

Commissioning AES67 in Your Plant

By **Andy Calvanese**
Vice President/Engineering
Wheatstone Corporation

Revision 1 – September 2018

There's been a great deal of talk about AES67 since this interoperability standard was ratified by the Audio Engineering Society in 2013. By now, just about everyone knows that AES67 is a standard for streaming audio between differing audio-over-IP systems, and that all the major AoIP vendors offer AES67 as part of their system.

But, that's as far as it's gone for most broadcasters – essentially a new standard still sitting on the dealer lot like all those new Corvettes.

Now the question is: What would a real-world AES67 commissioning scenario look like?

We decided to take AES67 out for a spin to find out. Earlier this summer we did a trial run of AES67 through a large WheatNet-IP system staged at the Wheatstone factory in New Bern, NC, during what we call a BLADEFest. (BLADEs are the I/O access units that make up the WheatNet-IP audio network). We do BLADEFests periodically to test our system under real-world conditions, and with AES67 getting to be as real as it gets, we added in a few AES67 devices while we were at it.

The WheatNet-IP system included 12 radio and TV consoles, 62 hardware BLADEs (I/O access units), 100 software BLADEs, talent stations, SideBoards, Smart Switch panels, and software, including three different vendors' automation systems. Then we added in AES67 devices from Genelec, Ward-Beck, Dante, and Axia. All of this was tied together by new model Cisco and Dell switches.

We routed audio from one device to the next until our signal path passed in a big daisy chain through every single piece of gear. Just cabling and configuring the system took days! Then, with our engineers monitoring closely, we ran the system through a series of automated torture tests that included completely rebooting the system and verifying proper operation afterward. We're happy to say that after more than 160 reboots, not a single connection failure or loss of audio occurred.

Just as important, we came away from this BLADEFest with some good information that broadcasters can use when commissioning AES67 in their plants.

You'll Need One of These

One of the key differences between various AoIP systems is timing, and it's an important difference. The AES67 standard was created to keep disparate audio devices synchronized so audio can play back without clicks and pops due to sample rate and timing mismatches. For example, Wheatstone's I/O BLADEs in the WheatNet-IP audio network all synchronize their internal clocking to a special signal that the system distributes to every BLADE. We call that signal the "metronome" and we send it around the network many times per second to keep all of the individual clocks in BLADEs in sync with each other. We developed this method of synchronization back in 2004 when we designed WheatNet-IP. Axia did something on their own with Livewire for the same purpose, and other vendors did yet something else. We each did our own thing because back in the day, there was nothing suitable for the purpose; we had to invent something to make it work.

Without AES67, WheatNet-IP BLADEs can't synchronize audio with an Axia xNode and vice versa. Now with AES67, we have a method to share synchronization amongst devices. We can use the standard timing protocol that has come out in recent years, or IEEE-1588, also known as the Precision Time Protocol, or PTP. In fact, the standard specifies a particular version of this protocol known as PTPv2.

For an AoIP system to maintain timing and stay synchronized with other AES67 devices, the system timing must be controlled by PTPv2. For that to occur there must be some device in the system that serves the role as PTPv2 timing generator to which all other devices slave their timing. Generally, that role is filled by a specialized PTP master clock device because the PTPv2 protocol is so precise that in the best circumstances (PTPv2 master clock synced to GPS for absolute timing reference and PTP aware switches used for reference signal distribution) timing accuracy of better than 1 microsecond can be achieved. An ordinary crystal oscillator in a PC or I/O device is nowhere near accurate and stable enough for this performance level, hence the need for standalone master clock generators.

Sub-microsecond timing accuracy is not required to maintain click free audio however, so if your main concern is clean audio and you're not worried about absolute timing accuracy you can dispense with the added expense of PTP aware switches and use a basic PTPv2 master clock. Some AoIP devices will actually provide a PTP reference clock signal themselves; however, in our experience their timing accuracy is typically poor.

Bottom line:

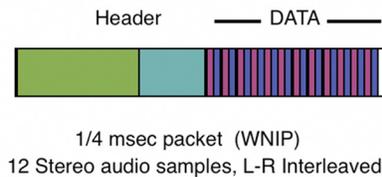
To use AES67 devices, your system must have a PTPv2 clock reference device.

