Ozmosis BIOS Guide – individuelles Anpassen & Erklärung des Aufbaus

Beitrag von „kuckkuck“ vom 6. Oktober 2017, 21:08

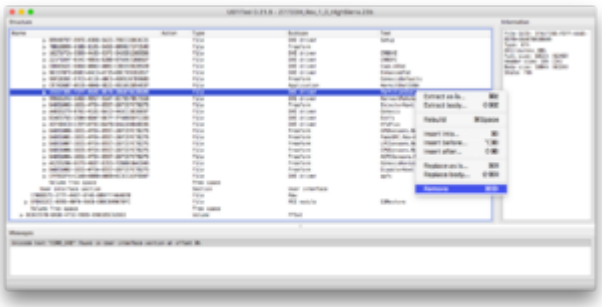
Das Ausmisten

Da ihr ja jetzt wisst wozu welche Komponente zuständig ist, können wir uns ans ausmisten machen.
Das ganze geht super einfach mit ein paar Klicks. Zuerst müssen wir uns aber im klaren sein, was wir brauchen und was wir wollen.

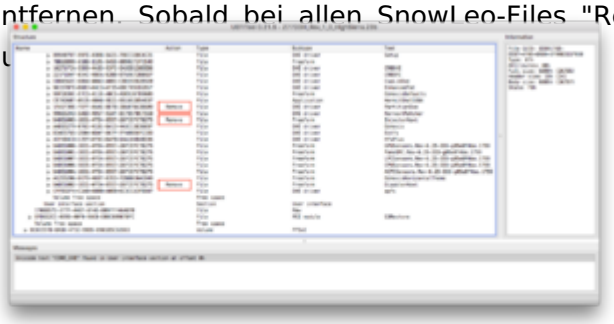
Machen wir mal das Beispiel mit SnowLeopard: Muss auf diesem Hacky Snowleopard laufen/wird es jemals benutzt werden?
Wenn deine Antwort "nein" ist, können wir alle Komponenten die für SnowLeo integriert wurden schon einmal entfernen:
`PartitionDXE`, `InjectorKext`, und `DisablerKext`

Das ab jetzt beschriebene Prozedere ist identisch für alles andere das entfernt werden soll. Wichtig ist nur, dass alle zur Ozmosis Minimalkonfiguration zählenden Komponenten nicht entfernt werden!
Wer also ExtFS, Shell, Kexts (wie das meist unbenutzte VoodooHDA) oder anderes entfernen will, kann dies mit diesem Guide ebenfalls tun.

Mit UEFI Tool können Komponenten mit einem Rechtsklick --> "Remove" entfernt werden. Das sieht dann so aus:



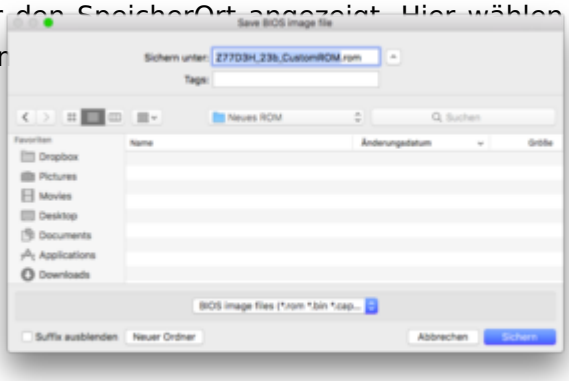
Klicken wir jetzt auf "Remove", wird die Datei noch nicht sofort aus dem ROM entfernt. Dies passiert immer erst sobald wir speichern! Das gibt uns jedoch wiederum die Möglichkeit direkt mehrere Komponenten auszuwählen, und alle SnowLeopard Dateien in einem Rutsch zu entfernen. Sobald bei allen SnowLeo-Files "Remove" ausgewählt wurde, sollte das ganze so au



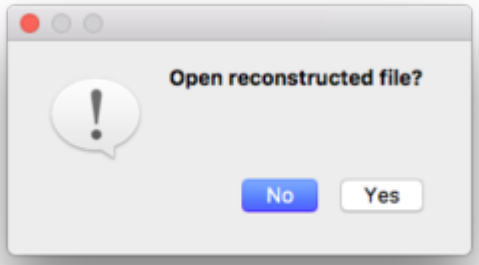
Man beachte dabei die kleinen Einträge in der

Spalte Action, die in diesem Fall "Remove" heißen (Rot markiert)
Zeit für unseren ersten Test, um zu sehen ob das auch so klappt.

