Take My Layout, AppleHDA!

As soon as Mountain Lion turned GM last year, I went about making a super-vanilla hackintosh (all Apple native kexts, some minor patches for device compatibility). Everything worked, except audio. The usual forums were bare due to the recent release, so I went about trying to apply the Lion methods of getting an Realtek ALC898 working in Mountain Lion. It didn't work. Things in AppleHDA had changed. Layout XML files, that contain audio pathmap parameters, were now gzip compressed. Adding your own custom gzipped layout simply wouldn't work like it used to. In Lion (10.7), it was possible to modify the "layout-id" variable on your HDEF device in your DSDT, then drop the appropriate custom named layout file in the AppleHDA kext. For example: You would set "layout-id" in DSDT to "**0x79 0x03 0x00 0x00**" and then drop a layout889.xml (your custom pathmap) into AppleHDA (note that 0x79 0x03 is 889(dec) using a little-endian byte order).

I couldn't figure it out. I tried disassembling AppleHDA and trying to find where things were going wrong based on the vague "Sound assertion" messages spewed into the syslog. Eventually, I gave up.

This weekend however, I revisited the problem. This time, on 10.8.4. There's a lot more information floating about this time around, but no one had _really_ solved the issue, just worked around it. These days, to get audio working in Mountain Lion, you change your layout-id in DSDT to a layout that _already_ exists in AppleHDA. For example: layout1.xml.zlib already exists in AppleHDA, so your "layout-id" will be 0x01 0x00 0x00 0x00, then you override layout1.xml.zlib with your own pathmap.

I never even bothered to try this, since it would require you to then replace one of Apple's pathmap files with your own. Essentially, you override a system file that might get used by other audio codecs (NVIDIA/Intel/AMD HDMI audio, maybe?) At the time, I really wanted to keep my custom pathmap separate, so it could run side-by-side with all of the others…in the spirit of being "vanilla". Chances are, the pathmap you override won't be used, but I was never satisfied with the solution. I decided to re-investigate. Here I detail how I went from a syslog message, to figuring out why AppleHDA rejects your custom gzipped layout

digitaldj.net
Computer scientist. Gastronome. Avid traveller. Purveyor of logic

```
localhost kernel[0]: Sound assertion in AppleHDADriver at line 1118
```

