RAVENNA

V1.0

RTP Payload Format for AES3

This document describes a suggested RTP payload format for transparent transport of AES3 audio data.

ALC NetworX

ALC NetworX GmbH

Am Loferfeld 58
81249 Munich
GERMANY

Fon: +49 (89) 44236777-0
Fax: +49 (89) 44236777-1

@: info@alcnetworx.de

www.alcnetworx.de

Version information:

V1.0 – 2012-02-14

Any information contained in this document may be changed or amended at any point in time without further notice. The document may also be replaced as a whole.

RAVENNA - RTP Payload Format for AES3

DRAFT 1.0

TOC

1 MOTIVATION

The AES3 interface has originally been defined as a transport for stereo (or 2-channel) PCM audio with a word length from 16 to 24 bit. It is very similar to the S/P-DIF format used in consumer devices, so that the discussion here applies to both. The IEC 60958 standard covers both, even though the "professional" implementation is maintained by the AES, and both standards bodies strive to keep the two standards in sync with each other.

AES3 not only multiplexes two channels of audio into a single data stream, but also provides for clock synchronization and offers low-bandwidth sideband data that serves to transmit control information and additional user information along with the audio data.

Over time many ways have been developed to incorporate AES3 formatted data in other types of transport. One of the earliest examples is AES10 (MADI), but many other examples have followed, notably SDI embedded audio (up to 16 channels of AES3 audio multiplexed into a video data stream). In recognition of this, the AES has recently restructured the AES3 standard to better separate the physical transport from the logical format (AES3-2009), and has given some guidelines on how to incorporate AES3 data into other transports (AES-2id).

The AES3 interface has also been used to transport data other than PCM formatted audio. One prominent example is Dolby E data, which employs psychoacoustic data compression techniques to transport 6 or 8 channels of surround audio in one single AES3 formatted data stream. The result requires encoders and decoders on sending and receiving ends in order to convert between PCM format and the compressed format. The Dolby E signal can therefore not be interpreted correctly by most devices. It can be routed, however, through most devices that offer a transparent data path, which makes a use case for RAVENNA networks.

This situation raises the question how to incorporate AES3 formatted data in a payload format for RTP that allows preservation of the control and user data in addition to the audio, and allows adequate compatibility detection before connecting to a stream.

2 ALTERNATIVES FOR DISCUSSION

2.1 Plain 32-bit transparent format (L32)

One may call this format "L32" with the implication that it carries 32-bit audio samples in two's complement format in the same manner as in the L24 and L16 payload definitions. The channel status and user bits would be placed in the least significant bits of the word.

Advantages:

+ Simplicity, both in definition and in implementation.
+ 32-bit alignment, leading to efficient code in payload codec.

Disadvantages:

- No standard assignment exists for status and user bits. We would have to define where they go in the sample word, and the definition cannot match all existing hardware.
- No distinction exists between real 32-bit audio and AES3 encoded audio. How does the receiver distinguish that?
- Some information that is important for determining whether a receiver is compatible with the stream is carried in the status bits and cannot be known to the receiver before connecting to the stream. Particularly, how does one prevent a receiver node from receiving a Dolby E stream when it is not able to handle this format (i.e. think about a loudspeaker)?
- No compatibility with L24 or L16 - the stream can only be received by nodes that support L32.

2.2 Separate stream for AES3 sideband data

The main audio could be transmitted using an L24 or L16 format, and a separate stream could be used to transmit the channel status and user bits. We would need a definition of the payload format for sideband data only.

Advantages:

+ Compatibility with L24 and L16 codecs. Nodes who don't care about sideband data just receive the raw audio and need not understand the AES3 sideband payload format.

Disadvantages:

- More complicated, because two (or even more) separate streams are needed which must be re-multiplexed at the receiving end.
- Information needed to determine receiver compatibility is still only in a stream, and worse, it is in a different stream than the audio stream.

2.3 AES3 sideband data in the RTP header extension

AES3 sideband data could be put into a header extension field of the RTP header. The payload format would still be L24 or L16, retaining compatibility with receivers that aren't interested in AES3 sideband data.

Advantages:

- + Compatibility with L16 and L24 codecs, provided they can skip RTP header extensions (which they are required to anyway).
- + Only one stream is involved

Disadvantages:

- − Legality of this is very questionable. Header extensions are not meant for payload-specific data, and the RTP standard is quite clear that such payload-specific extension data belongs into the payload section and not into the header. This solution will most probably be rejected by the IETF.

2.4 Specific AES3 payload format

This could be formatted in the same way as the L32 format described above, but distinguishes itself from L32 in specifying that the lower 8 bits of each word contain no audio, but sideband data.

Advantages:

- + Clean solution that can't be mixed up with 32-bit raw PCM audio.
- + Efficiency maintained because of 32-bit alignment.
- + Only one stream.