Wir drücken also CMD+S oder gehen auf File --> "Save image file..." und uns wird ein Fenster für den SpeicherOrt angezeigt. Hier wählen wir einen Ordner und Dateinamen aus, was dann zur



Klicken wir jetzt auf "Sichern", erscheint nach



kurzer Zeit folgende Meldung:

Hier immer "Yes" auswählen, denn nur so können wir überprüfen ob alles richtig gelaufen ist!

Sobald das frisch gebackene ROM geöffnet ist, müssen wir auf die Sektion "Messages", am unteren Ende des UEFI Tools schauen. Stehen hier **keine Meldungen** hat alles gut funktioniert und unser ROM ist OK. In diesem Zustand könnte es jetzt direkt geflasht werden! Tauchen jedoch Meldungen auf, ist dies kein gutes Zeichen und es ist etwas schief gelaufen. Falls diese Meldungen nach einem weiteren Versuch wieder auftauchen, oder sie bereits im ROM aus der Datenbank erscheinen, könnte es sich um unwichtige Meldungen handeln. Habt ihr Fragen dazu, könnt ihr einfach eure Meldung in diesen Thread schreiben.

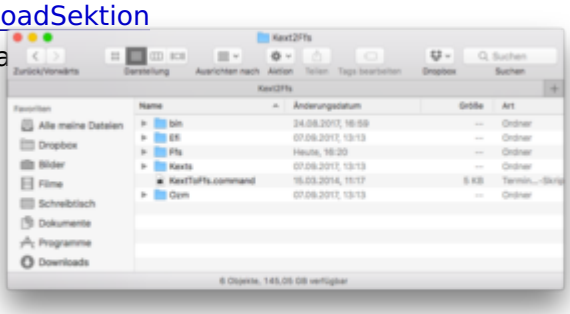
Aber wir sind noch nicht fertig, denn da geht noch einiges mehr!

Das Hinzufügen

Durch das Entfernen von ungenutzten Komponenten, kann Platz im ROM für neues geschaffen werden. Manche ROMs haben von Haus aus genug Platz um noch weiteres aufzunehmen, trotzdem lohnt es sich immer komplett ungenutzte Dateien aus dem ROM zu entfernen.

Praktisch an UEFITool ist, dass wir automatisch davor beschützt werden bestimmte Fehler zu machen. So kann zB nicht zu viel in das ROM eingefügt werden. Versuchen wir eine Datei hinzuzufügen, die den Speicherplatz sprengt, wird uns das Tool beim Speichern eine Fehlermeldung ausgeben und das ROM nicht speichern lassen. Wir brauchen uns also nicht darum kümmern, dass das BIOS zu voll wird, denn wenn wir etwas hinzufügen wollen, was zu groß ist, werden wir das schon merken. Dazu aber später mehr.

Wie kann man jetzt Dateien zum ROM hinzufügen?
Dies geht nur mit Dateien die im Flash-File-System - Format vorliegen, kurz `.ffs`
Eine Defaults im Plist Format, lässt sich also nicht ohne weiteres zum ROM hinzufügen...

Um Dateien zu FFS zu konvertieren, wird häufig das Tool "Kext2FFS" benutzt, welches ihr hier herunterladen könnt: DownloadSektion
Heruntergeladen und entpa  einfachen Finder Ordner handelt, der so aussieht:

Jeder Unterordner hat eine eigene Funktion:
In `bin` sind alle Executables untergebracht, die das Kext2FFS Skript verwendet. Dieser Ordner ist für uns relativ uninteressant.

In `EFI`, `Kexts` und `Ozm` können wir unsere Dateien hinterlegen, die zu FFS umgewandelt werden sollen.

