

AAC码流结构解析

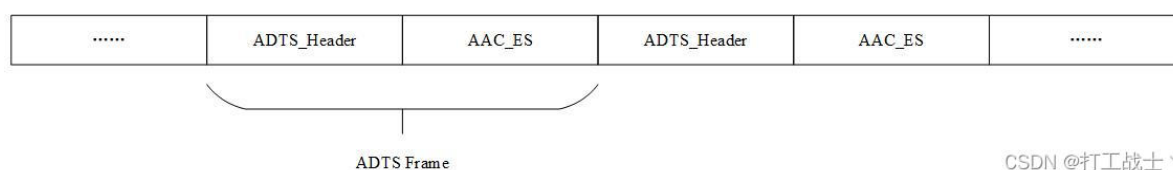
1.AAC格式介绍

音频编码AAC（Advanced Audio Coding）有两种封装格式，音频数据交换格式ADIF（Audio Data Interchange Format）和音频数据传输流ADTS（Audio Data Transport Stream）。二者的区别主要是：

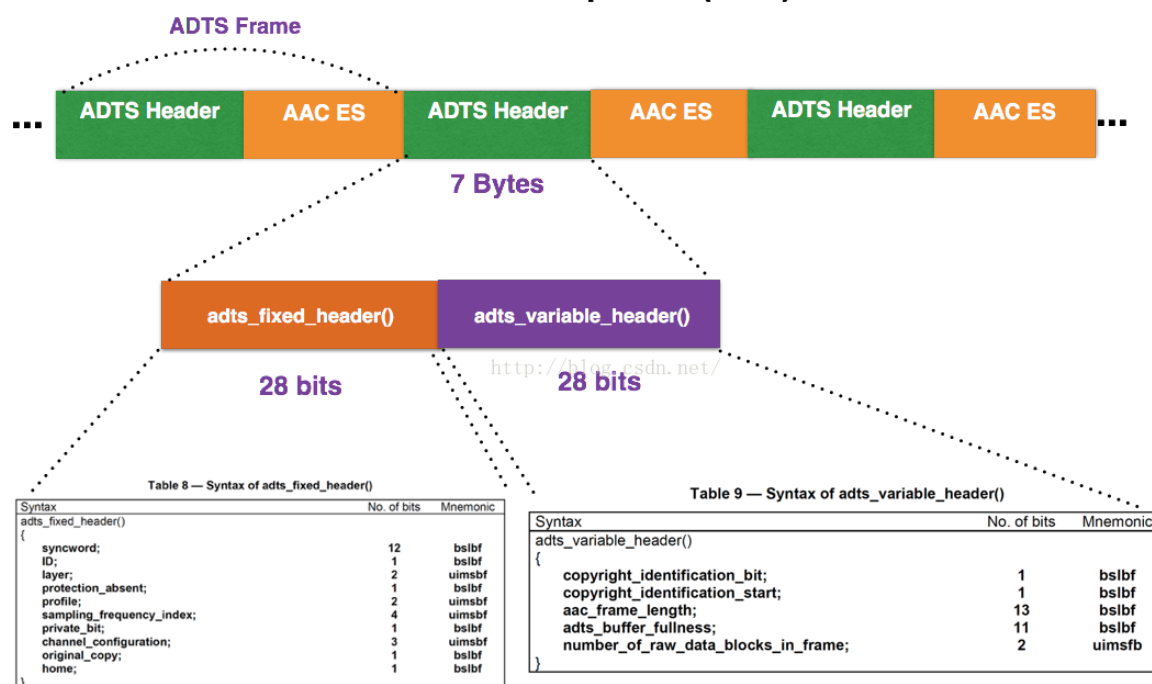
- ADIF只有一个头部，只能在开始的位置进行解码，主要用于本地存储；
- ADTS的每个帧都有一个头部，可以在音频流的任意位置进行解码，常用于流媒体传输

现在用的比较多的主要是ADTS，接下来介绍一下ADTS的结构

AAC原始码流是由一个个ADTS帧组成的，其结构如下图：



ADTS Sequence (AAC)



注:ADTS Header的长度可能为7字节或9字节. protection_absent=0时, 9字节. protection_absent=1时, 7字节.

ADTS_Header大小为一般是7个字节，如果需要对数据进行CRC校验，则会多出2个字节的校验码，那么长度则为9字节，里面包含了采样率、通道数、压缩级别以及帧长度等信息。ADTS_Header分为固定头部和可变头部，均为28bits。固定头部在帧与帧之间是一样的，而可变头部在帧与帧之间可能不同。

接下来介绍一下固定头部的信息。

Syntax	No. of bits	Meaning
Syncword	12	同步字，固定为0xFFF，表示帧开始
ID	1	MPEG标识符，0表示MPEG-4，1表示MPEG-2
layer	2	总是00
protection_absent	1	是否无码校验（CRC），0表示有码，1表示无码
profile	2	使用哪个级别的AAC，级别越高压缩率越大
sampling_frequency_index	4	采样率索引
private_bit	1	是否存在私有数据
channel_configuration	3	声道数
original_copy	1	表示原始音频数据是否进行了拷贝，为1表示该音频数据是原始的，为0表示该音频数据是经过拷贝的
home	1	

接下来介绍一下AAC级别索引表。

index	profile
0	Main profile
1	Low Complexity profile (LC)
2	Scalable Sampling Rate profile (SSR)
3	(reserved)

接下来介绍一下采样率索引表

sampling_frequency_index	value
0x0	96000 Hz
0x1	88200 Hz
0x2	64000 Hz
0x3	48000 Hz
0x4	44100 Hz
0x5	32000 Hz
0x6	24000 Hz
0x7	22050 Hz
0x8	16000 Hz
0x9	12000 Hz
0xa	11025Hz
0xb	8000 Hz
0xc	7350 Hz
0xd	reserved
0xe	reserved
0xf	escape value

接下来介绍一下可变头部的信息

Syntax	No. of bits	Meaning
copyright_identification_bit	1	音频数据是否受到版权保护
copyright_identification_start	1	编码默认值为0，解码忽略此值
aac_frame_length	13	ADTS帧长度，包括ADTS_Header和AAC_ES
adts_buffer_fullness	11	默认是0x7FF，表示码率可变的码流
number_of_raw_data_blocks_in_frame	2	表示ADTS帧中有 number_of_raw_data_blocks_in_frame + 1个AAC原始数据块

结构如下图所示



AAC流数据

Offset	Length	ID	Layer	Sampling Rate	Profile	Channel Configuration
0x0000000000000000	371	1	0	44100 Hz	1	2-LF-RF
0x0000000000000173	372	1	0	44100 Hz	1	2-LF-RF
0x00000000000002E7	371	1	0	44100 Hz	1	2-LF-RF
0x000000000000045A	372	1	0	44100 Hz	1	2-LF-RF
0x00000000000005CE	371	1	0	44100 Hz	1	2-LF-RF
0x0000000000000741	372	1	0	44100 Hz	1	2-LF-RF
0x00000000000008B5	371	1	0	44100 Hz	1	2-LF-RF
0x0000000000000A28	372	1	0	44100 Hz	1	2-LF-RF
0x0000000000000B9C	371	1	0	44100 Hz	1	2-LF-RF
0x0000000000000D0F	372	1	0	44100 Hz	1	2-LF-RF
0x0000000000000E93	371	1	0	44100 Hz	1	2-LF-RF
0x0000000000000FF5	372	1	0	44100 Hz	1	2-LF-RF
0x000000000000116A	371	1	0	44100 Hz	1	2-LF-RF
0x00000000000012D0	372	1	0	44100 Hz	1	2-LF-RF
0x0000000000001451	371	1	0	44100 Hz	1	2-LF-RF
0x00000000000015C4	372	1	0	44100 Hz	1	2-LF-RF
0x0000000000001738	371	1	0	44100 Hz	1	2-LF-RF
0x00000000000018AB	372	1	0	44100 Hz	1	2-LF-RF
0x0000000000001A1F	371	1	0	44100 Hz	1	2-LF-RF
0x0000000000001B92	372	1	0	44100 Hz	1	2-LF-RF
0x0000000000001D06	371	1	0	44100 Hz	1	2-LF-RF
0x0000000000001E79	372	1	0	44100 Hz	1	2-LF-RF
0x0000000000001FE0	371	1	0	44100 Hz	1	2-LF-RF
0x0000000000002160	372	1	0	44100 Hz	1	2-LF-RF
0x00000000000022D4	396	1	0	44100 Hz	1	2-LF-RF
0x0000000000002462	445	1	0	44100 Hz	1	2-LF-RF

每行是一个ADTS Frame

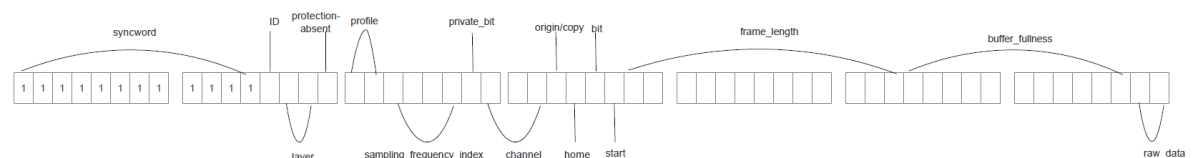
ADTS Header

```
adts_frame()
├─ adts_fixed_header()
│   ├── syncword          : 0x0FFF (12 bits)
│   ├── ID                 : 1 (1 bit)
│   ├── layer              : 0 (2 bits)
│   ├── protection_absent  : 1 (1 bit)
│   ├── profile            : 1 [Low Complexity profile (LC)] (2 bits)
│   ├── sampling_frequency_index : 4 [44100 Hz] (4 bits)
│   ├── private_bit        : 0 (1 bit)
│   ├── channel_configuration : 2 [2 - LF RF] (3 bits)
│   ├── original/copy      : 0 (1 bit)
│   └── home               : 0 (1 bit)
└─ adts_variable_header()
    ├── adts_error_check()
    └─ raw_data_block()
```

Hex View

```
0x0000000000000000  FF F9 50 80 2E 7F FC 21 1C 05 20 7E 22 90 3F 11  yaP...01...~".?
0x0000000000000010  FF FC 00 00 00 00 00 00 00 00 00 00 00 00 00  00
0x0000000000000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00
0x0000000000000030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00
0x0000000000000040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00
0x0000000000000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00
0x0000000000000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00
0x0000000000000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00
0x0000000000000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00
0x0000000000000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00
```

图解ADTS Header 数据格式一般为7个字节



ADTS header中有ADTS frame的大小，但是根据上面同步字咱们可以看出来，这些数据并不是以字节为单位连续排列的，而是按位排列的，从这一张图中可以很清晰的看到，frame_length存储在第4个字节的后两位，第5个字节，第6个字节的前三位