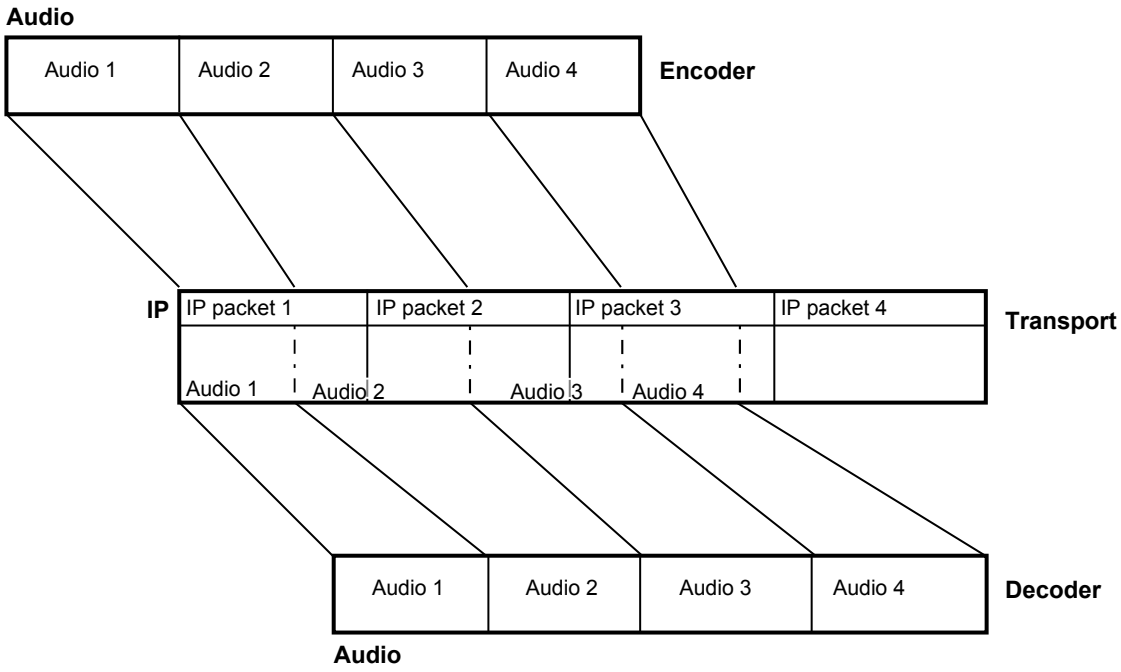
Audio over IP

1 Transport of audio via IP packets

In the average case of audio data transport via IP, one IP packet carries exactly one audio frame. Depending on frame length and packet size there is a specific overhead in each IP packet. An exception can occur if the data transport isn't working correctly due to different possible reasons, e.g. network problems. In this situation, if the IP packets are refused and have to be sent again, the audio frames are allocated to the IP packets in a different way.

1.1 One IP packet carries more than one frame

This would be the normal way of audio transport in a situation described above. As you can see in the following figure, the first IP packet carries not only the first audio frame but also parts of the second one. The second IP-packet carries the rest of Audio 2 and parts of Audio 3. IP-packet 3 carries the rest of Audio 3 and Audio 4 etc..



1.2 One frame is carried by more than one IP packet

In specific situations, e.g. if the bit rate is high and the sample rate is low, the frame length can be larger than the packet size. In this case the first IP packet would only carry parts of the first audio and IP packet 2 the rest etc..

In both situations it is necessary to get access to the frame header to be able to read the information about frame length, bit rate, sample rate etc.. How to get this information is dealt with in the following chapter.

2 Header information

The structure of the headers for MPEG Layer2/3, AAC/CT-aacPlus and AAC Low Delay are described in the following chapters.

2.1 MPEG Layer 2 / Layer 3 frame header

```
                              {++++ ++++ ++++--------------------------- Synch
                               |||| |||| |||| +------------------------- Ident
                               |||| |||| |||| |++----------------------- Layer
                               |||| |||| |||| |||+---------------------- Protection
                               |||| |||| |||| |||| ++++----------------- Bitrate
                               |||| |||| |||| |||| |||| ++------------- SamplFreq
                               |||| |||| |||| |||| |||| ||+------------ Padding
                               |||| |||| |||| |||| |||| |||+----------- Private
                               |||| |||| |||| |||| |||| |||| ++--------- Mode
                               |||| |||| |||| |||| |||| |||| ||++------ Modeextension
                               |||| |||| |||| |||| |||| |||| |||| +---- Copyright
                               |||| |||| |||| |||| |||| |||| |||| |+--- Original/Copy
                               |||| |||| |||| |||| |||| |||| |||| ||++- Emphasis
                               |||| |||| |||| |||| |||| |||| |||| ||||}
HEADER_MASK      = $FFFE0000;{1111 1111 1111 1110 0000 0000 0000 0000}
HEADER_MPEG1     = $FFFC0000;{1111 1111 1111 1100 0000 0000 0000 0000}
HEADER_MPEG1_L3  = $FFFA0000;{1111 1111 1111 1010 0000 0000 0000 0000}
HEADER_MPEG2_L2  = $FFF40000;{1111 1111 1111 0100 0000 0000 0000 0000}
HEADER_MPEG2_L3  = $FFF20000;{1111 1111 1111 0010 0000 0000 0000 0000}
HEADER_MPEG25_L  = $FFE20000;{1111 1111 1110 0010 0000 0000 0000 0000}

SAMPLFREQ_MASK   = $00000C00;{0000 0000 0000 0000 0000 1100 0000 0000}
BITRATE_MASK     = $0000F000;{0000 0000 0000 0000 1111 0000 0000 0000}
PADDING_MASK     = $00000200;{0000 0000 0000 0000 0000 0010 0000 0000}
MODE_MASK        = $000000C0;{0000 0000 0000 0000 0000 0000 1100 0000}
MODE_EXT_MASK    = $00000030;{0000 0000 0000 0000 0000 0000 0011 0000}
PROTECTION_MASK  = $00010000;{0000 0000 0000 0001 0000 0000 0000 0000}
PRIVATEBIT_MASK  = $00000100;{0000 0000 0000 0000 0000 0001 0000 0000}
```

