DJ Link Packet Analysis

James Elliott
Deep Symmetry, LLC

April 29, 2016

Abstract

The protocol used by Pioneer professional DJ equipment to communicate and coordinate performances can be monitored to provide useful information for synchronizing other software, such as light shows and sequencers. By creating a "virtual CDJ" that sends appropriate packets to the network, other devices can be induced to send packets containing even more useful information about their state. This article documents what has been learned so far about the protocol, and how to accomplish these tasks.

1 Mixer Startup

When the mixer starts up, after it obtains an IP address (or gives up on doing that and self-assigns an address), it sends out what look like a series of packets[1] simply announcing its existence to UDP port 50000 on the broadcast address of the local network.

These have a data length[2] of 37 bytes, appear roughly every 300 milliseconds, and have the content shown in Figure 1.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Header	51	73	70	74	31	57	6d	4a	4f	4c	**0a**	00					Name
	Device Name (padded with 00)																Name
	01	02	00	25	02												

Figure 1: Initial announcement packets from Mixer

[1]The packet capture described in this analysis can be found at `https://github.com/brunchboy/dysentery/raw/master/doc/assets/powerup.pcapng`

[2]Values within packets are shown in hexadecimal, while packet lengths and byte offsets are discussed in decimal.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Header	51	73	70	74	31	57	6d	4a	4f	4c	**00**	00					Name
	Device Name (padded with 00)																Name
	01	02	00	2c	N	02	MAC address										

Figure 2: First-stage Mixer device number assignment packets

The tenth byte (inside what is labeled the header) is bolded because its value changes in the different types of packets which follow.

After about three of these packets are sent, another series of three begins. It is not clear what purpose these packets serve, because they are not yet asserting ownership of any device number; perhaps they are used when CDJs are powering up as part of the mechanism the mixer can use to tell them which device number to use based on which network port they are connected to?

In any case, these three packets have a data length of 44 bytes, are again sent to UDP port 50000 on the local network broadcast address, at roughly 300 millisecond intervals, and have the content shown in Figure 2.

The value N at byte 36 is 1, 2, or 3, depending on whether this is the first, second, or third time the packet is sent.

After these comes another series of three numbered packets. These appear to be claiming the device number for a particular device, as well as announcing the IP address at which it can be found. They have a data length of 50 bytes, and are again sent to UDP port 50000 on the local network broadcast address, at roughly 300 millisecond intervals, with the content shown in Figure 3.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Header	51	73	70	74	31	57	6d	4a	4f	4c	**02**	00					Name
	Device Name (padded with 00)																Name
	01	02	00	32	IP address				MAC address						D	N	
	02	01															

Figure 3: Second-stage Mixer device number assignment packets

I identify these as claiming/identifying the device number because the value D at byte 46 is the same as the device number that the mixer uses to identify itself (`0x21`) and the same is true for the corresponding

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Header	`51`	`73`	`70`	`74`	`31`	`57`	`6d`	`4a`	`4f`	`4c`	**`04`**	`00`					Name
	Device Name (padded with `00`)																
	`01`	`02`	`00`	`26`	D	N											

Figure 4: Final-stage Mixer device number assignment packets

packets seen from the CDJs (they use device numbers 2 and 3, as they are connected to those ports/channels on the mixer).

As with the previous series of three packets, the value N at byte 47 takes on the values 1, 2, and 3 in the three packets.

These are followed by another three packets, perhaps the last stage of claiming the device number, again at 300 millisecond intervals, to the same port 50000. These shorter packets have 38 bytes of data and the content shown in Figure 4.

As before the value D at byte 36 is the same as the device number that the mixer uses to identify itself (`0x21`) and N at byte 37 takes on the values 1, 2, and 3 in the three packets.

Once those are sent, the mixer seems to settle down and send what looks like a keep-alive packet to retain presence on the network and ownership of its device number, at a less frequent interval. These packets are 54 bytes long, again sent to port 50000 on the local network broadcast address, roughly every second and a half. They have the content shown in Figure 5.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Header	`51`	`73`	`70`	`74`	`31`	`57`	`6d`	`4a`	`4f`	`4c`	**`06`**	`00`					Name
	Device Name (padded with `00`)																
	`01`	`02`	`00`	`36`	D	`02`	MAC address						IP address				
	`01`	`00`	`00`	`00`	`02`	`00`											

Figure 5: Mixer keep-alive packets

2 CDJ Startup

When a CDJ starts up the procedure and packets are nearly identical, with groups of three packets sent at 300 millisecond intervals to port 50000 of the local network broadcast address. The only difference between Figure 6 and Figure 1 is the final byte, which is `0x01` for the CDJ, and was `0x02` for the mixer.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Header	51	73	70	74	31	57	6d	4a	4f	4c	**0a**	00					Name
	Device Name (padded with 00)																
	01	02	00	25	01												

Figure 6: Initial announcement packets from CDJ

Similarly, the next series of three packets from the CDJ are nearly identical to those from the mixer. The only difference between Figure 7 and Figure 2 is byte 37 (immediately after the packet counter N), which again is `0x01` for the CDJ, and was `0x02` for the mixer.