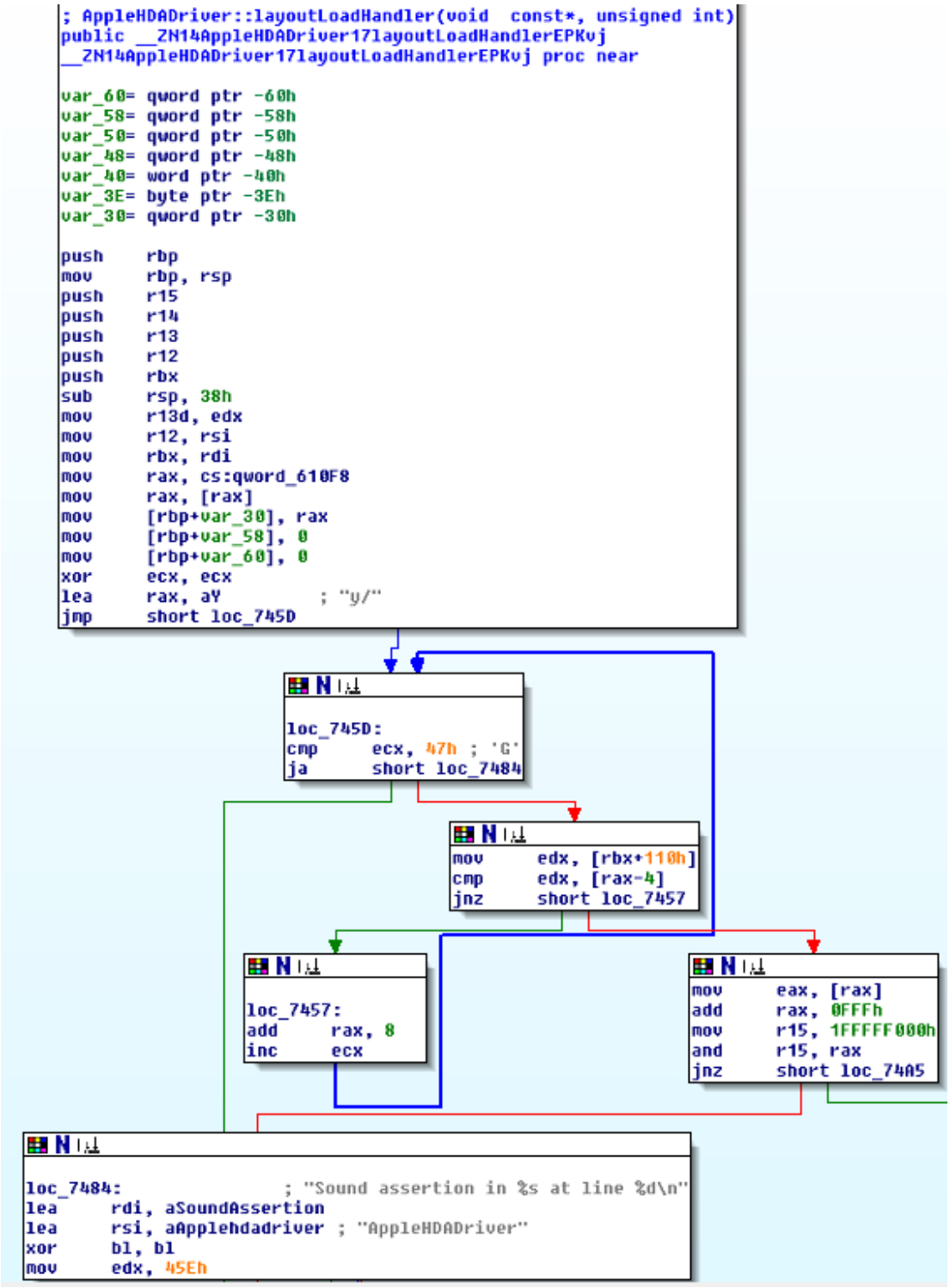
Okay, line 1118 huh. I don't have the source, so let's fire up IDA. Maybe… search for "Sound assertion":

Address	Function	Instruction
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	loc_1893: ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	loc_18AB: ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	loc_1EF0: ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"
__text:00000000000...	__ZN20AppleHDACod...	lea rdi, aSoundAssertion ; "Sound assertion in %s at line %d\n"

Fantastic, which could it be!? I guess I could narrow it down with the line number. Looks like the line number gets moved into edx for the format string. Guess I can do a search for **mov edx, 45e** (1118d).

```
loc_7484:                    ; "Sound assertion in %s at line %d\n"
lea     rdi, aSoundAssertion
lea     rsi, aApplehdadriver ; "AppleHDADriver"
xor     bl, bl
mov     edx, 45Eh
```


Okay! Now let's take a look at what instructions brought us here.



Looks like we're in a function **layoutLoadHandler** Seems appropriate. Looks like it enters some sort of loop and breaks out when it finds something, and if it doesn't we get the Sound assertion error. But what the hell is it looping over?! This is where I got stuck last year. Last time, I tried modifying some instructions to jump out of the loop and force success… which was only met with a kernel panic *to the face*. Then it finally hit me. What is that just before we jump into the loop? I'm referring to the **"y/"**. Looks like we're

