**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11 MPEG/M54362**

**June 2020, Online**

|  |  |
| --- | --- |
| **Source** | **Philips** |
| **Status** | **Input document** |
| **Title** | **[MIV] Comments on the 5th working draft** |
| **Author** | Bart Kroon, Bart Sonneveldt |

# Introduction

This contribution provides:

1. a general review of MIV WD5 [N19212], and
2. one or more solutions for the lack of access units in the special atlas ([issue #40](http://mpegx.int-evry.fr/software/MPEG/MIV/Specs/23090-12/issues/40)).

# General review of MIV WD5

## Remark on viewing space definition

In the General section of F.3.1. (Viewing space SEI payload semantics) it is now suggested that the viewing space exceedance is handled using fading:

“*It is based on the possibility given to the end device to compute a fading index between 0 (no fading) and 1 (full fading) of inclusiveness inside this viewing space. The end device application can use this index to implement a fade out when the viewport leaves the viewing space*.”

While there are a variety of handling options specified in F.3.3. We propose to use a more neutral formulation like:

*“It is based on the possibility to give the end device the opportunity to handle viewing space exceedance. A viewing space inclusiveness factor can be computed where 0 indicates fully inside and 1 indicates fully outside. The end device application can use this factor to take a viewers’ transient, from inside the viewing space to outside, into account.”*

# Lack of access units in the special atlas

## Description of the problem

The idea of NAL\_FOC was to give a frame number to AAPS updates. The structure would be like this assuming an update of camera extrinsics at frame 40:

AD(vuh\_atlas\_id=0x3F): AAPS[VPL\_INITLIST] FOC[40] AAPS[VPL\_UPD\_EXT]

According to Clause 7.4.5.3.4 of V3C DIS d85 [MPEG/N19329]:

An access unit consists of one coded atlas with nal\_layer\_id equal to 0, zero or more ACL NAL units with nal\_layer\_id greater than 0 and zero or more non-ACL NAL units. The association of ACL NAL units to coded atlases is described in subclause 7.4.5.3.5.

This is a problem because there is no ACL in the above example and hence there is no access unit. Even this is probably invalid:

AD(vuh\_atlas\_id=0x3F): AAPS[VPL\_INITLIST]

Similarly, there was the idea that we could have SEI messages w/o ATL in regular atlases:

AD(vuh\_atlas\_id=0): ASPS AFPS ATL(foc=0) FOC(foc=10) PREFIX\_NSEI ATL(foc=32)

The FOC starts a new ACL, but the ACL needs to have an ATL, so the FOC and ATL are part of the access unit and they cannot have a different FOC.

## Proposed solution 1

Make NAL\_FOC an ACL NAL unit that is confined to the special atlas.

Use **ath\_atlas\_frm\_order\_cnt\_lsb** to skip frames in the regular atlases.

### Clause 7.4.5.2 in part 5

|  |  |  |  |
| --- | --- | --- | --- |
| 48 | NAL\_FOC | Frame order count  frame\_order\_count\_rbsp( ) | ~~non-~~ACL |

When vuh\_atlas\_id is in the range 0 .. 0x3E, then there shall be no NAL unit with nal\_unit\_type equal to NAL\_FOC.

When vuh\_atlas\_id is equal to 0x3F, then each access unit includes a NAL unit with nal\_unit\_type equal to NAL\_FOC.

### Clause 7.4.5.3.4 in part 5

Let firstBlAFrmNalUnit be the first ACL NAL unit of a coded atlas frame with nal\_layer\_id equal to 0. The first of any of the following NAL units preceding firstBlAFrmNalUnit and succeeding the last ACL NAL unit preceding firstBlAFrmNalUnit, if any, specifies the start of a new access unit:

* NAL unit with nal\_unit type NAL\_FOC with nal\_layer\_id equal to 0 (when present)

(...)

The order of the coded atlas frames and non-ACL NAL units within an access unit shall obey the following constraints:

* (...)
* When a frame order count NAL unit is present, it shall be the ~~first NAL unit in the coded atlas access-unit. There shall be at most one frame order count~~ only ACL NAL unit in the ~~any~~ coded atlas access unit.

## Proposed solution 2

Like solution 1 but remove NAL\_FOC from Part 5 and specify in Part 12 using a better name. This gives us some more time to refine the special atlas.

### Clause 7.3.6.10 Frame order count RBSP syntax

Remove FOC RBSP from part 5 and add to part 12 as APL RBSP:

|  |  |
| --- | --- |
| atlas\_projection\_layer\_rbsp( ) { | **Descriptor** |
| **apl\_frm\_order\_cnt\_lsb** | u(v) |
| **apl\_extension\_flag** | u(1) |
| if( apl\_extension\_flag ) { |  |
| **apl\_extension\_8bits** | u(6) |
| } |  |
| if( apl\_extension\_8bits ) |  |
| while( more\_rbsp\_data( ) ) |  |
| apl\_extension\_data\_flag | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

### Clause 7.4.6.10 Frame order count RBSP semantics

Remove from part 5. Add to part 12.

### Clause 7.4.5.2

Part 5:

|  |  |  |  |
| --- | --- | --- | --- |
| 48 | NAL\_RSV\_ACL\_48 | Specified in ISO/IEC 23090-12 MIV | ~~non-~~ACL |

Part 12:

|  |  |  |  |
| --- | --- | --- | --- |
| 48 | NAL\_APL | Atlas Projection Layer  atlas\_projection\_layer\_rbsp( ) | ACL |

## Proposed solution 3

Like solution 2, but move view parameter (updates) out of AAPS and into APL:

|  |  |
| --- | --- |
| atlas\_projection\_layer\_rbsp( ) { | **Descriptor** |
| **apl\_atlas\_adaptation\_parameter\_set\_id** | ue(v) |
| **apl\_frm\_order\_cnt\_lsb** | u(v) |
| **apl\_miv\_view\_params\_list\_update\_mode** | u(2) |
| if( apl\_miv\_view\_params\_list\_update\_mode == VPL\_INITLIST) |  |
| miv\_view\_params\_list( ) |  |
| else if( apl\_miv\_view\_params\_list\_update\_mode == VPL\_UPD\_EXT) |  |
| miv\_view\_params\_update\_extrinsics( ) |  |
| else if( apl\_miv\_view\_params\_list\_update\_mode == VPL\_UPD\_INT) |  |
| miv\_view\_params\_update\_intrinsics( ) |  |
| else if( apl\_miv\_view\_params\_list\_update\_mode == VPL\_EXT\_INT) { |  |
| miv\_view\_params\_update\_extrinsics( ) |  |
| miv\_view\_params\_update\_intrinsics( ) |  |
| } |  |
| **apl\_extension\_flag** | u(1) |
| if( apl\_extension\_flag ) { |  |
| **apl\_extension\_8bits** | u(6) |
| } |  |
| if( apl\_extension\_8bits ) |  |
| while( more\_rbsp\_data( ) ) |  |
| apl\_extension\_data\_flag | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

|  |  |
| --- | --- |
| aaps\_miv\_extension ( ) { | **Descriptor** |
| **aame\_omaf\_v1\_compatible\_flag** | u(1) |
| **~~aame\_miv\_view\_params\_list\_update\_mode~~** | ~~u(2)~~ |
| ~~if( aame\_miv\_view\_params\_list\_update\_mode == VPL\_INITLIST~~~~)~~ |  |
| ~~miv\_view\_params\_list( )~~ |  |
| ~~else if( aame\_miv\_view\_params\_list\_update\_mode == VPL\_UPD\_EXT~~~~)~~ |  |
| ~~miv\_view\_params\_update\_extrinsics( )~~ |  |
| ~~else if( aame\_miv\_view\_params\_list\_update\_mode == VPL\_UPD\_INT~~~~)~~ |  |
| ~~miv\_view\_params\_update\_intrinsics( )~~ |  |
| ~~else if( aame\_miv\_view\_params\_list\_update\_mode == VPL\_EXT\_INT~~~~) {~~ |  |
| ~~miv\_view\_params\_update\_extrinsics( )~~ |  |
| ~~miv\_view\_params\_update\_intrinsics( )~~ |  |
| ~~}~~ |  |
| } |  |

## Proposed solution 4

Like solution 3, but instead of the 0x3F trick, define a new V3C unit type “MIV\_PD” having an projection\_sub\_bitstream( ). This can improve the clarify of the standard but may cause problems at the Systems level.

# Recommendations

Solve issue #40.

Modify the text Viewing space SEI payload semantics such that the handling method is not specified.