However it appears that in this capture the CDJ skips the second stage of claiming a device number, probably because it is configured to be automatically assigned a device number based on the port of the mixer to which it is connected, and we cannot see a packet that the mixer sent it assigning it that device number. Instead, it jumps right to the end of the third and final stage, sending a single 38-byte packet with header byte 10 set to tt 04 (which identified the three packets of the third stage when the mixer was starting up), with content identical to Figure 4.

Even though the value of N is `01`, this is the only packet in this series that the CDJ sends. It would probably behave differently if configured to assign its own device number (behaving like we saw the mixer behave in claiming its device number).

The CDJ then moves to the keep-alive stage, sending out 54-byte packets with the content shown in Figure 8.

As seems to always be the case when comparing mixer and CDJ packets, the difference between this and Figure 5 is that byte 37 (following the device number D) has the value `01` rather than `02`, and the same is true of the second-to-last byte in each of the packets. (Byte

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Header	51	73	70	74	31	57	6d	4a	4f	4c	**00**	00					Name
	Device Name (padded with 00)																
	01	02	00	2c	N	01	MAC address										

Figure 7: First-stage CDJ device number assignment packets

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Header	`51`	`73`	`70`	`74`	`31`	`57`	`6d`	`4a`	`4f`	`4c`	**`06`**	`00`					Name
	Device Name (padded with `00`)																Name
	`01`	`02`	`00`	`36`	D	`01`	MAC address						IP address				
	`01`	`00`	`00`	`00`	`01`	`00`											

Figure 8: CDJ keep-alive packets

52 is `01` in Figure 8 and `02` in Figure 5.

3 Tracking BPM and Beats

For some time now, Afterglow[3] has been able to synchronize its light shows with music being played on Pioneer equipment by observing packets broadcast by the mixer to port 50001. Until recently, however, it was not possible to tell which player was the Master, so there was no way to determine the down beat (the start of each measure). This section will be expanded and more details provided as Afterglow is updated to take advantage of the discoveries described in the next section.

Until then, here is a summary of what is currently done. A socket is opened and bound to port 50001. Whenever a packet from the mixer is received on this socket, if the length is 96 bytes, it is known to contain beat and BPM information. The current BPM can be obtained as:

$$\frac{byte[90] \times 256 + byte[91]}{100}$$

These packets are sent on each beat, and the current beat number (1, 2, 3 or 4) is sent in $byte[92]$. However, the beat number is *not* synchronized with the master player, and so it is not useful for much. We expect to make use of the Virtual CDJ technique to determine the actual beat number soon.

4 Creating a Virtual CDJ

Although some useful information can be obtained simply by watching broadcast traffic on a network containing Pioneer gear, in order to

[3] `https://github.com/brunchboy/afterglow#afterglow`

get important details it is necessary to cause the gear to send you information directly. This can be done by simulating a “Virtual CDJ”.[4]

To do this, bind a UDP server socket to port 50002 on the network interface on which you are receiving DJ-Link traffic, and start sending keep-alive packets to port 50000 on the broadcast address as if you were a CDJ. Follow the structure shown in Figure 8, but use the actual MAC and IP addresses of the network interface on which you are receiving DJ-Link traffic, so the devices can see how to reach you.

You can use a value like 5 for D (the device/player number) so as not to conflict with any actual players you have on the network, and any name you would like. As long as you are sending these packets roughly every 1.5 seconds, the other players and mixers will begin sending packets directly to the socket you have opened on port 50002.

We are just beginning to analyze all the information which can be gleaned from these packets, but here is what we know so far.[5]

4.1 Mixer Status Packets

Packets from the mixer will have a length of 56 bytes and the content shown in Figure 9.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	51	73	70	74	31	57	6d	4a	4f	4c	**29**					
16	Device Name (padded with 00)															01
32	00	D	00	14	D	00	00	d0	00	10	00	00	80	00	BPM	
48	00	10	00	00	00	09	00	X	B							

Figure 9: Mixer status packets

Packets coming from a DJM-2000 nexus connected as the only mixer on the network contain a value of 33 (`0x21`) for their Device Number D (bytes 33 and 36).

