Grunde die DSDT gar nicht mehr dumpen und patchen muss sondern nur diesen Kext verwenden braucht?

Beitrag von „mhaeuser“ vom 12. Dezember 2017, 16:32

> Zitat von Sascha_77
>
> Krasse Sache. Kann man damit auch die IDs von rebrandeten WLan-Karten umbiegen?
> Bzw. heisst das generell, das man im Grunde die DSDT gar nicht mehr dumpen und patchen muss sondern nur diesen Kext verwenden braucht?

Hab keine Ahnung von rebrandeten Karten, aber, falls es wirklich nur um die Eigenschaft "device-id" im DT geht, dann wird das sehr wahrscheinlich gehen.
Und nein, die DSDT sollte man immer noch patchen, weil nicht jeder Patch nur Eigenschaften hinzufügt.
Wer lieber keine Kexts nutzen will, kann das ganze auch auf UEFI-Ebene mit dem DevicePathPropertyDatabase-Protokoll lösen

---

Beitrag von „Sascha_77“ vom 12. Dezember 2017, 16:58

Genau, es geht sich nur um die ID's bzw. SubID's. Dann werd ich das mal testen. Braucht statt 3 dann nur noch 1 Kext.

Beitrag von „Brumbaer“ vom 12. Dezember 2017, 17:46

Ich bezweifle, dass eine FakeID funktioniert, außer vielleicht in besonderen Fällen.

Die Routine wird angestossen, wenn das Gerät initialisiert wurde. das geschieht zusammen mit anderen Untergeräten und Treibern. Es ist anzunehmen, dass die Auswahl der potentiellen Geräte/Treiber schon erfolgt ist.

Es könnte allerdings funktionieren wenn zwischen dem Gerät und dem "Ziel des Fakes" ein paar Stufen liegen.

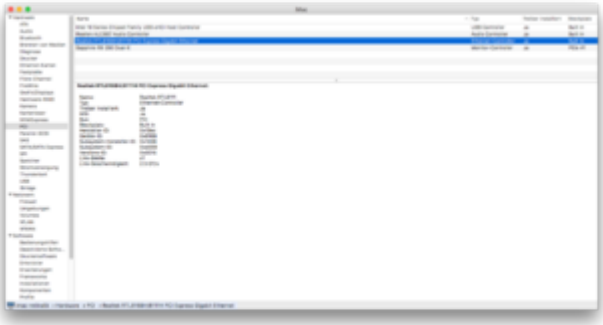
So funktioniert z.B. das "Faken" der Plugin Id um X86Platform zu laden, da zu dem Zeitpunkt zu dem Entschieden wird ob es oder die SMC Platform geladen wird, die Plug In Id schon gesetzt wird.

Es stellt sich auch die Frage ob die für den Matching Prozess der "Kinder" verwendete ID tatsächlich die Werte aus den Properties sind oder ob dazu Werte verwendet werden und die Properties nur deren "Ansicht" sind.

Lange Rede kurzer Sinn, ich denke man müsste eine aufwändigere Methode realisieren um FakeIDs erzeugen zu können.

Beitrag von „Harper Lewis“ vom 12. Dezember 2017, 18:40

Feine Sache, besten Dank dafür!



Beitrag von „Dr.Stein“ vom 12. Dezember 2017, 18:51

Ich glaub der Typ hat bei sich die absolut perfekten Vorzeigehackis stehen 😛

Beitrag von „DSM2“ vom 12. Dezember 2017, 20:36

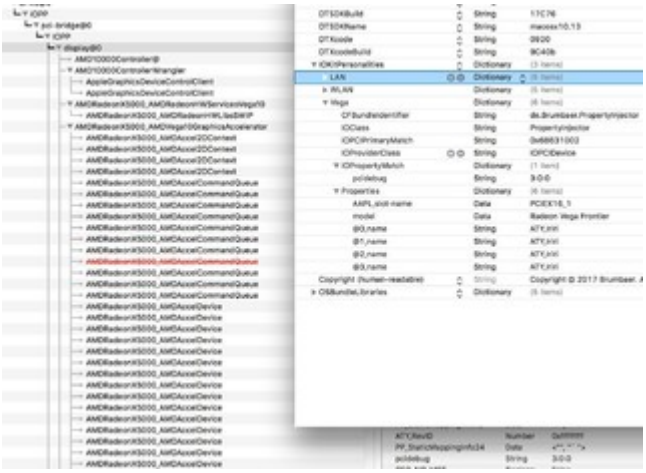
Wie ist das bei zwei Vega's ?
Sobald AAPL,slot-name 16_1 gesetzt ist, denkt die zweite Karte ebenfalls das sie in 16_1 steckt.

Beitrag von „Brumbaer“ vom 13. Dezember 2017, 01:09

[edit]Sorry, das bild war falsch[/edit]

Es gibt mehrere Ansätze, einer ist sich im IORegistryExplorer ein Property zu suchen, in dem sich die beiden Karten unterscheiden und dies zur Selection heranzuziehen.

