|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  9th Meeting: Geneva, CH, 27 April – 7 May 2012 | Document: JCTVC-I0046 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **APS partial update through conditional replacement** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Stephan Wenger, Jill Boyce 433 Hackensack Ave., Hackensack, N.J. 07601, USA | Tel: Email: | +1-415-713-5473 (Stephan) [stewe@stewe.org](mailto:stewe@stewe.org) [jill@vidyo.com](mailto:jill@vidyo.com) |
| *Source:* | Vidyo, Inc. | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This document proposes (again) an update mechanism for parameter sets, specifically the APS, through conditional replacement.

# Introduction

For two meeting cycles, the need for a partial update mechanism for (at least) the Adaptation Parameter Set, or a mechanism that can serve a similar purpose, has been widely acknowledged. However, this technically rather unexciting area has become subject to strong disagreements between a number of proponents, and no progress was made towards the selection of one (or more) partial update mechanisms.

This document re-proposes the quite possibly simplest and least impactful partial update mechanism so far discussed and proposed. It is not the most elegant mechanism, and it may well be the least efficient one, at least by judging what was on the table in San Jose. However, including it is certainly better than doing nothing with respect to partial update, and in our opinion also with respect to codec performance and, perhaps even more importantly, future-proofness and flexibility.

To recapitulate, partial update refers to mechanisms that allow parts of a parameter set to change, without requiring complete (re-)transmission of those parts that haven’t changed. Two general approached have been proposed before, one being the subject of this document, and the other using various forms of “referencing” mechanisms (be they from parameter sets of the same type, or from a master parameter set and wherein the “slave” parameter set may be split up into several mini parameter sets).

It should be noted that parameter set update through referencing—be it according to the ZTE/Vidyo/BBC proposal JCTVC-H0069, the Qualcomm proposal JCTVC-H0505, or anything roughly similar—and update through conditional replacement could, at least in theory, co-exist. With the present parameter set complexity, this is in our opinion overkill. However, let’s not forget that parameter sets carry, among other things, profile information. Adding new profiles is the primary versioning mechanism of JCT-VC. Therefore, at least if we were considering extending “partial update” to high level parameter sets such as the SPS, we would need to get things right NOW—we can’t retrofit changes later without breaking backward compatibility—a lesson painfully learned in the context of H.264 and SVC. That would speak in favour of adding as much flexibility as we can now, even if it were inducing a little pain to those who currently do not believe an update mechanism is required for their applications.

Parameter set update through conditional replacement works by

* Identifying each block of parameters in a set that semantically belong together,
* Making the presence of each block conditional to a value of a flag,
* And change the parameter set activation process such that when a parameter set is activated, a block is marked at not present, and the block was present at the previous activation of the parameter set (previous block), the content of the previous block stays activated.

Later sections of this document attempt to identify relevant blocks of parameters in the APS only. It would be sensible to allow for conditional replacement also in other blocks of other parameter sets, such as the VUI in the SPS. However, nothing like this is proposed herein, so not to overload the document. It would be simple to create such mechanisms in real-time during the meeting, if JCT-VC were so inclined.

# Proposal

## Overview

It is proposed to add conditional replacement to the following blocks of parameters (blocks henceforth) in the APS:

* Scaling list,
* Sample Adaptive Offset (SAO-) parameters,
* Adaptive Loop Filter (ALF-) parameters,
* and everything else specified (currently only the deblocking filter parameters, but other information that is small—a few octets at most—could be added into this block).

The extension mechanism remains unchanged.

We do not propose to put small sets of parameters that belong together (such as, at present, the deblocking filter info) in its own “block”. Including these few octets into each APS does not do much harm, and conditional replacement update would not bring much gain. However, we are also not opposed in putting those deblocking parameters in their own block. We leave them out mostly to illustrate that not all parameters necessarily must reside in a block associated with a coding tool; it is perfectly acceptable to enable conditional replacement only for those blocks that are (or can be) of such a significant size that re-sending them with every APS (essentially for every picture) is detrimental for the coding efficiency.

We do not specifically care about the sequencing of these parameter blocks in the syntax, except that (obviously) the extension block should be the last block in the syntax.

We note that that the existing flags gating certain blocks in the APS appear to have side effects that go beyond the signalling of presence of the parameters block in the APS. For example, the aps\_adaptive\_loop\_filter\_flag signals not only the presence of loop filter parameters, but also enables the loop filter for the picture in question. This binding essentially disallows re-using previously transmitted loop filter parameters for a future picture. Whenever we spotted such a (what we perceive) bug, we corrected it by adding a “presence” flag.

## Syntax changes

Table 1 shows the APS syntax diagram according to this proposal.

Highlighted in yellow are changes in the syntax (newly defined flags). Highlighted in cyan are those presence flags for which the semantics are changed without affecting the syntax.

