RevisionID in IOHDACodecDevice ändern (PropertyInjector.kext)

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

Richtig verstehen: “Service”

Service, Programm, Treiber...

Dummerweise ist der Sprachgebrauch nicht uneindeutig. 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.