Complete AppleHDA Patching Guide

By EMlyDinEsH
November 20, 2012 in AppleHDA

Share Followers 70

EMlyDinEsH
Retired
1.1k

Posted November 20, 2012

Hi friends,

This is a complete AppleHDA patching guide for the Audio codec in the Notebooks (but theory works for Desktop too). I'm writing this guide based on my knowledge and experience gained myself doing so many patches and details from the web. Hoping this could help many people looking to patch their AppleHDA by themselves. I did my best to make this as simple as possible and will try to improve as the time passes.

Also I don't take credit for all this information entirely except for appreciation on writing this guide. The credit will go for all these people: king, Master Chief, RevoGirl, toleda, bcc9, TimeWalker and many others who contributed to AppleHDA patching.

I'm using the Audio codec ALC269 and HD3000 HDMI audio which i've in my notebook for explaining the process. And I'm gonna break this guide into four chapters since its gonna be a very lengthy guide because of trying to provide insight on everything. Hope everyone can find this easy and helpful. Lets get started!

The Four Chapters are:

```
1. Calculating Codec verb commands and PathMaps
2. Patching XML(Platforms and layout) files.
3. AppleHDA Binary Patch
4. HDMI Audio Patch(coming soon)
```

CHAPTER -1 CALCULATING CODEC VERB COMMANDS AND PATHMAPS:

*******Calculation of codec verb commands and PathMaps explanation*******
*****Part 1*****
****Section 1****
Getting the codec dump
First, we need get the dump of your audio codec from linux. So, get into any of the Linux distributions of either Live USB/CD (or) Full install.
And enter the following command in terminal to get the dump in text format at Desktop.

```
cat /proc/asound/card0/codec#0 > ~/Desktop/codec_dump.txt
(or)
cat /proc/asound/card0/codec#1 > ~/Desktop/codec_dump.txt
(or)
cat /proc/asound/card0/codec#2 > ~/Desktop/codec_dump.txt
```

****Section 2****
Analyzing the codec dump and extracting relevant information

We need the following details from the codec dump:

```
1. Codec
2. Address
3. Vendor Id (Convert this hex value into decimal value)
4. Pin Complex Nodes with Control Name
5. Audio Mixer/Selector Nodes
6. Audio Output Nodes
7. Audio Input Nodes
```

Above details for the example ALC269 from the codec dump are below:

```
1. CODEC : Realtek ALC269VB

2. ADDRESS : 0

3. VENDOR ID : Hex: 0x10ec0269 [Decimal: 283902569]
```

4. PIN COMPLEX NODES WITH CONTROL NAMES :

```
Node 0x14 [Pin Complex] wcaps 0x40018d: Stereo Amp-Out
```

```
Amp-Out caps: ofs=0x00, nsteps=0x00, stepsize=0x00, mute=1
Amp-Out vals: [0x00 0x00]
Pincap 0x00010014: OUT EAPD Detect
EAPD 0x2: EAPD
Pin Default 0x99130110: [Fixed] Speaker at Int ATAPI
Conn = ATAPI, Color = Unknown
DefAssociation = 0x1, Sequence = 0x0
Misc = NO_PRESENCE
Pin-ctls: 0x40: OUT
Unsolicited: tag=00, enabled=0
Connection: 2
0x0c* 0x0d

Node 0x18 [Pin Complex] wcaps 0x40018f: Stereo Amp-In Amp-Out
Control: name="Mic Boost Volume", index=0, device=0
ControlAmp: chs=3, dir=In, idx=0, ofs=0
Amp-In caps: ofs=0x00, nsteps=0x03, stepsize=0x2f, mute=0
Amp-In vals: [0x00 0x00]
Amp-Out caps: ofs=0x00, nsteps=0x00, stepsize=0x00, mute=1
Amp-Out vals: [0x80 0x80]
Pincap 0x00001734: IN OUT Detect
Vref caps: HIZ 50 GRD 80
Pin Default 0x04a11820: [Jack] Mic at Ext Right
Conn = 1/8, Color = Black
DefAssociation = 0x2, Sequence = 0x0
Pin-ctls: 0x24: IN VREF_80
Unsolicited: tag=08, enabled=1
Connection: 1
0x0d

Node 0x19 [Pin Complex] wcaps 0x40008b: Stereo Amp-In
Control: name="Internal Mic Boost Volume", index=0, device=0
ControlAmp: chs=3, dir=In, idx=0, ofs=0
Amp-In caps: ofs=0x00, nsteps=0x03, stepsize=0x2f, mute=0
Amp-In vals: [0x03 0x03]
Pincap 0x00001724: IN Detect
Vref caps: HIZ 50 GRD 80
Pin Default 0x99a3092f: [Fixed] Mic at Int ATAPI
Conn = ATAPI, Color = Unknown
DefAssociation = 0x2, Sequence = 0xf
Misc = NO_PRESENCE
Pin-ctls: 0x24: IN VREF_80
Unsolicited: tag=00, enabled=0

Node 0x21 [Pin Complex] wcaps 0x40018d: Stereo Amp-Out
Control: name="Headphone Playback Switch", index=0, device=0
ControlAmp: chs=3, dir=Out, idx=0, ofs=0
Amp-Out caps: ofs=0x00, nsteps=0x00, stepsize=0x00, mute=1
Amp-Out vals: [0x00 0x00]
Pincap 0x0000001c: OUT HP Detect
Pin Default 0x0421101f: [Jack] HP Out at Ext Right
Conn = 1/8, Color = Black
DefAssociation = 0x1, Sequence = 0xf
Pin-ctls: 0xc0: OUT HP
Unsolicited: tag=04, enabled=1
Connection: 2
0x0c 0x0d*
```