In `EFI` können .efi Dateien gelegt werden. Dazu zählen zB FileSystem-Treiber wie APFS.efi, UEFI Applications wie Shell.efi oder auch DXE-Treiber wie Ozmosis.efi selber. Bei Problemen nach der Umwandlung und anschließenden Integration ins Rom, lohnt sich manchmal die Nutzung folgender Version: Kext2FFS

Diesen Ordner braucht Otto NormalVerbraucher nicht häufig, da die meisten eigentlichen .efi Dateien, auch bereits als .ffs gefunden werden können, so zB Ozmosis oder der HFSPlus, die man häufig direkt als Ozmosis.ffs oder HFSPlus.ffs finden kann.

Weiter gehts mit dem Ordner `Kexts`. Wie der Name schon sagt, können wir hier jede beliebige Kernel-Extension zu FFS konvertieren lassen, um sie danach ins BIOS zu integrieren.
Hier aber nochmal die Warnung: In das BIOS sollten nur die nötigsten Kexts eingefügt werden, da sie das Hochfahren verlangsamen können. Was sich gut anbietet, ist mit diesem Tool die neueste FakeSMC sowie die zugehörigen Plugins nach .ffs zu konvertieren, um diese ins BIOS einzufügen. Wer will kann sich auch einen Lan Treiber ins BIOS packen, um Lan bei Neuinstallationen zu haben, man beachte aber, dass dies wie oben erwähnt nicht immer die beste Idee ist.

Schließlich gibt es noch den Ordner `Ozm` mit dem sich bei dieser Kext2FFS-Version .plist Dateien nach FFS umwandeln lassen. (Der Ordner kann in manchen Fällen auch "OzmDefault" lassen und neben dem Ordner "Ozm" stehen. In diesem Fall einfach "Ozm" ignorieren und "OzmDefault" nutzen.)
Dieser Ordner wird in erster Linie dazu benutzt um eine Defaults.plist nach FFS umzuwandeln. Falls man seine eigene Defaults in sein ROM integrieren will, muss diese in oben genannten Ordner gelegt werden und das Ende (Suffix) .plist besitzen.

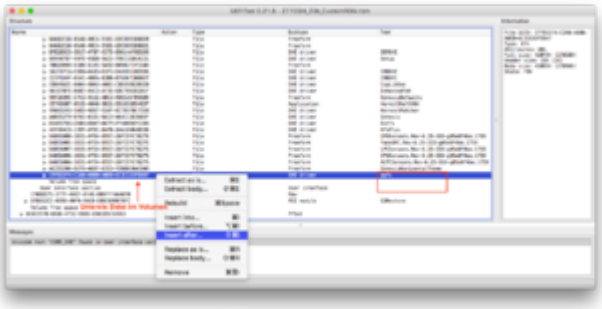
Jetzt gibt es nur noch 2 Dinge in Kext2FFS: Der `Ffs` Ordner und der `KextToFfs.command`:
Der `KextToFfs.command` ist das Skript, dass alle unsere hinterlegten Dateien nach .ffs umwandelt, sobald wir es ausführen. Das Skript läuft dabei im Terminal und gibt uns noch ein paar Informationen aus.

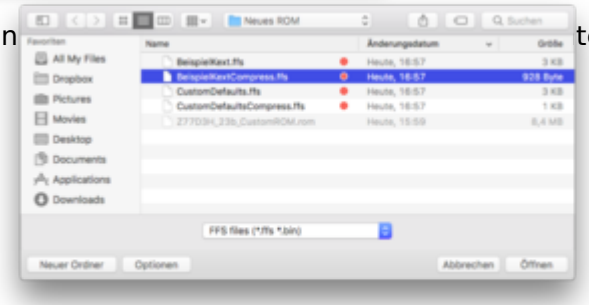
Sobald der Befehl dann ausgeführt wurde (doppelklick), erscheinen in dem Ordner `Ffs` in dem jeweiligen Unterordner `Efi`, `Kext` oder `Ozm` unsere erstellten FFS Dateien, jeweils einmal in normal, und einmal in einer komprimierten, kleineren Form, im UnterOrdner `Compress`. Die Dateien werden dabei nach dem OriginalDatei-Namen benannt. Heißt unsere Original-Datei also "Spongebob", wird die erstellte Datei ebenfalls "Spongebob" heißen, und das auch im UEFITool und boot.log!