Hier ist **pcidebug** zu sehen.
Du modifizierst den IOKitPersonality Eintrag so dass dieses Property auch zur Identifikation herangezogen wird.
Das machst du mit **IOPropertyMatch** und dem zugehörigen Untereintrag mit dem Namen des Properties (**pcidebug**). Darauf achten, dass Name, Type und Wert des Untereintrags auch stimmen.



Dann kopierst du die komplette Personality und änderst der Wert von pcidebug auf den Wert der zweiten Karte.

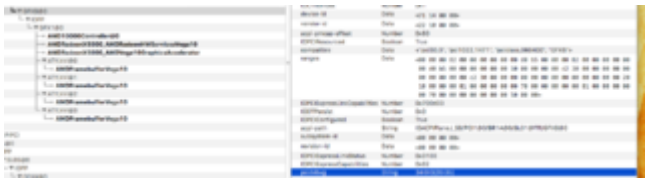
Möglicherweise kann man das eleganter über die Slots selbst lösen, habe aber gerade leine Zeit das zu testen.

Beitrag von „DSM2“ vom 13. Dezember 2017, 07:36

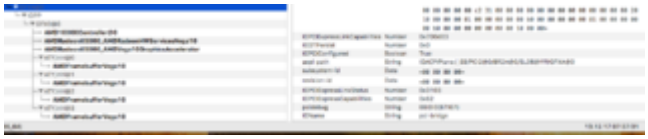
Also entweder ich mache etwas falsch oder aber es will einfach nicht.
Tippe eher auf das erste...
Könntest du dir das vielleicht mal anschauen @Brumbaer ?

Hab folgende Werte für pcidebug genutzt

GPU:1



GPU:2



Problem ist das sobald die zweite Karte eingetragen ist, beide Karten auf 16_3 stecken.

Beitrag von „Brumbaer“ vom 13. Dezember 2017, 07:54

@DSM2
Guten Morgen,
ich hatte gestern Nacht noch das Bild geändert. Das IOPropertyMatch gehört nicht in die Properties, sondern eine Ebene höher.

Ich habe es in deinem Kext angepasst.

Es wird trotzdem nicht funktionieren, da du die pcidebug Werte von GFXB verwendest, die Patches sich aber auf die jeweils zwei Einträge tiefer liegenden GFX0 und GFX1 beziehen. Deren pcidebug Werte sind andere. Da ich sie von hier aus nicht sehen kann, habe ich sie auch

nicht angepasst.

PropertyInjector.zip

Beitrag von „DSM2“ vom 13. Dezember 2017, 08:18

Guten Morgen und danke dir!
Ok, dann werde ich gleich mal auf die Suche nach den korrekten werten gehen.

EDIT: Also irgendwie finde ich keine anderen pcidebug werte ausser die ich vorher eingetippt hab.

Komisch ist auch das wenn IOPropertyMatch nicht in Properties drin ist die Karten als AMD RX xxx laufen obwohl im Kext ja anders benannt.

Beitrag von „Brumbaer“ vom 13. Dezember 2017, 08:46

Hier:

Wichtig ist, dass du den Wert vom richtigen Device nimmst.

Beitrag von „DSM2“ vom 13. Dezember 2017, 09:12

Oh man, ich bin heute morgen wohl echt blind!
Jedenfalls habe ich den Wert nicht gesehen.

Danke dir, jetzt passt alles

Beitrag von „Cheesy“ vom 13. Dezember 2017, 09:29

Hey,
erstmal muss ich wirklich den Hut vor dir ziehen. Kenne mich mit solchen Dingen mal gar nicht aus. Wie du in meinem Profil sehen kannst, besitze ich auch eine Vega. Jedoch eine 56er.
Zur Zeit benutze ich 10.13.1 (ohne Sicherheitsupdate), da mit diesem mein Hacki nicht mehr schlafen wollte.
Meine Frage ist nun:

> Zitat
>
> Framebuffer heißt ATY,Iriri

Hat die 56er den gleichen, oder wie bekomme ich das raus. Würde mich heute Abend gerne einmal daran versuchen. Des Weiteren benutze ich eine angepasste DSDT von einem anderen Forumsmitglied. Kann ich diese weiternutzen, oder muss diese entfernt werden?

Danke schon einmal für eine Antwort

Grüße aus Augsburg

Beitrag von „Brumbaer“ vom 13. Dezember 2017, 10:00

Der PropertyInjector wird nach der DSDT aktiviert und "arbeitet" mit dem was ihm die DSDT "hinterlässt". Also DSDT und PropertyInjector zusammen sind kein Problem.

In diesem Thread bitte nur Fragen zur Methode, nicht zu Properties und deren Auswirkung.

Beitrag von „kuckkuck“ vom 13. Dezember 2017, 13:20

Der Injector modifiziert also die ACPI Tabellen, die geladen werden?

