

# Tree syntax for field sequence SEI

Geneva

April 2012

JCTVC I-393

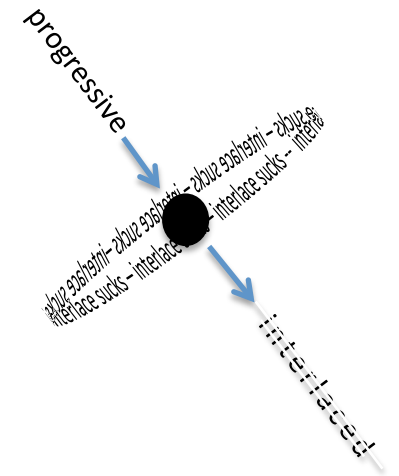
# First, to clear the air..

No one in their right mind would use interlace for new captures, transport, display formats.

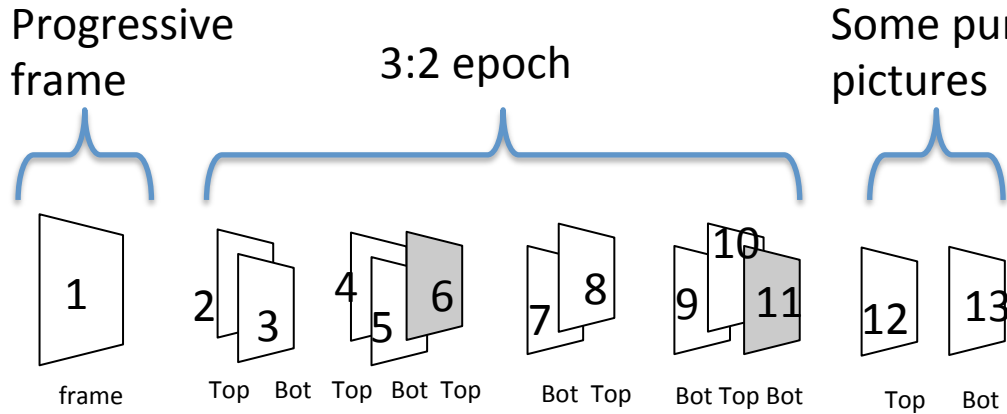
In fact, interlace “sucks” so bad, light cannot escape

..or at least every other line...

..but, it is needed for legacy sources



# Core concept example: PAFF -> field seq.

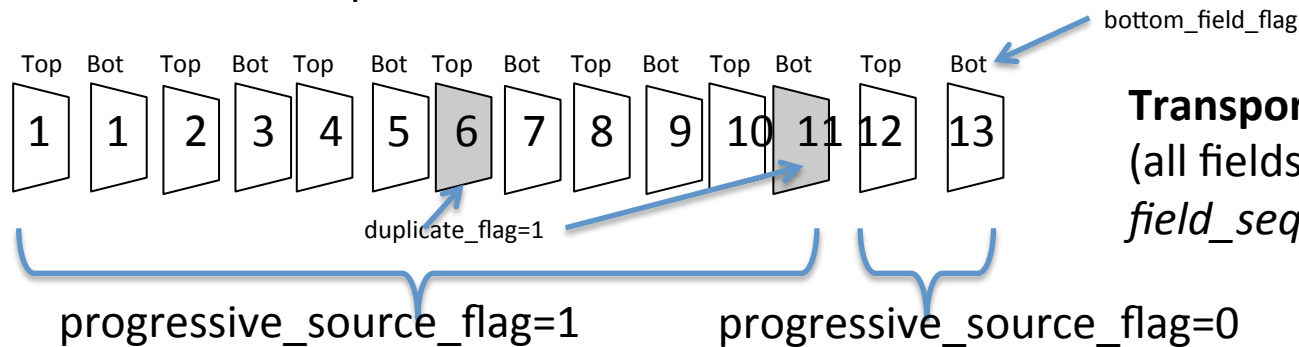


## Source

(example: mixed content.. An extreme case)