Haben wir jetzt also unsere Dateien, die eingebaut werden sollen, im FFS Format, können wir uns auch schon ans einbinden machen. In den meisten Fällen spricht nichts dagegen einfach die `Compress`-Variante einzufügen, da diese häufig wesentlich kleiner ist.

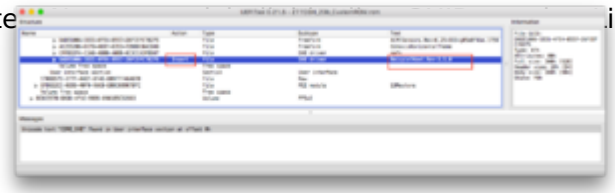
In meinem Fall werde ich nun eine Custom- Defaults.plist und eine BeispielKext einfügen. Dafür haben ich diese mit Kext2FFS konvertiert und die .ffs Dateien in meinen Ordner verschoben.

Sobald UEFI-Tool geöffnet ist, gilt wieder das Standardprozedere: BIOS-File auf das ProgrammFenster ziehen, nach `CORE_DXE` suchen und zu den Ozmosis Komponenten scrollen.
Hier angekommen müssen wir jetzt nur noch die Dateien einfügen. Dafür wählen wir die unterste Ozmosis-Datei im ausgewählten Volume aus, machen einen Rechtsklick und wählen "Insert after...":

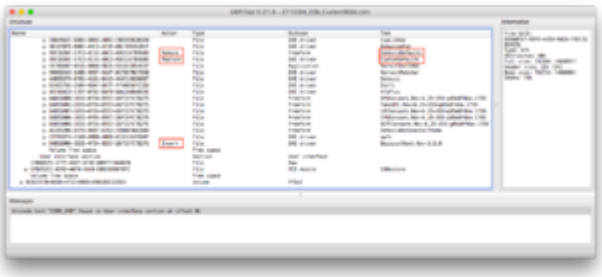


Im erscheinenden 

teien in der Compress-Version und klicken öffnen:

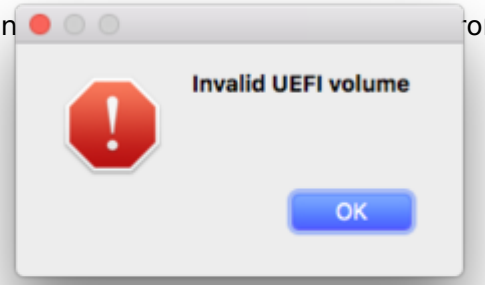
Dadurch wird die Datei mit unserem gewählte  iste hinzugefügt und die Action "Insert" hinterlegt:

Bevor wir speichern, ersetzen wir jetzt noch die Standard `OzmosisDefaults`, mit unserer Defaults.plist, in meinem Fall CustomDefaults.
Dazu suchen wir die Datei mit dem Namen "OzmosisDefaults", machen einen Rechtsklick und wählen jetzt "**Replace as is...**"!
Im erscheinenden Fenster wählen wir jetzt unsere Defaults.ffs und bestätigen das ganze. Daraus resultiert folgende Gesamtsituation:



Jetzt das ganze, wie bereits gelernt, speichern und das neu erstellte ROM auf Messages überprüfen. Wenn alles gut gelaufen ist, haben wir jetzt ein Custom angepasstes ROM.

Sind die Dateien



oß, erhalten wir eine Warnung, die wie folgt aussieht: --> das ROM wird nicht gespeichert.

Sowie die Warnung in Messages: `reconstructVolume: root volume can't be grown`. Die Komponenten sind also zu groß und es muss entweder ausgemistet, oder verzichtet werden.

Auf diese Art und Weise könnt ihr sowohl euer ROM personalisieren, als auch Updaten wenn es neue Versionen von bestimmten Komponenten gibt. Erhält zum Beispiel Ozmosis ein Update, oder wird APFS aktualisiert, könnt ihr mit der "Replace as is..."-Funktion ganz einfach euer ROM updaten und daraufhin neu flashen.

~~Soweit erst mal dazu, in nächster Zeit werde ich evtl noch eine Ergänzung veröffentlichen. Bis dahin~~

👍

Weiter gehts: Ozmosis BIOS selber bauen