INTERNATIONAL STANDARD

ISO/IEC 14496-15

Second edition
2010-06-01

Information technology — Coding of audio-visual objects —

Part 15: Advanced Video Coding (AVC) file format

Technologies de l'information — Codage des objets audiovisuels —

Partie 15: Format de fichier de codage vidéo avancé (AVC)

Reference number
ISO/IEC 14496-15:2010(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 14496-15 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 14496-15:2004) which has been technically revised. It also incorporates the Amendments ISO/IEC 14496-15:2004/Amd.1:2006, ISO/IEC 14496-15:2004/Amd.2:2008, and the Technical Corrigenda ISO/IEC 14496-15:2004/Cor.1:2006, ISO/IEC 14496-15;2004/Cor.2:2006 and ISO/IEC 14496-15:2004/Cor.3:2009.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

— *Part 1: Systems*

— *Part 2: Visual*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Delivery Multimedia Integration Framework (DMIF)*

— *Part 7: Optimized reference software for coding of audio-visual objects* [Technical Report]

— *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*

— *Part 9: Reference hardware description* [Technical Report]

— *Part 10: Advanced Video Coding*

— *Part 11: Scene description and application engine*

— *Part 12: ISO base media file format*

— *Part 13: Intellectual Property Management and Protection (IPMP) extensions*

— *Part 14: MP4 file format*

— *Part 15: Advanced Video Coding (AVC) file format*

— *Part 16: Animation Framework eXtension (AFX)*

— *Part 17: Streaming text format*

— *Part 18: Font compression and streaming*

— *Part 19: Synthesized texture stream*

— *Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)*

— *Part 21: MPEG-J Graphics Framework eXtension (GFX)*

— *Part 22: Open Font Format*

— *Part 23: Symbolic Music Representation*

— *Part 24: Audio and systems interaction*

— *Part 25: 3D Graphics Compression Model*

— *Part 26: Audio conformance*

— *Part 27: 3D Graphics conformance*

Introduction

The Advanced Video Coding (AVC) standard, jointly developed by the ITU-T and ISO/IEC JTC 1/SC 29/WG 11 (MPEG), offers not only increased coding efficiency and enhanced robustness, but also many features for the systems that use it. To enable the best visibility of, and access to, those features, and to enhance the opportunities for the interchange and interoperability of media, this part of ISO/IEC 14496 defines a storage format for video streams compressed using AVC.

This part of ISO/IEC 14496 defines a storage format based on, and compatible with, the ISO Base Media File Format (ISO/IEC 14496-12 and ISO/IEC 15444-12), which is used by the MP4 file format (ISO/IEC 14496-14) and the Motion JPEG 2000 file format (ISO/IEC 15444-3) among others. This part of ISO/IEC 14496 enables AVC video streams to

— be used in conjunction with other media streams, such as audio,

— be used in an MPEG-4 systems environment, if desired,

— be formatted for delivery by a streaming server, using hint tracks, and

— inherit all the use cases and features of the ISO Base Media File Format on which MP4 and MJ2 are based.

This part of ISO/IEC 14496 may be used as a standalone specification; it specifies how AVC content shall be stored in an ISO Base Media File Format compliant format. However, it is normally used in the context of a specification, such as the MP4 file format, derived from the ISO Base Media File Format, that permits the use of AVC video.

The ISO Base Media File Format is becoming increasingly common as a general-purpose media container format for the exchange of digital media, and its use in this context should accelerate both adoption and interoperability.

Extensions to the ISO Base Media File Format are defined here to support the new systems aspects of the AVC codec.

This International Standard defines the storage for plain AVC, SVC, and MVC video streams, where ‘plain AVC’ refers to the main part of ISO/IEC 14496-10, excluding Annex G (Scalable Video Coding) and Annex H (Multiview Video Coding); SVC refers to ISO/IEC 14496-10 when the techniques in Annex G (Scalable Video Coding) are in use, and MVC refers to ISO/IEC 14496-10 when the techniques in Annex H (Multiview Video Coding) are in use. Specific techniques are introduced for handling of scalable and multiview streams, enabling their use, and assisting the extraction of subsets of scalable and multiview streams.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of a patent.

The ISO and IEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured the ISO and IEC that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with the ISO and IEC. Information may be obtained from the companies listed in Annex G.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified in Annex G. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Information technology — Coding of audio-visual objects —

Part 15:
Advanced Video Coding (AVC) file format

1 Scope

This part of ISO/IEC 14496 specifies the storage format for AVC (ISO/IEC 14496-10) video streams.

The storage of AVC content uses the existing capabilities of the ISO base media file format but also defines extensions to support the following features of the AVC codec.

— Switching pictures:
to enable switching between different coded streams and substitution of pictures within the same stream.

— Sub-sequences and layers:
provides a structuring of the dependencies of a group of pictures to provide for a flexible stream structure (e.g. in terms of temporal scalability and layering).

— Parameter sets:
the sequence and picture parameter set mechanism decouples the transmission of infrequently changing information from the transmission of coded macroblock data. Each slice containing the coded macroblock data references the picture parameter set containing its decoding parameters. In turn, the picture parameter set references a sequence parameter set that contains sequence level decoding parameter information.

The file format for storage of SVC content, as defined in Annexes A to E, and the file format for storage of MVC content, as defined in Annexes B to F, use the existing capabilities of the ISO base media file format and the plain AVC file format (i.e. the file format specified in Clauses 2 to 5 not including SVC and MVC supports specified in Annexes A to F). In addition, the following new extensions, among others, to support SVC- and/or MVC-specific features are specified.

— Scalable or multiview grouping:
a structuring and grouping mechanism to indicate the association of NAL units with different types and hierarchy levels of scalability.

— Aggregator:
a structure to enable efficient scalable grouping of NAL units by changing irregular patterns of NAL units into regular patterns of aggregated data units.

— Extractor:
a structure to enable efficient extraction of NAL units from other tracks than the one containing the media data.

— Temporal metadata statements:
structures for storing time-aligned information of media samples.

— AVC compatibility:
a provision for storing an SVC or MVC bitstream in an AVC compatible manner, such that the AVC compatible base layer can be used by any plain AVC file format compliant reader.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14496-1:2001, *Information technology — Coding of audio-visual objects — Part 1: Systems*

ISO/IEC 14496-10, *Information technology — Coding of audio-visual objects — Part 10: Advanced Video Coding*

ISO/IEC 14496-12, *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*[1)]

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 14496-10, this part of ISO/IEC 14496, A.2, F.2 and the following apply.

3.1.1
parameter set
sequence parameter set or a picture parameter set, as defined in ISO/IEC 14496-10

NOTE This term is used to refer to both types of parameter sets.

3.1.2
parameter set elementary stream
elementary stream containing samples made up of only sequence and picture parameter set NAL units synchronized with the video elementary stream

3.1.3
video elementary stream
elementary stream containing access units made up of NAL units for coded picture data

3.2 Abbreviated terms

AVC Advanced Video Coding. Where contrasted with SVC or MVC in this International Standard, this term refers to the main part of ISO/IEC 14496-10, including neither Annex G (Scalable Video Coding) nor Annex H (Multiview Video Coding)

FF File Format

HRD Hypothetical Reference Decoder

IDR Instantaneous Decoding Refresh

MVC MultiviewVideo Coding [refers to ISO/IEC 14496-10 when the techniques in Annex H (Multiview Video Coding) are in use]

NAL Network Abstraction Layer

PPS Picture Parameter Set

1) ISO/IEC 14496-12 is technically identical to ISO/IEC 15444-12.

ROI	Region-Of-Interest
SEI	Supplementary Enhancement Information
SPS	Sequence Parameter Set
SVC	Scalable Video Coding [refers to ISO/IEC 14496-10 when the techniques in Annex G (Scalable Video Coding) are in use]
VCL	Video Coding Layer

4 Extensions to the ISO Base Media File Format

4.1 Introduction

The technologies originally documented in clause 4 are now defined in ISO/IEC 14496-12:2008 (technically identical to ISO/IEC 15444-12:2008).

4.2 File identification

See subclause 6.3 in ISO/IEC 14496-12.

4.3 Independent and Disposable Samples Box

See subclause 8.6.4 in ISO/IEC 14496-12 for the definition of this box.

4.4 Sample groups

See subclause 8.9 in ISO/IEC 14496-12.

4.5 Random access recovery points

See subclause 10.1 in ISO/IEC 14496-12.

4.6 Representation of new structures in movie fragments

See subclause 8.9.4 in ISO/IEC 14496-12.

5 AVC elementary streams and sample definitions

This clause specifies the elementary stream and sample structure used to store AVC visual content.

5.1 Elementary stream structure

AVC specifies a set of Network Abstraction Layer (NAL) units, which contain different types of data. This subclause specifies the format of the elementary streams for storing such AVC content. Two types of elementary streams are defined for this purpose (see also Figure 1):

- **Video Elementary Streams** shall contain all video coding related NAL units (i.e. those NAL units containing video data or signaling video structure) and may contain non-video coding related NAL units such as SEI messages and access unit delimiter NAL units. Other NAL units that are not expressly prohibited may be present, and if they are unrecognized should be ignored (e.g. not placed in the output buffer while accessing the file).

- **Parameter set elementary streams** shall not contain video coding related NAL units (i.e. those NAL units containing video data or signalling video structure), and would normally contain only sequence parameter sets, picture parameter sets and sequence parameter set extension NAL units.

Using these stream types, AVC content shall be stored in either one or two elementary streams:

- **Video elementary stream only**: In this case, sequence and picture parameter set NAL units shall be stored in the sample descriptions of this track. Sequence and picture parameter set NAL units shall not be part of AVC samples within the stream itself.

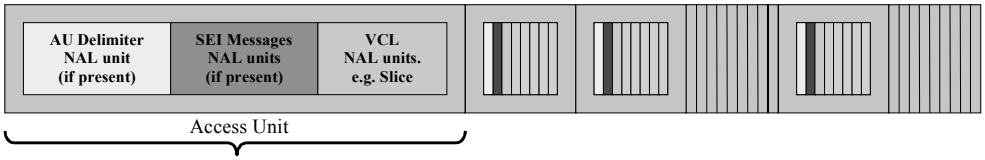
- **Video elementary stream and parameter set elementary stream**: In this case, sequence and picture parameter set NAL units shall be transmitted only in the parameter set elementary stream and shall neither be present in the sample descriptions nor the AVC samples of the video elementary stream.

The types of NAL units that are allowed in each of the video and parameter set elementary streams are specified in Table 1.