Disadvantages:

- − No backwards compatibility with L24 and L16
- − Bit position of sideband data cannot be defined such that it is compatible with all hardware.

2.5 Payload format for AM824 formatted data

In conjunction with Firewire (IEEE1394) there is a definition for AM824 formatted data standardized in IEC 61883. Among other things, AM824 provides an encoding for AES3 data including the sideband data.

Advantages:

- + Starting point for standardization is there in the form of IEC 61883 and an old and expired IETF draft.

- + More general solution that reaches beyond AES3 data.
- + 32-bit aligned data, single stream.
- + Some compatibility with Firewire, and hence also AVB

Disadvantages:

- − More complicated, as it allows non-audio data formats as part of the AM824 definition, which need to be handled appropriately.

2.6 Encapsulation in MPEG-2 transport stream

There is a payload format for MPEG-2 transport streams (RFC 2250), and SMPTE 302M defines a method of encapsulating AES3 in an MPEG-2 transport stream. Combining the two format definitions renders a payload format for AES3 that is already standardized.

Advantages:

- + Standard already available, no new effort needed

Disadvantages:

- − Encapsulation inside MPEG-2 TS adds quite a lot of unnecessary complexity
- − MPEG-TS can contain quite a lot more data that the receiver may not be able to interpret

3 RELATIONSHIP WITH AND IMPACT ON SDP SIGNALIZATION

3.1 What is the problem?

In RTP streaming systems, a receiver wants to know all relevant information about a stream before it actually connects to it. Relevant information is what needs to be known in order to properly decode and use the stream-data. A situation where one finds out that decoding the stream is impossible only after reception has started is not desirable.

In AES3, information that is important for proper decoding of the content is encoded as sideband information in the channel status blocks of the data stream. As such it is only accessible when receiving the RTP packets. The RTP packets, however, are available only after having connected to the stream.

We need to identify what this relevant information is, and define a way how it is put into the SDP description. This will allow a receiver to decide in advance whether the stream is acceptable or not. Unfortunately, there are still potential problems associated with this:

- The sideband information can change during a running stream. If this information is replicated in the SDP description, an inconsistency results. It is unclear what the reaction of the system should be.
- There may be situations where the transmitter cannot determine the relevant information with any degree of reliability. For example it may receive the data from the outside and be unable to determine the sideband data before the actual streaming starts. Consider for example a device that receives a continuous AES3 signal and converts it into a stream. How is this device supposed to behave? What if the AES3 signal on the input switches modes mid-way?

3.2 A possible solution

One way of dealing with this problem can be to provide optional entries in SDP that restrict and further describe the data that can be sent as the payload. If there are no such restrictions recorded in the SDP description, the receiver must expect anything, and examine the sideband data in the running stream in order to find out about the actual encoding and interpretation of the data.

If there are restrictions specified in the SDP description, the receiver can expect the stream to comply with them. A stream that violates these restrictions is in error, and the receiver cannot be blamed for misinterpreting it. In other words, the SDP takes precedence over the sideband data contained in the stream.

Such restrictions may be (among others):

- PCM encoding or non-PCM encoding of the audio data
- Actual word length of the audio data
- Emphasis

- Channel mode (stereo, 2-channel, mono, single channel double frequency, …)
- …

A syntax for describing these in the SDP document needs to be developed.

4 PROPOSAL

4.1 Guiding thoughts

We believe that raw 32-bit audio (i.e. a 32-bit two's complement number as the sample value) is different from 24-bit audio with AES3 sideband data, hence they should be treated as distinct payload types. If not, misinterpretation is likely in a receiver, with unpredictable and potentially weird results.

We also believe that there is a lot to be said in favor of the AM824 encoding, particularly when we envisage future nodes which are AVB compatible and RAVENNA compatible at the same time. AM824 provides for full AES3 transparency and offers additional data types which might be of interest to us in the future (for example MIDI).

Our proposal therefore is to define a payload format for AM824 encoded data, and use this for providing AES3 transparency from end to end.

4.2 Payload format definition

The AM824 data type is defined in IEC61883-6. It divides the 32-bit word into a 24-bit data field and an 8-bit label. The label forms the most significant byte in the word. The label encodes the data type, and in the case of AES3 (actually IEC 60958) it also encodes the sideband data. The meaning and encoding of the data field depends on the label. In case of AES3 the data field holds the sample value as defined in AES3. The mapping between AES3 and AM824 is direct and straightforward.

We propose to stay close to the L16 and L24 formats with this payload type, with the same data arrangement, i.e. channel interleaved with the samples in network byte order (big-endian). No CIP header should be included, just the samples in AM824 format. The CIP header would only duplicate information that is available from other sources, hence there is no sensible use for it. This is different from the earlier "Fairman-draft" which attempted to define a payload format for complete IEC61883 type packets including CIP header, a much more general approach that is also more complex.