Wird dabei aktiv die _DSM Methode verändert oder gesetzt? Würde sich die Kext folglich so anpassen lassen, dass sie ebenfalls auch anderen Code als nur die _DSM Methode setzt, eine Methode _INI zum Beispiel?

Beitrag von „Brumbaer“ vom 13. Dezember 2017, 13:49

Nein, der Injector verändert die ACPI-Tabellen nicht, er ändert nur die Properties in der IORegistry.
Deshalb kann man mit dem Injector keine ACPI Tabelleneinträge und somit auch keine ACPI Methoden ändern.

Die Annahme, dass Kexte mit ACPI Tabellen und Werten arbeiten ist in den meisten Fällen falsch.
Mit Ausnahme der AppleACPIIrgendetwas Treiber, verwenden Treiber normalerweise keine ACPI Tabellen oder deren Werte.
AppleACPIIrgendetwas Treiber kopieren die für sie relevanten Werte aus den ACPI Tabellen und speichern sie in der IORegistry und mit diesen Werten arbeiten dann die anderen Treiber, weshalb der Injector funktioniert ohne die ACPI Tabellen anzufassen. Ob die Werte aus einer ACPI Tabelle oder einem Zufallsgenerator stammen ist den Treibern völlig egal.

Natürlich muss die Änderung der Properties abgeschlossen werden bevor sie verwendet werden. Da die Injection nach dem Start(aus Treiber-Sicht) des Gerätes erfolgt, kann die Änderung für bestimmte Operationen zu spät erfolgen.

Das klingt schlimmer als es ist, da im Gegensatz zur landläufigen Vorstellung ein physikalisches Gerät wie eine Graphikkarte aus Systemsicht aus mehreren Geräten und Services (Treiber) besteht und bis z.B. der Framebuffer der Grafikkarte initialisiert wird, hatte der Injector schon Gelegenheit die nötigen Werte zu ändern.

Davon abgesehen sind Werte, die die Initialisierung nicht beeinflussen, wie Slot-name, model etc., völlig unkritisch und können jederzeit geändert werden. Da sie jedesmal neu gelesen werden.

Es gibt immer mal wieder ein Kext, das nicht die IORegistry benutzt, sondern die Werte anders bestimmt, da funktioniert der Injector nicht. Dass passiert auch Clover manchmal, dann funktioniert z.B. das Setzen einer FakeId durch Clover nicht und muss durch FakePCIID erfolgen.

Beitrag von „kuckkuck“ vom 13. Dezember 2017, 14:08

Top, vielen Dank für die Erklärung 👍

Beitrag von „Cheesy“ vom 13. Dezember 2017, 20:13

Hey,
habe jetzt mal das Kext auf meine Bedürfnisse angepasst. Funktioniert soweit super.
Eine Frage habe ich jedoch, ich kann mit XCode model und AAPL,slot-name nicht anpassen. Sobald ich es editiere kommt eine Fehlermeldung (siehe Screenshot).
Die IOPCIPrimaryMatch ließ eine Editierung ohne Probleme zu. Was mache ich falsch?

Danke schon mal für evtl Hilfe

Grüße aus Augsburg

Beitrag von „Harper Lewis“ vom 13. Dezember 2017, 20:15

Das sollte funktionieren: Typ von "Data" auf "String" umstellen, Text eintippen, danach wieder auf "Data" zurückstellen.

Beitrag von „Cheesy“ vom 13. Dezember 2017, 20:19

@Harper Lewis

Danke das wars.

@Brumbaer

Nochmal vielen lieben Dank. Wenn jetzt in Bayern wärst, würd ich ein Bier springen lassen (aber keine Berliner Weiße

Grüße aus Augsburg

Beitrag von „grt“ vom 14. Dezember 2017, 11:56

@Harper_Lewis für den tipp. an dem fehler hab ich auch schon diverse male festgehangen

---

Beitrag von „schluden“ vom 14. Dezember 2017, 13:07

Wahnsinn. Das wird mein Blackscreen Problem in 10.13.xx lösen. Das weiß ich.
Aber ich bleibe bei 10.12.6. Bis ich einen Monitor besitze, der HDMI Audio ausgibt

