H.264(六)序列参数集（SPS）

在H.264标准协议中规定了多种不同的NAL Unit类型，其中类型7表示该NAL Unit内保存的数据为Sequence Paramater Set。在H.264的各种语法元素中，SPS中的信息至关重要。如果其中的数据丢失或出现错误，那么解码过程很可能会失败。SPS及后续将要讲述的图像参数集PPS在某些平台的视频处理框架（比如iOS的VideoToolBox等）还通常作为解码器实例的初始化信息使用。

> ## 1. 什么是SPS
>
> 如上文所述，SPS即Sequence Paramater Set，又称作序列参数集。SPS中保存了一组编码视频序列(Coded video sequence)的全局参数。所谓的编码视频序列即原始视频的一帧一帧的像素数据经过编码之后的结构组成的序列。**而每一帧的编码后数据所依赖的参数保存于图像参数集中。**
>
> 在简单的H.264编解码的Demo中，一个SPS和PPS的NAL Unit通常位于整个码流的起始位置。但在某些特殊情况下，在码流中间也可能出现这两种结构，主要原因可能为
>
> 1. 解码器需要在码流中间开始解码；
> 2. 编码器在编码的过程中改变了码流的参数（如图像分辨率等）；

2. SPS的结构

在我们这个实例中，SPS NAL Unit中的二进制内容为：

```
42 00 1e e8 58 58 98 80
```

为了让后续的解码过程可以使用SPS中包含的参数，必须对其中的数据进行解析。其中H.264标准协议中规定的SPS格式位于文档的7.3.2.1.1部分，如下图所示：

seq_parameter_set_data( ) {	C	Descriptor
profile_idc	0	u(8)
constraint_set0_flag	0	u(1)
constraint_set1_flag	0	u(1)
constraint_set2_flag	0	u(1)
constraint_set3_flag	0	u(1)
constraint_set4_flag	0	u(1)
constraint_set5_flag	0	u(1)
reserved_zero_2bits /* equal to 0 */	0	u(2)
level_idc	0	u(8)
seq_parameter_set_id	0	ue(v)
if(profile_idc == 100 || profile_idc == 110 || profile_idc == 122 || profile_idc == 244 || profile_idc == 44 || profile_idc == 83 || profile_idc == 86 || profile_idc == 118 || profile_idc == 128) {		
chroma_format_idc	0	ue(v)
if(chroma_format_idc == 3)		
separate_colour_plane_flag	0	u(1)
bit_depth_luma_minus8	0	ue(v)
bit_depth_chroma_minus8	0	ue(v)
qpprime_y_zero_transform_bypass_flag	0	u(1)
seq_scaling_matrix_present_flag	0	u(1)
if(seq_scaling_matrix_present_flag)		
for(i = 0; i < ((chroma_format_idc != 3) ? 8 : 12); i++) {		
seq_scaling_list_present_flag[i **]**	0	u(1)
if(seq_scaling_list_present_flag[i])		
if(i < 6)		
scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	0	
else		
scaling_list(ScalingList8x8[i − 6], 64, UseDefaultScalingMatrix8x8Flag[i − 6])	0	
}		
}		

log2_max_frame_num_minus4	0	ue(v)
pic_order_cnt_type	0	ue(v)
if(pic_order_cnt_type == 0)		
log2_max_pic_order_cnt_lsb_minus4	0	ue(v)
else if(pic_order_cnt_type == 1) {		
delta_pic_order_always_zero_flag	0	u(1)
offset_for_non_ref_pic	0	se(v)
offset_for_top_to_bottom_field	0	se(v)
num_ref_frames_in_pic_order_cnt_cycle	0	ue(v)
for(i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++)		
offset_for_ref_frame[i]	0	se(v)
}		
max_num_ref_frames	0	ue(v)
gaps_in_frame_num_value_allowed_flag	0	u(1)