loading some constant address into **rax**. What the hell is at **“aY”**?

```
__const:0000000000060402                db    0
__const:0000000000060403                db    0
__const:0000000000060404 aY             db 'y/',0               ; DATA XREF: AppleHDADriver::layoutLoadHandler(
__const:0000000000060407                align 8
__const:0000000000060408                db    1
__const:0000000000060409                db    0
__const:000000000006040A                db    0
__const:000000000006040B                db    0
__const:000000000006040C                db  87h ; ç
__const:000000000006040D                db    4
__const:000000000006040E                db    1
__const:000000000006040F                db    0
__const:0000000000060410                db  0Ah
__const:0000000000060411                db    0
__const:0000000000060412                db    0
__const:0000000000060413                db    0
__const:0000000000060414                db  7Fh ; ■
__const:0000000000060415                db  38h ; 8
__const:0000000000060416                db    0
__const:0000000000060417                db    0
__const:0000000000060418                db  0Bh
__const:0000000000060419                db    0
__const:000000000006041A                db    0
__const:000000000006041B                db    0
__const:000000000006041C                db  7Ah ; z
__const:000000000006041D                db  4Eh ; N
__const:000000000006041E                db    1
__const:000000000006041F                db    0
__const:0000000000060420                db  0Ch
__const:0000000000060421                db    0
__const:0000000000060422                db    0
__const:0000000000060423                db    0
__const:0000000000060424                db  7Ch ; |
__const:0000000000060425                db  1Bh
__const:0000000000060426                db    0
__const:0000000000060427                db    0
__const:0000000000060428                db  0Dh
__const:0000000000060429                db    0
__const:000000000006042A                db    0
```

Looks like a whole load of nothing. It isn’t, by the way. Align 8….is this an array?! 8 bytes per element? Oh yes, it is. Let’s see what that gives us every 8 bytes:

```
01000000 (1) 87040100
0A000000 (10) 7F380000
0B000000 (11) 7A4E0100
0C000000 (13) 7C1B0000
```

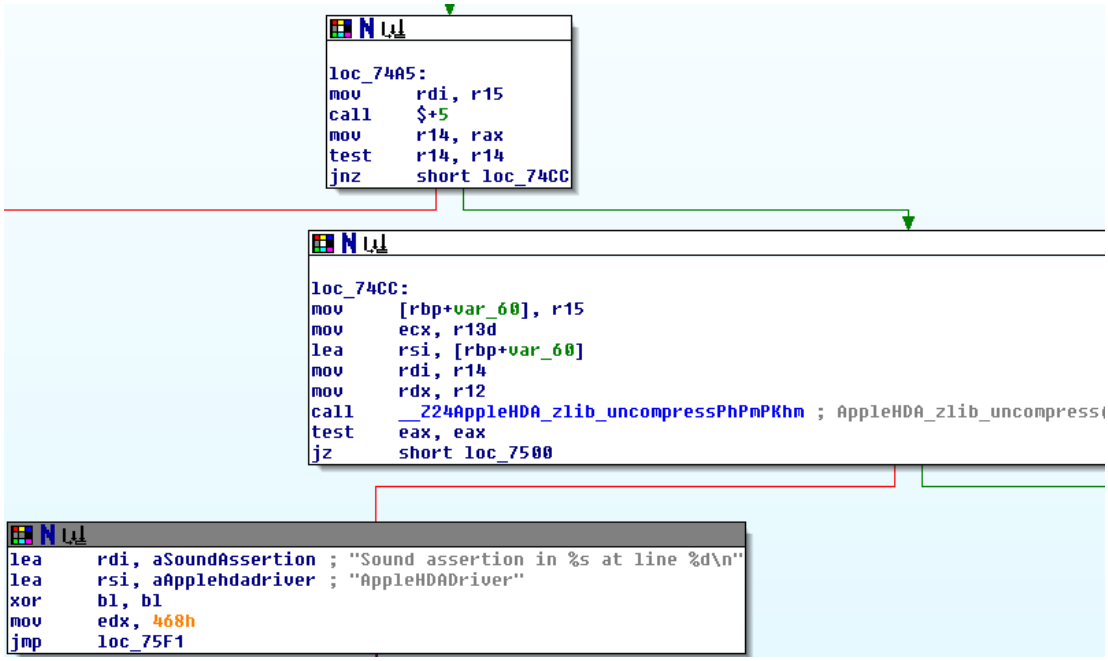
Turns out, the first 4 bytes of each element happen to match up exactly to the same layout-id files that Apple provide in the **AppleHDA.kext/Contents/Resources** folder (i.e. layout1.xml.zlib, layout10.xml.zlib, layout11.xml.zlib etc.) At this point, I thought I’d cracked it. I could simply modify this table in the AppleHDA binary, and chuck in my own custom layout with a number that isn’t in use. I went ahead and set my HDEF layout-id to **0x06 0x00 0x00 0x00**, created a layout6.xml.zlib with my pathmap and gave it a shot.