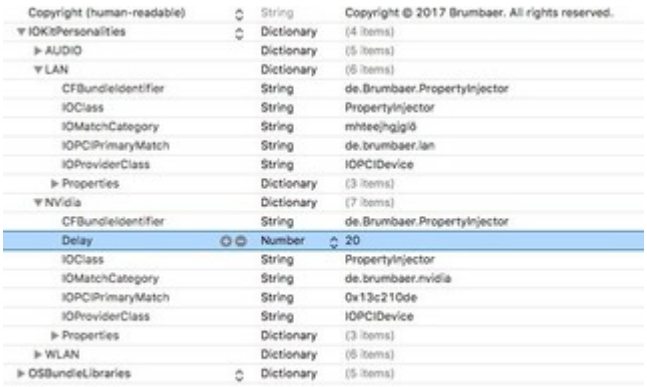
---

Beitrag von „Brumbaer“ vom 7. Januar 2018, 18:33

Version 1.1

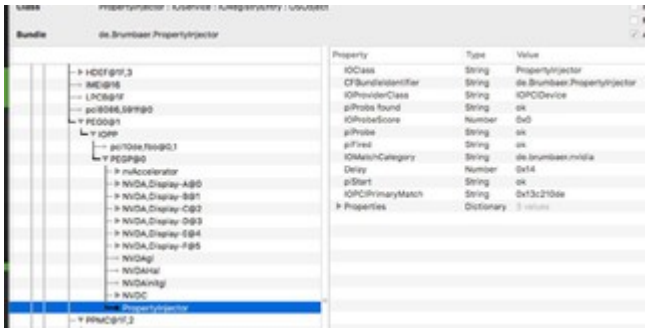
Der NVidia Grafikkarten Treiber verwendet auch das **model** Property um den Namen der Karte zu setzen.
Der Name wird allerdings nicht vom IOPCIDevice-Service eingetragen, sondern der Controller

Treiber ändert das Property des IOPCIDevice-Service.
Der Controller Treiber macht das genauso, wie das PropertyInjector Kext. Dummerweise macht der Controller Treiber das nach dem PropertyInjector Kext. Also egal was wir in das **model** Property schreiben, der Controller Treiber überschreibt es.
Deshalb gibt es eine neue Option in Form einer Eigenschaft namens **Delay** für das IOKitProperty, das nach ein paar Sekunden die Properties einfach nochmal setzt.
Boah Ey das klingt kompliziert.
Ok, erst mal sehen wohin das Property **Delay** gehört.
In den **IOKitProperties** des (Info.plist Files des) Kextes sind die Geräte(Services) gelistet für die Properties geändert werden sollen. Die Eigenschaft **Delay** soll für jedes Gerät einzeln gesetzt werden können. Es muss also in den Einträgen der Geräte deren Eigenschaften nochmal geändert werden sollen stehen.
Langer Satz deshalb ein Beispiel:



Man sieht nur der Eintrag NVidia (für die Grafikkarte) enthält einen **Delay** Eintrag.
Der Eintrag ist vom Typ **Number** und gibt die Anzahl Sekunden an nach der die Properties noch einmal geschrieben werden. In diesem Falle nach 20 Sekunden.
Es gibt allerdings ein Problem. Um verzögert schreiben zu können muss ein Treiber installiert werden. Um zu vermeiden, dass dieser Treiber einen der anderen Treibern, ersetzt muss er ein **IOMatchCategory** Property bekommen, das sich von allen **IOMatchCategory** Properties der anderen Treiber unterscheidet. Man kann z.B. den umgekehrten Namen der eigenen Webseite mit angehängter Kennung verwenden z.B. **de.brumbaer.nvidia** oder sonst irgendwas, was vermutlich kein anderer verwendet. Der **IOMatchCategory** Eintrag hat keinen Einfluss auf Einträge ohne **Delay** Eintrag, weshalb es keine Auswirkungen hat, dass er auch im Eintrag für das **LAN** Gerät auftaucht.
Das war's.

Im **IORegistryEditor** kann man den PropertyInjector Treiber(Service) dann als Kind des Gerätes sehen. Er enthält neben den Werten für dieses Gerät aus der Info.plist des PropertyInjector Kextes noch ein paar Status Variablen.



Und zu guter letzt noch das Kext:
PropertyInjector.kext.zip

Beitrag von „ammoune78“ vom 10. Januar 2018, 18:21

Hallo, sehr gute Arbeit 👍 , aber ich habe es versucht, aber es funktioniert nicht.
Können Sie bitte erklären, wie Sie es mit Xcode vorbereiten?

Vielen Dank 

Beitrag von „DSM2“ vom 18. Januar 2018, 02:57

Guten Abend!

Ich hätte noch eine Frage bezüglich des Kextes @Brumbaer
Um folgendes geht es mir dabei:

Ich habe ein neues Board für welches ich ebenfalls ein Kext erstellt habe, mein Problem ist jedoch das er mal alles korrekt aus dem Kext einließt mit den jeweiligen Slots und mal sind sie doch falsch hinterlegt.
Kann das eventuell damit zu tun haben wann er das ganze reinlädt?
Komisch ist wie gesagt das es mal alles genau passt und mal eben nicht...

Oder hab ich einen bösen Fehler drin ?

Beitrag von „Brumbaer“ vom 18. Januar 2018, 12:25

Passiert das mitten im Betrieb oder liegt jedes mal ein Neustart und/oder ein Sleep/Wake dazwischen ?

Kannst du mir bitte zwei IORegs schicken
eine aufgenommen, wenn es funktioniert und
eine aufgenommen, wenn es nicht funktioniert.