2.2 AAC/CT-aacPlus-Header

```
                                {++++ ++++ ++++-------------------------- Synch
                                 |||| |||| |||| +------------------------ Ident
                                 |||| |||| |||| |++---------------------- Layer
                                 |||| |||| |||| |||+--------------------- Protect. absent
                                 |||| |||| |||| |||| ++------------------ Profile
                                 |||| |||| |||| |||| ||++ ++------------- SamplFreq
                                 |||| |||| |||| |||| |||| ||+------------ Private
                                 |||| |||| |||| |||| |||| |||+ ++-------- Chan. Config.
                                 |||| |||| |||| |||| |||| |||| ||+------- Original/Copy
                                 |||| |||| |||| |||| |||| |||| |||+------ Home
                                 |||| |||| |||| |||| |||| |||| |||| ||||}
HEADER_MPEG2_AAC     = $FFF80000;{1111 1111 1111 1000 0000 0000 0000 0000}
PROTECTION_MASK_AAC  = $00010000;{0000 0000 0000 0001 0000 0000 0000 0000}
PROFILE_MASK_AAC     = $0000C000;{0000 0000 0000 0000 1100 0000 0000 0000}
SAMPLFREQ_MASK_AAC   = $00003C00;{0000 0000 0000 0000 0011 1100 0000 0000}
PRIVATEBIT_MASK_AAC  = $00000200;{0000 0000 0000 0000 0000 0010 0000 0000}
MODE_MASK_AAC        = $000001C0;{0000 0000 0000 0000 0000 0001 1100 0000}
ORIGINAL_MASK_AAC    = $00000020;{0000 0000 0000 0000 0000 0000 0010 0000}
HOME_MASK_AAC        = $00000010;{0000 0000 0000 0000 0000 0000 0001 0000}
```

2.3 AAC Low Delay

```
                                {++++ ++++ +++---------------------------- Synch
                                 |||| |||| |||+ ++++ ++++ ++++----------- Length
                                 |||| |||| |||| |||| |||| |||| +---------- use same stream
                                 |||| |||| |||| |||| |||| |||| |+--------- audio mux vers.
                                 |||| |||| |||| |||| |||| |||| |||| ||||
                                 |||| |||| |||| |||| |||| |||| |||| ||||
                                 |||| |||| |||| |||| |||| |||| |||| ||||
                                 |||| |||| |||| |||| |||| |||| |||| ||||
                                 |||| |||| |||| |||| |||| |||| |||| ||||
                                 |||| |||| |||| |||| |||| |||| |||| ||||
                                 |||| |||| |||| |||| |||| |||| |||| ||||}
HEADER_MPEG4_AAC_LD = $56E00000;{0101 0110 1110 0000 0000 0000 0000 0000}
HEADER_MASK_AAC_LD  = $FFE00000;{1111 1111 1110 0000 0000 0000 0000 0000}
LENGTH_MASK_AAC_LD  = $001FFF00;{0000 0000 0001 1111 1111 1111 0000 0000}
```

3 3010 Signal

The signal of the 3010 has to be embedded into the MSC-channel by the multiplexer and the FAC- and SDC-channel has to be supplied with the multiplex parameters.

COFDM/QAM has to be used for the channel-coding and modulation along with time interleaving and forward error correction.

4 The use of the various protocols

Audio via IP can be done with several protocols in combination with different payload formats.
The remote command that decide which one to use is “com_connect” (see Communication Reference Manual, 3.2.8 Connect Centauri to a Remote Unit)

4.1 TCP

A “t” is necessary in front of the IP address to activate TCP.
A TCP connection is always a bidirectional connection, which means that encoder and decoder is being started. The far end device starts its encoder and decoder appropriate to the settings in the calling device. The different algorithms are identified by unique port assignment so that no port infomation must be specified with the connect command. TCP is specified for unicast only.
e.g.: com_connect t10.0.0.11
TCP isn’t an appropriate protocol for audio transmission and is no longer supported!