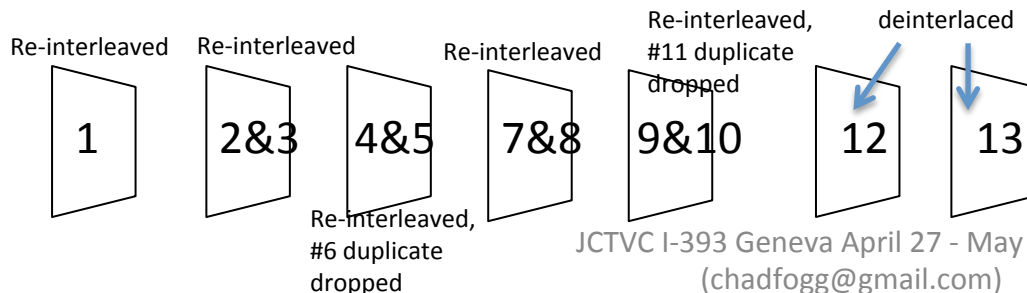
*Picture numbers are to show mapping from source to field input to encoder.. It is not POC, frame\_num, etc.*

## Pre-mux prior to HEVC encode



## Transport & encoding

(all fields in a field sequence)  
*field\_sequence\_flag=1*



## Progressive display

# Key, independent flags in field SEI

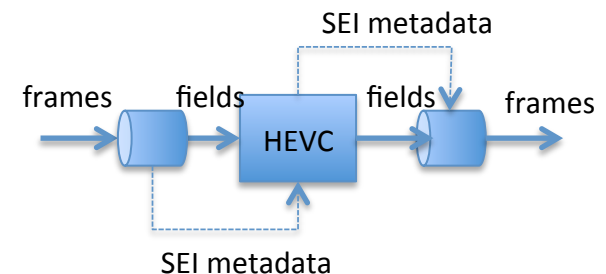
**field\_sequence\_flag:** how fields/frames are packed for transport through HEVC codec

**progressive\_source\_flag:** indicates whether content is progressive or interlaced

**duplicate\_flag:** indicates coded picture is duplicated / can be dropped

# Benefits of the current SEI

- Pass-through concept that allows mixed progressive & interlace content to tunnel through HEVC without changes to Video Coding Layer (VCL), including HRD (no phantom/non-existent pictures in HRD!)
- Signals content type (interlaced or progressive via *progressive\_source\_flag*) separately from picture structure transport (field or frame pictures via *field\_sequence\_flag*)
- Pure interlaced can be sent cleanly as field sequences.
- For fixed equipment, such as those linked by SDI (e.g. HD-D5), fields can be encapsulated (interleaved) in frames, or progressive frames can be encapsulated (extracted) in fields, then re-assembled back into source format according to the SEI flags.
- Duplicate\_flag identifies coded pictures that were synthesized by, for example, stream splicing, editing, or 3:2 pulldown



# The current SEI: flag vector style section D.1.19 of HEVC CD (I-0030)

field_indication( payloadSize ) {	Descriptor
sequence_type_flag	u(1)
progressive_source_flag	u(1)
bottom_field_flag	u(1)
top_field_first_flag	u(1)
duplicate_flag	u(1)
reserved_zero_3bits /* equal to 0 */	u(3)
}	