5. AUDIO MIXER/SELECTOR NODES :

```
Node 0x0b [Audio Mixer] wcaps 0x20010b: Stereo Amp-In
Amp-In caps: ofs=0x17, nsteps=0x1f, stepsize=0x05, mute=1
Amp-In vals: [0x97 0x97] [0x97 0x97] [0x97 0x97] [0x97 0x97] [0x97 0x97]
Connection: 5
0x18 0x19 0x1a 0x1b 0x1d

Node 0x0c [Audio Mixer] wcaps 0x20010b: Stereo Amp-In
Amp-In caps: ofs=0x00, nsteps=0x00, stepsize=0x00, mute=1
Amp-In vals: [0x00 0x00] [0x00 0x00]
Connection: 2
0x02 0x0b

Node 0x0d [Audio Mixer] wcaps 0x20010b: Stereo Amp-In
Amp-In caps: ofs=0x00, nsteps=0x00, stepsize=0x00, mute=1
Amp-In vals: [0x00 0x00] [0x00 0x00]
Connection: 2
0x03 0x0b

Node 0x0f [Audio Mixer] wcaps 0x20010c: Mono Amp-In
```

Forums Unread Sign In Sign Up More

```
Connection: 2
0x02 0x0b

Node 0x22 [Audio Selector] wcaps 0x30010b: Stereo Amp-In
Amp-In caps: N/A
Amp-In vals: [0x00 0x00] [0x00 0x00] [0x00 0x00] [0x00 0x00] [0x00 0x00] [0x00 0x00] [0x00 0x00]
Connection: 7
0x18* 0x19 0x1a 0x1b 0x1d 0x0b 0x12

Node 0x23 [Audio Mixer] wcaps 0x20010b: Stereo Amp-In
Amp-In caps: ofs=0x00, nsteps=0x00, stepsize=0x00, mute=1
Amp-In vals: [0x80 0x80] [0x00 0x00] [0x80 0x80] [0x80 0x80] [0x80 0x80] [0x80 0x80]
Connection: 6
0x18 0x19 0x1a 0x1b 0x1d 0x0b
```

6. AUDIO OUTPUT NODES :

```
Node 0x02 [Audio Output] wcaps 0x1d: Stereo Amp-Out
Control: name="Speaker Playback Volume", index=0, device=0
ControlAmp: chs=3, dir=Out, idx=0, ofs=0
Device: name="ALC269VB Analog", type="Audio", device=0
Amp-Out caps: ofs=0x57, nsteps=0x57, stepsize=0x02, mute=0
Amp-Out vals: [0x41 0x41]
Converter: stream=5, channel=0
PCM:
rates [0x560]: 44100 48000 96000 192000
bits [0xe]: 16 20 24
formats [0x1]: PCM
Node 0x03 [Audio Output] wcaps 0x1d: Stereo Amp-Out
Control: name="Headphone Playback Volume", index=0, device=0
ControlAmp: chs=3, dir=Out, idx=0, ofs=0
Amp-Out caps: ofs=0x57, nsteps=0x57, stepsize=0x02, mute=0
Amp-Out vals: [0x41 0x41]
Converter: stream=5, channel=0
PCM:
rates [0x560]: 44100 48000 96000 192000
bits [0xe]: 16 20 24
formats [0x1]: PCM
```