4.2 UDP

A “u” is necessary in front of the IP address to activate UDP.
A UDP connection is always a unidirectional connection. The calling device (encoder) controlls the far end device (decoder).The different algorithms are recognized by unique port assignment so that no port infomation must be specified with the connect command.
With UDP unicast and multicast are possible. Encoder and decoder can be started independantly.
If a unicast address is given to the connect command the encoder is being started automatically.
The far end being called is starting ist appropriate decoder automatically.
If a multicast address is given to the connect command, the encoder is being started if the “dependency” is configured to “local” otherwise the decoder joins the multicast group.
e.g.: com_connect u10.1.1.11 (Encoder is being started sending to 10.1.1.11)
e.g. com_connect u225.1.1.11 (Encoder is being started sending to 225.1.1.11 if configured to “local”. If configured to “remote” the decoder joins 225.1.1.11)
UDP isn’t an appropriate protocol for audio transmission and is no longer supported!

4.3 RTP/RTCP

If “r’” is preceded the IP address RTP is used. RTP is also used if nothing was specified.
Like UDP RTP is a unidirectional connection. The different algorithms are identified by the use of the RTP payload type (if possible) or the use of the RTP Header Extensions. Within that special header the payload is described. (The use of Header Extensions for codec identification is proprietary and may be incompatible to codecs from other vendors).
With RTP unicast and multicast are possible. Encoder and decoder can be started independantly.
The behaviour when the encoder or decoder is started is the same as with UDP.
e.g.: com_connect 10.1.1.11 (Encoder is being started sending to 10.1.1.11(Port 5004)
e.g. com_connect 225.1.1.11 (Encoder is being started sending to 225.1.1.11(Port 5004) if configured to “local”. If configured to “remote” the decoder joins 225.1.1.11)
RTP uses port 5004 for codec aggregate 1, 5006 for aggregate 2, etc.
RTP is also used if you preced a “RTP://” but with some differences:
a) You have to specify if the connection is a “sending”(encoder) or a “receiving” connection.
b) You can choose the port freely. If left blank, port 5004 is taken as the default port for codec aggrgate 1 (see above).
c) The audio is used without header extension! So you should use algorithms for the encoder that can be recognized by the RTP payload type
e.g. com_connect 1 rtp://10.1.1.11:5006 (Encoder is sending to 10.1.1.11, Port 5006)
com_connect 2 rtp://224.1.1.11 (Decoder is receiving from 224.1.1.11, Port 5004)

The following algorithms have defined RTP payload types (see RFC 1890):
G.711(a-law, μ-law)
G.722
Linear(16 Bit, 44100Hz, Mono)
Linear(16 Bit, 44100Hz, Stereo),
MPEG-I & MPEG-II Audio(Mpeg-Layer2, Mpeg-Layer3, AAC-Mpeg2, MP3Pro)

4.4 HTTP

HTTP describes only the way SDP(Session Description Protocol, RFC2327) informations are exchanged. The media transport protocol is RTP. To activate HTTP you have to add “http://” to the connect command.
It must be specified if the encoder or decoder should be used.
If using the encoder a port can be specified where the encoder has to its media to. The default port is 5004. If the encoder is started it creates a SDP file in the WWW directory
That SDP file can be used by decoders to learn about the encoder parameters.
A decoder has different possibilities to get that SDP file.
One way is via HTTP, where you have to specify the file name of the encoders SDP file.
e.g.: com_connect 2 http://10.1.1.11/stream1.sdp
causes to get stream1.sdp from 10.1.1.11 and start the decoder with the settings found in that file.
No port must be specifed for the decoder connect.

4.5 SAP

Another way for the decoder to get the SDP information is SAP. (The media transport protocol is RTP) If using SAP all encoders send their SDP information to a special address where all decoders can have a look at.
SAP is activated if “sap://” is used with the connect command.
It must be specified if the encoder or decoder should be used.
If using the encoder a port can be specified where the encoder has to sent its media to. The default port is 5004.The encoder creates a SDP file in the WWW directory if started
e.g. com_connect 1 sap://225.1.1.11
Makes the encoder start streaming to 225.1.1.11, Port 5004.
The SAP information is sent to 224.2.127.254, Port 9875(RFC2974).
From that address the decoder can retrieve SDP informations.
If using the decoder the address can be left blank. In that case the decoder grabs the fist SAP announcement available. If an address is specified the Decoder grabs the first announcement from that address. To choose among different SAP announcements from the same address a Hash-ID must be specified.
e.g. com_connect 2 sap://10.1.1.11:45123
causes the decoder to listen to the SAP address waiting for a stream from 10.1.1.11 with the hash ID 45123.
The hash ID is connection unique. If a running encoder is being rebooted it will setup its encoder automatically. But with a different hash ID! So that a listening decoder, where a hash ID is specified cannot resynchronize!