No go, but progress! We’re out of the loop.

digitaldj.net

```
localhost kernel[0]: Sound assertion in AppleHDADriver at line 1128
```

Here we go again, 1128d = 468h



Okay, now we're failing at inflating the zlib file. How is this possible? I made sure it was valid; it was. For reference, the zlib uncompress function source can be found here: http://www.opensource.apple.com/source/zlib/zlib-43/zlib/uncompr.c At this point, I was a little stuck. I decided to go back and take a look at the other 4 bytes of each element in the array. What on earth are they?

```
01000000 (1) 87040100 (66,695 d)
0A000000 (10) 7F380000 (14,463 d)
0B000000 (11) 7A4E0100 (85,626 d)
0C000000 (13) 7C1B0000 (7,036 d)
```

I figured it had to somehow do with the zlib uncompress function. The method definition is:

```
int ZEXPORT uncompress (dest, destLen, source, sourceLen)
Bytef *dest;
uLongf *destLen;
const Bytef *source;
uLong sourceLen;
```

digitaldj.net

Computer scientist. Gastronome. Avid traveller. Purveyor of logic.

So, uh, length? file size?! Oh. Turns out it's a map of layout-id to uncom-

pressed layout file size! Bingo. I modified the table for 06 to be the size of my inflated XML layout. It worked, finally. AppleHDA fired up with no sound assertions, and everything works as per 10.7 with a custom HDEF layout-id of **0x06** (not 0x01 !). Of course, for this to work, you also have to patch one of the unused device IDs in AppleHDA to your own audio controller's ID (e.g. Realtek ALC889 is 0x10EC0889), which forces AppleHDA to use the generic codec, but this is already well documented in the hackintosh world.

In a nutshell, if you want to do this yourself. Find a **layoutX.xml.zlib** in AppleHDA that you absolutely do not need to use. Note down its **ID (X)** and its **inflated (uncompressed) file size**. Search the AppleHDA binary for **XX XX XX XX YY YY YY YY**, where XX XX XX XX is your layout-id and YY YY YY YY is the uncompressed size **IN LITTLE-ENDIAN BYTE ORDER**.

For example,
layout16394.xml.zlib is 66,695 bytes UNCOMPRESSED
Therefore,
layout-id = 16394 (decimal) = 0x00 0x00 0x40 0x0A (hex big-endian) = **0x0A 0x40 0x00 0x00 (hex little-endian)**
filesize = 66,695 (decimal) = 0x00 0x01 0x04 0x87 (hex big-endian) = **0x87 0x04 0x01 0x00 (hex little-endian)**
And you would search for in **AppleHDA.kext/Contents/MacOS/AppleHDA**:
0x0A 0x40 0x00 0x00 0x87 0x04 0x01 0x00
Replace with your own layout's ID and size

Well, that was satisfying. Although this still isn't absolutely vanilla, it makes it a little more modular. I'm fairly sure this method of customization is a first.

It turns out gzipping the layouts and hard-coding their sizes into AppleHDA is part of Apple's effort to speed up the boot time of Mountain Lion, which is cool and all, but I'm sure that preventing custom use of AppleHDA was also part of the reason.

July 9, 2013 admin

digitaldj.net

2 thoughts on "Take My Layout, AppleHDA!"

eddy

July 9, 2013 at 10:45 pm

my lord. awesomeness.

thanks for the research and information.

Brandon

July 4, 2014 at 12:21 pm

I just want to drop by and say how awesome this article was. Thanks. :)

No spam links, promise. This is a real thank you.

digitaldj.net

Computer scientist. Gastronome. Avid traveller. Purveyor of logic.