You'll Need Packet Structure

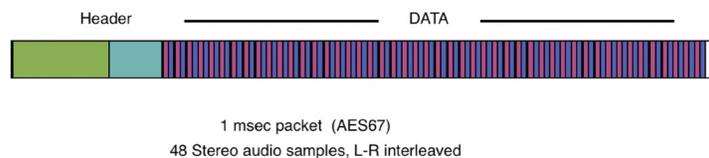
The short description on how AoIP systems work is that the transmitting device creates a packet header using the standard IP protocols and the payload gets filled with pieces of digital audio data which will get put back together by the receiving device to recreate the audio signal.

It's important that the transmitting device and receiving device use the same system for filling and decoding the payload. There are many different ways this can be done:

1. Fill the payload with one audio sample of data; for instance, left channel, 24 bits. Send another stream with another 24 bits, in this case the right channel. This would work, but it's a very, very inefficient and difficult method to implement because it would take 96,000 packets per second to make one stereo audio stream sampled at 48K.
2. Fill the packet with 12 samples of 24 bits of left channel audio and then 24 bits of right channel audio.
3. Fill the packet with 12 samples of 24-bit audio, interleaving left and right channels, as shown below.



4. Fill the packet with 48 samples of 24-bit audio, interleaved, as shown here



5. Fill the packet with 64 samples of 16-bit audio, interleaved.
6. Fill the packet with 96 samples of 24-bit audio, mono.

You can see the possibilities are nearly endless. Therefore, what AES67 does is to specify a single packet structure that must be used; this is the 1msec packet timing you see in all of the specifications and it equates to 48 samples left-right interleaved in a stereo stream. By the way, AES67 goes on to specify some additional packet structures that could be used because there is no one right answer for the ideal structure. The problem is that larger packet structures are more efficient for the network infrastructure because fewer packets are needed to stream, but larger packets induce greater latency because it takes longer to fill a big packet with 48K audio data samples than a small one. In fact, this is why WheatNet-IP audio network uses gigabit Ethernet connections and ¼ msec packet timing as our default to keep latency to a minimum.

Configuring the stream

Streams that are intended to be sent to more than one device take advantage of a standard IP mechanism, multicasting, to maximize network efficiency. In this protocol every stream is given a unique multicast address to identify it on the network so standard switches can send its packets to the devices that want them, but not to devices that don't want them. The number of available multicast addresses is very large and individual device manufacturers could choose any of these multicast addresses to use in their system.

Ports provide a mechanism in IP networks of pre-identifying the type of information that will be contained in the payload of a packet. For instance, an SMTP email message will be directed to port 25 so the receiving device automatically knows to forward the payload data to the email application without having to decode the actual payload to figure out what it contains.

AoIP is similar in that it is desirable to forward the payload data directly to the AoIP streaming application. Of course, different vendors have chosen their own ports to use: AES67 specifies a standard port, 5004, that all devices should be capable of using.

Bottom line:

By providing specific configuration details for AoIP streams, AES67 makes it possible for devices that adhere to these details to stream audio between them. Since vendors of AoIP devices may have existing systems that were built before there was an AES67 standard and will want to maintain streaming capability with their own equipment, there must be a mechanism to specify these details on a case by case basis as the needs arise. In WheatNet-IP, this is done via the Navigator software application. Other AoIP vendors have their own mechanisms they use to alter their device configuration parameters and you will need to be familiar with and use each vendor's methods for the AES67 devices you have in your system.

Map It Out

Since AES67 only specifies stream content parameters and does nothing to manage stream discovery and control, these functions must be managed manually. AES67 does not include a standard way for AoIP streams to be identified. There is no way for a device from vendor A to "see" or know about a stream from vendor B unless both just happen to be using the same stream discovery mechanism. The current work-around is to specifically tell device B that there is a stream from device A available with such and such parameters that it should start receiving. This is the purpose of the SDP (Session Description Protocol) mentioned in the AES67 standard. Not only does the standard specify the parameters in an AES67 stream, it also specifies how those parameters should be presented. Therefore, the system engineer becomes the audio router by manually calling out which devices should listen to which streams.