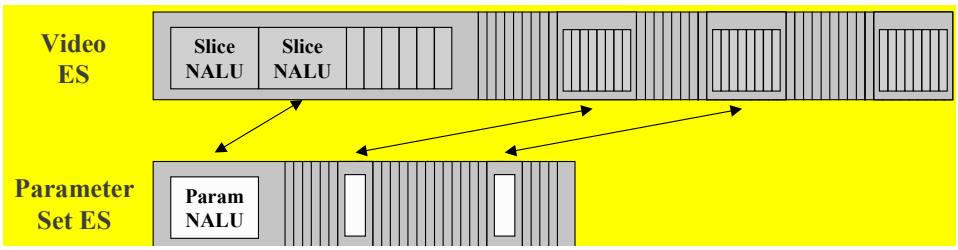
Table 1 — NAL unit types in elementary streams

Value of nal_unit_type	Description	Video elementary stream	Parameter set elementary stream
0	Unspecified	Not specified by this part of ISO/IEC 14496	Not specified by this part of ISO/IEC 14496
1	Coded slice of a non-IDR picture slice_layer_without_partitioning_rbsp( )	Yes	No
2	Coded slice data partition A slice_data_partition_a_layer_rbsp( )	Yes	No
3	Coded slice data partition B slice_data_partition_b_layer_rbsp( )	Yes	No
4	Coded slice data partition C slice_data_partition_c_layer_rbsp( )	Yes	No
5	Coded slice of an IDR picture slice_layer_without_partitioning_rbsp( )	Yes	No
6	Supplemental enhancement information(SEI) sei_rbsp( )	Yes. Except for the Sub-sequence, layering or Filler SEI messages	Only 'declarative' SEIs should be present
7	Sequence parameter set (SPS) seq_parameter_set_rbsp( )	No. If parameter set elementary stream is not used, SPS shall be stored in the Decoder Specific Information.	Yes
8	Picture parameter set (PPS) pic_parameter_set_rbsp( )	No. If parameter set elementary stream is not used, PPS shall be stored in the Decoder Specific Information.	Yes
9	Access unit delimiter (AU Delimiter) access_unit_delimiter_rbsp( )	Yes	No
10	End of sequence end_of_seq_rbsp()	Yes	No
11	End of stream end_of_stream_rbsp()	Yes	No
12	Filler data (FD) filler_data_rbsp( )	No	No
13	Sequence parameter set extension seq_parameter_set_extension_rbsp( )	No. If parameter set elementary stream is not used, Sequence Parameter Set Extension shall be stored in the Decoder Specific Information.	Yes
14…18	Reserved	Not specified by this part of ISO/IEC 14496	Not specified by this part of ISO/IEC 14496
19	Coded slice of an auxiliary coded picture without partitioning slice_layer_without_partitioning_rbsp( )	Yes	No

20…23	Reserved	Not specified by this part of ISO/IEC 14496	Not specified by this part of ISO/IEC 14496
24 – 31	Unspecified	Not specified by this part of ISO/IEC 14496	Not specified by this part of ISO/IEC 14496



(a) Single video elementary stream containing NAL units



(b) Synchronized video and parameter sets with arrows denoting synchronization between streams

Figure 1 — AVC elementary stream structure

5.2 Sample and Configuration definition

5.2.1 Introduction

AVC sample: An AVC sample is an access unit as defined in ISO/IEC 14496-10, 7.4.1.2.

AVC parameter set sample: An AVC parameter set sample is a sample in a parameter set stream which shall consist of those parameter set NAL units that are to be considered as if present in the video elementary stream at the same instant in time.

5.2.2 Canonical order and restrictions

The AVC elementary stream is stored in the ISO Base Media File Format in a *canonical* format. The canonical format is as *neutral* as possible so that systems that need to customize the stream for delivery over different transport protocols — MPEG-2 Systems, RTP, and so on — should not have to *remove* information from the stream while being free to *add* to the stream. Furthermore, a canonical format allows such operations to be performed against a known initial state.

The canonical stream format is an AVC elementary stream that satisfies the following conditions:

- **Video data NAL units (Coded Slice, Coded Slice Data Partition A, Coded Slice Data Partition B, Coded Slice Data Partition C, Coded Slice IDR Pictures)**: All slice and data partition NAL units for a single picture shall be contained with the sample whose decoding time and composition time are those of the picture. Each AVC sample shall contain at least one video data NAL unit of the primary picture.

- **SEI message NAL units**: All SEI message NAL units shall be contained in the sample whose decoding time is that before which the SEI messages come into effect instantaneously, or in the parameter set arrays. The order of SEI messages within a sample is as defined in ISO/IEC 14496-10, 7.4.1.2. This means that the SEI messages for a picture shall be included in the sample containing that picture and that SEI messages pertaining to a sequence of pictures shall be included in the sample containing the first picture of the sequence to which the SEI message pertains.

- **Access unit delimiter NAL units**: The constraints obeyed by access unit delimiter NAL units are defined in ISO/IEC 14496-10, 7.4.1.2.3.

- **Parameter sets***:* If a parameter set elementary stream is used, then the sample in the parameter stream shall have a decoding time equal or prior to when the parameter set(s) comes into effect instantaneously. This means that for a parameter set to be used in a picture it must be sent prior to the sample containing that picture or in the sample for that picture.

  NOTE Parameter sets are stored either in the sample descriptions of the video stream or in the parameter set stream, but never in both. This ensures that it is not necessary to examine every part of the video elementary stream to find relevant parameter sets. It also avoids dependencies of indefinite duration between the sample that contains the parameter set definition and the samples that use it. Storing parameter sets in the sample descriptions of a video stream provides a simple and static way to supply parameter sets. Parameter set elementary streams on the other hand are more complex but allow for more dynamism in the case of updates. Parameter sets may be inserted into the video elementary stream when the file is streamed over a transport that permits such parameter set updates.

- The sequence of NAL units in an elementary stream and within a single sample must be in a valid decoding order for those NAL units as specified in ISO/IEC 14496-10.

- **Parameter set track:** A sync sample in a parameter set track indicates that all parameter sets needed from that time forward in the video elementary stream are in that or succeeding parameter stream samples. Also there shall be a parameter set sample at each point a parameter set is updated. Each parameter set sample shall contain exactly the sequence and picture parameter sets needed to decode the relevant section of the video elementary stream.

  NOTE The use of a parameter set track in the file format does not require that a system delivering AVC content use a separate elementary stream for parameter sets. Instead, implementations may choose to map parameter sets to in-band parameter set NAL units in the video elementary stream or use some out-of-band delivery mechanism defined by the transport layer.

- **All timing information is external to stream.** Picture Timing SEI messages that define presentation or composition timestamps may be included in the AVC video elementary stream, as this message contains other information than timing, and may be required for conformance checking. However, all timing information is provided by the information stored in the various sample metadata tables, and this information over-rides any timing provided in the AVC layer. Timing provided within the AVC stream in this file format should be ignored as it may contradict the timing provided by the file format and may not be correct or consistent within itself.

  NOTE This constraint is imposed due to the fact that post-compression editing, combination, or re-timing of a stream at the file format level may invalidate or make inconsistent any embedded timing information present within the AVC stream.

- **Sub-sequence and layering SEI messages.** Sub-sequence or layering SEI messages shall not occur in the AVC elementary stream. Specifically, the sub-sequence information, sub-sequence layer characteristics, and sub-sequence characteristics SEI messages shall not occur in the stored AVC video elementary stream. Instead, all such information is stored as external metadata as described in 5.3.12.

- **Redundant picture:** NAL units within a single access unit shall be ordered in non-decreasing order of redundant picture count (redundant_pic_cnt).

- **Slice groups:** NAL units within a primary coded picture or a redundant coded picture shall be ordered in non-decreasing order of slice group identifier. Within the same slice group, slices shall be ordered by their first Macroblock location (first_mb_in_slice in the slice header).

  NOTE Slice groups are stored in a canonical order to ease hinting, and to make it easier to find a primary picture within a sample.

- **No start codes.** The elementary streams shall not include start codes. As stored, each NAL unit is preceded by a length field as specified in 5.2.3; this enables easy scanning of the sample's NAL units. Systems that wish to deliver, from this file format, a stream using start codes will need to reformat the stream to insert those start codes.

- **No filler data.** Video data is naturally represented as variable bit rate in the file format and should be filled for transmission if needed. Filler Data NAL units and Filler Data SEI messages shall not be present in the file format stored stream.

  NOTE The removal of Filler Data NAL units, start codes, zero_byte syntax elements, SEI messages or Filler Data SEI messages may change the bit-stream characteristics with respect to conformance with the HRD when operating the HRD in CBR mode as specified in ISO/IEC 14496-10, Annex C.

5.2.3 AVC sample structure definition

This subclause defines structure for the samples of AVC streams. Samples are externally framed and have a size supplied by that external framing. An example of the structure of an AVC sample is depicted in Figure 2.

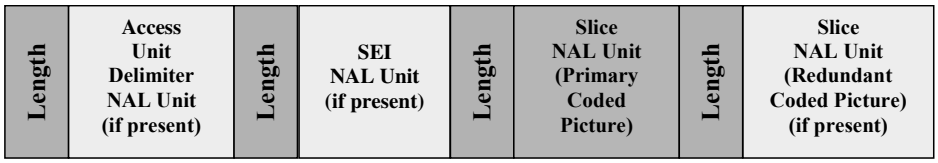


Figure 2 — The structure of an AVC sample

An AVC access unit is made up of a set of NAL units. Each NAL unit is represented with a:

- *Length*: Indicates the length in bytes of the following NAL unit. The length field can be configured to be of 1, 2, or 4 bytes.

- *NAL Unit*: Contains the NAL unit data as specified in ISO/IEC 14496-10.

5.2.4 Decoder configuration information

This subclause specifies the decoder configuration information for ISO/IEC 14496-10 video content.

5.2.4.1 AVC decoder configuration record

This record contains the size of the length field used in each sample to indicate the length of its contained NAL units as well as the initial parameter sets. This record is externally framed (its size must be supplied by the structure which contains it).

This record contains a version field. This version of the specification defines version 1 of this record. Incompatible changes to the record will be indicated by a change of version number. Readers must not attempt to decode this record or the streams to which it applies if the version number is unrecognised.

Compatible extensions to this record will extend it and will not change the configuration version code. Readers should be prepared to ignore unrecognised data beyond the definition of the data they understand (e.g. after the parameter sets in this specification).

When used to provide the configuration of

- a parameter set elementary stream,
- a video elementary stream used in conjunction with a parameter set elementary stream,

the configuration record shall contain no sequence or picture parameter sets (`numOfSequenceParameterSets` and `numOfPictureParameterSets` shall both have the value 0).

The values for AVCProfileIndication, AVCLevelIndication, and the flags which indicate profile compatibility must be valid for all parameter sets of the stream described by this record. The level indication must indicate a level of capability equal to or greater than the highest level indicated in the included parameter sets; each profile compatibility flag may only be set if all the included parameter sets set that flag. The profile indication must indicate a profile to which the entire stream conforms. If the sequence parameter sets are marked with different profiles, and the relevant profile compatibility flags are all zero, then the stream may need examination to determine which profile, if any, the stream conforms to. If the stream is not examined, or the examination reveals that there is no profile to which the stream conforms, then the stream must be split into two or more sub-streams with separate configuration records in which these rules can be met.

Explicit indication can be provided in the AVC Decoder Configuration Record about the chroma format and bit depth used by the avc video elementary stream. The parameter '`chroma_format_idc`' present in the sequence parameter set in AVC specifies the chroma sampling relative to the luma sampling. Similarly the parameters '`bit_depth_luma_minus8`' and '`bit_depth_chroma_minus8`' in the sequence parameter set specify the bit depth of the samples of the luma and chroma arrays. The values of `chroma_format_idc, bit_depth_luma_minus8`' and '`bit_depth_chroma_minus8`' must be identical in all sequence parameter sets in a single AVC configuration record. If two sequences differ in any of these values, two different AVC configuration records will be needed. If the two sequences differ in color space indications in their VUI information, then two different configuration records are also required.

The array of sequence parameter sets, and the array of picture parameter sets, may contain SEI messages of a 'declarative' nature, that is, those that provide information about the stream as a whole. An example of such an SEI is a user-data SEI. Such SEIs may also be placed in a parameter set elementary stream. NAL unit types that are reserved in ISO/IEC 14496-10 and in this specification may acquire a definition in future, and readers should ignore NAL units with reserved values of NAL unit type when they are present in these arrays.

NOTE - this 'tolerant' behaviour is designed so that errors are not raised, allowing the possibility of backwards-compatible extensions to these arrays in future specifications.

When Sequence Parameter Set Extension NAL units occur in this record in profiles other than those indicated for the array specific to such NAL units (profile_idc not equal to any of 100, 110, 122, 144), they should be placed in the Sequence Parameter Set Array.

5.2.4.1.1 Syntax

```
aligned(8) class AVCDecoderConfigurationRecord {
   unsigned int(8) configurationVersion = 1;
   unsigned int(8) AVCProfileIndication;
   unsigned int(8) profile_compatibility;
   unsigned int(8) AVCLevelIndication;
   bit(6) reserved = '111111'b;
   unsigned int(2) lengthSizeMinusOne;
   bit(3) reserved = '111'b;
   unsigned int(5) numOfSequenceParameterSets;
   for (i=0; i< numOfSequenceParameterSets;  i++) {
      unsigned int(16) sequenceParameterSetLength ;
      bit(8*sequenceParameterSetLength) sequenceParameterSetNALUnit;
   }
   unsigned int(8) numOfPictureParameterSets;
   for (i=0; i< numOfPictureParameterSets;  i++) {
      unsigned int(16) pictureParameterSetLength;
      bit(8*pictureParameterSetLength) pictureParameterSetNALUnit;
   }
   if( profile_idc  ==  100  ||  profile_idc  ==  110  ||
       profile_idc  ==  122  ||  profile_idc  ==  144 )
   {
      bit(6) reserved = '111111'b;
      unsigned int(2) chroma_format;
      bit(5) reserved = '11111'b;
      unsigned int(3) bit_depth_luma_minus8;
      bit(5) reserved = '11111'b;
      unsigned int(3) bit_depth_chroma_minus8;
      unsigned int(8) numOfSequenceParameterSetExt;
      for (i=0; i< numOfSequenceParameterSetExt; i++) {
         unsigned int(16) sequenceParameterSetExtLength;
         bit(8*sequenceParameterSetExtLength) sequenceParameterSetExtNALUnit;
      }
   }
}
```

5.2.4.1.2 Semantics

`AVCProfileIndication` contains the profile code as defined in ISO/IEC 14496-10.

`profile_compatibility` is a byte defined exactly the same as the byte which occurs between the profile_IDC and level_IDC in a sequence parameter set (SPS), as defined in ISO/IEC 14496-10.

`AVCLevelIndication` contains the level code as defined in ISO/IEC 14496-10.

`lengthSizeMinusOne` indicates the length in bytes of the `NALUnitLength` field in an AVC video sample or AVC parameter set sample of the associated stream minus one. For example, a size of one byte is indicated with a value of 0. The value of this field shall be one of 0, 1, or 3 corresponding to a length encoded with 1, 2, or 4 bytes, respectively.

`numOfSequenceParameterSets` indicates the number of SPSs that are used as the initial set of SPSs for decoding the AVC elementary stream.

`sequenceParameterSetLength` indicates the length in bytes of the SPS NAL unit as defined in ISO/IEC 14496-10.

`sequenceParameterSetNALUnit` contains a SPS NAL unit, as specified in ISO/IEC 14496-10. SPSs shall occur in order of ascending parameter set identifier with gaps being allowed.

`numOfPictureParameterSets` indicates the number of picture parameter sets (PPSs) that are used as the initial set of PPSs for decoding the AVC elementary stream.

`pictureParameterSetLength` indicates the length in bytes of the PPS NAL unit as defined in ISO/IEC 14496-10.

`pictureParameterSetNALUnit` contains a PPS NAL unit, as specified in ISO/IEC 14496-10. PPSs shall occur in order of ascending parameter set identifier with gaps being allowed.

`chroma_format` contains the chroma_format indicator as defined by the chroma_format_idc parameter in ISO/IEC 14496-10.

`bit_depth_luma_minus8` indicates the bit depth of the samples in the Luma arrays. For example, a bit depth of 8 is indicated with a value of zero (BitDepth = 8 + bit_depth_luma_minus8). The value of this field shall be in the range of 0 to 4, inclusive.

`bit_depth_chroma_minus8` indicates the bit depth of the samples in the Chroma arrays. For example, a bit depth of 8 is indicated with a value of zero (BitDepth = 8 + bit_depth_luma_minus8). The value of this field shall be in the range of 0 to 4, inclusive.

`numOfSequenceParameterSetExt` indicates the number of Sequence Parameter Set Extensions that are used for decoding the AVC elementary stream.

`sequenceParameterSetExtLength` indicates the length in bytes of the SPS Extension NAL unit as defined in ISO/IEC 14496-10.

`sequenceParameterSetExtNALUnit` contains a SPS Extension NAL unit, as specified in ISO/IEC 14496-10.

5.3 Derivation from ISO Base Media File Format

5.3.1 Introduction

This subclause defines how the plain AVC file format is derived from the ISO Base Media File Format [ISO/IEC 14496-12].

Table 2 summarizes the correspondences between the sets of terminology used in AVC Video and the ISO Base Media File Format.

Table 2 — Correspondence of terms in AVC and ISO Base Media File Format

AVC	ISO Base Media File Format
-	Movie
Stream	Track
Access Unit	Sample

5.3.2 AVC File type and identification

Conformance with this part of ISO/IEC 14496 is indicated by the presence of the brand of a specification that permits the inclusion of AVC content, in the compatible brands list of the `FileTypeBox` as defined in ISO/IEC 14496-12. The file extension normally matches the major brand.

AVC content may be used in an MPEG-4 context. If the extensions documented in Clause 4 are not used or their support is not required in the decoder, then the brands ‘mp41’ or ‘mp42’ may be used. If the extensions are required, then the brand ‘avc1’ should be used. In this case, in a file with extension “.mp4”, the major brand may be ‘avc1’.

Readers conformant to this part of ISO/IEC 14496 should read the file if a suitable brand occurs in the compatible-brands list. Other structures and/or track types, defined in specifications other than that identified by the brand, may be present, and these may be ignored by a reader conformant with the specification identified by the brand.

5.3.3 AVC Track Structure

In the terminology of ISO/IEC 14496-12, AVC tracks (both video and parameter set tracks) are video or visual tracks. They therefore use:

a) a handler_type of 'vide' in the HandlerBox;

b) a video media header `'vmhd'`;

c) and, as defined below, a derivative of the `VisualSampleEntry`.

A video stream is represented by one or more video tracks in a file.

If there is more than one track representing scalable aspects of a single stream, then they form alternatives to each other, and the field '`alternate_group`' should be used, or the composition system used should select one of them, as appropriate. See 8.10.3 "Track Selection Box" of ISO/IEC 14496-12 for informative labelling of why tracks are members of alternate groups.

5.3.4 AVC Video Stream Definition

This subclause defines the sample entry and sample format for AVC video elementary streams.

5.3.4.1 Sample description name and format

5.3.4.1.1 Definition

Box Types: 'avc1', 'avc2', 'avcC', 'm4ds','btrt'
Container: Sample Table Box ('stbl')
Mandatory: An 'avc1' or 'avc2' sample entry is mandatory
Quantity: One or more sample entries may be present

An AVC visual sample entry shall contain an AVC Configuration Box, as defined below. This includes an `AVCDecoderConfigurationRecord`, as defined in 5.2.4.1.

An optional `MPEG4BitRateBox` may be present in the AVC visual sample entry to signal the bit rate information of the AVC video stream. Extension descriptors that should be inserted into the Elementary Stream Descriptor, when used in MPEG-4, may also be present.

Multiple sample descriptions may be used, as permitted by the ISO Base Media File Format specification, to indicate sections of video that use different configurations or parameter sets.

The sample entry name 'avc1' may only be used when the entire stream is a compliant and usable AVC stream as viewed by an AVC decoder operating under the configuration (including profile and level) given in the AVCConfigurationBox. The file format specific structures that resemble NAL units (see Annex B) may be present but must not be used to access the AVC base data; that is, the AVC data must not be contained in Aggregators (though they may be included within the bytes referenced by the additional_bytes field) nor referenced by Extractors.

The sample entry name 'avc2' may only be used when Extractors or Aggregators (Annex B) are required to be supported, and an appropriate Toolset is required (for example, as indicated by the file-type brands). This sample entry type indicates that, in order to form the intended AVC stream, Extractors must be replaced with the data they are referencing, and Aggregators must be examined for contained NAL Units. Tier grouping may be present.

5.3.4.1.2 Syntax

```
   // Visual Sequences
class AVCConfigurationBox extends Box('avcC') {
   AVCDecoderConfigurationRecord() AVCConfig;
}

class MPEG4BitRateBox extends Box('btrt'){
   unsigned int(32) bufferSizeDB;
   unsigned int(32) maxBitrate;
   unsigned int(32) avgBitrate;
}

class MPEG4ExtensionDescriptorsBox extends Box('m4ds') {
   Descriptor Descr[0 .. 255];
}

class AVCSampleEntry() extends VisualSampleEntry ('avc1'){
   AVCConfigurationBox  config;
   MPEG4BitRateBox ();              // optional
   MPEG4ExtensionDescriptorsBox (); // optional
}

class AVC2SampleEntry() extends VisualSampleEntry ('avc2'){
   AVCConfigurationBox  avcconfig;
   MPEG4BitRateBox bitrate;             // optional
   MPEG4ExtensionDescriptorsBox descr;  // optional
   extra_boxes          boxes;          // optional
}
```

5.3.4.1.3 Semantics

`Compressorname` in the base class `VisualSampleEntry` indicates the name of the compressor used with the value `"\012AVC Coding"` being recommended (\012 is 10, the length of the string as a byte).

`config` is defined in 5.2.4. If a separate parameter set stream is used, numOfSequenceParameterSets and numOfPictureParameterSets must both be zero.

`Descr` is a descriptor which should be placed in the `ElementaryStreamDescriptor` when this stream is used in an MPEG-4 systems context. This does not include `SLConfigDescriptor` or `DecoderConfigDescriptor`, but includes the other descriptors in order to be placed after the `SLConfigDescriptor`.

`bufferSizeDB` gives the size of the decoding buffer for the elementary stream in bytes.

`maxBitrate` gives the maximum rate in bits/second over any window of one second.

`avgBitrate` gives the average rate in bits/second over the entire presentation.

5.3.4.2 Sample format

This subclause defines the format of a sample in an AVC video elementary stream. The syntax of a sample is configured via the decoder specific configuration for the AVC elementary stream. If the coded representations of a picture contain redundant pictures, then the constraints obeyed by the order of the coded pictures are as defined in ISO/IEC 14496-10, 7.4.1.2.3.

5.3.4.2.1 Syntax

```
aligned(8) class AVCSample
{
   unsigned int PictureLength = sample_size; //Size of AVCSample from
SampleSizeBox
   for (i=0; i<PictureLength; )     // to end of the picture
   {
      unsigned int((AVCDecoderConfigurationRecord.LengthSizeMinusOne+1)*8)
         NALUnitLength;
      bit(NALUnitLength * 8) NALUnit;
      i += (AVCDecoderConfigurationRecord.LengthSizeMinusOne+1) + NALUnitLength;
   }
}
```

5.3.4.2.2 Semantics

`NALUnitLength` indicates the size of a NAL unit measured in bytes. The length field includes the size of both the one byte NAL header and the EBSP payload but does not include the length field itself.

`NALUnit` contains a single NAL unit. The syntax of a NAL unit is defined in ISO/IEC 14496-10 and includes both the one byte NAL header and the variable length encapsulated byte stream payload.

5.3.5 AVC parameter set stream definition

This subclause defines the sample entry and sample format for AVC parameter set streams.

5.3.5.1 Sample description name and format

5.3.5.1.1 Definition

Box Types: ‘avcp’
Container: Sample Table Box (‘stbl’)
Mandatory: Yes
Quantity: One or more sample entries may be present

An AVC parameter stream sample entry shall contain an AVC Parameter Stream Configuration Box, as defined below.

5.3.5.1.2 Syntax

```
class AVCParameterSampleEntry() extends VisualSampleEntry (‘avcp’){
   AVCConfigurationBox  config;
}
```

5.3.5.1.3 Semantics

`Compressorname` in the base class `VisualSampleEntry` indicates the name of the compressor used with the value `"\016AVC Parameters"` being recommended (\016 is 14, the length of the string as a byte).

`config` is defined in 5.2.4. `numOfSequenceParameterSets` and `numOfPictureParameterSets` must both be zero.

5.3.5.2 Sample format

This subclause defines the sample format for AVC Parameter set streams. An AVC parameter set sample contains only one or more sequence, picture parameter set, or sequence parameter set extension NAL units.

5.3.5.2.1 Syntax

```
aligned(8) class AVCParameterSample
{
   unsigned int PictureLength = sample_size;
            //Size of AVCParameterSample from SampleSizeBox
   for (i=0; i<PictureLength; )     // to end of the picture
   {
      unsigned int((AVCDecoderConfigurationRecord.LengthSizeMinusOne+1)*8)
         NALUnitLength;
      bit(NALUnitLength * 8) NALUnit;
      i += (AVCDecoderConfigurationRecord.LengthSizeMinusOne+1) + NALUnitLength;
   }
}
```

5.3.5.2.2 Semantics

`NALUnitLength` indicates the size of a NAL unit measured in bytes. The length field includes the size of both the one byte NAL header and the EBSP payload but does not include the length field itself.

`NALUnit` contains a single NAL unit. The syntax of a NAL unit is defined in ISO/IEC 14496-10 and includes both the one byte NAL header and the variable length encapsulated byte stream payload.

5.3.5.3 Track reference

A track reference of type 'avcp' in the video elementary stream track reference table, referencing the parameter set stream, is used to connect from the video elementary stream to the parameter set elementary stream.

5.3.6 Template fields used

The ISO Base Media File Format defines a number of fields which have default values but which may be defined for use by specific sub-systems. Tracks containing AVC data use the following template fields:

a) `alternate_group` in the `TrackHeaderBox` (see 5.3.13 on stream switching).

b) template field '`depth`' in the VisualSampleEntry to document the presence of alpha.

`depth` takes one of the following values

`0x18` – the video sequence is in colour with no alpha

`0x28` – the video sequence is in grayscale with no alpha

`0x20` – the video sequence has alpha (gray or colour)

5.3.7 Visual width and height

The width and height fields in a `VisualSampleEntry` must correctly document the cropped frame dimensions (visual presentation size) of the AVC stream that is described by that entry. The width and height fields do *not* reflect any changes in size caused by SEI messages such as pan-scan. The visual handling of SEI messages such as pan-scan is both optional and terminal-dependent. If the width and height of the sequence changes, then a new sample description is needed.

Note that the visual size in the SPS may be either frame or field size; in the sample entry, it is always the frame size.

The width and height fields in the track header may not be the same as the width and height fields in the one or more `VisualSampleEntry` in the video track. As specified in the ISO Base Media File Format, if normalized visual presentation is needed, all the sequences are normalized to the track width and height for presentation.

5.3.8 Parameter sets

This subclause applies when a separate parameter set stream is not used.

Each AVC sample description, which contains the AVC video stream decoder specific information, includes a group of SPSs and PPSs. This group of parameter sets functions much like a codebook. Each parameter set has an identifier, and each slice references the parameter set it was coded against using the parameter set's identifier.

In the file format each configuration of parameter sets is represented separately. A parameter set cannot be updated without causing a different sample description to be used. For example, suppose that samples 1 to 4 use PPSs identified as 1, 2, 3 and a single SPS identified as 1. At sample 5 a new value of PPS 2 is required but PPSs 1 and 3 remain unaltered and are used until sample 10. In this case, the sample description for samples 1 through 4 is the same and contains the initial values of PPSs 1, 2, 3 and SPS 1. At sample 5 the sample description must change to a second sample description, which contains the updated value for PPS 2 *as well as* the original values of PPSs 1 and 3 and SPS 1. This second sample description is used for samples 5 through 10.

Systems wishing to send SPS or PPS updates will need to compare the two configurations to find the differences in order to send the appropriate parameter set updates.

NOTE 1 It is recommended that when several parameter sets are used and parameter set updating is desired, a separate parameter set elementary stream be used.

NOTE 2 Decoders conforming to this specification are required to support both parameter sets stored in separate elementary streams as well as parameter sets stored in the AVC sample description entries, unless restricted by another specification using this one.

5.3.9 Decoding time (DTS) and composition time (CTS)

Samples are stored in the file format in decoding order. If picture reordering is not used and decoding and composition times are the same, then presentation is the same as decoding order and only the time-to-sample ‘stts’ table is used. Note that any kind of picture may be reordered in AVC video, not only B-pictures.

If decoding time and composition time differ, the composition time-to-sample ‘ctts’ table is also used in conjunction with the 'stts' table.

5.3.10 Sync sample (IDR)

An AVC sample is considered as a random access point if ALL of the following conditions are met:

- The video data NAL units in the sample indicate that the primary picture contained in the sample is an instantaneous decoding refresh (IDR) picture.

- All SPSs and PPSs needed to decode the video data NAL units in the sample of the IDR picture are contained in the decoder configuration of the video elementary stream or in a separate parameter set elementary stream sample.

A parameter set elementary stream sample is a random access point if and only if all parameter sets required by the associated video elementary stream from the time of the parameter set sample forward are supplied, in the parameter set stream, before they are required by the associated video elementary stream.

5.3.11 Shadow sync

The use of the shadow sync table to indicate alternate encodings of a sample for random access are supported as defined in the ISO Base Media File Format. A shadow sync shall indicate a sample that is a random access point as specified in 5.3.10.

While the use of shadow sync for AVC content is supported for backward compatibility reasons, this use is deprecated and use of the mechanisms defined in 5.3.13 is recommended.

5.3.12 Layering and sub-sequences

5.3.12.1 Introduction

Streams may be constructed so that the referential dependencies between samples allow only subsets of the samples to be sent to the decoder. This mechanism is called *thinning* a stream. Thinning discards entire sets of samples using knowledge of what other sets of pictures this set of pictures depends on and what picture sets in turn depend on it.

The referential dependencies between samples in a stream are structured into *layers* and *sub-sequences*. Samples in higher layers can only depend on samples in lower layers. Layers are numbered, and the samples are organized such that a sample in layer N has no dependencies on samples in layers greater than N.

Sub-sequences are as defined in the Annex D of ISO/IEC 14496-10. Dependency relations between sub-sequences represent the dependency structure of a stream. Each sub-sequence belongs to one and only one layer. A sample shall reside in one layer and in one sub-sequence only.

Layering and sub-sequence information is represented in the file format to allow systems reading the files to understand the ways in which stream thinning may be achieved without having to examine the dependency structure of every sample.

Layer and sub-sequences are represented in the AVC file format as Sample Group. An AVC file shall contain zero or one instance of a `SampleGroupBox` (per track) with a `grouping_type` equal to 'avll'. This `SampleGroupBox` instance represents the assignment of samples in a track to layers. An accompanying instance of the SampleGroupDescriptionBox with the same grouping type shall, if it exists, contain `AVCLayerEntry` sample group entries describing the layers. Similarly, an AVC file shall contain zero or one instance of a `SampleGroupBox` (per track) with a `grouping_type` equal to ‘avss’. This `SampleGroupBox` instance represents the assignment of samples in a track to sub-sequences. An accompanying instance of the SampleGroupDescriptionBox with the same grouping type shall, if it exists, contain `AVCSubSequenceEntry` sample group entries describing the sub-sequences.

5.3.12.2 Sub-sequence description entry

Group Types: ‘avss'
Container: Sample Group Description Box ('sgpd')
Mandatory: No
Quantity: Zero or more.

A sub-sequence description entry is a sample group entry that describes a sub-sequence. A sub-sequence is a set of samples in a track belonging to the same layer. A sub-sequence depends on another sub-sequence if and only if there exists a sample in the sub-sequence that is directly referentially dependent on some sample in the other sub-sequence. All samples in a sub-sequence shall directly depend only on (i.e., refer to) other samples within the same sub-sequence or samples in the sub-sequences on which is it dependent. A sub-sequence can depend on zero or more sub-sequences in the lower layers. A sub-sequence shall not depend on any other sub-sequence in the same or higher layer.

At most one partition of an AVC stream into layers shall exist in the AVC file format; that is, there is either zero or one instances of the sample group boxes (`SampleGroupBox`, `SampleGroupDescriptionBox`) per track concerning the grouping of samples into layers and sub-sequences.

5.3.12.2.1 Syntax

```
aligned(8) class DependencyInfo
{
      unsigned int(8)   subSeqDirectionFlag;
      unsigned int(8)   layerNumber;
      unsigned int(16)  subSequenceIdentifier;
}

class AVCSubSequenceEntry () extends VisualSampleGroupEntry ('avss')
{
      unsigned int(16) subSequenceIdentifer;
      unsigned int(8)  layerNumber;
      unsigned int(1)  durationFlag;
      unsigned int(1)  avgRateFlag;
      unsigned int(6)  reserved = 0;
      if (durationFlag)
         unsigned int(32) duration;
      if (avgRateFlag)
      {
         unsigned int(8)  accurateStatisticsFlag;
         unsigned int(16) avgBitRate;
         unsigned int(16) avgFrameRate;
      }
      unsigned int(8) numReferences;
      DependencyInfo dependency[numReferences];
   }
}
```

5.3.12.2.2 Semantics

`subSeqDirectionFlag`, `layerNumber` and `subSequenceIdentifier` within the `DependencyInfo` class identify a sub-sequence that is used as a reference for this sub-sequence. Only direct, not indirect, referential dependencies shall be identified. The identified sub-sequence has sub-sequence identifier equal to `subSequenceIdentifier` and resides in the layer having the layer number equal to `layerNumber`. If `subSeqDirectionFlag` is 0, the sub-sequence used as a reference for this sub-sequence is the closest sub-sequence among all the candidate sub-sequences whose first picture precedes the first picture of this sub-sequence in decoding order and which resides in the indicated layer and has the indicated sub-sequence identifier; 'closest' means that among all the candidate sub-sequences the first picture of the referenced sub-sequence is the closest to the first picture of this sub-sequence in decoding order. If `subSeqDirectionFlag` is equal to 1, the sub-sequence used as a reference for this sub-sequence is the closest sub-sequence among all the candidate sub-sequences whose first picture succeeds the first picture of this sub-sequence in decoding order and which resides in the indicated layer and has the indicated sub-sequence identifier; 'closest' has the same meaning as above.

`subSequenceIdentifier` gives the identifier for the sub-sequence.

`layerNumber` gives the layer number to which the sub-sequence belongs.

`durationFlag` equal to 0 indicates that the duration of the target sub-sequence is not specified. Otherwise, a value of 1 indicates that the `duration` field indicates the duration of this sub-sequence.

`avgRateFlag` equal to 0 indicates that the average bit rate and the average frame rate of the target sub-sequence are unspecified. Otherwise, a value of 1 indicates that the average rate characteristics are described by the `accurateStatisticsFlag`, `avgBitRate`, and `avgFrameRate` fields.

`duration` indicates the duration of the target sub-sequence in clock ticks of a 90-kHz clock.

`accurateStatisticsFlag` indicates how reliable the values of `avgBitRate` and `avgFrameRate` are. `accurateStatisticsFlag` equal to 1 indicates that `avgBitRate` and `avgFrameRate` are rounded from statistically correct values. `accurateStatisticsFlag` equal to 0 indicates that `avgBitRate` and `avgFrameRate` are estimates and may deviate somewhat from the correct values.

`avgBitRate` gives the average bit rate in (1000 bits)/second of this sub-sequence. All NAL units of this sub-sequence are taken into account in the calculation. In the following, B is the number of bits in all NAL units in the sub-sequence. t_1 is the decoding timestamp of the first picture of the sub-sequence (in decoding order), and t_2 is the decoding timestamp of the last picture of the sub-sequence (in decoding order). Then, the avgBitRate is calculated as follows provided that $t_1 \neq t_2$: avgBitRate = round(B ÷ ((t_2 - t_1) * 1000)). If $t_1 = t_2$, avgBitRate shall be 0.

`avgFrameRate` gives the average frame rate in units of frames/(256 seconds) of this sub-sequence. All NAL units of this sub-sequence are taken into account in the calculation. The average frame rate is calculated according to the presentation timestamp of the frame. In the following, C is the number of frames in the sub-sequence. t_1 is the presentation timestamp of the first picture of the sub-sequence (in decoding order), and t_2 is the presentation timestamp (in seconds) of the last picture of the sub-sequence (in decoding order). Then, the avgFrameRate is calculated as follows provided that $t_1 \neq t_2$: avgFrameRate = round(C * 256 ÷ (t_2 – t_1)). If $t_1 = t_2$, `avgFrameRate` shall be 0. Value zero indicates an unspecified frame rate.

`numReferences` gives the number of sub-sequences directly referenced in this sub-sequence. `dependency` is an array of `DependencyInfo` structures giving the identifying referenced sub-sequences.

5.3.12.3 Layer description entry

Group Types: 'avll'
Container: Sample Group Description Box ('sgpd')
Mandatory: No
Quantity: Zero or more.

A layer sample group entry defines the layer information for all samples in a layer. Layers are numbered with non-negative integers. Layers are ordered hierarchically based on their dependency on each other: A layer having a larger layer number is a higher layer than a layer having a smaller layer number. The layers are ordered hierarchically based on their dependency on each other so that a layer does not depend on any higher layer and may depend on lower layers. The lowest layer is numbered as zero and other layers are given consecutive numbers. In other words, layer 0 is independently decodable, pictures in layer 1 may be predicted from layer 0, pictures in layer 2 may be predicted from layers 0 and 1, etc.

5.3.12.3.1 Syntax

```
class AVCLayerEntry() extends VisualSampleGroupEntry ('avll')
{
   unsigned int(8)  layerNumber;
   unsigned int(8)  accurateStatisticsFlag;
   unsigned int(16) avgBitRate;
   unsigned int(16) avgFrameRate;
}
```

5.3.12.3.2 Semantics

`layerNumber` gives the number of this layer with the base layer being numbered as zero and all enhancement layers being numbered as one or higher with consecutive numbers.

`accurateStatisticsFlag` indicates how reliable the values of `avgBitRate` and `avgFrameRate` are. `accurateStatisticsFlag` equal to 1 indicates that `avgBitRate` and `avgFrameRate` are rounded from statistically correct values. `accurateStatisticsFlag` equal to 0 indicates that `avgBitRate` and `avgFrameRate` are estimates and may deviate somewhat from the correct values.

`avgBitRate` gives the average bit rate in units of 1000 bits per second. All NAL units in this and lower sub-sequence layers are taken into account in the calculation. The average bit rate is calculated according to the decoding timestamp. In the following, B is the number of bits in all NAL units in this and lower sub-sequence layers. t_1 is the decoding timestamp of the first picture in this and lower sub-sequence layers in the presentation order, and t_2 is the decoding timestamp of the latest picture in this and lower sub-sequence layers in the presentation order. Then, `avgBitRate` is calculated as follows

provided that $t_1 \neq t_2$: `avgBitRate` = round(B ÷ ((t_2 – t_1) * 1000))). If $t_1 = t_2$, `avgBitRate` shall be 0. Value zero indicates an unspecified bit rate.

`avgFrameRate` gives the average frame rate in units of frames/(256 seconds). All NAL units in this and lower sub-sequence layers are taken into account in the calculation. In the following, C is the number of frames in this and lower sub-sequence layers. t_1 is the presentation timestamp of the first picture in this and lower sub-sequence layers in presentation order, and t_2 is the presentation timestamp of the latest picture in this and lower sub-sequence layers in the presentation order. Then, the `avgFrameRate` is calculated as follows provided that $t_1 \neq t_2$: avgFrameRate = round(C * 256 ÷ (t_2 – t_1)). If $t_1 = t_2$, `avgFrameRate` shall be 0. Value zero indicates an unspecified frame rate.

5.3.13 Alternate streams and switching pictures

In typical streaming scenarios, one of the key requirements is to scale the bit rate of the compressed data in response to changing network conditions. The simplest way to achieve this is to encode multiple streams with different bandwidths and quality settings for representative network conditions. The server can then switch amongst these pre-coded streams in response to network conditions. In earlier standards, switching between streams is only possible at I-pictures, because the pictures can only be switched when there are no dependencies on prior pictures for reconstruction.

AVC includes supports for SP-pictures and SI-pictures ("switching pictures") that allow switching from one stream to another while still supporting inter coding of switching pictures. Figure 3 shows how SP pictures are used to switch between two different bit streams.

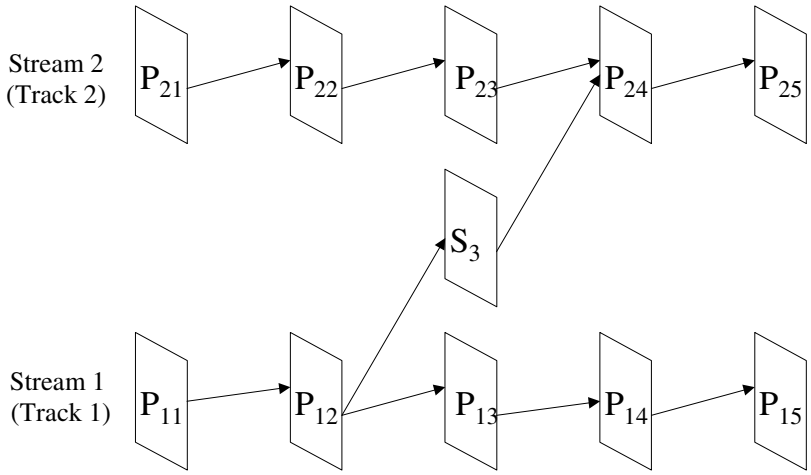


Figure 3 — Stream switching

In the file format, switching pictures are stored in *switching picture* tracks, which are tracks separate from the track that is being switched from and the track being switched to. Switching picture tracks can be identified by the existence of a specific required track reference in that track. A switching picture is an alternative to the sample in the destination track that has exactly the same decoding time. If all switching pictures are SI pictures, then no further information is needed.

If any of the pictures in the switching track are SP pictures, then two extra pieces of information may be needed. First, the source track that is being switched from must be identified by using a track reference (the source track may be the same track as the destination track). Second, the dependency of the switching picture on the samples in the source track may be needed, so that a switching picture is only used when the pictures on which it depends have been supplied to the decoder.

This dependency is represented by means of an optional extra sample table. There is one entry per sample in the switching track. Each entry records the relative sample numbers in the source track on which the switching picture depends. If this array is empty for a given sample, then that switching sample contains an SI picture. If the dependency box is not present, then only SI-frames shall be present in the track.

A switching sample may have multiple coded representations with different dependencies. For AVC video, the multiple representations of a switching sample are stored in different switching tracks (i.e. access unit). For example, one switch track might contain a SP-picture representation dependent on some earlier samples, used for stream switching, while another switch track may contain another representation as an SI-picture, used for random access.

5.3.13.1 Alternate group

The ISO Base Media File Format (but not the version one specification of the MPEG-4 file format, which is branded as 'mp41') supports the use of what is called *alternate tracks.* Each track can optionally specify an *alternate group* (in the track header box) that groups together alternate encodings of the same content. Thus, each alternate bit-stream can be stored as a separate track and related together as alternate tracks. All the tracks which form a group which may be switched between, but not the tracks containing the switching pictures, must be a member of an `alternate_group` with a non-zero group identifier.

An alternate group is not needed if there is only one primary track, with a switching track. This switching track may contain SI pictures, or SP pictures for trick modes or error resilience, which predict both from and to the same track.

5.3.13.2 Track references

The switching track must be linked to the track into which it switches (the destination track) by a track reference of type ‘swto’ in the switching picture track.

If the switching track contains SP pictures, the switching track must be linked to the track from which it switches (the source track) by a track reference of type ‘swfr’ in the switching picture track.

5.3.13.3 Sample dependency

This subclause defines the dependencies of each switching sample on sample(s) in the source track. This table is only needed in a switching track that has a source (‘swfr’) track dependency.

5.3.13.3.1 Definition

Box Type: ‘sdep’
Container: Sample Table ‘stbl’
Mandatory: No
Quantity: Zero or exactly one.

This box contains the sample dependencies for each switching sample. The dependencies are stored in the table, one record for each sample. The size of the table, `sample_count` is taken from the `sample_count` in the Sample Size Box ('stsz') or Compact Sample Size Box (‘stz2’).

5.3.13.3.1.1 Syntax

```
aligned(8) class SampleDependencyBox
   extends FullBox('sdep', version = 0, 0) {
   for (i=0; i < sample_count; i++){
      unsigned int(16) dependency_count;
      for (k=0; k < dependency_count; k++) {
         signed int(16) relative_sample_number;
      }
   }
}
```

5.3.13.3.1.2 Semantics

`dependency_count` is an integer that counts the number of samples in the source track on which this switching sample directly depends.

`relative_sample_number` is an integer that identifies a sample in the source track. The relative sample numbers are encoded as follows. If there is a sample in the source track with the same decoding time, it has a relative sample number of 0. Whether or not this sample exists, the sample in the source track which immediately precedes the decoding time of the switching sample has relative sample number –1, the sample before that –2, and so on. Similarly, the sample in the source track which immediately follows the decoding time of the switching sample has relative sample number +1, the sample after that +2, and so on.

5.3.14 Random access recovery points

The AVC codec includes the concept of a 'gradual decoding refresh' or random access recovery point. This is signalled in the bit-stream using the recovery point SEI message. This message is found at the beginning of the random access, and indicates how much data must be decoded subsequent to the access unit at the position of the SEI message before the recovery is complete.

This concept of gradual recovery is supported in the file format also by using RollRecoveryEntry Groups [4.5]. In order that the group membership marks the sample containing the SEI message the `'roll-distance'` is constrained to being only positive (i.e. a post-roll).

Note carefully that the roll-group counts samples in the file format; this may not match the way that the distances are represented in the SEI message.

Within a stream, it is necessary to mark the beginning of the pre-roll, so that a stream decoder may start decoding there. However, in a file, when performing random access, a deterministic search is desired for the closest preceding frame which can be decoded perfectly (either a sync sample, or the end of a pre-roll).

5.3.15 Hinting

Note that what the hint tracks call "B frames" are actually 'disposable' pictures or non-reference pictures as defined in ISO/IEC 14496-10.

5.3.16 Definition of a sub-sample for AVC

For the use of the sub-sample information box (8.7.7 of ISO/IEC 14496-12) in an AVC stream, a sub-sample is defined as one or more contiguous NAL units within a sample and having the same value of the following fields; `RefPicFlag`, `RedPicFlag` and `VclNalUnitFlag`. Each sub-sample includes both NAL unit(s) and their preceding NAL unit length field(s). The presence of this box is optional; however, if present in a track containing AVC data, it shall have the semantics defined here.

The subsample_priority field shall be set to a value in accordance with the specification of this field in ISO/IEC 14496-12.

The discardable field shall be set to 1 only if this sample can still be decoded if this sub-sample is discarded (e.g. the sub-sample consists of an SEI NAL unit, or a redundant coded picture).

The reserved field is defined for AVC as follows:

```
unsigned int(1) RefPicFlag;
unsigned int(1) RedPicFlag;
unsigned int(1) VclNalUnitFlag;
unsigned int(29) reserved = 0;
```

`RefPicFlag` equal to 0 indicates that all the NAL units in the sub-sample have `nal_ref_idc` equal to 0. `RefPicFlag` equal to 1 indicates that all the NAL units in the sub-sample have `nal_ref_idc` greater than 0.

`RedPicFlag` equal to 0 indicates that all the NAL units in the sub-sample have `redundant_pic_cnt` equal to 0. `RedPicFlag` equal to 1 indicates that all the NAL units in the sub-sample have `redundant_pic_cnt` greater than 0.

`VclNalUnitFlag` equal to 0 indicates that all NAL units in the sub-sample are non-VCL NAL units. Value 1 indicates that all NAL units in the sub-sample are VCL NAL units.

Annex A
(normative)

SVC elementary stream and sample definitions

A.1 Introduction

This Annex specifies the storage format of SVC data. It extends the definitions of the storage format of AVC in clause 5.

A.2 Terms and definitions

For the purpose of Annex A through Annex E, inclusive, the following terms and definitions apply. The definitions in ISO/IEC 14496-10 (including Annex G) also apply.

A.2.1
aggregator
in-stream structures using a NAL unit header including a NAL unit header extension, with a NAL unit type equal to 30

NOTE Aggregators are used to group NAL units belonging to the same sample.

A.2.2
AVC base layer
maximum subset of a scalable bitstream that is AVC compatible, i.e. a bitstream not using any of the functionality of ISO/IEC 14496-10 Annex G

NOTE 1 The AVC base layer is represented by AVC VCL NAL units and associated non-VCL NAL units.

NOTE 2 The AVC base layer itself can be a temporal scalable bitstream.

A.2.3
AVC NAL units
AVC VCL NAL units and their associated non-VCL NAL units in a bitstream

A.2.4
AVC VCL NAL unit
NAL units with type 1 to 5 (inclusive)

A.2.5
extraction path
set of operations on the original bitstream, each yielding a subset bitstream, ordered such that the complete bitstream is first in the set, and the base layer is last, and all the bitstreams are in decreasing complexity (along one of the scalability axes, such as resolution), and where every bitstream is a valid operating point

NOTE An extraction path may be represented by the values of priority_id in the NAL unit headers. Alternatively an extraction path can be represented by the run of tiers or by a set of hierarchically dependent tracks.

A.2.6
extractor
in-stream structure using a NAL unit header including a NAL unit header extension, with a NAL unit type equal to 31.

NOTE Extractors contain instructions on how to extract data from other tracks. Logically an Extractor can be seen as a 'link'. While accessing a track containing Extractors, the Extractor is replaced by the data it is referencing.

A.2.7
in-stream structure
structure residing within sample data

A.2.8
operating point
subset of a scalable bitstream, representing a particular spatial resolution, temporal resolution, and quality

NOTE 1 Each operating point consists of all the data needed to decode this particular bitstream subset.

NOTE 2 In an SVC stream an operating point can be represented either by (i) specific values of DTQ (dependency_id, temporal_id and quality_id) or (ii) specific values of P (priority_id) or (iii) combinations of them (e.g. PDTQ). Note that the usage of priority_id is defined by the application. In an SVC file a track represents one or more operating points. Within a track tiers may be used to define multiple operating points.

A.2.9
prefix NAL unit
NAL units with type 14

NOTE Prefix NAL units provide scalability information about AVC VCL NAL units and filler data NAL units. Prefix NAL units do not affect the decoding process of a legacy AVC decoder. The behaviour of a legacy AVC file reader as a response to prefix NAL units is undefined.

A.2.10
scalable layer
layer
set of VCL NAL units with the same values of dependency_id, quality_id, and temporal_id and the associated non-VCL NAL units. A scalable layer with any of dependency_id, quality_id, and temporal_id not equal to 0 enhances the video by one or more scalability levels in at least one direction (temporal, quality or spatial resolution)

NOTE SVC uses a "layered" encoder design which results in a bitstream representing "coding layers". In some publications the 'base layer' is the first quality layer of a specific coding layer. In some publications the base layer is the scalable layer with the lowest priority. The SVC file format uses "scalable layer" or "layer" in a general way for describing nested bitstreams (using terms like AVC base layer or SVC enhancement layer).

A.2.11
scalable layer representation
scalable layer representation refers to the bitstream subset that is required for decoding the scalable layer, and consists of the scalable layer itself and all the scalable layers on which the scalable layer depends

NOTE A scalable layer representation is also referred to as the representation of the scalable layer.

A.2.12
sub-picture
proper subset of coded slices of a layer representation

A.2.13
sub-picture tier
tier that consists of sub-pictures

NOTE Any coded slice that is not included in the tier representation of a sub-picture tier is not to be referred to in inter prediction or inter-layer prediction for decoding of the sub-picture tier.

A.2.14
SVC enhancement layer
layer that specifies a part of a scalable bitstream that enhances the video

NOTE 1 An SVC enhancement layer is represented by SVC VCL NAL units and the associated non-VCL NAL units and SEI messages.

NOTE 2 Usually an SVC enhancement layer represents a spatial or coarse-grain scalability (CGS) coding layer (identified by a specific value of dependency_id).

A.2.15
SVC NAL unit
SVC VCL NAL units and their associated non-VCL NAL units in an SVC stream

A.2.16
SVC stream
let the greatest value of dependency_id of all the operating points represented by DTQ (dependency_id, temporal_id and quality_id) combinations be equal to mDid, and the set of all the operating points with dependency_id equal to mDid be mOpSet. SVC stream refers to the bitstream represented by the operating point for which dependency_id is equal to mDid, temporal_id is the greatest temporal_id value among mOpSet, and quality_id is the greatest quality_id value among mOpSet

NOTE The term “SVC stream” is referenced by ‘decoding/accessing the entire stream’ in this document. There may be NAL units which are not required for decoding this operating point.

A.2.17
SVC VCL NAL unit
NAL units with type 20, and NAL units with type 14 when the immediately following NAL units are AVC VCL NAL units

NOTE SVC VCL NAL units do not affect the decoding process of a legacy AVC decoder.

A.2.18
tier
defines a set of operating points within a track, providing information about the operating points and instructions on how to access the corresponding bitstream portions (using maps and groups)

NOTE 1 A tier represents one or more scalable layers of an SVC bitstream.

NOTE 2 The term “tier” is used to avoid confusion with the frequently used term layer. A tier represents a subset of a track and represents an operating point of an SVC bitstream. Tiers in a track subset the entire track, no matter whether the track references another track by extractors.

A.2.19
tier representation
representation of the tier
refers to the bitstream subset that is required for decoding the tier, and consists of the tier itself and all the tiers on which the tier depends

A.3 Elementary stream structure

SVC streams are stored in accordance with 5.1, with the following definition of an SVC video elementary stream:

- **SVC Video Elementary Streams** shall contain all video coding related NAL units (i.e. those NAL units containing video data or signalling video structure) and may contain non-video coding related NAL units such as SEI messages and access unit delimiter NAL units. Also Aggregators (see B.2) or Extractors (see B.3) may be present. Aggregators and Extractors shall be processed as defined in this International Standard (e.g. shall not directly be placed in the output buffer while accessing the file). Other NAL units that are not expressly prohibited may be present, and if they are unrecognized they should be ignored (e.g. not placed in the output buffer while accessing the file).

SVC streams may also be stored using associated parameter set streams, if needed.

For SVC streams, Table 1 is updated as follows; only entries where the definition for SVC differs from AVC, are shown.

Table A.1 — NAL Unit Types in SVC and AVC Streams

Value of nal_unit _type	Description	AVC video elementary stream	SVC video elementary stream	Parameter set elementary stream
14	Prefix NAL unit in scalable extension prefix_nal_unit_rbsp( )	Not specified	Yes	No
15	Subset sequence parameter set subset_seq_parameter_set _rbsp( )	Not specified	No. If parameter set elementary stream is not used, Subset SPS shall be stored in the Decoder Specific Information.	Yes
20	Coded slice in scalable extension slice_layer_ extension_rbsp( )	Not specified	Yes	No
24 – 29	Not specified	Not specified	Not specified	Not specified
30	Aggregator	Not specified	Yes	No
31	Extractor	Not specified	Yes	No

NOTE slice_layer _extension_rbsp was previously called slice_layer_in_scalable_extension_rbsp.

There may be AVC VCL NAL units, SVC VCL NAL units and other NAL units, i.e. non-VCL NAL units, present in an SVC video elementary stream. Additionally, there may be Aggregator NAL units and Extractor NAL units present in an SVC video elementary stream.

An AVC VCL NAL unit in an SVC video elementary stream conforming to one or more profiles specified in Annex G of ISO/IEC 14496-10 shall be immediately preceded by a prefix NAL unit containing the scalability information for the AVC VCL NAL unit. In this file format an AVC VCL NAL unit and the immediately preceding prefix NAL unit are logically seen as one NAL unit: the prefix NAL unit provides the scalability information and the AVC VCL NAL unit provides the NAL unit type and payload.

A.4 Use of the plain AVC file format

The SVC file format is an extension of the plain AVC file format defined in this International Standard.

Subclause 5.3.12 is defined for use with plain AVC streams. Its use with SVC streams is deprecated.

A.5 Sample and configuration definition

A.5.1 Introduction

SVC Sample: An SVC sample is also an access unit as defined in 7.4.1.2 of ISO/IEC 14496-10.

A.5.2 Canonical order and restrictions

A.5.2.1 Restrictions

The following restrictions apply to SVC data in addition to the requirements in 5.2.2.

- **SVC coded slice NAL units** (Coded slices in scalable extension): All SVC coded slice NAL units for a single instant in time shall be contained in the sample whose composition time is that of the picture represented by the access unit. An SVC sample shall contain at least one AVC or SVC VCL NAL unit.

- **Prefix NAL units** (Prefix NAL unit in scalable extension): Each prefix NAL unit is placed immediately before the corresponding AVC VCL NAL unit, providing scalability information about the AVC VCL NAL unit.

  NOTE Prefix NAL units may also be associated with filler data NAL units, which are not present in elementary streams.

- **Aggregators/Extractors**: The order of all NAL units included in an Aggregator or referenced by an Extractor is exactly the decoding order as if these NAL units were present in a sample not containing aggregators or extractors. After processing the Aggregator or the Extractor, all NAL units must be in valid decoding order as specified in ISO/IEC 14496-10.

A.5.2.2 Decoder configuration record

When the decoder configuration record defined in 5.2.4.1 is used for a stream which can be interpreted as either an SVC or AVC stream, the AVC decoder configuration record shall reflect the properties of the AVC compatible base layer, e.g. it shall contain only parameter sets needed for decoding the AVC base layer.

A parameter set stream may be used with SVC streams, as with AVC streams. In that case, parameter sets shall not be included in the decoder configuration record.

Sequence parameter sets are numbered in order of storage from 1 to `numOfSequenceParameterSets` or `numOfPictureParameterSets` respectively. Sequence and Picture parameter sets stored in this record in a file may be referenced using this 1-based index by the `InitialParameterSetBox`.

The `SVCDecoderConfigurationRecord` is structurally identical to an `AVCDecoderConfigurationRecord`. However, the reserved bits preceding and succeeding the `lengthSizeMinusOne` field are re-defined. The syntax is as follows:

```
aligned(8) class SVCDecoderConfigurationRecord {
   unsigned int(8) configurationVersion = 1;
   unsigned int(8) AVCProfileIndication;
   unsigned int(8) profile_compatibility;
   unsigned int(8) AVCLevelIndication;
   bit(1) complete_represenation;
   bit(5) reserved = '11111'b;
   unsigned int(2) lengthSizeMinusOne;
   bit(1) reserved = '0'b;
   unsigned int(7) numOfSequenceParameterSets;
   for (i=0; i< numOfSequenceParameterSets; i++) {
      unsigned int(16) sequenceParameterSetLength ;
      bit(8*sequenceParameterSetLength) sequenceParameterSetNALUnit;
   }
   unsigned int(8) numOfPictureParameterSets;
   for (i=0; i< numOfPictureParameterSets; i++) {
      unsigned int(16) pictureParameterSetLength;
      bit(8*pictureParameterSetLength) pictureParameterSetNALUnit;
   }
}
```

The semantics of the fields `AVCProfileIndication`, `profile_compatibility`, and `AVCLevelIndication` differ from the `AVCDecoderConfigurationRecord` as follows:

The fields `AVCProfileIndication`, `AVCLevelIndication` carry the profile and level indications, respectively, indicating the profile and level of the entire scalable stream in this track. They, and the `profile_compatibility` field, must have values such that a conforming SVC decoder is able to decode bitstreams conforming to the profile, level and profile compatibility flags indicated in any of the sequence parameter sets or subset sequence parameter sets contained in this record.

The semantics of other fields are as follows, or are as defined for an `AVCDecoderConfigurationRecord`:

`complete_representation` is set on a minimal set of tracks that contain a portion of the original encoded scalable stream, as defined in A.6.1. Other tracks may be removed from the file without loss of any portion of the original encoded bitstream, and, once the set of tracks has been reduced to only those in the complete subset, any further removal of a track removes a portion of the encoded information.

`numOfSequenceParameterSets` indicates the number of SPSs and subset SPSs that are used for decoding the SVC elementary stream. The value of `numOfSequenceParameterSets` shall be in the range of 0 to 64, inclusive.

`SequenceParameterSetLength` indicates the length in bytes of the SPS or subset SPS NAL unit.

`SequenceParameterSetNALUnit` contains a SPS or subset SPS NAL unit. SPSs shall occur in order of ascending parameter set identifier with gaps being allowed. Subset SPSs shall occur in order of ascending parameter set identifier with gaps being allowed. Any SPS shall occur before all the subset SPSs, if any.

A.6 Derivation from the ISO base media file format

A.6.1 SVC track structure

A scalable video stream is represented by one or more video tracks in a file. Each track represents one or more operating points of the scalable stream. A scalable stream may, of course, be further thinned, if desired.

There is a minimal set of one or more tracks that, when taken together, contain the complete set of encoded information. All these tracks shall have the flag "`complete_representation`" set in all their sample entries. This group of tracks that form the complete encoded information are called the "complete subset".

Let the lowest operating point be the one of all the operating points represented by DTQ (dependency_id, temporal_id and quality_id) combinations that has the least values of dependency_id, temporal_id and quality_id, respectively. The track that has the flag "`complete_representation`" set and contains the lowest operating point shall be nominated as the 'scalable base track'. All the other tracks that are part of the same scalable encoded information shall be linked to this base track by means of a track reference of type '`sbas`' (scalable base). The complete encoded information can be retained when the tracks included in the "complete subset" are retained; all other tracks shall be extractions, subsets, copies or re-orderings of the complete subset.

NOTE An alternate group may also include completely independent bitstreams, as well as alternative operating points of the same bitstream. The SVC tracks in the alternate group must be examined to see how many scalable base tracks are identified.

NOTE "A scalable bitstream" may require more than one track to represent it (consider a stream with a low-resolution, low-frame-rate base layer, and a high resolution enhancement layer, and a high frame-rate enhancement layer, but missing the data for high resolution high frame-rate). However, such a scalable bitstream is typically a non-conforming bitstream.

All the tracks sharing the same scalable base track must share the same timescale.

A.6.2 Data sharing and extraction

Different tracks may logically share data. This sharing can take one of the following two forms:

a) The sample data is copied from one track into another track (and possibly compacted or re-interleaved with other data, such as audio). This creates larger overall files, but the low bit rate data may be compacted and/or interleaved with other material, for ease of extraction.
b) There may be instructions on how to perform this copy at the time that the file is read.

For the second case, Extractors (defined in B.3) are used.

A.6.3 SVC video stream definition

A.6.3.1 Sample description name and format

A.6.3.1.1 Definition

Types: 'avc2', 'avcC', 'svc1','svcC','seib'
Container: Sample Table Box ('stbl')
Mandatory: Either the avc1, or avc2 or svc1 box is mandatory.
Quantity: One or more sample entries may be present

If an SVC elementary stream contains a usable AVC compatible base layer, then an AVC visual sample entry ('avc1' or 'avc2') shall be used. Here, the entry shall contain initially an AVC Configuration Box, possibly followed by an SVC Configuration Box as defined below. The AVC Configuration Box documents the Profile, Level and Parameter Set information pertaining to the AVC compatible base layer as defined by the `AVCDecoderConfigurationRecord`. The SVC Configuration Box documents the Profile, Level and Parameter Set information pertaining to the entire stream containing the SVC compatible enhancement layers as defined by the `SVCDecoderConfigurationRecord`, stored in the SVCConfigurationBox.

For all sample entries except for 'svc1', i.e. 'avc1' and 'avc2', the width and height fields in the sample entry document the AVC base layer. For an 'svc1' sample entry, the width and height document the resolution achieved by decoding the entire stream.

If the SVC elementary stream does not contain a usable AVC base layer, then an SVC visual sample entry ('svc1') shall be used. The SVC visual sample entry shall contain an SVC Configuration Box, as defined below. This includes an `SVCDecoderConfigurationRecord`, as defined in this International Standard.

The `lengthSizeMinusOne` field in the SVC and AVC configurations in any given sample entry shall have the same value.

A priority assignment URI provides the name (in the URI space) of a method used to assign priority_id values. When it occurs in an AVC or SVC sample entry, exactly one URI shall be present, that documents the priority_id assignments in the stream. The URI is treated here as a name only; it should be de-referenceable, though this is not required. File readers may be able to recognize some methods and thereby know what stream extraction operations based on priority_id would do.

Extractors or aggregators may be used for SVC VCL NAL units in 'avc1', 'avc2' or 'svc1' tracks. The 'extra_boxes' in an 'avc2' sample entry may be an SVCConfigurationBox, ScalabilityInformationSEIBox, SVCPriorityAssignmentBox or other extension boxes.

> NOTE When AVC compatibility is indicated, it may be necessary to indicate an unrealistic level for the AVC base layer, to accommodate the bit rate of the entire stream, because all the NAL units are considered as included in the AVC base layer and hence may be fed to the decoder, which is expected to discard those NAL unit it does not recognize. This case happens when the 'avc1' sample entry is used and both AVC and SVC configurations are present.

Either or both of a ScalabilityInformationSEIBox or SVCConfigurationBox may be present in an 'avc1' sample entry. In this case the AVCSVCSampleEntry definition below applies.

`Compressorname` in the base class `VisualSampleEntry` indicates the name of the compressor used, with the value `"\012SVC Coding"` being recommended (\012 is 10, the length of the string "SVC coding" in bytes).

The parameter sets required to decode a NAL unit that is present in the sample data of a video stream, either directly or by reference from an Extractor, shall be present in the decoder configuration of that video stream or in the associated parameter set stream (if used).

The following table shows for a video track all the possible uses of sample entries, configurations and the SVC tools (excluding timed metadata, which is always used in another track):

sample entry name	with configuration records	Meaning
'avc1'	AVC Configuration Only	A plain AVC track without SVC NAL units; Extractors, aggregators, and tier grouping shall not be present.
'avc1'	AVC and SVC Configurations	An SVC track with both AVC and SVC NAL units; Extractors and aggregators may be present; Extractors shall not reference AVC NAL units; Aggregators shall not contain but may reference AVC NAL units; Tier grouping may be present.
'avc2'	AVC Configuration Only	A plain AVC track without SVC NAL units; Extractors may be present and used to reference AVC NAL units; Aggregators may be present to contain and reference AVC NAL units; Tier grouping may be present.
'svc1'	SVC Configuration	An SVC track without AVC NAL units; Extractors may be present and used to reference both AVC and SVC NAL units; Aggregators may be present to contain and reference both AVC and SVC NAL units; Tier grouping may be present.

