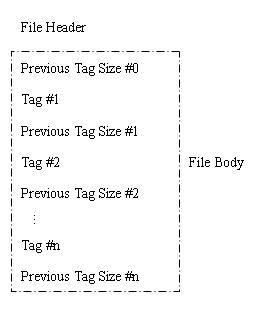
# FLV文件格式解析

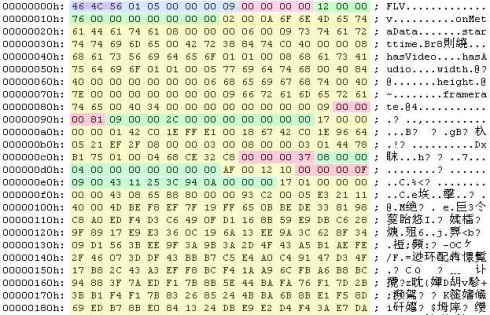
FLV（Flash Video）是现在非常流行的流媒体格式，由于其视频文件体积轻巧、封装播放简单等特点，使其很适合在网络上进行应用，目前主流的视频网站无一例外地使用了FLV格式。另外由于当前浏览器与Flash Player紧密的结合，使得网页播放FLV视频轻而易举，也是FLV流行的原因之一。

    FLV视频格式是Adobe公司设计开发的，目前已经免费开放，现在的版本是v10。下面我们就了解一下FLV文件格式。

    FLV是流媒体封装格式，我们可以将其数据看为二进制字节流。总体上看，FLV包括文件头（File Header）和文件体（File Body）两部分，其中文件体由一系列的Tag及Tag Size对组成。因此一个FLV文件看上去是下面的结构：

[](http://photo.blog.sina.com.cn/showpic.html#blogid=48f93b530100eyoe&url=http://s6.sinaimg.cn/orignal/48f93b53g74f5adde2875&690)

    其中，Previous Tag Size紧跟在每个Tag之后，占4个字节表示一个UI32类型的数值，表示前面一个Tag的大小。需要注意的是，Previous Tag Size #0的值总是为0。Tag类型包括视频、音频和Script，且每个Tag只能包含一种类型的数据。下面我们看看File Header和Tag的具体结构，为了方便大家理解，我贴一个实际FLV文件的字节流图片，图片中把不同部分区分了颜色。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=48f93b530100eyoe&url=http://s1.sinaimg.cn/orignal/48f93b53g74f5d8e518d0&690)

1. **File Header结构**

|  |  |
| --- | --- |
| FLV Header Field | 描述 |
| 文件类型(3byte) | 'F' (0x46) 'L' (0x4C) 'V' (0x56) |
| 版本(1byte) | File Version(For example 0x01 for FLV version 1) |
| 流信息(1byte) | UB[7]~UB[3]、UB[2]=1 Audio,UB[1]总为0，UB[0] = 1 Video |
| Header长度(4bytes) | 整个文件头的长度，一般是9（3+1+1+4），有时候后面还有些别的信息，就不是9了 |

|  |  |  |
| --- | --- | --- |
| 文件类型 | 3bytes | 总是FLV（0x46 0x4C 0x56），否则... |
| 版本 | 1byte | 一般是0x01，表示FLV version 1 |
| 流信息 | 1byte | 倒数第一bit是1表示有视频，倒数第三bit是1表示有音频，其他都应该是0（有些软件如flvtool2可能造成倒数第四bit是1，不过也没发现有什么不对） |
| header长度 | 4bytes | 整个文件头的长度，一般是9（3+1+1+4），有时候后面还有些别的信息，就不是9了 |

    File Header在当前版本中总是由9个字节组成，如图中蓝色区域所示。

    第1-3字节为文件标识（Signature），总为“FLV”（0x46 0x4C 0x56），如图中紫色区域。

    第4字节为版本，目前为1（0x01）。

    第5个字节的前5位保留，必须为0。

    第5个字节的第6位表示是否存在音频Tag。

    第5个字节的第7位保留，必须为0。

    第5个字节的第8位表示是否存在视频Tag。

    第6-9个字节为UI32类型的值，表示从File Header开始到File Body开始的字节数，版本1中总为9。

1. **File Body结构**

|  |  |  |
| --- | --- | --- |
| FLV File Body Field | Type | Comment |
| PreviousTagSize0 | UI32 | Always 0 |
| Tag1 | FLVTAG | First tag |
| PreviousTagSize1 | UI32 | Size of previous tag, including its header. For FLV version 1,this value is 11 plus the DataSize of the previous tag. |
| Tag2 | FLVTAG | Second tag |
| … | … | … |
| PreviousTagSizeN-1 | UI32 | Size of second-to-last tag |
| TagN | FLVTAG | Last tag |
| PreviousTagSizeN | UI32 | Size of last tag |