This is easy enough until you realize that because the different devices can't "see" each other, it is entirely possible that they may be using address settings already in use elsewhere.

Bottom line:

The first step in commissioning AES67 is to map out an IP and stream multicast address plan that assures every stream on every device will have a unique address available, keeping in mind that multicast traffic is not normally routed across subnets.

Multicast addresses are in the form of 239.xxx.yyy.zzz. Each AoIP vendor has their own way of allocating addresses for each stream. WheatNet-IP does it automatically, but the user can change the starting address and range if needed. Axia uses “Livewire channel numbers” but can also accept a manually entered address. Dante will let the user specify the xxx octet of the address but automatically generates the yyy and zzz without any user control of what it chooses. Other equipment may do this yet differently.

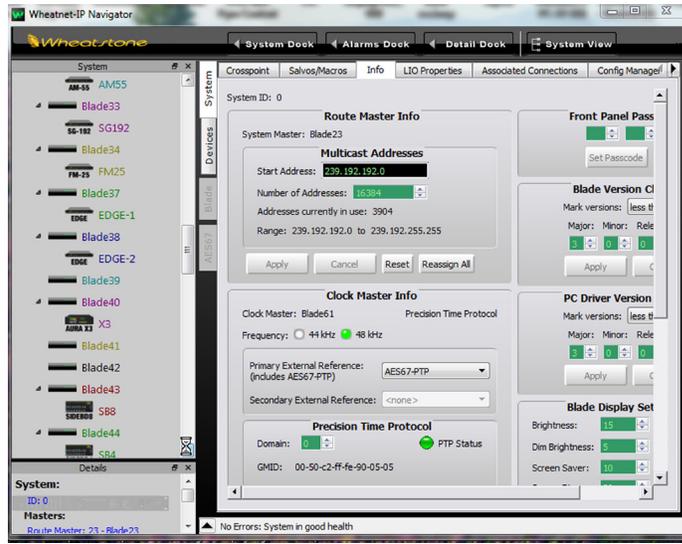
To make your stream multicast address plan you must know all the different devices that will be in the system and how they allocate multicast stream addresses. Start with the devices that are least common or least flexible in specifying or changing multicast addresses and isolate them in an address range well removed for what the majority of your devices will be.

As an example, if you have a WheatNet-IP system with 50 BLADEs and you want to add two Dante devices to it, you would give the Dante device streams a multicast address of .192 for the second octet and check what multicast addresses Dante auto-assigns to the third and fourth octets. Then have the WheatNet-IP devices auto-generate multicast addresses starting lower or higher than the Dante addresses to make sure there are no WheatNet-IP streams assigned to the Dante addresses.

It's important to go through this effort to create a plan first, because as you add AES67 devices to your system, you will be doing a lot of hand entry specifying stream addresses devices are using to transmit on and which stream address devices are using to receive. If you go through all this effort only to discover that some other device is using that same address, you will have to start all over. Besides, having your multicast address plan in place will be useful later when you are routing AES67 streams in your network; you'll refer to it over and over again.

BLADEFest Test Run

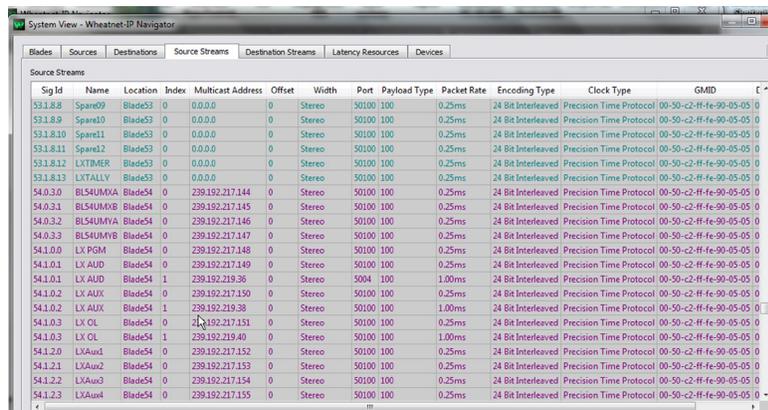
To set up our test run of AES67, and with a multicast plan in place, we opened the Systems Info tab in the Navigator application. Most AoIP systems have an administration application; ours is called Navigator. Here is what our screen looked like:



In our case, because we already had the PTPv2 master clock installed and running, Navigator had detected it and showed its status in the Master Clock Section, as you see in the above screen grab. You can also see that we set the system sample rate to 48kHz; unless you know the device sample rate, you should set your system sample rate to 48kHz, as AES67 does not require devices to support 44.1kHz and many do not.