其中的每一个语法元素及其含义如下：

> **(1). profile_idc：**
>
> 标识当前H.264码流的profile。我们知道，H.264中定义了三种常用的档次profile：
>
> 基准档次：baseline profile;
> 主要档次：main profile;
> 扩展档次：extended profile;
> 在H.264的SPS中，第一个字节表示profile_idc，根据profile_idc的值可以确定码流符合哪一种档次。判断规律为：
>
> profile_idc = 66 → baseline profile;
> profile_idc = 77 → main profile;
> profile_idc = 88 → extended profile;
> 在新版的标准中，还包括了High、High 10、High 4:2:2、High 4:4:4、High 10 Intra、High 4:2:2 Intra、High 4:4:4 Intra、CAVLC 4:4:4 Intra等，每一种都由不同的profile_idc表示。
>
> 另外，constraint_set0_flag ~ constraint_set5_flag是在编码的档次方面对码流增加的其他一些额外限制性条件。
>
> 在我们实验码流中，profile_idc = 0x42 = 66，因此码流的档次为baseline profile。
>
> 参考：https://blog.csdn.net/xiaojun111111/article/details/52090185

> **(2). level_idc**
>
> 标识当前码流的Level。编码的Level定义了某种条件下的最大视频分辨率、最大视频帧率等参数，码流所遵从的level由level_idc指定。
>
> 当前码流中，level_idc = 0x1e = 30，因此码流的级别为3。

(3). seq_parameter_set_id

表示当前的序列参数集的id。通过该id值，图像参数集pps可以引用其代表的sps中的参数。

(4). log2_max_frame_num_minus4

这个句法元素主要是为读取另一个句法元素 frame_num 服务的，frame_num 是最重要的句法元素之一，它标识所属图像的解码顺序。可以在句法表看到， fram-num 的解码函数是 ue（v），函数中的 v 在这里指定：

v = log2_max_frame_num_minus4 + 4
从另一个角度看，这个句法元素同时也指明了 frame_num 的所能达到的最大值：
MaxFrameNum = 2(log2_max_frame_num_minus4 + 4)

变量 MaxFrameNum 表示 frame_num 的最大值，在后文中可以看到，在解码过程中它也是一个非常重要的变量。

值得注意的是 frame_num 是循环计数的，即当它到达 MaxFrameNum 后又从 0 重新开始新一轮的计数。 解码器必须要有机制检测这种循环， 不然会引起类似千年虫的问题，在图像的顺序上造成混乱。

(5). pic_order_cnt_type

指明了 poc (picture order count) 的编码方法， poc 标识图像的播放顺序。由于H.264 使用了 B 帧预测，使得图像的解码顺序并不一定等于播放顺序，但它们之间存在一定的映射关系。 poc 可以由 frame-num 通过映射关系计算得来，也可以索性由编码器显式地传送。 H.264 中一共定义了三种 poc 的编码方法，这个句法元素就是用来通知解码器该用哪种方法来计算 poc。 而以下的几个句法元素是分别在各种方法中用到的数据。

在如下的视频序列中本句法元素不应该等于 2:

一个非参考帧的接入单元后面紧跟着一个非参考图像(指参考帧或参考场)的接入单元
两个分别包含互补非参考场对的接入单元后面紧跟着一个非参考图像的接入单元.
一个非参考场的接入单元后面紧跟着另外一个非参考场,并且这两个场不能构成一个互补场对

(6). log2_max_pic_order_cnt_lsb_minus4

指明了变量 MaxPicOrderCntLsb 的值:
MaxPicOrderCntLsb = 2(log2_max_pic_order_cnt_lsb_minus4 + 4)
该变量在 pic_order_cnt_type = 0 时使用。

(7). max_num_ref_frames

指定参考帧队列可能达到的最大长度，解码器依照这个句法元素的值开辟存储区，这个存储区用于存放已解码的参考帧， H.264 规定最多可用 16 个参考帧，本句法元素的值最大为 16。值得注意的是这个长度以帧为单位，如果在场模式下，应该相应地扩展一倍。