FLV TagHeader have the following format：

|  |  |  |
| --- | --- | --- |
| FLVTAG Field | Type | Comment |
| TagType | UI8 | Type of this tag. Values are：  8:audio    9:video   18:script data |
| DataSize | UI24 | Length of the data in the Data field |
| Timestamp | UI24 | Time in milliseconds at which the data in the this tag applies. The valve is relative to the first tag in the FLV file, which always has a timestamp of 0. |
| TimestampExtended | UI8 | Extension of Timestamp field to form a UI32 value. This field represents the upper 8 bits, while the previous Timestamp filed represents the lower 24 bits of the time in milliseconds. |
| StreamID | UI24 | Always 0 |

In playback, the time sequencing of FLV tags depends on the FLV timestamps only. Any timing mechanisms built into the playload data format are ignored.

|  |  |  |
| --- | --- | --- |
| previoustagsize | 4bytes | 前一个tag的长度，第一个tag就是0 |
| tag类型 | 1byte | 8 -- 音频tag  9 -- 视频tag  18 -- 脚本tag |
| 数据区长度 | 3bytes |  |
| 时间戳 | 3bytes | 单位毫秒，如果是脚本tag就是0 |
| 扩展时间戳 | 1byte | 作为时间戳的高位 |
| streamsID | 3bytes | 总是0（不知道干啥用） |
| 数据区 |  |  |

**三、Previous Tag Size结构**

    如图中红色位置所示。

**四、Tag结构**

    Tag包括Tag Header和Tag Data两部分。不同类型的Tag的Header结构是相同的，但是Data结构各不相同。如图中绿色和黄色区域分别表示Tag Header数据和Tag Data数据。

    当前版本的Tag Header结构占用11个字节。

    第1个字节表示Tag类型，包括音频（0x08）、视频（0x09）和script data（0x12），其他类型值被保留。

    第2-4字节为UI24类型的值，表示该Tag Data部分的大小。

    第5-7字节为UI24类型的值，表示该Tag的时间戳（单位为ms），第一个Tag的时间戳总是0。

    第8个字节为时间戳的扩展字节，当24位数值不够时，该字节作为最高位将时间戳扩展为32位值。

    第9-11字节为UI24类型的值，表示stream id，总是0。

 后面的字节为Tag Data数据，Data的大小由第2-4字节的数值指示，根据第1个字节指示的Tag类型，按照不同的结构解析Tag Data。下面分别介绍。

**五、Audio Tag Data结构**

|  |  |  |
| --- | --- | --- |
| audio信息 | 1byte | 前四位bits表示音频格式:   * 0 -- 未压缩 * 1 -- ADPCM * 2 -- MP3 * 5 -- Nellymoser 8kHz momo * 6 -- Nellymoser   下面两位bits表示samplerate:   * 0 -- 5.5kHz * 1 -- 11kHz * 2 -- 22kHz * 3 -- 44kHz   下面一位bit表示每个采样的长度：   * 0 -- snd8Bit * 1 -- snd16Bit   下面一位bit表示类型：   * 0 -- sndMomo * 1 -- sndStereo |
| audio数据区 | 不定 |  |

    音频Tag开始的第1个字节包含了音频数据的参数信息，从第2个字节开始为音频流数据。

    第1个字节的前4位的数值表示了音频数据格式，各值代表的意义大家可以查阅Adobe公司发布的FLV格式说明文档（可以到我的磁盘空间的“资料”目录去下载 <http://e.ys168.com/?tinyfun> ）。

    第1个字节的第5-6位的数值表示采样率，0 = 5.5 kHz，1 = 11 kHz，2 = 22 kHz，3 = 44 kHz。

    第1个字节的第7位表示采样精度，0 = 8bits，1 = 16bits。

    第1个字节的第8位表示音频类型，0 = mono，1 = stereo。

**六、Video Tag Data结构**

|  |  |  |
| --- | --- | --- |
| video信息 | 1byte | 前四位bits表示类型：   * 1 -- keyframe * 2 -- inner frame * 3 -- disposable inner frame (H.263 only)   后四位bits表示编码器id：   * 2 -- Seronson H.263 * 3 -- Screen video * 4 -- On2 VP6 * 5 -- On2 VP6 without channel * 6 -- Screen video version 2 |
| video数据区 | 不定 |  |