The current tempo in beats-per-minute identified by the mixer can be obtained as:

$$\frac{byte[46] \times 256 + byte[47]}{100}$$

[4] Thanks are due to Diogo Santos for discovering the trick of creating a virtual CDJ in order to receive detailed status information from other devices.

[5] Examples of packets discussed in this section can be found in the capture at `https://github.com/brunchboy/dysentery/raw/master/doc/assets/to-virtual.pcapng`

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>0</td><td>51</td><td>73</td><td>70</td><td>74</td><td>31</td><td>57</td><td>6d</td><td>4a</td><td>4f</td><td>4c</td><td>0a</td><td colspan="5"></td></tr>
<tr><td>16</td><td colspan="15">Device Name (padded with 00)</td><td>01</td></tr>
<tr><td>32</td><td>03</td><td>D</td><td>00</td><td>b0</td><td>D</td><td>00</td><td>01</td><td>A</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
<tr><td>48</td><td>00</td><td>00</td><td>00</td><td>t_1</td><td>00</td><td>00</td><td>t_2</td><td>t_3</td><td>00</td><td>00</td><td>t_4</td><td>t_5</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
<tr><td>64</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>t_6</td><td>t_7</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
<tr><td>80</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
<tr><td>96</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>01</td><td>00</td><td>t_8</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
<tr><td>112</td><td>00</td><td>00</td><td>00</td><td>03</td><td>00</td><td>U</td><td>00</td><td>00</td><td>01</td><td>00</td><td>00</td><td>P_1</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
<tr><td>128</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>01</td><td>00</td><td>F</td><td>ff</td><td>7e</td><td>00</td><td colspan="3">$Pitch_1$</td></tr>
<tr><td>144</td><td colspan="2">l_1</td><td colspan="2">BPM</td><td>7f</td><td>ff</td><td>ff</td><td>ff</td><td>00</td><td colspan="3">$Pitch_2$</td><td>00</td><td>P_2</td><td>l_2</td><td>ff</td></tr>
<tr><td>160</td><td>00</td><td>00</td><td colspan="2">$Beat$</td><td colspan="2">Mem</td><td>B_b</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
<tr><td>176</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>10</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td></tr>
<tr><td>192</td><td>00</td><td colspan="3">$Pitch_3$</td><td>00</td><td colspan="3">$Pitch_4$</td><td colspan="4">$Packet$</td><td>0f</td><td>00</td><td>00</td><td>00</td></tr>
<tr><td>208</td><td>00</td><td>00</td><td>00</td><td>00</td><td colspan="12"></td></tr>
</table>

Figure 10: CDJ status packets

This value is labeled *BPM* in Figure 9.

Mixer status packets are sent on each beat, and the current beat number (1, 2, 3 or 4) is sent in *byte*[55], labeled *B*. However, the beat number is *not* synchronized with the master player, and so it is not useful for much. The beat number should be determined, when needed, from packets that are sent by the master player.

The value at *byte*[54], labeled *X*, has an unknown meaning. It seems to start out with the value `0x00`, and then change when a player starts playing to the value `0xff`, but it may well do other things as well.

4.2 CDJ Status Packets

Packets from a CDJ will have a length of 212 bytes and the content shown in Figure 10.

The Device Number in *D* (bytes 33 and 36) is the Player Number as displayed on the CDJ itself. In the case of this capture, the CDJs were assigned Player Numbers 2 and 3.

The activity flag A at byte 39 seems to be 0 when the player is idle, and 1 when it is playing, searching, or loading a track.

There are a number of bytes, labeled t_1 through t_7, whose purpose is as yet undetermined. They are all zero when there is no track loaded, but take different values when a track is loaded. Byte 106, labeled t_8, seems to tick back and forth between the value 4 and 6 when a track is loaded, but the rate at which it does this is not the same as the BPM, and it is rather mysterious.

Byte 117, labeled U, appears to have the value 1 whenever USB media is present in any player on the network, whether or not the Link option is chosen in the other players, and 0 otherwise. I don't know if that is true of SD media as well; this needs more investigation.

