Subject: The Patchbay - please explain
Posted by dnafe on Mon, 03 May 2010 10:15:57 GMT
View Forum Message <> Reply to Message

Hi all

I've been off Paris for a while and am getting back to business this summer and the one thing I've always had trouble with is the damn patchbay routing.

Would someone please explain the ins and outs, the mislabeling and the principles of routing in the Patchbay...I'm positive it would make a great Wiki article.

Thanks

Subject: Re: The Patchbay - please explain
Posted by kerryg on Mon, 03 May 2010 17:38:18 GMT
View Forum Message <> Reply to Message

Heya Don - some of it exists already.

It's not really mislabeled so much as misleadingly (or "non-intuitively") labelled.

The first confusing thing: using "hardware" and "software" objects together

The basic gist is as follows: the Patch Bay really contains two classes of objects - stuff that represents actual PARIS hardware (ie "IF-2" or "MEC Master A"), and representations of parts of the PARIS software (all of this "software" class stuff has the prefix "MIXER").

Because there's no real visual distinction between those classes of objects, confusion arises as people try to deal with two types of objects that look similar yet behave differently.

When you're patching between software objects (Mixer A, Mixer A FX, Mixer A Insert), it's just like in the real world - "ins" get patched to "outs".

When you're patching between non-software objects (no "MIXER" prefix), it's also just like in the real world - "ins" get patched to "outs".

But when you're patching *across the hardware/software boundary*, ie MEC MODULES A to MIXER A - suddenly "ins" go to "ins" and "outs" go to "outs".

Think of this "cross boundary" patching as "mapping"; you're now mapping your software objects to actual inputs and outputs in your hardware.

Rule of thumb: look at the prefixes.

If both say MIXER, - "ins to outs",
If neither say MIXER - "ins to outs".

If one says MIXER and the other doesn't - "ins to ins"/"outs to outs".

The next confusing thing: inconsistent naming

Don't ask me why, but things are named differently in your Patch Bay than their counterparts in your Mixer Window. As they exist, the names are far more "correct" than they are "helpful".

Here's some clarification:

"MIXER A Insert" is just your CHANNEL STRIP INSERTS. This is where you route signal to/from the EDS plugin named "External" when you insert it in the relevant strip.

You can route to and from those INSERTS all day long but they won't become "active" until you insert that "External" plugin into the relevant channel strip.

"MIXER A FX", on the other hand, is actually your Auxes one through eight. To make it even more non-intuitive, they're labelled "1 through 16", whereas in your Mixer Window the auxes are labelled as eight right/left pairs.

Rule of thumb:

MIXER A Inserts are a way of giving your Mixer Window the equivalent of the channel insert jacks on the back of a regular mixer (although it won't start working until you also insert an "External" plugin in the EDS slot for that strip).

MIXER A FX corresponds to similar insert points for your Mixer Window's AUXES (this also requires the insertion of an "External" plugin).

Finally:

- You can "mult" to multiple software inputs from one jack (ie, you could feed all sixteen channel strips from one input). But the reverse is not true: any given input can only receive a single source.

- You're limited to sixteen i/o per MEC Module A (or B or C, etc). You can fill as many slots in your MEC as you like and leave them hardwired to gear so you can patch it in with a mouseclick - but only sixteen can be active at a time.

- You have to "close the loop" with circular routing (ie looping out to an FX unit. You can't just patch one-way - you have to patch back to a return as well, even if you don't need it (eg a monitor send).

- And of course you can't patch between submixes. You can do this via hardware cables, of course.

As you've no doubt found over the years, it's actually a killer design, extremely useful and flexible;

all you'll have to do is once again get your head around the apparently-interrupted-in-the-middle nature of both the labeling and the execution (here I am writing a tutorial on the damn thing, and I struggle with it myself often enough).