#### 7.3.2.5 Adaptation parameter set RBSP syntax

|  |  |
| --- | --- |
| aps\_rbsp( ) { | Descriptor |
| **aps\_id** | ue(v) |
| **aps\_scaling\_list\_data\_present\_flag** | u(1) |
| if( aps\_scaling\_list\_data\_present\_flag ) |  |
| scaling\_list\_param( ) |  |
| **aps\_deblocking\_filter\_flag** | u(1) |
| if(aps\_deblocking\_filter\_flag) { |  |
| **disable\_deblocking\_filter\_flag** | u(1) |
| if( !disable\_deblocking\_filter\_flag ) { |  |
| **beta\_offset\_div2** | se(v) |
| **tc\_offset\_div2** | se(v) |
| } |  |
| } |  |
| **aps\_sao\_param\_presence\_flag** | u(1) |
| if( aps\_sao\_param\_presence\_flag ) { |  |
| **aps\_sao\_interleaving\_flag** | u(1) |
| if(!aps\_sao\_interleaving\_flag ) { |  |
| **aps\_sample\_adaptive\_offset\_flag** | u(1) |
| if( aps\_sample\_adaptive\_offset\_flag ) |  |
| aps\_sao\_param( ) |  |
| } |  |
| } |  |
| **aps\_adaptive\_loop\_filter\_flag** | u(1) |
| if( aps\_adaptive\_loop\_filter\_flag ) |  |
| **aps\_alf\_param\_presence\_flag** | u(1) |
| if( aps\_alf\_param\_presence\_flag ) |  |
| alf\_param( ) |  |
| **aps\_extension\_flag** | u(1) |
| if( aps\_extension\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **aps\_extension\_data\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

Table 1: APS syntax diagram

## Semantics changes

The changes we deem necessary in the APS semantics section (7.4.2.5) are shown below. Specifically, in the semantics description of each of the flags that gate a block (highlighted in cyan or yellow the syntax diagram of table 1) a qualifier is added to the extent that if a certain block is not present, it shall be predicted from the same block of the previous aps, if any.

#### 7.4.2.5 Adaptation parameter set RBSP semantics

**aps\_id** identifies the adaptation parameter set that is referred to in the slice header. The value of aps\_id shall be in the range of 0 to TBD, inclusive.

**aps\_scaling\_list\_data\_present\_flag** equal to 1 specifies that the scaling list parameters exist in this APS, equal to 0 specifies that scaling list parameters do not exist in this APS. If the scaling list parameters do not exist in this APS, the scaling list parameters of the previous APS, if any, remain activated.

**aps\_deblocking\_filter\_flag** equal to 1 specifies that deblocking parameters are present in the APS. aps\_deblocking\_filter\_flag equal to 0 specifies that deblocking parameters do not exist in this APS.

**aps\_sao\_param\_presence\_flag** equal to 1 specifies that sample adaptive offset parameters are present in the APS; equal to 0 specifies that sample adaptive offset parameters are not present in this APS, and the sample adaptive offset parameters of the previously activated APS, if any, remain activated.

**aps\_sao\_interleaving\_flag** equal to 1 specifies that the SAO parameters are interleaved in slice data for slices referring to the current APS; equal to 0 specifies that the SAO parameters are in APS for slices referring to the current APS. When there is no active APS, aps\_sao\_interleaving\_flag is inferred to be 0.

**aps\_sample\_adaptive\_offset\_flag** equal to 1 specifies that the SAO is on for slices referring to the current APS; equal to 0 specifies that the SAO is off for slices referring to the current APS. When there is no active APS, the aps\_sample\_adaptive\_offset\_flag value is inferred to be 0.

**aps\_adaptive\_loop\_filter\_flag** equal to 1 specifies that the ALF is on for slices referring to the current APS; equal to 0 specifies that the ALF is off for slices referring to the current APS. When there is no active APS, the aps\_adaptive\_loop\_filter\_flag value is inferred to be 0.

**aps\_alf\_param\_presence\_flag** equal to 1 specifies that adaptive loop filter parameters are present in the APS; equal to 0 specifies that adaptive loop filter parameters are not present in this APS, and the adaptive loop filter parameters of the previously activated APS, if any, remain activated.

**aps\_extension\_flag** equal to 0 specifies that no aps\_extension\_data\_flag syntax elements are present in the picture parameter set RBSP syntax structure. aps\_extension\_flag shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard. The value of 1 for aps\_extension\_flag is reserved for future use by ITU‑T | ISO/IEC. Decoders shall ignore all data that follow the value 1 for aps\_extension\_flag in a picture parameter set NAL unit.

**aps\_extension\_data\_flag** may have any value. Its value does not affect decoder conformance to profiles specified in this Recommendation | International Standard.

# Patent rights declaration(s)

**Vidyo may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**