# Basic four cases (D.2.19)

**Table D-1**

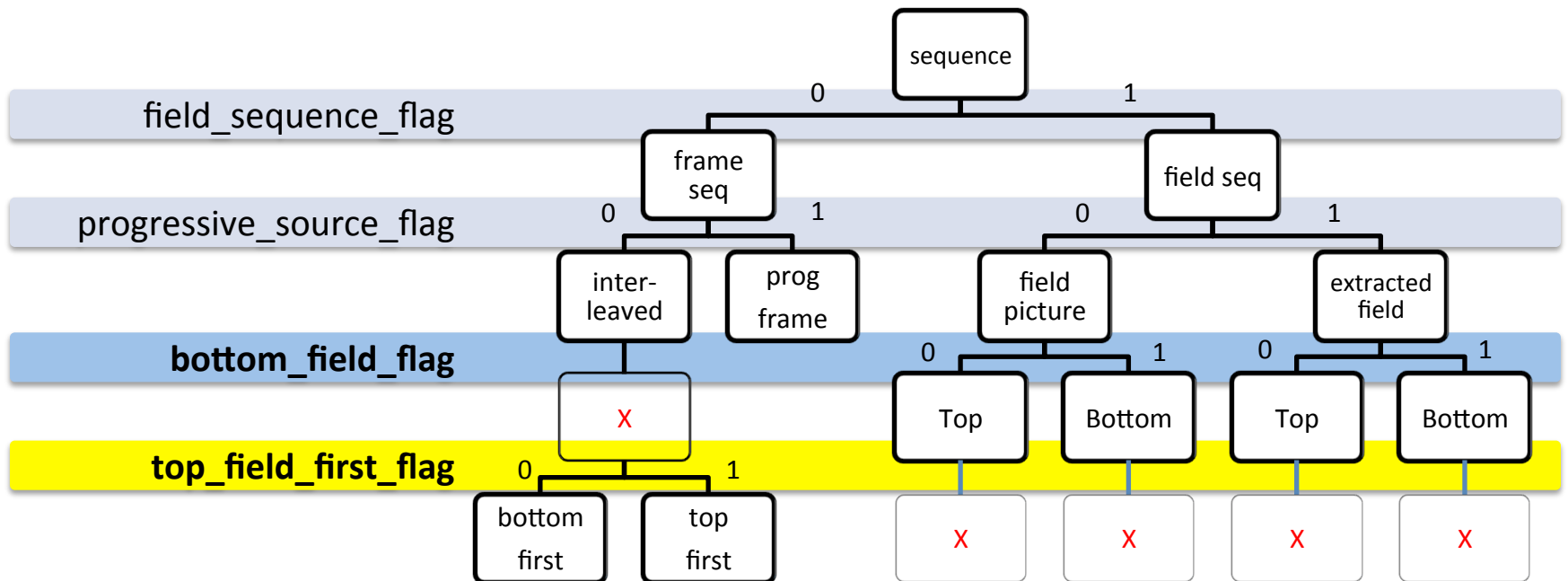
<b>Interpretation</b>	<b>sequence_type_flag</b>	<b>progressive_source_flag</b>
Picture is progressive frame	0	1
Picture is interleaved fields coded in frame picture	0	0
Picture is field	1	0
Picture is field extracted from progressive frame	1	1

# Problem: some combinations are not meaningful (indicated by “X”)

Use case	field_sequence	progressive_source	bottom_field	top_field_first
Interleaved fields, bottom field output first	0	0	X	0
Interleaved fields, top field output first	0	0	X	1
(pure progressive) frame	0	1	X	X
Top field picture (pure interlaced)	1	0	0	X
Bottom field picture (pure interlaced)	1	0	1	X
Top field extracted from a progressive frame	1	1	0	X
Bottom field extracted from a progressive frame	1	1	1	X



# Meaningful use cases



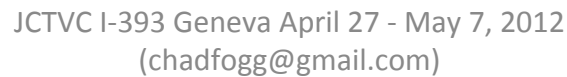
# Observations

`top_field_first` and `bottom_field` are mutually exclusive:

- *bottom\_field* (field parity) always needed in field sequences (`field_sequence_flag==1`) to signal field parity; `bottom_field` is not meaningful in frame sequences (`field_sequence_flag==0`).
- *top\_field\_first* only needs to be signalled in frame sequences containing interlaced content (`field_sequence_flag==0 && progressive_source_flag==0`), the so-called “interleaved fields,” to know which interleaved field (top or bottom) should be output first
- Neither *top\_field\_first* nor *bottom\_field* needed in pure progressive frame sequences (`field_sequence_flag==0 && progressive_source_flag==1`)

**Possible solution:** eliminate combinations that are not meaningful by making them impossible to signal via a conditional tree syntax, thus helping HEVC specification readers from harming themselves.

(no functional changes to current SEI)



# Proposed syntax

field_indication( payloadSize ) {	Descriptor
field_sequence_flag	u(1)
progressive_source_flag	u(1)
duplicate_flag	u(1)
if( field_sequence_flag==1 )	
bottom_field_flag	u(1)
else if( progressive_source==0)	
top_field_first_flag	u(1)
else	
reserve_zero_1bit	u(1)
reserved_zero_4bits /* equal to 0 */	u(4)
}	