(8). gaps_in_frame_num_value_allowed_flag

这个句法元素等于 1 时，表示允许句法元素 frame_num 可以不连续。当传输信道堵塞严重时，编码器来不及将编码后的图像全部发出，这时允许丢弃若干帧图像。在正常情况下每一帧图像都有依次连续的 **frame_num 值**，解码器检查到如果 frame_num 不连续，便能确定有图像被编码器丢弃。这时，解码器必须启动错误掩藏的机制来近似地恢复这些图像，因为这些图像有可能被后续图像用作参考帧。当这个句法元素等于 0 时，表不允许 frame_num 不连续，即编码器在任何情况下都不能丢弃图像。这时， H.264 允许解码器可以不去检查 frame_num 的连续性以减少计算量。这种情况下如果依然发生 frame_num 不连续，表示在传输中发生丢包，解码器会通过其他机制检测到丢包的发生，然后启动错误掩藏的恢复图像。

(9). pic_width_in_mbs_minus1

本句法元素加 1 后指明图像宽度，以宏块为单位：

PicWidthInMbs = pic_width_in_mbs_minus1 + 1

通过这个句法元素解码器可以计算得到亮度分量以像素为单位的图像宽度：

PicWidthInSamplesL = PicWidthInMbs * 16

从而也可以得到色度分量以像素为单位的图像宽度：

PicWidthInSamplesC = PicWidthInMbs * 8

以上变量 PicWidthInSamplesL、 PicWidthInSamplesC 分别表示图像的亮度、色度分量以像素为单位的宽。

H.264 将图像的大小在序列参数集中定义，意味着可以在通信过程中随着序列参数集动态地改变图像的大小，在后文中可以看到，甚至可以将传送的图像剪裁后输出。

frame_width = 16 × (pic_width_in_mbs_minus1 + 1);

(10). pic_height_in_map_units_minus1

本句法元素加 1 后指明图像高度：

PicHeightInMapUnits = pic_height_in_map_units_minus1 + 1

PicSizeInMapUnits = PicWidthInMbs * PicHeightInMapUnits

图像的高度的计算要比宽度的计算复杂，因为一个图像可以是帧也可以是场，从这个句法元素可以在帧模式和场模式下分别计算出出亮度、色度的高。值得注意的是，这里以 map_unit 为单位， map_unit的含义由后文叙述。

PicHeightInMapUnits = pic_height_in_map_units_minus1 + 1;

(11). frame_mbs_only_flag

标识位，说明宏块的编码方式。当该标识位为0时，宏块可能为帧编码或场编码；该标识位为1时，所有宏块都采用帧编码。根据该标识位取值不同，PicHeightInMapUnits的含义也不同，为0时表示一场数据按宏块计算的高度，为1时表示一帧数据按宏块计算的高度。

按照宏块计算的图像实际高度FrameHeightInMbs的计算方法为：

FrameHeightInMbs = (2 − frame_mbs_only_flag) * PicHeightInMapUnits

(12). mb_adaptive_frame_field_flag

指明本序列是否属于帧场自适应模式。 mb_adaptive_frame_field_flag等于 1 时表明在本序列中的图像如果不是场模式就是帧场自适应模式，等于 0 时表示本序列中的图如果不是场模式就是帧模式。。表 列举了一个序列中可能出现的编码模式：
a. 全部是帧，对应于 frame_mbs_only_flag =1 的情况。
b. 帧和场共存。 frame_mbs_only_flag =0, mb_adaptive_frame_field_flag =0
c. 帧场自适应和场共存。 frame_mbs_only_flag =0, mb_adaptive_frame_field_flag =1
值得注意的是，帧和帧场自适应不能共存在一个序列中。

(13). direct_8x8_inference_flag

标识位，用于B_Skip、B_Direct模式运动矢量的推导计算。

(14). frame_cropping_flag

标识位，说明是否需要对输出的图像帧进行裁剪。

(15). vui_parameters_present_flag

指明 vui 子结构是否出现在码流中， vui 的码流结构在附录中指明，用以表征视频格式等额外信息。