Once the PTPv2 clock is running, the system is compatible with AES67 and it's possible to begin connecting AES67 devices to the network.

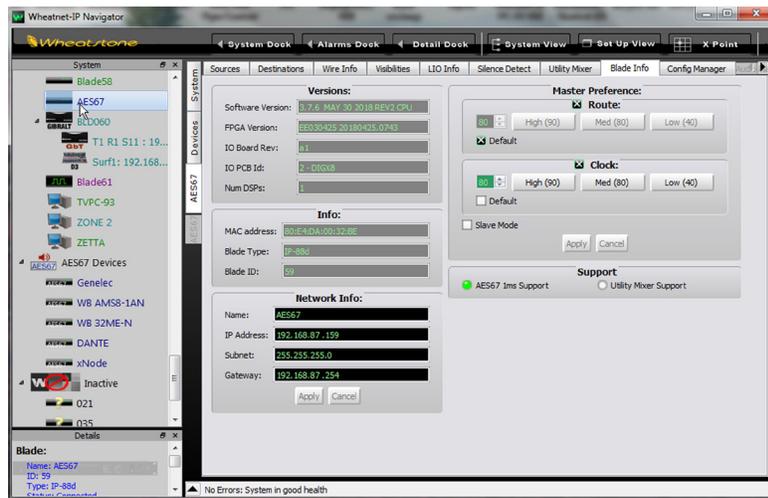
Our next step was to make note of the streams and address allocations within WheatNet-IP. You'll need this to "tell" AES67 devices the information for the WheatNet-IP streams you want them to receive. This Navigator screen shows all of the WheatNet-IP streams available in the system and gives information about their assigned IP and multicast addresses, ports, and packet timing.



Sig Id	Name	Location	Index	Multicast Address	Offset	Width	Port	Payload Type	Packet Rate	Encoding Type	Clock Type	GMID
531.8.8	Spare09	Blade53	0	0.0.0.0	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
531.8.9	Spare10	Blade53	0	0.0.0.0	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
531.8.10	Spare11	Blade53	0	0.0.0.0	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
531.8.11	Spare12	Blade53	0	0.0.0.0	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
531.8.12	LXTIMER	Blade53	0	0.0.0.0	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
531.8.13	LXTALLY	Blade53	0	0.0.0.0	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.0.3.0	BLS4UMXA	Blade54	0	239.192.217.144	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.0.3.1	BLS4UMDB	Blade54	0	239.192.217.145	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.0.3.2	BLS4UMYA	Blade54	0	239.192.217.146	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.0.3.3	BLS4UMPB	Blade54	0	239.192.217.147	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.0.0	LX PGM	Blade54	0	239.192.217.148	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.0.1	LX AUD	Blade54	0	239.192.217.149	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.0.1	LX AUD	Blade54	1	239.192.219.36	0	Stereo	5004 100	1.00ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.0.2	LX AUX	Blade54	0	239.192.217.150	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.0.2	LX AUX	Blade54	1	239.192.219.38	0	Stereo	50100 100	1.00ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.0.3	LX CL	Blade54	0	239.192.217.151	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.0.3	LX CL	Blade54	1	239.192.219.40	0	Stereo	50100 100	1.00ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.2.0	LXAux1	Blade54	0	239.192.217.152	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.2.1	LXAux2	Blade54	0	239.192.217.153	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.2.2	LXAux3	Blade54	0	239.192.217.154	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0
54.1.2.3	LXAux4	Blade54	0	239.192.217.155	0	Stereo	50100 100	0.25ms	24 Bit Interleaved	Precision Time Protocol	00-50-c2-ff-fe-90-05-05	0

Because AES67 does not specify stream discovery and connection management, we manually reassigned multicast addresses to AES67 receiving devices. In the previous screen shown, for example, you'll see that WheatNet-IP configured the AES67 device BL54UMIXA to transmit on 239.192.217.144.