7. AUDIO INPUT NODES :

```
Node 0x08 [Audio Input] wcaps 0x10011b: Stereo Amp-In
Control: name="Capture Switch", index=0, device=0
Control: name="Capture Volume", index=0, device=0
Device: name="ALC269VB Analog", type="Audio", device=0
Amp-In caps: ofs=0x0b, nsteps=0x1f, stepsize=0x05, mute=1
Amp-In vals: [0x1e 0x1e]
Converter: stream=1, channel=0
SDI-Select: 0
PCM:
rates [0x560]: 44100 48000 96000 192000
bits [0xe]: 16 20 24
formats [0x1]: PCM
Connection: 1
0x23

Node 0x09 [Audio Input] wcaps 0x10011b: Stereo Amp-In
Control: name="Capture Switch", index=1, device=0
Control: name="Capture Volume", index=1, device=0
Amp-In caps: ofs=0x0b, nsteps=0x1f, stepsize=0x05, mute=1
Amp-In vals: [0x00 0x00]
Converter: stream=0, channel=0
SDI-Select: 0
PCM:
rates [0x560]: 44100 48000 96000 192000
bits [0xe]: 16 20 24
formats [0x1]: PCM
Connection: 1
0x22
```

*****Part 2*****
****Section 1****
Extracting the values 'Pin Default', 'EAPD' and 'Node ID' from the Pin Complex Nodes:
We have analyzed and got the relevant details from the codec dump in part 1. Now, we try to get the values of Pin Default, EAPD and Node ID from the Pin Complex nodes with Control Name extracted above.

For the example ALC269:-
Pin Complex Nodes with Control Name

```
Node 14 : Pin Default 0x99130110, EAPD: 0x02
Node 19 : Pin Default 0x99a3092f
Node 21 : Pin Default 0x0421101f
```

Section 2
Extracting the verb data:
We will get the verb data from the Pin Default values of Nodes.

Pin default values must be read from right to left. And we will take two digits from it and write it down from left to right like below explained for ALC269.

```
Node 14:
Pin Default value: 0x99130110
Extracted verb data: "10 01 13 99"

Node 18:
Pin Default value: 0x04a11820
Extracted verb data: "20 18 a1 04"

Node 19:
Pin Default value: 0x99a3092f
Extracted verb data: "2f 09 a3 99"

Node 21:
Pin Default value: 0x0421101f
Extracted verb data: "1f 10 21 04"
```

Now, we need to correct the above verb data according to verbs info explained in the second post.

```
at Node 14: 10 01 13 99 [ Correction 99->90(Note 1)]
at Node 18: 20 18 a1 04 [ Correction 18->10(Note 2)]
at Node 19: 2f 09 a3 99 [ Correction 2f->20(Note 3) 09->01(Note 2) 99->90(Note 1)]
at Node 21: 1f 10 21 04 [ Correction 1f->10(Note 3)]
at Node 14 EAPD : 02 (Note 5)

Corrected Verb Data:
Code:
at Node 14: 10 01 13 90
at Node 18: 20 10 a1 04
at Node 19: 20 01 a3 90
at Node 21: 10 10 21 04
at Node 14 EAPD : 02
```

Verb data after default association corrections: *Note 4*

```
at Node 14: 10 01 13 90
at Node 18: 20 10 a1 04
at Node 19: 30 01 a3 90
at Node 21: 40 10 21 04
at Node 14 EAPD : 02
```

Final Verb data after Mic corrections: *Note 6 and Note 7*

```
Node 14: 10 01 13 90
Node 18: 20 10 81 04
Node 19: 30 01 a0 90
Node 21: 40 10 21 04
Node 14 EAPD : 02
```

Note 1: Location correction (9x)
Apple uses the location value as Built in Device - N/A instead of ATAPi in their codecs, which is always '0 ' for the location bit for the Integrated devices(Speakr, Int Mic). This is optional and audio will work if we use default codec value also. Just changing this to be more like Apple codec so we can avoid any future issues.

Ex:- at Node 14: 10 01 13 90 (Speaker) [Changed from 9 to 0]
at Node 19: 30 01 a0 90 (Int Mic) [Changed from 9 to 0]

Note 2: Jack color and sense capability correction (xx)
For internal devices like speakers etc., we use the jack color value as '0'(unknown) and need to Jack sense value of '1'.
For external devices like Headphones etc., we use the jack color value as '1'(black) and need to Jack sense value of '0'.
In this jack color may be optional, but Jack sense must use the values i've explained.

Note 3: Sequence correction (Ax)
We have to set this value to '0' for every device because Apple don't use analog multi out.
We have corrected above 2f and 1f values to 20 and 10 by replacing f with 0 in our example, so this Sequence number(Second digit) value must be always 0 for every node and A is the associate bit value of the node which is unique for each node.

Note 4: Default association correction (x0)
The Default association bit in the codec verbs must not match with other devices, so the association bit must be unique for all the devices.