Beitrag von „anonymous_writer“ vom 2. Februar 2018, 14:28

Hallo @Brumbaer,
kann man eigentlich mit dieser Methode auch die Geräteeigenschaften eines I2C Trackpads ändern?
Es geht um ein ELAN 1200 Trackpad.
Da der VoodooI2C Kext mein Trackpad nicht ladet kann würde ich gerne den Hexadecimal APIC pin Wert von aktuell HEX 0x55 = DEC 85 ändern auf eine Wert unter HEX 0x2F.

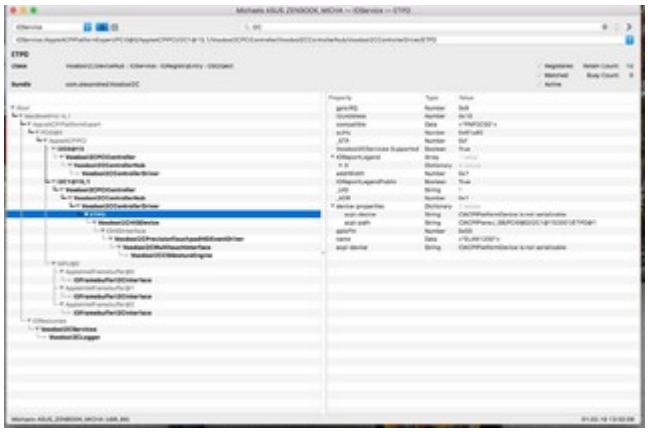
Code

1. VoodooGPIOSunrisePointLP:: pin 85 cannot be used as IRQ

Keine Ahnung ob das mit dieser Methode überhaut geht. 😕
Danke dir schon mal im Voraus für deine Mühe und diesen Thread.

Gruß Michael.

PS: es geht um den gpioPin > 0x55



Beitrag von „DSM2“ vom 2. Februar 2018, 14:33

@Brumbaer : Da lag immer ein Reboot dazwischen, komischerweise funktioniert es mittlerweile wie es soll.
Keine Ahnung was da genau los war... Jedenfalls wurde nichts danach verändert...

Beitrag von „Brumbaer“ vom 2. Februar 2018, 17:31

> Zitat von anonymous writer
>
> Hallo @Brumbaer,
> kann man eigentlich mit dieser Methode auch die Geräteeigenschaften eines I2C Trackpads ändern?
> Es geht um ein ELAN 1200 Trackpad.
> Da der VoodooI2C Kext mein Trackpad nicht ladet kann würde ich gerne den Hexadecimal APIC pin Wert von...

Ändern solltest du den Wert können, aber ob die Änderung rechtzeitig erfolgt um Wirkung zu zeigen, musst du ausprobieren.
Kommt der Wert schon in einem Gerät weiter oben in der Hierarchie vor, oder taucht er hier zum ersten Mal auf ?
Falls er schon früher auftaucht solltest du ihn dort ändern und schauen ob die Änderung durchgereicht wird.

Beitrag von „anonymous_writer“ vom 26. Juni 2018, 09:39

Hallo @Brumbaer,
ich wollte dir mal zurückmelden das ich den PropertyInjector.kext inzwischen auf meinen Zenbook und dem G4 Rechner nutze.
Mir ist jetzt erst klar geworden was der größte Voreil dieses Kextes ist. Dadurch das durch den Kext direkt das Gerät über die Device und Vendor ID angesprochen wird funktionieren damit eingerichtete Geräte unabhängig von jeglichen DSDT, SSDT und Clover Patches.

Benötigt der Kext irgendwann ein Update oder ist es so das wenn er einmal funktioniert immer funktionieren wird? Meine Frage ist bezogen auf den Quellcode des Kextes und zukünftige OSX Versionen.

Nochmals danke für das erstellen des Kextes. 

PS: Unter Mojave funktionier der Kext bis jetzt auch bestens.

Beitrag von „Brumbaer“ vom 26. Juni 2018, 11:40

Die Frage ist schwer zu beantworten.

Ohne grundlegende Änderungen an der Geräteverwaltung ist nicht mit der Notwendigkeit eines Updates auf Grund des Funktionsprinzipes zu rechnen.
Solange eine 10 am Anfang der OS Version steht ist eine Änderung sehr unwahrscheinlich.

Da Apple die „Security Schraube“ stetig anzieht mag es dazu kommen, dass der Einsatz von Third Party Kexten erschwert wird, was ein Update erforderlich machen könnte.
Ob das passiert und ob es eine Hintertür geben wird steht in den Sternen.