The next step was to determine whether or not we needed to use packet timing translation with the AES67 devices. WheatNet-IP uses 1/4 ms packet timing for minimum latency. Most AES67 devices also support 1/4 ms packet timing but some (namely Dante) do not.



For those devices that do not use 1/4 ms packet timing, we enabled the AES67 1ms Support functionality for the associated BLADE, as shown in the above screen.

With these few settings, we then were able to start connecting AES67 devices.

Example 1: Bringing in audio streams from an Axia xNode using 1/4ms packet timing.

To set up an xNode IP address into the WheatNet-IP subnet, we simply opened up the PC browser and logged onto the xNode. We then set the IP address for the xNode to one that is typically unused.

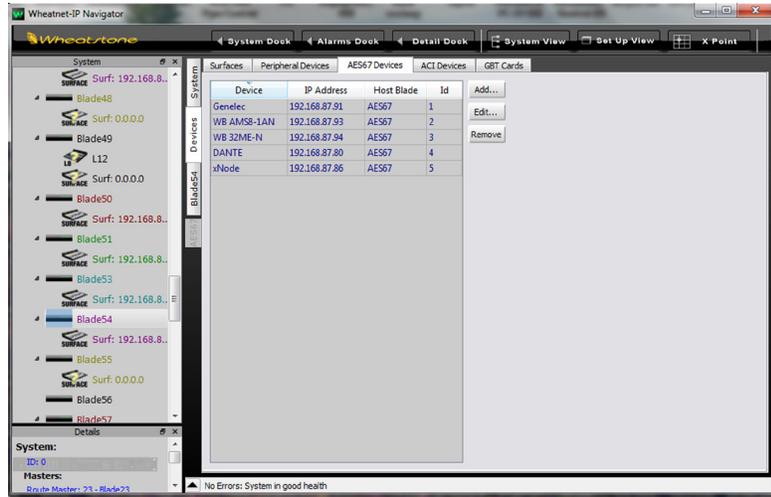
IP settings:

Host name:

NET1 Network address:

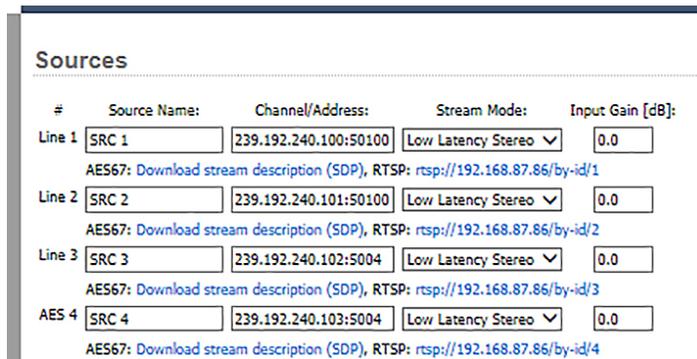
NET2 Network address:

We then set up IP addresses for all of our devices in the network. Below is a WheatNet-IP audio network screen showing a number of different AES67 devices that we added to the system.