Here, We have same association bit for both Speakers and Headphone as "1" and for both Mic's as "2", so we have to correct this default association bit. Most importantly the association bit must be in sequence and serially like assigning 1,2...x to the Nodes in sequence.
For example:- '1' to node 14, '2' to node 18 in sequence, '3' to Node 19 in sequence and '4' to Node 21 in sequence to previous node 19.

Note 5: EAPD existence
We have to look carefully at the output nodes like Speaker and Headphone, since in some codecs there is an External Amplifier(EAPD) to power up/down the Speaker to save power. We need to use EAPD command to wake up the node to get the sound otherwise we won't get sound even though its recognized.

Note 6: Internal Mic correction
If you did not get the "Ambient noise reduction" working for the Internal mic then you have to correct the "Connection type" bit field to Unknown Code Value(i.e, 0). In our example, we have the value 'a3' for the mic and corrected to 'a0'.

Note 7: External Mic correction
Here, we have to correct the external mic verb data to LineIn, otherwise the external mic won't work with AppleHDA.So, the verb data needs to be corrected at the position where it tells what kind of device it is.

Existing verb data for External Mic is "ax", where 'a' tells its Mic In. Now, we change this to Line in with 'ax' is replaced by '81' .

For more information about the codec verbs info:Click Here**.**

*****Part 3*****
Calculating the Codec verb commands :

```
Formulae for calculating the Codec verbs is:
Codec Address + NodeID + Verb Commands + Verb data
```

We have the Codec Address value from the Section 2 of Part 1.
We have the NodeID's values from the above Section 1 of Part 2.
Verb Commands are standard and they are 71c, 71d, 71e, 71f and 70c for EAPD
We have the verb data from the Section 2 of Part 2.

Calculation of codec Verb commands for the example ALC269:

```
For the Pin Complex Node 14:
0 + 14 + 71c + 10 = 01471c10
0 + 14 + 71d + 01 = 01471d01
0 + 14 + 71e + 13 = 01471e13
0 + 14 + 71f + 90 = 01471f90
```

```
For the Pin Complex Node 18:
0 + 18 + 71c + 20 = 01871c20
0 + 18 + 71d + 10 = 01871d10
0 + 18 + 71e + 81 = 01871e81
0 + 18 + 71f + 04 = 01871f04
```

```
For the Pin Complex Node 19:
0 + 19 + 71c + 30 = 01971c30
0 + 19 + 71d + 01 = 01971d01
0 + 19 + 71e + a0 = 01971ea0
0 + 19 + 71f + 90 = 01971f90
```

```
For the Pin Complex Node 21:
0 + 21 + 71c + 40 = 02171c40
0 + 21 + 71d + 10 = 02171d10
0 + 21 + 71e + 21 = 02171e21
0 + 21 + 71f + 04 = 02171f04
```

In our ALC269 example, there is an EAPD at Node 14 of Speaker. So, we need to calculate the verb command for this EAPD and use in our patch to get sound from speaker.

```
For EAPD at Node 14:
0 + 14 +70c + 02 = 01470c02
```

Now, we assemble the calculated codec verb commands so we can use for AppleHDA patch.

```
<01471c10 01471d01 01471e13 01471f99 01470c02
01871c20 01871d10 01871ea1 01871f04
01971c30 01971d01 01971ea3 01971f99
02171c40 02171d10 02171e21 02171f04>
```

DISABLING THE NODES (PIN COMPLEX) THAT ARE NOT USED:

We should disable the nodes that the codec is not using, so we can avoid any issues from them. Use the following the verb data for disabling those nodes and calculate verb commands.

Verb data for disabled node:

```
F0 00 00 40
```

Final codec verb commands with disabled nodes are:

```
<01271cf0 01271d00 01271e00 01271f40
```

```
01971c30 01971d10 01971ea0 01971f90
01a71cf0 01a71d00 01a71e00 01a71f40
01b71cf0 01b71d00 01b71e00 01b71f40
01d71cf0 01d71d00 01d71e00 01d71f40
01e71cf0 01e71d00 01e71e00 01e71f40
02171c40 02171d10 02171e21 02171f04>
```

*****Part 4*****
Calculating the PathMaps:

For calculating the PathMaps, we have to carefully follow the connections mentioned in every node from the analyzed relevant nodes information in the Section 2 of Part 1.

For output Devices, the PathMap follows this pattern
We have to find a Pin Complex node, an Audio Mixer node (optional for some codecs) and finally an Audio output node.

```
Pin Complex -> Audio Mixer -> Audio Output
(or)
Pin Complex ->  Audio Output
```

For Input Devices, the PathMap follow this pattern
We have to find a Pin Complex node, an Audio Mixer/Selector node (optional for some codecs) and finally an Audio input node.

```
Pin Complex -> Audio Selector/Mixer -> Audio Input
(or)
Pin Complex -> Audio Input
```

Output device PathMap calculation:
Lets first calculate the PathMaps for the output devices speaker and Headphone in our example ALC269.