Beitrag von „anonymous_writer“ vom 26. Juni 2018, 11:48

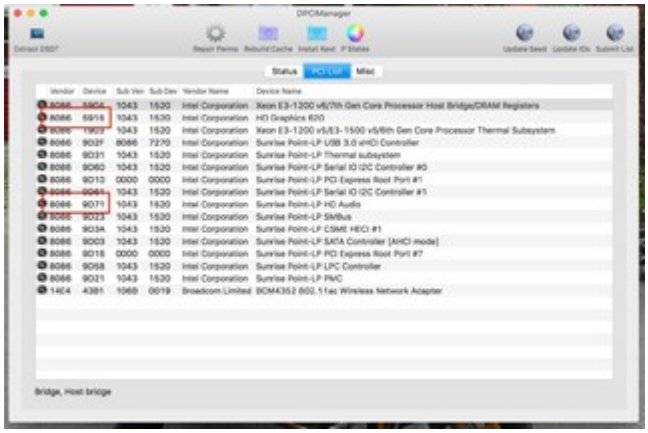
Dann warten wir mal was passiert.
Diese Art mit dem Kext ist jedenfalls eine sehr coole und effiziente Art Geräte anzusprechen und zusätzliche Eigenschaften einzuspielen.

Beitrag von „DerGiftzwerg“ vom 29. Juni 2018, 16:57

Ist der Kext immer auf die Hardware bezogen oder generell für alle nutzbar?

Beitrag von „anonymous_writer“ vom 29. Juni 2018, 20:17

Der Kext ist bezogen auf die Vendor und Device ID des Gerätes und funktioniert dann mit gleichen ID's.



Beitrag von „DerGiftzwerg“ vom 29. Juni 2018, 20:23

Das heißt, ich stelle ihn einmal ein und dann sollte er erstmal immer gehen?

Beitrag von „anonymous_writer“ vom 29. Juni 2018, 20:36

Ganz genau. Die Werte welche du injectest müßen natürlich zu dem Device passen.

Beitrag von „macdesignerin“ vom 1. Oktober 2018, 12:12

@Brumbaer Hallo, ich wünsche einen schönen Tag

Ich bin gerade beim ergänzen meiner PCI Geräteliste und dabei angekommen, Thunderbolt zum laufen zu bekommen. Es funktioniert auch, in der PCI Liste wird mir aber der Typ unbekannt angezeigt. Wie kann ich den in die plist eintragen. Kannst du mir da weiterhelfen?

Beitrag von „Noir0SX“ vom 1. Oktober 2018, 12:19

Das setzen von `device_type` sollte funktionieren

Beitrag von „apfelnico“ vom 1. Oktober 2018, 12:28

Eine SSDT könnte auch helfen. Zumal da weitere Dinge fehlen. Der XHCI im TB-Strang ist auch nicht weiter deklariert, hängt angeblich an „Airport“. Hier hilft es, die _SUN (Slot Unit Number) zu entfernen. Denn Apple ist diese bekannt und darauf sitzt am originalen Mac eben Airport.

Beitrag von „macdesignerin“ vom 1. Oktober 2018, 12:40

@apfelnico grüß dich. Ja du hast recht.
Ich habe deine EFI aus dem Mojave Thread probiert, bekomme die aber nicht zum laufen. Unsere Hardware ist ja fast identisch ASUSP Prime Deluxe x299. Ich hab jetzt eine 7940x drauf, habe auch die TSCAdjustReset.kext editiert (auf 27), bleibt aber trotzdem beim booten hängen. Hast du das 1503 Bios drauf oder noch den Vorgänger (wahrscheinlich hakt die DSDT). Und habe natürlich nur eine Vega drin mit 4 Ports (3xDP, 1xHDMI).
Wenn du mir weiterhelfen könntest wäre das super.

Beitrag von „apfelnico“ vom 1. Oktober 2018, 12:44

Bin ab heute zwei Wochen im Urlaub. Gern danach. Bekommen wir hin.

Beitrag von „macdesignerin“ vom 1. Oktober 2018, 12:47

@apfelnico super, ich wünsch dir einen schönen Urlaub.
hat Zeit, die Kiste läuft ja halbwegs ordentlich.

Beitrag von „Brumbaer“ vom 1. Oktober 2018, 15:05

Wie schon erwähnt sollte device_type helfen.

_SUN kannst du auch über den Injector ändern, allerdings nicht löschen. Muss man mal ausprobieren ob es einen geeigneten Wert gibt. Vielleicht entspricht ein leerer Data Type auch einem nicht vorhanden Eintrag.