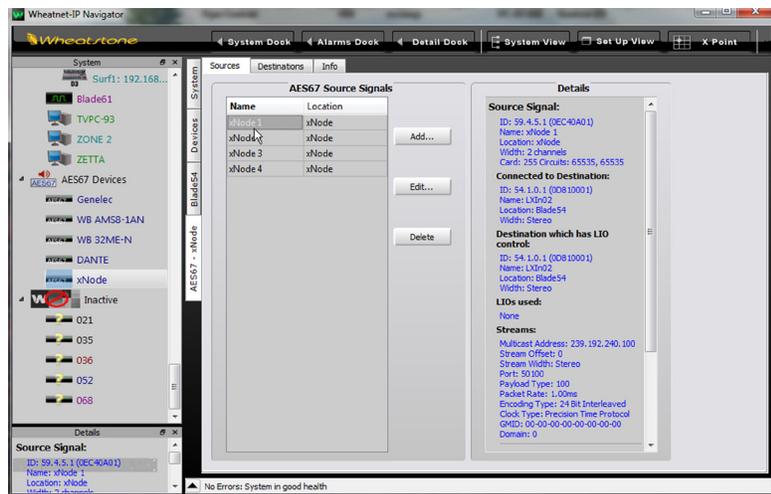
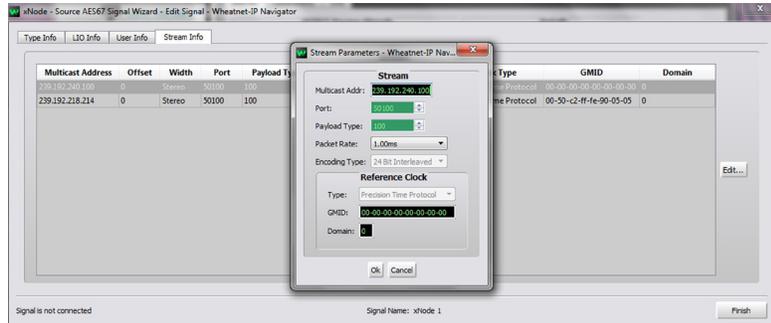
Once we added AES67 devices to the network, we then specified the particular audio streams between WheatNet-IP and each AES67 device.

Here, we've configured the four xNode channels to make AES67 compatible streams.

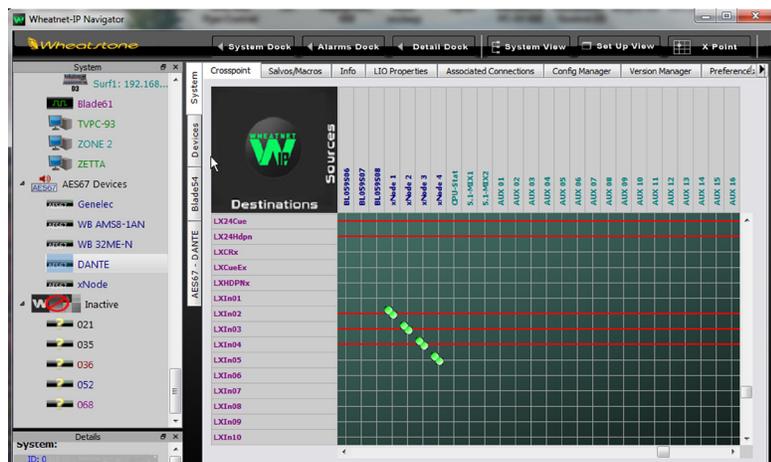


We then made the necessary configurations on the WheatNet-IP side.

Here, we defined the streams to match the xNode sources.



After we defined all the streams in WheatNet-IP for AES67 devices, our screen looked like the one shown above. Below is another view of the AES67 devices on the Navigator crosspoint grid showing which sources connect to which destinations in the WheatNet-IP system.



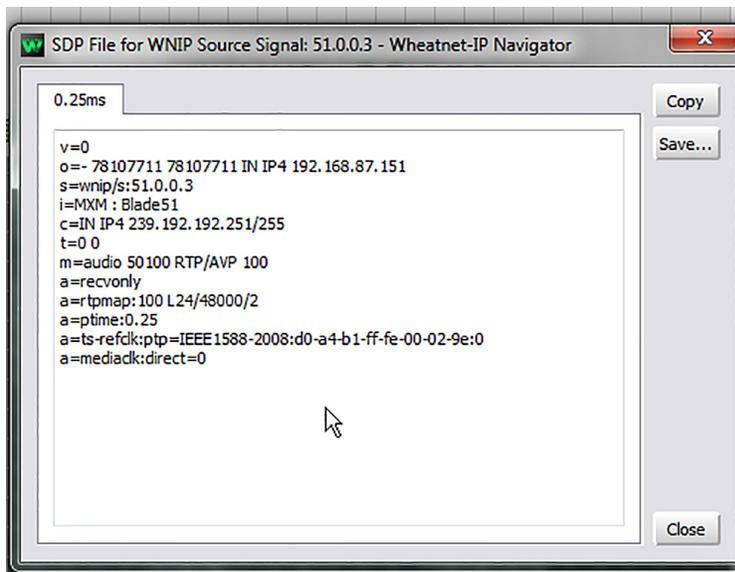
You'll see in this crosspoint grid that the AES67 device stream appears as a source signal in the system and WheatNet-IP destinations connect to it by clicking on the crosspoint. WheatNet-IP "recognizes" packet timing for each specific connection request, either 1 ms or ¼ ms.