According to the output device PathMap pattern, first we need to find Pin Complex node. In our ALC269 example, the output device 'Speaker' is located at the Pin Complex Node 0x14 with the Control Name "Speaker Playback Switch". Write down this Pin Complex node value in hex and decimal value.

```
0x14 , Decimal - 20
```

Now, take a look at the Connection in the Node 0x14.
We have two connections to nodes 0x0c and 0x0d, but the connection 0x0c has an asterisk symbol(*) indicating that this node 0x14 has a real connection to the node 0x0c which is an Audio Mixer needed according to our PathMaps pattern, so write down this node value in sequence

```
0x14->0x0c, Decimal - 20->12
```

Again, take a look at the node 0x0c we reached from the node 0x14. Here, we have two more nodes 0x02 and 0x0b without asterisk symbol(*) indicating the path where it goes. So, we have to figure out ourself and choose one from the two nodes. To solve this, just take a look at the two Nodes 0x02 and 0x0b.
For the output device PathMap, we already have two nodes out of three required except Audio Output Node. In our case 0x02 node is the Audio Output, so write this value in the sequence

```
0x14->0x0c->0x02, Decimal - 20->12->2
```

Now, we have calculated PathMap for the output device speaker.

Similarly, take a look at the Pin Complex Node 0x21 which is also an another output device "HeadPhone Playback switch". If we calculate the PathMap just like speaker and we will get

```
0x21->0x0d->0x03, Decimal - 33->13->3
```

Now, we have calculated PathMap for the output device Headphone.

NOTE:
If both Pin Complex nodes have connection to same Audio output node then try to use the other output node we analyzed from codec dump and test. sometimes using same output node for both also works.

Input device PathMaps calculation:
Calculating the pathMaps for input devices is little different from the output devices because connections are not mentioned and doesn't follow similar to output nodes. So, in this case we have go through the PathMap pattern from "Audio input" to "Pin Complex node" instead from "Pin Complex node" to "Audio Input" like we did for Output devices.

Like this,

1. First go to the Audio input node and look at the connections to the nodes Audio Mixer/Selector from here.
2. Follow the connection which will lead us to Audio Mixer/Selector
3. At Audio Mixer/Selector, Look at the Connections for the "*" symbol on Pin complex node
4. If it contains the symbol "*" then that is the Pin Complex node it is connected to, so got the PathMap of Pin Comple Node->Audio Mixer/Selector->Audio Input we needed.
5. If it does not contain then compare the results of the other Audio Input node ->Audio Mixers/Selector for the symbol "*". Even that Audio Mixer/Selector node also don't has symbol "*" then try to experiment with the available options of Pin complex node to Audio Selector/Mixer nodes (or) look at the Pin Complex node connections to Audio Selector/Mixer nodes.

Lets calculate the PathMaps for the example of ALC 269.

I've followed the Audio input (Node#9) Connection to Audio Selector(Node#22) then to Audio Input (Node#18) because of the symbol "*" on that node indicating default path.

Again, followed Audio input (Node#8) Connections to Audio Mixer(Node#23). But, i did not find the symbol "*" here to show the path. However, i've followed to Pin complex (Node#19) because its the only Input device we left with and even the connections of the Node#22 contains a Connection to Node#19 as well.

```
0x08->0x023->0x19 => 0x19->0x23->0x08
```

PathMaps for the Output and Input Devices of ALC269:

```
Pin Complex->Audio Mixer->Audio Output (Hex values)
Speaker : 20-> 12-> 2 (0x14->0x0c->0x02)
HeadPhone : 33-> 13-> 3 (0x21->0x0d->0x03)

Pin Complex->Audio Selector/Mixer->Audio Input(Hex values)
Internal Mic : 25-> 35-> 8 (0x19->0x23->0x08)
External Mic : 24-> 34-> 9 (0x18->0x22->0x09)
```

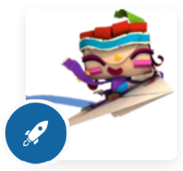
*******END OF CHAPTER-1*******
After following all the above steps, we get the verb commands and PathMaps for the codec.

Follow the below links for the next chapter:
*****CHAPTER-2*****
LINK: XML Files patching
*****CHAPTER-3*****
LINK: AppleHDA Binary Patching
*****CHAPTER-4*****
LINK: HDMI audio Patch

Hoping this could help some people in patching their AppleHDA.