Beitrag von „iMarc“ vom 26. Oktober 2018, 19:02

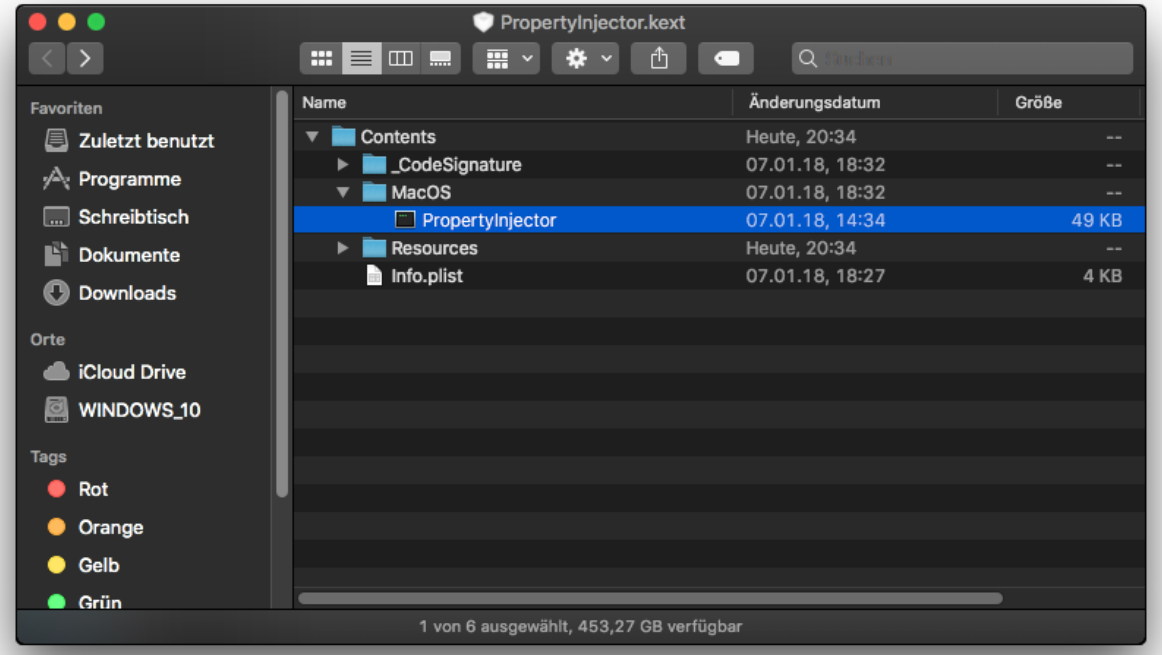
ahoi!

hat damit schon mal jemand versucht eine usb device id zu spoofen? bekomm das irgendwie nicht hin, sollte doch möglich sein oder?

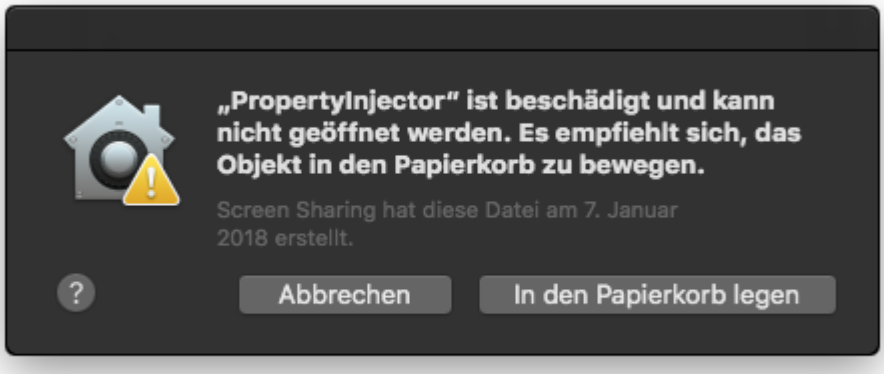
Beitrag von „Mork vom Ork“ vom 26. Oktober 2018, 20:41

Brumbaer :

darf ich fragen, was der CODE-part Deines Kexts macht?



Wann immer ich versuche, diesen via Hopper Dissasembler zu öffnen, bekomme ich folgende Fehlermeldung:



Würde für mich bedeuten, dass ich diesen Part eigentlich auch rausschmeissen kann. Liege ich mit dieser Annahme falsch?

Meine bisherigen Erfahrungen mit Injector-Kexts zeigten bislang immer nur eine Info.plist, die entsprechend angepasst werden musste.

Wenn bei Deinem Kext hier tatsächlich ausgeführter/ausführbarer Code hinterlegt ist, was macht dieser dann genau?

Beitrag von „Brumbaer“ vom 28. Oktober 2018, 00:01

Welchen Hopper verwendest du ? Ich habe es mit Hopper 4 probiert und es funktioniert ohne Probleme.

Egal.

Es ist ein normaler Treiber. Er hat IOPersonalities mit verschiedenen Matches. Wenn ein Match zutrifft wird er gestartet und bekommt die Parameter übergeben und die trägt er dann in der IORegistry bei dem entsprechenden Gerät ein.

Das ganze funktioniert nur, wenn das Gerät mit dem gematched wird auch registerService aufgerufen wird. Ohne registerService gibt es kein matching.

Ids sind kritisch, weil diese u.U. nicht aus der Registry, sondern direkt vom Gerät gelesen werden. Die Id im Gerät kann nicht geändert werden.

Allerdings greift ein Programmierer für gewöhnlich nicht direkt auf die Register zu, sondern verwendet einen IOKIt Befehl zur Abfrage der Id vom Gerät.