A Word About SDP files

The AES67 standard provides a way for stream transmitters to encapsulate all of the multicast address/IP address/port/packet timing/format information about their AES67 streams in a standardized way; this is known as the SDP file.

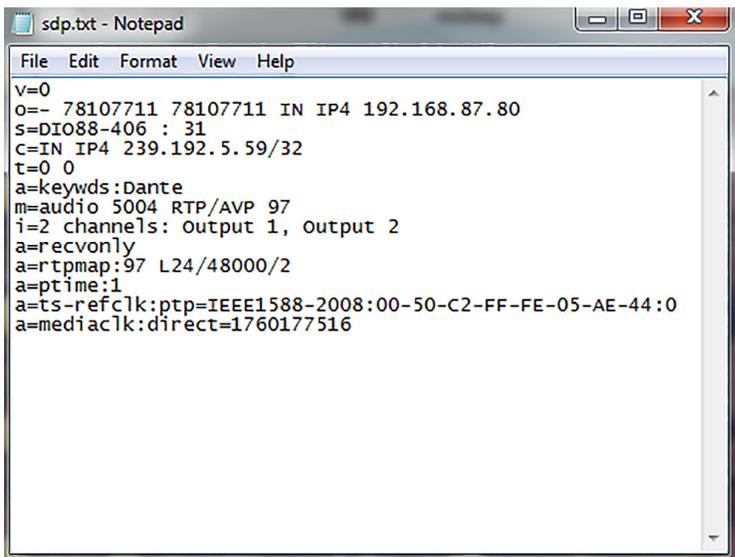
Some AES67 devices will not let you manually manage streaming details as we have done above, although these devices often can ingest these details in the form of the SDP file. In these cases, we created SDP files for the WheatNet-IP streams by simply right-clicking on the desired source stream's name in the Navigator crosspoint grid and opening a window that let us create the file.

Here are a few sample SDP files from WheatNet-IP and Dante showing multicast address, packet timing, sample rate and stream formats.



```

SDP File for WNIP Source Signal: 51.0.0.3 - Wheatnet-IP Navigator
0.25ms
v=0
o=- 78107711 78107711 IN IP4 192.168.87.151
s=wnip/s:51.0.0.3
i=MXM : Blade51
c=IN IP4 239.192.192.251/255
t=0 0
m=audio 50100 RTP/AVP 100
a=recvonly
a=rtpmap:100 L24/48000/2
a=ptime:0.25
a=ts-refclk:ptp=IEEE1588-2008:d0-a4-b1-ff-fe-00-02-9e:0
a=mediaclk:direct=0
  
```



```

sdp.txt - Notepad
File Edit Format View Help
v=0
o=- 78107711 78107711 IN IP4 192.168.87.80
s=DIO88-406 : 31
c=IN IP4 239.192.5.59/32
t=0 0
a=keywds:Dante
m=audio 5004 RTP/AVP 97
i=2 channels: output 1, output 2
a=recvonly
a=rtpmap:97 L24/48000/2
a=ptime:1
a=ts-refclk:ptp=IEEE1588-2008:00-50-C2-FF-FE-05-AE-44:0
a=mediaclk:direct=1760177516
  
```

In Closing

The methods vendors use are quite varied and may take a bit of studying to understand in order to commission AES67 in the plant, but the principles are the same:

- Provide a PTPv2 master clock source by using a Master Clock.
- Assure all devices are on the same IP subnet as multicasting does not normally cross subnet boundaries.
- Configure the desired multicast addresses, port, packet timing, and payload type for Source streams.
- Configure Destinations with the stream details for the desired stream to receive.

Here's to successful interoperating with AES67!

Some screen shots of various vendors configuration screens for commissioning AES67:

