



Information technology — Coding of audio-visual objects — Part 10: Advanced Video Coding

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Codage des objets audiovisuels —

Partie 10: Codage visuel avancé

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 14496-10:2012 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

In G.7.4.3.4, replace the following:

"When `quality_id` is equal to 0, the NAL units with `DQId` equal to `ref_layer_dq_id` shall have `discardable_flag` equal to 0.

The variable `MaxRefLayerDQId` is set equal to the maximum value of `ref_layer_dq_id` for the slices of the current layer representation."

with

"When `quality_id` is equal to 0, the NAL units with `DQId` equal to `ref_layer_dq_id` shall have `discardable_flag` equal to 0.

When `ref_layer_dq_id` is greater than or equal to 0, it is a requirement of bitstream conformance that the layer representation with `DQId` equal to `ref_layer_dq_id` is present in the bitstream.

The variable MaxRefLayerDQId is set equal to the maximum value of ref_layer_dq_id for the slices of the current layer representation."

Replace H.8.5.1 with the following text:

H.8.5.1 Derivation process for required anchor view components

This process is recursively invoked to derive the set of required anchor view components for a specified view.

Input to this process is a variable viewId, representing a view with view_id equal to viewId, with its corresponding view order index denoted by vOldx.

Outputs of this process are a possibly updated VOIdxList, and additional invocations of the derivation process based on the inter-view dependency for anchor view components in the view with view_id equal to viewId as specified in the sequence parameter set MVC extension.

The following ordered steps are specified:

1. Add vOldx to VOIdxList if the same value is not already included in VOIdxList.
2. Depending on num_anchor_refs_I0[vOldx] and num_anchor_refs_I1[vOldx], the following applies.
 - If both num_anchor_refs_I0[vOldx] and num_anchor_refs_I1[vOldx] are equal to 0, terminate this process.
 - Otherwise (num_anchor_refs_I0[vOldx] or num_anchor_refs_I1[vOldx] is not equal to 0), the following ordered steps are specified:
 - a. When num_anchor_refs_I0[vOldx] is not equal to 0, invoke the process specified in subclause H.8.5.1 for each viewId equal to anchor_ref_I0[vOldx][i] for all i in the range of 0 to num_anchor_refs_I0[vOldx] – 1, inclusive, in ascending order of i.
 - b. When num_anchor_refs_I1[vOldx] is not equal to 0, invoke the process specified in subclause H.8.5.1 for each viewId equal to anchor_ref_I1[vOldx][i] for all i in the range of 0 to num_anchor_refs_I1[vOldx] – 1, inclusive, in ascending order of i.

Replace H.8.5.2 with the following text:

H.8.5.2 Derivation process for required non-anchor view components

This process is recursively invoked to derive the set of required non-anchor view components for a specified view.

Input to this process is a variable viewId, representing a view with view_id equal to viewId, with its corresponding view order index denoted by vOldx.

Outputs of this process are a possibly updated VOIdxList and additional invocations of the derivation process based on the inter-view dependency for non-anchor view components in the view with view_id equal to viewId as specified in the sequence parameter set MVC extension.

The following ordered steps are specified:

1. Add vOldx to VOIdxList if the same value is not already included in VOIdxList.
2. Depending on num_non_anchor_refs_I0[vOldx] and num_non_anchor_refs_I1[vOldx], the following applies.
 - If both num_non_anchor_refs_I0[vOldx] and num_non_anchor_refs_I1[vOldx] are equal to 0, terminate this process.
 - Otherwise (num_non_anchor_refs_I0[vOldx] or num_non_anchor_refs_I1[vOldx] is not equal to 0), the following ordered steps are specified:

- a. When `num_non_anchor_refs_l0[ vOldx ]` is not equal to 0, invoke the process specified in subclause H.8.5.2 for each `viewId` equal to `non_anchor_ref_l0[ vOldx ][ i ]` for all `i` in the range of 0 to `num_non_anchor_l0[ vOldx ] - 1`, inclusive, in ascending order of `i`.
- b. When `num_non_anchor_refs_l1[ vOldx ]` is not equal to 0, invoke the process specified in subclause H.8.5.2 for each `viewId` equal to `non_anchor_ref_l1[ vOldx ][ i ]` for all `i` in the range of 0 to `num_non_anchor_l1[ vOldx ] - 1`, inclusive, in ascending order of `i`.

Replace H.8.5.3 with the following text:

H.8.5.3 Sub-bitstream extraction process

It is requirement of bitstream conformance that any sub-bitstream that is the output of the process specified in this subclause with `pldTarget` equal to any value in the range of 0 to 63, inclusive, `tldTarget` equal to any value in the range of 0 to 7, inclusive, `viewIdTargetList` consisting of any one or more values of `viewIdTarget` identifying the views in the bitstream, shall be conforming to this Recommendation | International Standard.

NOTE 1 – A conforming bitstream contains one or more coded slice NAL units with `priority_id` equal to 0 and `temporal_id` equal to 0.

NOTE 2 – It is possible that not all operation points of sub-bitstreams resulting from the sub-bitstream extraction process have an applicable `level_idc` or `level_idc[ i ]`. In this case, each coded video sequence in a sub-bitstream must still conform to one or more of the profiles specified in Annex A and Annex H, but may not satisfy the level constraints specified in subclauses A.3 and H.10.2, respectively.

Inputs to this process are:

- a variable `pldTarget` (when present),
- a variable `tldTarget` (when present),
- a list `viewIdTargetList` consisting of one or more values of `viewIdTarget` (when present).

Outputs of this process are a sub-bitstream and a list of `VOldx` values `VOldxList`.

When `pldTarget` is not present as input to this subclause, `pldTarget` is inferred to be equal to 63.

When `tldTarget` is not present as input to this subclause, `tldTarget` is inferred to be equal to 7.

When `viewIdTargetList` is not present as input to this subclause, there shall be one value of `viewIdTarget` inferred in `viewIdTargetList` and the value of `viewIdTarget` is inferred to be equal to `view_id` of the base view.

The sub-bitstream is derived by applying the following operations in sequential order:

1. Let `VOldxList` be empty and `minVOldx` be the `VOldx` value of the base view.
2. For each value of `viewIdTarget` included in `viewIdTargetList`, invoke the process specified in subclause H.8.5.1 with the value of `viewIdTarget` as input.
3. For each value of `viewIdTarget` included in `viewIdTargetList`, invoke the process specified in subclause H.8.5.2 with the value of `viewIdTarget` as input.
4. Mark all VCL NAL units and filler data NAL units for which any of the following conditions are true as "to be removed from the bitstream":
 - `priority_id` is greater than `pldTarget`,
 - `temporal_id` is greater than `tldTarget`,
 - `view_id` is not in the `viewIdTargetList`.
5. Remove all access units for which all VCL NAL units are marked as "to be removed from the bitstream".
6. Remove all VCL NAL units and filler data NAL units that are marked as "to be removed from the bitstream".