In the same way as in L16 and L24, the channel count is encoded in the SDP, and there must be an integral number of samples for each channel in a packet. The receiver derives the number of samples from the packet size.

Each channel should keep its data type during streaming, for example it should not switch between IEC 60958 and MIDI mid-stream. Different channels in the same stream may, however, differ in data type. For example, it is ok to have a mixture of MIDI and IEC 60958 channels within the same stream.

Note that we use the term "data type" in an informal way here, since several label values can denote the same data type. Generally, a data type is denoted by a range of label values.

IEC 60958 type data does not necessarily come as linear PCM. It may just as well be in Dolby E format, or it may be in one of the compressed formats defined by IEC 61937. In the general case, one can only determine the actual format by looking at the sideband data contained within the stream, i.e. by looking at the channel status bits encoded in the label bytes. A stream receiver should therefore not blindly assume that the data is valid PCM, but instead behave in a way similar to what an AES3 receiver would do. The SDP stream description can contain restricting information, however, that overrides the settings in the channel status bits, and can tell the receiver in advance what encoding to expect.

4.3 MIME type

The new format should be given the MIME type "AM824", which needs to be registered with the IANA.

4.4 SDP description

The encoding within an SDP description is very similar to L16 and L24. In addition to the AM824 encoding, sampling rate and channel count need to be given to specify the media format representation. Here's an example:

```
m=audio 49230 RTP/AVP 97

a=rtpmap:97 AM824/48000/8
```

Restrictions can be given as format-specific attributes, and the set of attributes that are supported is yet to be defined. There will be quite a few, in order to cover the rather large variety of encodings that can be carried within the AM824 and IEC 60958 formats. It is as of now an open question how this set of attributes can be defined such that future extensions of IEC 60958 or AM824 can be covered in a backwards compatible way.

Restrictions given inform a receiver in advance what to expect from the stream. A restriction effectively overrides some encoding in the AM824 labels and/or in the IEC 60958 channel status block of a channel. Without a restriction, the receiver has to look into the data stream to determine a particular setting, whereas with a restriction it may assume the setting without looking at the data stream. Without a restriction, the receiver also has to be prepared for mid-stream changes to the settings.

Restrictions offer the possibility to check stream properties in advance, before connecting to a stream, and hence reduce the chance of making a nonsensical connection that can not be interpreted properly by the receiver. A source device, however, may not have full knowledge of the actual stream format details in advance, so it may not be able to determine a full set of restrictions for the SDP description.

5 IMPLEMENTATION NOTE

5.1 Forwarding of sideband data to the audio hardware

Inside a node, the sideband data needs to be made available to the hardware. For example, a receiving node must be able to pick the channel and user bits in order to insert them into an AES3 data stream, if the signal is to be output on such an interface. How to do that is strictly out of scope for an AM824 payload definition, but it makes sense to discuss it here in order to complete the chain for the entire application.

Most systems use the low-order 8 bits in an I2S signal to convey such sideband data. ADCs and DACs with up to 24-bit resolution typically ignore those extra 8 bits, so no harm is done when extra information is transmitted on them. Unfortunately, there is no standard for this method, and each manufacturer decides on the formatting and the semantics of the lower 8 bits autonomously. As a consequence, various mutually incompatible systems are in use.

Our proposal is to stick closely to the AM824 format to minimize the work for the RAVENNA streaming core, and leave most of the interpretation to the hardware that receives or sends the I2S streams. In particular, the flag byte, which is the MSB in AM824, should be rotated to the LSB position. Processors with a rotate-instruction can do this directly.

In order to provide backwards compatibility for plain I2S, we propose to invert bit 6 in the flag byte. For a node that sends I2S signals, this means that no special handling is necessary as the flag 0x40, which means "plain 24-bit linear audio", gets translated to all zeroes in the lower I2S byte.

For a receiver of I2S signals, a zero lower byte means "plain 24-bit linear audio", exactly what it would mean in standard I2S. If an AES3 signal is to be formed from that, the channel status and user bits would have to be provided locally by the audio hardware, as they were not received from the RAVENNA stream.

If the low byte in the I2S frame is not zero, it may be in various different formats according to AM824, and the hardware would have to decode the byte, taking into account that bit 6 is inverted from what AM824 defines. Typically, it would mute the output if it finds a format it doesn't support.

Document information:

Authors: Stefan Heinzmann, Andreas Hildebrand

Editor: Andreas Hildebrand

Version: 1.0

Last revision: 2017-07-13 by Andreas Hildebrand

File: AM824 - RTP payload format for AES3.docx