Other Links that provide AppleHDA Patching details which I used as reference for this guide:
Reference link 1
Reference link 2

EMlyDinEsH



Retired

💬 1.1k

Posted November 20, 2012

Author ♥ 4

CHAPTER -2 PATCHING XML FILES:

*****Platforms and Layoutxx file Patch Explanation*****

COMPRESSING AND UNCOMPRESSING ZLIB FILES OF PLATFORM AND LAYOUT XML:
From 10.8 or later, the xml files are compressed to zlib format. We have to uncompress them to edit the files. After editing, again we have to compress it back to zlib.

For Compressing and uncompressing, use the attached perl script and below commands in terminal:

```
for uncompressing
perl zlib.pl inflate layout28.xml.zlib > layout28.xml

for compressing
perl zlib.pl deflate layout28.xml > layout28.xml.zlib
```

Layoutxx.xml file Patching

You can either use the attached xml files (or) can choose any one of the layout xml file from the apple Resources directory inside AppleHDA kext that matches Inputs and outputs of your codec and try this only if you want to experiment.

I'm using the Layout28.xml of Apple and edited to the values of ALC269. One of the reason to choose layout28 is because its used in MacBookPro8,1 and works very well. The other layout id's which also works for some codecs are '1' and '12' in hackintosh.

Note: *We removed the tags External Mic, SPDIF from the Apple layout28 xml file since they are not needed for our example ALC269.*

In the Layout xml file, we have to edit the following information.

1. LayoutID

```
<key>LayoutID</key>
<integer>28</integer>
```

2. CodecID [ALC 269 Vendor id decimal value =283902569]

```
<key>CodecID</key>
<array>
  <integer>283902569</integer>
</array>
```

3. Edit the Inputs key like below for External Mic(LineIn) and Internal Mic in Notebooks

```
    <string>Mic</string>
    <string>LineIn</string>
</array>
```

4. Edit the "IntSpeaker" key like below and remove all the signal processing elements.
Note: set MuteGPIO to 0 (or) remove this if its not working [not supported by some codecs]

For Realtek:

```
<key>IntSpeaker</key>
<dict>
    <key>MaximumBootBeepValue</key>
    <integer>48</integer>
    <key>MuteGPIO</key>
    <integer>0</integer>
</dict>
```

For others:

```
 <key>IntSpeaker</key>
<dict>
    <key>MaximumBootBeepValue</key>
    <integer>110</integer>
</dict>
```

5. Edit the "LineIn" key like below.
For Realtek:

```
<key>LineIn</key>
<dict>
     <key>MuteGPIO</key>
     <integer>1342242840</integer>
</dict>
```

For Others:

```
<key>Mic</key>
<dict/>
```

6. Edit the "Mic" key like below.
For Realtek:

```
<key>Mic</key>
<dict>
     <key>MuteGPIO</key>
     <integer>1342242841</integer>
</dict>
```

For Others:

```
<key>LineIn</key>
<dict/>
```

7. Edit the Outputs key like below for Speakers and Headphone in Notebooks

```
<key>Inputs</key>
<array>
  <string>Headphone</string>
  <string>IntSpeaker</string>
</array>
```

8. Edit the PathMapID tag at the end of the file with the PathMapID value used in Platforms xml file.

```
<key>PathMapID</key>
<integer>269</integer>
```

Note: SignalProcessing elements for Mic and Speaker are not supported by some codecs, so i've removed it. But can provide some good audio if used but not sure, so try to experiment with this later after getting audio working. I've attached xml files with the SignalProcessing working fine in ALC269 for speaker and Mic in Realtek and IDT for your reference, you can get more from Apple xml files.

Platforms.xml Patching:

This file contains the Mapping of Controls to its nodes giving a path. These path maps are contained in the key tag "PathMaps".

Note: *You can use the Platforms xml file i've attached which has all the PathMaps of Apple codec are removed and has only one PathMap of ALC269 in order to make it easy for editing instead of Apple file.*

We need to add our pathMaps to this root key which is a mapping of our Pin Complex's(O/P & I/P) to its Output/Input controls.