    视频Tag也用开始的第1个字节包含视频数据的参数信息，从第2个字节开始为视频流数据。

    第1个字节的前4位的数值表示帧类型。

    第1个字节的后4位的数值表示视频编码ID，1 = JPEG（现已不用），2 = Sorenson H.263，3 = Screen video，4 = On2 VP6，5 = On2 VP6 with alpha channel，6 = Screen video version 2。

**七、Script Tag Data结构**

    该类型Tag又通常被称为Metadata Tag，会放一些关于FLV视频和音频的参数信息，如duration、width、height等。通常该类型Tag会跟在File Header后面作为第一个Tag出现，而且只有一个。

    一般来说，该Tag Data结构包含两个AMF包。AMF（Action Message Format）是Adobe设计的一种通用数据封装格式，在Adobe的很多产品中应用，简单来说，AMF将不同类型的数据用统一的格式来描述。第一个AMF包封装字符串类型数据，用来装入一个“onMetaData”标志，这个标志与Adobe的一些API调用有，在此不细述。第二个AMF包封装一个数组类型，这个数组中包含了音视频信息项的名称和值。具体说明如下，大家可以参照图片上的数据进行理解。

第一个AMF包：

第1个字节表示AMF包类型，一般总是0x02，表示字符串，其他值表示意义请查阅文档。

第2-3个字节为UI16类型值，表示字符串的长度，一般总是0x000A（“onMetaData”长度）。后面字节为字符串数据，一般总为“onMetaData”。

第二个AMF包：

第1个字节表示AMF包类型，一般总是0x08，表示数组。

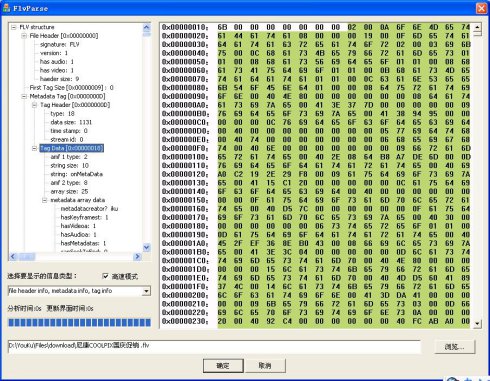
第2-5个字节为UI32类型值，表示数组元素的个数。后面即为各数组元素的封装，数组元素为元素名称和值组成的对。表示方法如下：

第1-2个字节表示元素名称的长度，假设为L。后面跟着为长度为L的字符串。

第L+3个字节表示元素值的类型。后面跟着为对应值，占用字节数取决于值的类型。

Metadata tag数据区组成:   
//-------------------------------------   
object由名字和数据组成:   
--------------------------------------------   
object名字类型开始标志 0x02   
object结束  0x02/000009   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
object名字:   
名字大小 2个字节/4个字节名字的长度 名字   
------------------   
object数据:   
数据类型 1个字节   
  0 = Number type   
  1 = Boolean type   
  2 = String type   
  3 = Object type   
  4 = MovieClip type   
  5 = Null type   
  6 = Undefined type   
  7 = Reference type   
  8 = ECMA array type   
  10 = Strict array type   
  11 = Date type   
  12 = Long string type   
数组长度   
  If Type = 8, 4个字节的数据大小   
     
数据值  If Type = 0, DOUBLE(8个字节的double数据)   
  If Type = 1, UI8   
  If Type = 2, 2字节长度的字符串   
  If Type = 3, object数据[n]   
  If Type = 4, SCRIPTDATASTRING 定义MovieClip的路径   
  If Type = 7, UI16   
  If Type = 8,数据变量[ECMAArrayLength] 变量的结束标志是000009   
  If Type = 10,数据变量[n]   
  If Type = 11,日期类型   
  If Type = 12,4字节长度的字符串   
数据结束  If Type = 3,object数据结束标志  Object 和array 的结束标志   
  If Type = 10,array数据结束标志   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
日期类型是由下面组成:   
------------------------   
时间值  DateTime类型由8个字节的毫秒从Jan 1, 1970 UTC开始   
时间偏移 本地的时区和标准时间的分钟的偏移

    OK，到此FLV文件结构讲完了，希望我已经讲清楚了：）另外我自己写了一个小工具，用它可以查看一个FLV文件的结构，并且能够分析各个字段对应字节的具体值，截图如下。目前还有一些小问题需要完善，后面我会加一个使用说明。想用的朋友可以到我磁盘空间的“工具”目录去下载。（<http://tinyfun.ys168.com/> ）

[](http://photo.blog.sina.com.cn/showpic.html#blogid=48f93b530100eyoe&url=http://s12.sinaimg.cn/orignal/48f93b53g74f5e29d9f3b&690)