FakePCIId biegt diese Routine für ein PCIe Gerät um. Deshalb kann es auch in Fällen noch helfen in denen Clover oder der PropertyInjector scheitern.

Beitrag von „Mork vom Ork“ vom 28. Oktober 2018, 12:19

Danke für die Antwort:

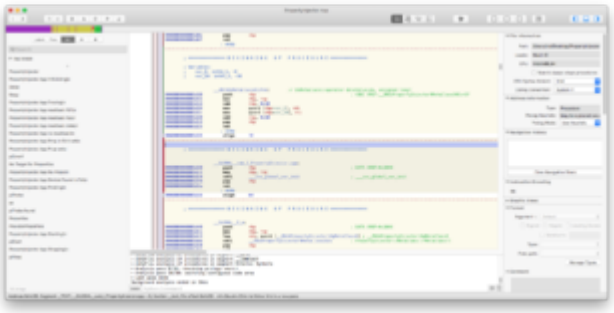
auch ich verwende Hopper in der neuesten Version 4.x (lizenzierte Edition).

Ich habe mir den Kext sowohl hier aus diesem Thread gezogen, als auch die gepostete Version von MacDesignerin aus dem "AMD RX580" Thread. Habe die Dateien jeweils unter macOS und unter WINDOWS10 gedownloaded,

aber wann immer ich versuche, diese in HOPPER zu öffnen, bekomme ich die von mir zuvor bereits angesprochene Fehlermeldung. Strange...

Aber: er funktioniert. Ich wollte mir halt nur mal den CODE-Part ansehen.

EDIT: ich habe die ZIP-Datei mal unter WINDOWS entpackt und mir unter macOS die entpackte Kext von der WINDOWS-Platte kopiert: nun kann ich den Code-Part unter HOPPER v4 korrekt auslesen. DANKE.



Beitrag von „kuckkuck“ vom 28. Oktober 2018, 12:21

Gib der Datei mal ein Suffix (zB .txt), öffne sie dann nativ (Doppelklick, das wäre dann im TextEditor) und probiers danach nochmal über Hopper. Da mag häufig die macOS Decompression von ZIPs nicht so richtig.

Beitrag von „Noir0SX“ vom 28. Oktober 2018, 12:31

Geht bei mir sogar mit der Demo Version

Beitrag von „Squallsnext“ vom 3. November 2018, 20:33

Hallo Brumbaer,

danke für den super Guide. Ich habe deine Anleitung soweit befolgt und konnte eine guten Leistungschub aus meine Karte rausholen können. Nur leider ist an meinem zweiten Monitor (zweiter HDMI Ausgang) jetzt ein Pink-Stich. Kann mir einer erklären was ich ändern kann das der HDMI Ausgang auch funktioniert?

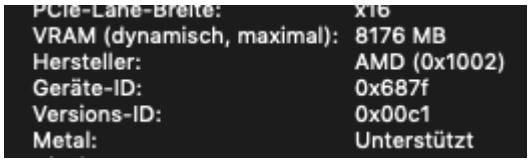
PropertyInjector.kext.zip

Beitrag von „Noir0SX“ vom 3. November 2018, 20:40

Egal was passiert sollte das <string>0x687f1002)</string> schonmal raus

Beitrag von „Squallsnext“ vom 3. November 2018, 20:43

> Zitat von Brumbaer
>
> GeräteId 0x6863 und Hersteller 0x1002 zusammengefasst zu einem Wert 0x68631002.



Laut Guide nicht?

Update:

Die klammer, habe es gesehen. Ich teste.

Beitrag von „Noir0SX“ vom 3. November 2018, 20:45

Bei Dir taucht aber ne Klammer auf) . Bin zwar nicht auf macOS, glaube aber mein Editor täucht nicht

Beitrag von „Squallsnext“ vom 3. November 2018, 20:47

Habe es draußen. Daran lag es leider nicht

Beitrag von „ductator“ vom 3. November 2018, 20:55

Pinkstich auf HDMI-Ausgang könnte eventuell ein Problem mit dem Farbprofil des Monitors sein.

Probier's mal damit: https://www.mathewinkson.com/2...ty-of-an-external-monitor

Dadurch wird ein RGB Profil erzwungen, im Idealfall löst es das Problem und man hat auch einen größeren Farbraum.

Beitrag von „Squallsnext“ vom 3. November 2018, 21:08

Danke, du bist mein Held. Ich war skeptisch habe es dennoch gewagt.

Geiler Tipp funktioniert zu 100%

Beitrag von „BS9“ vom 20. Dezember 2018, 01:01

Nabend,

mal eine Frage: Ich habe den Kext ebenfalls benutzt um die Ports für meine RX 560 zu definieren. Hat auch super geklappt, nur bewirkt die Änderung, dass bestimmte Programme

nicht mehr starten (werden direkt geschlossen ohne Fehlermeldung), unter anderem Spotify und Dota2.

Jemand eine Idee warum das passiert?