Follow the pattern i've explained below and edit the values of your codec nodes PathMaps calculated in the first post for each input and output. After editing then add the PathMap pattern of yours inside the file Platforms.xml at the end after the key tag of PathMapID like below.

```
</dict>
<<< Your pathMaps >>
```

The pattern for the PathMap is

```
<dict>
     <key>PathMap</key>
            <array>
                <array>
                    <array>
                      Data of Input1 (LineIn)
                    </array>
                    <array>
                      Data of Input2 (Mic)
                    </array>
                </array>
                <array>
                    <array>
                      Data of Output1 (Speaker)
                    </array>
                    <array>
                      Data of Output2 (Headphone)
                    </array>
                </array>
            </array>
            <key>PathMapID</key>
            <integer>[PathMapID used in layout#]</integer>
     </dict>
```

Input Data pattern:

```
<array>
    <dict>
        <key>Amp</key>
        <dict>
            <key>Channels</key>
            <array>
                <dict>
                    <key>Bind</key>
                    <integer>1</integer>
                    <key>Channel</key>
                    <integer>1</integer>
                </dict>
                <dict>
                    <key>Bind</key>
                    <integer>2</integer>
                    <key>Channel</key>
                    <integer>2</integer>
                </dict>
            </array>
            <key>MuteInputAmp</key>
            <true/>
            <key>PublishMute</key>
            <true/>
            <key>PublishVolume</key>
            <true/>
            <key>VolumeInputAmp</key>
            <true/>
        </dict>
        <key>NodeID</key>
        <integer>[Input Node#]</integer>
    </dict>
    <dict>
        <key>NodeID</key>
        <integer>[Audio Mixer/Selector Node#]</integer>
    </dict>
    <dict>
        <key>Boost</key>
        <integer>[Boost value# 1-3]</integer>
        <key>NodeID</key>
        <integer>[Pin complex Node#]</integer>
    </dict>
</array>
```

Input Data pattern without Audio Mixer/Selector:

Note: Some codecs doesn't need (or) use Audio Mixer/Selector node, so in order to get them working we should remove it from the pattern. Mostly this has been seen from IDT and Conexant codecs so far by me.

```
<array>
```

```
        <dict>
            <key>Channels</key>
            <array>
                <dict>
                    <key>Bind</key>
                    <integer>1</integer>
                    <key>Channel</key>
                    <integer>1</integer>
                </dict>
                <dict>
                    <key>Bind</key>
                    <integer>2</integer>
                    <key>Channel</key>
                    <integer>2</integer>
                </dict>
            </array>
            <key>MuteInputAmp</key>
            <true/>
            <key>PublishMute</key>
            <true/>
            <key>PublishVolume</key>
            <true/>
            <key>VolumeInputAmp</key>
            <true/>
        </dict>
        <key>NodeID</key>
        <integer>[Input Node#]</integer>
    </dict>
    <dict>
        <key>Boost</key>
        <integer>[Boost value# 1-3]</integer>
        <key>NodeID</key>
        <integer>[Pin complex Node#]</integer>
    </dict>
</array>
```

Output data pattern:

```
<array>
    <dict>
        <key>Amp</key>
        <dict>
            <key>MuteInputAmp</key>
            <false/>
            <key>PublishMute</key>
            <true/>
            <key>PublishVolume</key>
            <true/>
            <key>VolumeInputAmp</key>
            <false/>
        </dict>
        <key>NodeID</key>
        <integer>[Pin complex Node#]</integer>
    </dict>
    <dict>
        <key>Amp</key>
        <dict>
            <key>MuteInputAmp</key>
            <true/>
            <key>PublishMute</key>
            <true/>
            <key>PublishVolume</key>
            <true/>
            <key>VolumeInputAmp</key>
            <false/>
        </dict>
        <key>NodeID</key>
        <integer>[Audio Mixer Node#]</integer>
    </dict>
    <dict>
        <key>Amp</key>
        <dict>
            <key>Channels</key>
            <array>
                <dict>
                    <key>Bind</key>
                    <integer>1</integer>
                    <key>Channel</key>
```

```
                        <key>Bind</key>
                        <integer>2</integer>
                        <key>Channel</key>
                        <integer>2</integer>
                    </dict>
                </array>
                <key>MuteInputAmp</key>
                <true/>
                <key>PublishMute</key>
                <true/>
                <key>PublishVolume</key>
                <true/>
                <key>VolumeInputAmp</key>
                <false/>
            </dict>
            <key>NodeID</key>
            <integer>[Output Node#]</integer>
        </dict>
    </array>
```

Output Data pattern without Audio Mixer:

Note: Some codecs doesn't need (or) use Audio Mixer, so in order to get them working we should remove it from the pattern. Mostly this has been seen from IDT and Conexant codecs so far by me.

```
<array>
    <dict>
        <key>Amp</key>
        <dict>
            <key>MuteInputAmp</key>
            <false/>
            <key>PublishMute</key>
            <true/>
            <key>PublishVolume</key>
            <true/>
            <key>VolumeInputAmp</key>
            <false/>
        </dict>
        <key>NodeID</key>
        <integer>[Pin complex Node#]</integer>
    </dict>
    <dict>
        <key>Amp</key>
        <dict>
            <key>Channels</key>
            <array>
                <dict>
                    <key>Bind</key>
                    <integer>1</integer>
                    <key>Channel</key>
                    <integer>1</integer>
                </dict>
                <dict>
                    <key>Bind</key>
                    <integer>2</integer>
                    <key>Channel</key>
                    <integer>2</integer>
                </dict>
            </array>
            <key>MuteInputAmp</key>
            <true/>
            <key>PublishMute</key>
            <true/>
            <key>PublishVolume</key>
            <true/>
            <key>VolumeInputAmp</key>
            <false/>
        </dict>
        <key>NodeID</key>
        <integer>[Output Node#]</integer>
    </dict>
</array>
```

Layout12_XML_Files_ref.zip

Unavailable

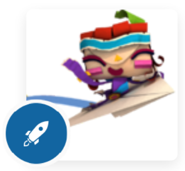
Layout28_XML_Files_ref.zip

Unavailable

zlib.zip

Unavailable

EMlyDinEsH



Retired

1.1k

Posted November 20, 2012

Codec Verbs Info

071CXY

```
X = Default Association
Values: 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d and f

Y = Sequence
Values: Always set this to '0' because Apple dont use analog multi outputs in their codec.
```

071DXY

```
X = Color: Color of the jack
Values:
Unknown 0
Black 1
Grey 2
Blue 3
Green 4
Red 5
Orange 6
Yellow 7
Purple 8
Pink 9
Reserved A-D
White E
Other F

Y = Misc - Jack detect sensing capability
Values:
1 for Internal Devices(Speaker etc.,) and
0 for External Devices(Headphones etc.,)
```

071EXY

```
X = Default device - Intended use of the Jack
Values:
Speakers 1
HP Out 2
CD 3
SPDIF Out 4
Digital Other Out 5
Modem Line Side 6
Modem Handset Side 7
Line In 8
AUX 9
Mic In A
Telephony B
SPDIF In C
Digital Other In D
Reserved E
Other F

Y = Connection type - indicates the type of physical connection
Values:
Unknown 0
1/8 stereo/mono 1
1/4 stereo/mono 2
ATAPI internal 3
RCA 4
Optical 5
Other Digital 6
Other Analog 7
Multichannel Analog (DIN) 8
XLR/Professional 9
RJ-11 (Modem) A
Combination B
Other F
```

071FXY

```
X = Port Connectivity - indicates the external connectivity of the Pin Complex.

Software can use this value to know what Pin Complexes are connected to jacks, internal devices, or not connected at all.

00b - The Port Complex is connected to a jack (1/8, ATAPI, etc.).
01b - No physical connection for Port.
```

```
Y = Location
Location indicates the physical location of the jack or device to which the pin complex is connected. This allows software


Details:
Convert the 2 digit hex number to binary.
Pad the front with zeros to make it 8 dgits.

Example:
Code: 0x02 = binary 10 = 00000010 8 digit binary
Reading the bits from left to right:

Port Connectivity bits 7:6
------------------------------------------------------------
00 - Port is connected to a Jack

01 - No External Port -or- No physical connection for Port**

10 - Fixed Function/Built In Device (integrated speaker, mic, etc)

11 - Jack and Internal device are attached


Location Part 1 - bits 5:4
------------------------------------------------------------
00 - External on primary chassis

01 - Internal

10 - Separate chassis

11 - Other


Location Part 2 - bits 3:0
------------------------------------------------------------
The meaning depends on Location Part 1

00 0000****N/A

00 0001** Rear

00 0010** Front

00 0011** Left

00 0100** Right

00 0101** Top

00 0110** Bottom

00 0111** Special (Rear panel)

00 1000** Special (Drive bay)



01 0000** N/A

01 0111** Special (Riser)

01 1000** HDMI

01 1001** ATAPI



10 0000****N/A

10 0001** Rear

10 0010** Front

10 0011** Left
```

```
10 0101** Top

10 0110** Bottom



11 0000** N/A

11 0110** ?

11 0111** Inside Mobile Lid (example: mic)

11 1000** Outside Mobile Lid



************Bits

Hex******76 54 3210
-------------------
71cf01 = 00 00 0001 - Port has a jack - It is External - Rear Location

71cf02 = 00 00 0010 - Port has a jack - It is External - Front Panel Location

71cf59 = 01 01 1001 - No External Port - ATAPI

71cf18 = 00 01 1000 - Port has a jack - External - HDMI

71cf90 = 10 01 0000 - Built In Device - Internal - N/A

******** |**|**||||

******** |**|**|--------- Location part 2

******** |**|------------ Location part 1**

******** |--------------- Port Connectivity
```

This topic is now closed to further replies.