Byte 123, labeled P_1, appears to describe the current play mode. The values that have been seen so far, and their apparent meanings, are:

0 No track is loaded.

3 The player is playing normally.

4 The player is playing a loop.

5 The player is paused anywhere other than the cue point.

6 The player is paused at the cue point.

9 The player is searching forwards or backwards.

Byte 137, labeled F, is a bit field containing some very useful state flags, detailed in Figure 11.

7	6	5	4	3	2	1	0
1	Play	Master	Sync	On-Air	1	0	0

Figure 11: CDJ state flag bits

> We have not yet seen any other values for bits 0, 1, 2, or 7 in F, so we're unsure if they also carry meaning. If you ever find different values for them, please let us know by filing an Issue! `https://github.com/brunchboy/dysentery/issues`

There are four different places where pitch information appears in these packets: $Pitch_1$ at bytes 141–143, $Pitch_2$ at bytes 153–155, $Pitch_3$ at bytes 193–195, and $Pitch_4$ at bytes 197–199.

Each of these values represents a three-byte pitch adjustment percentage, where `0x100000` represents no adjustment (0%), `0x000000` represents slowing all the way to a complete stop (−100%, reachable only in Wide tempo mode), and `0x200000` represents playing at double speed (+100%).

Here is how the pitch adjustment percentage represented by $Pitch_1$ would be calculated:

$$100 \times \frac{(byte[141] \times 65536 + byte[142] \times 256 + byte[143]) - 1048576}{1048576}$$

We don't know why there are so many copies of the pitch information, or all circumstances under which they might differ from each other, but it seems that $Pitch_4$ always reports the actual location of the Pitch fader (unless Tempo Reset is active, effectively locking the Pitch fader to 0% and $Pitch_4$ to `0x100000`), and the others seem to represent the current effective pitch (for example, the pitch of the master player when sync is active).

The current BPM of the track (the BPM at the point that is currently being played, or at the location where the player is currently paused) can be found at bytes 146–147 (labeled BPM). It is a two-byte integer representing one hundred times the current track BPM. So, the current track BPM value to two decimal places can be calculated as:

$$\frac{byte[146] \times 256 + byte[147]}{100}$$

In order to obtain the actual playing BPM, this value must be multiplied by the current effective pitch, calculated from $Pitch_1$ as described above.

Because Rekordbox and the CDJs support tracks with variable BPM, this value can and does change while such tracks play. When no track is loaded, BPM has the value `0xffff`.

The meaning of values l_1 (bytes 144–145) and l_3 (byte 158) are not currently known. They may simply reflect whether a track is loaded or not: l_1 seems to have the value `0x7fff` when no track is loaded, and the value `0x8000` when a track is loaded, while l_2 and l_3 have the value 0 when no track is loaded and 1 when a track is loaded... but there is likely more going on here than we have yet figured out.

Byte 157 (labeled P_2) seems to communicate additional information about the current play mode, with the following meanings that we have found so far:

0 No track is loaded.

1 The player is paused or playing in Reverse mode.

9 The player is playing in Forward mode with jog mode set to Vinyl.

10 The player is playing in Forward mode with jog mode set to CDJ.

The beat counter (which counts each beat from 1 through the end of the track) is found in bytes 162–163, labeled *Beat*. (When the player is paused at the start of the track, this seems to hold the value 0, even though it is beat 1.) The counter *Bb* at byte 166 counts out the beat within each bar, cycling $1 \to 2 \to 3 \to 4$ repeatedly, and can be used to identify the down beat (as is used in the Master Player display on the CDJs as a mixing aid).

A countdown timer to the next saved memory point is available in bytes 164–165 (labeled *Mem*). If there is no saved memory point after the current play location in the track, or if it is further than 64 bars ahead, these bytes contain the value `0x01ff`. As soon as there are just 64 bars (256 beats) to go before the next memory point, this value becomes `0x0100`—this is the point at which the CDJ starts to display a countdown, which it displays as "63.4 Bars". As each beat goes by, this value decreases by 1, until the memory point is about to be reached, at which point the value is `0x0001` and the CDJ displays "0.1 Bars". On the beat on which the memory point was saved the value is `0x0000` and the CDJ displays "0.0 Bars". On the next beat, the value becomes determined by the next memory point (if any) in the track.

Bytes 200-203 seem to contain a 4-byte packet counter *Packet*, which is incremented for each packet sent by the player. (I am just guessing it is four bytes long, I have not yet watched long enough for the count to need more than the last three bytes).

> This analysis is incomplete; there are many bytes whose purpose is not yet known, not all of which are even mentioned here yet.