A.6.3.1.2 Syntax

```
class SVCConfigurationBox extends Box('svcC') {
   SVCDecoderConfigurationRecord() SVCConfig;
}

class ScalabilityInformationSEIBox extends Box('seib', size)
{
   unsigned int(8*size-64) scalinfosei;
}

class SVCPriorityAssignmentBox extends Box('svcP')
{
   unsigned int(8)   method_count;
   string PriorityAssignmentURI[method_count];
}

class AVCSVCSampleEntry() extends AVCSampleEntry (){
   SVCConfigurationBox  svcconfig;      // optional
   ScalabilityInformationSEIBox  scalability;  // optional
   SVCPriorityAssignmentBox   method;          // optional
}

class AVC2SVCSampleEntry() extends VisualSampleEntry ('avc2'){
   AVCConfigurationBox  avcconfig;
   SVCConfigurationBox  svcconfig;      // optional
   MPEG4BitRateBox bitrate;             // optional
   MPEG4ExtensionDescriptorsBox descr;  // optional
   ScalabilityInformationSEIBox  scalability;  // optional
   SVCPriorityAssignmentBox   method;          // optional
}

// Use this if the track is NOT AVC compatible
class SVCSampleEntry() extends VisualSampleEntry ('svc1'){
   SVCConfigurationBox     svcconfig;
   MPEG4BitRateBox bitrate;             // optional
   MPEG4ExtensionDescriptorsBox descr;  // optional
   ScalabilityInformationSEIBox  scalability;  // optional
   SVCPriorityAssignmentBox   method;          // optional
}
```

A.6.3.1.3 Semantics

`scalinfosei` contains an SEI NAL unit containing only a scalability information SEI message as specified in ISO/IEC 14496-10 Annex G. The 'size' field of the container box ScalabilityInformationSEIBox shall not be equal to 0 or 1.

`method_count` provides a count of the number of following URIs. This field must take the value 1 in an 'avc1', 'avc2' or 'svc1' sample entry.

`PriorityAssignmentURI` provides a unique name of the method used to assign prioritiy_id values. In the case of absence of this box, the priority assignment method is unknown.

A.6.4 SVC visual width and height

The visual width and height documented in a VisualSampleEntry of a stream containing SVC VCL NAL unit is the visual width and height of the AVC base layer, if the stream is described by a sample entry of type 'avc1' or 'avc2'; otherwise it is the visual width and height of decoded pictures by decoding the entire stream.

A.6.5 Sync sample (IDR)

For video data described by a sample entry of type ‘avc1’ or ‘avc2’, the sync sample table identifies IDR access units for both an AVC decoder, and an SVC decoder (if any) operating on the entire bitstream.

For video data described by a sample entry of type ‘svc1’, the sync sample table identifies IDR access units in the entire SVC bitstream.

> NOTE The sync sample table, if present, documents only access units that are IDR access units for both the AVC compatible base layer and the layer corresponding to decoding the entire bitstream contained in the track. In case documenting of layer-specific IDR access units is desired, the stream should be stored in separate tracks, e.g. two tracks, one containing the AVC base layer with a sample entry of type ‘avc1’, and the other containing the SVC enhancement layers with a sample entry of type ‘svc1’. However, extractors must then be used for tracks that are not the scalable base track.

A.6.6 Shadow sync

A shadow sync box shall not be used for video data described by an ‘svc1’ sample entry. Its use for AVC is deprecated.

A.6.7 Independent and disposable samples box

If it is used in a track which is both AVC and SVC compatible, then care should be taken that the statements are true no matter what valid subset of the SVC data (possibly only the AVC data) is used. The ‘unknown’ values (value 0 of the fields sample-depends-on, sample-is-depended-on, and sample-has-redundancy) may be needed if the information varies.

A.6.8 Random access recovery points

For video data described by a sample entry of type ‘avc1’ or ‘avc2’, the random access recovery sample group identifies random access recovery points for both an AVC decoder, and an SVC decoder (if any) operating on the entire bitstream.

> NOTE If the random access recovery points for the AVC decoder and the SVC decoder operating on the entire bitstream are not all aligned, the random access recovery points table will not document all of them. In this case, the stream can be stored in multiple tracks, e.g. two tracks, one containing the AVC base layer with a sample entry of type ‘avc1’, and the other containing the SVC enhancement layers with a sample entry of type ‘svc1’.

For video data described by a sample entry of type ‘svc1’, the random access recovery sample group identifies random access recovery in the entire SVC bitstream.

A.6.9 Hinting

Care should be taken when the SVC structures (aggregators or extractors) are in use and the track is hinted. These structures are defined only for use in the file format and should not be transmitted. In particular, a hint track that points at an extractor in a video track would cause the extractor itself to be transmitted (which is probably both incorrect and not the desired behaviour), not the data the extractor references. Hint tracks should normally directly reference NAL units specified in ISO/IEC 14496-10.

A.6.10 Definition of a sub-sample for SVC

This subclause extends the definition of a sub-sample for AVC in 5.3.16.

For the use of the sub-sample information box (8.7.7 of ISO/IEC 14496-12) in an SVC stream, a sub-sample is defined as one or more contiguous whole NAL units having the same values of the following fields: RefPicFlag, RedPicFlag, VclNalUnitFlag, IdrFlag, PriorityId, DependencyId, QualityId, TemporalId, UseRefBasePicFlag, DiscardableFlag and StoreBaseRepFlag, specified subsequently. Each sub-sample includes both NAL unit(s) and their preceding NAL unit length field(s). The presence of this box is optional; however, if present in a track containing SVC data, it shall have the semantics defined here.

As required in 5.3.16, the subsample_priority field shall be set to a value in accordance with the specification of this field in ISO/IEC 14496-12.

The reserved field is defined for SVC as follows:

```
unsigned int(1) RefPicFlag;
unsigned int(1) RedPicFlag;
unsigned int(1) VclNalUnitFlag;
unsigned int(5) reserved = 0;
unsigned int(1) reserved = 0;
unsigned int(1) IdrFlag;
unsigned int(6) PriorityId;
unsigned int(1) reserved = 0; // corresponding to no_inter_layer_pred_flag
unsigned int(3) DependencyId;
unsigned int(4) QualityId;
unsigned int(3) TemporalId;
unsigned int(1) UseRefBasePicFlag;
unsigned int(1) DiscardableFlag;
unsigned int(1) reserved = 0; // corresponding to output_flag
unsigned int(1) StoreBaseRepFlag;
unsigned int(1) reserved = 0;
```

For an AVC VCL NAL unit in an SVC context, the prefix NAL unit shall be grouped with the AVC VCL NAL unit in the same sub-sample, and its fields values apply to the AVC VCL NAL unit.

`RefPicFlag` equal to 0 indicates that all the NAL units in the sub-sample have nal_ref_idc equal to 0. `RefPicFlag` equal to 1 indicates that all the NAL units in the sub-sample have nal_ref_idc greater than 0.

`RedPicFlag` equal to 0 indicates that all the NAL units in the sub-sample have redundant_pic_cnt equal to 0. `RedPicFlag` equal to 1 indicates that all the NAL units in the sub-sample have redundant_pic_cnt greater than 0.

`VclNalUnitFlag` equal to 0 indicates that all NAL units in the sub-sample are non-VCL NAL units. Value 1 indicates that all NAL units in the sub-sample are VCL NAL units.

`IdrFlag` indicates the idr_flag value of the NAL units in the sub-sample. All the NAL units in the sub-sample shall have the same value of idr_flag.

`PriorityId` indicates the priority_id value of the NAL units in the sub-sample. All the NAL units in the sub-sample shall have the same value of priority_id.

`NoInterLayerPredFlag` indicates the no_inter_layer_pred_flag of the NAL units in the sub-sample. All the NAL units in the sub-sample shall have the same value of no_inter_layer_pred_flag.

`DependencyId` indicates the dependency_id value of the NAL units in the sub-sample. All the NAL units in the sub-sample shall have the same dependency_id value.

`QualityId` indicates the quality_id value of the NAL units in the sub-sample. All the NAL units in the sub-sample shall have the same quality_id value.

`TemporalId` indicates the temporal_id value of the NAL units in the sub-sample. All the NAL units in the sub-sample shall have the same temporal_id value.

`UseRefBasePicFlag` indicates the use_ref_base_pic_flag value of the NAL units in the sub-sample. All the NAL units in the sub-sample shall have the same value of use_ref_base_pic_flag.

`DiscardableFlag` indicates the discardable_flag value of the NAL units in the sub-sample. All the NAL units in the sub-sample shall have the same discardable_flag value.

NOTE that this is not the same definition as the discardable field in the sub-sample information box.

`StoreBaseRepFlag` indicates the store_base_rep_flag value of the NAL units in the sub-sample. All the NAL units in the sub-sample shall have the same value of store_base_rep_flag.

Annex B
(normative)

In-stream structures specific to SVC and MVC file formats

B.1 Introduction

Aggregators and Extractors are file format internal structures enabling efficient grouping of NAL units or extraction of NAL units from other tracks.

Aggregators and Extractors use the NAL unit syntax. These structures are seen as NAL units in the context of the sample structure. While accessing a sample, Aggregators must be removed (leaving their contained or referenced NAL Units) and Extractors must be replaced by the data they reference. Aggregators and Extractors must not be present in a stream outside the file format.

These structures use NAL unit types reserved for the application/transport layer by ISO/IEC 14496-10.

NOTE The following is from ISO/IEC 14496-10:

"NOTE – NAL unit types 0 and 24..31 may be used as determined by the application. No decoding process for these values of nal_unit_type is specified in this Recommendation | International Standard."

B.2 Aggregators

B.2.1 Definition

This subclause describes Aggregators, which enable NALU-map-group entries to be consistent and repetitive. (See Annex C).

Aggregators are used to group NAL units belonging to the same sample. Aggregators use the same NAL unit header as SVC VCL NAL units or MVC VCL NAL units, but with a different value of NAL unit type. When the svc_extension_flag of the NAL unit syntax (specified in 7.3.1 of ISO/IEC 14496-10) of an aggregator is equal to 1, the NAL unit header of SVC VCL NAL units is used for the aggregator. Otherwise, the NAL unit header of MVC VCL NAL units is used for the aggregator.

Aggregators can both aggregate, by *inclusion,* NAL units within them (within the size indicated by their length) and also aggregate, by *reference,* NAL units that follow them (within the area indicated by the additional_bytes field within them). When the stream is scanned by an AVC file reader, only the included NAL units are seen as "within" the aggregator. This permits an AVC file reader to skip a whole set of un-needed SVC VCL NAL units or MVC VCL NAL units when they are aggregated by inclusion. This also permits an AVC reader not to skip AVC NAL units but let them remain in-stream when they are aggregated by reference.

Aggregators can be used to group AVC base view NAL units. If these Aggregators are used in an 'avc1' track then an aggregator shall not use inclusion but reference of AVC base view NAL units (the length of the Aggregator includes only its header and the NAL units referenced by the Aggregator are specified by `additional_bytes`).

When the aggregator is referenced by either an extractor with data_length equal to zero, or by a Map sample group, the aggregator is treated as aggregating both the included and referenced bytes.

An Aggregator may include or reference Extractors. An Extractor may extract from Aggregators. An aggregator must not include or reference another aggregator directly; however, an aggregator may include or reference an extractor which references an aggregator.

When scanning the stream:

a) if the aggregator is unrecognized (e.g. by an AVC reader or decoder) it is easily discarded with its included content;
b) if the aggregator is not needed (i.e. it belongs to an undesired layer) it and its contents both by inclusion and reference are easily discarded (using its length and additional_bytes fields);
c) if the aggregator is needed, its header is easily discarded and its contents retained.

An aggregator is stored within a sample like any other NAL unit.

All NAL units remain in decoding order within an aggregator.

B.2.2 Syntax

```
class aligned(8) Aggregator (AggregatorSize) {
   NALUnitHeader();
   unsigned int i = sizeof(NALUnitHeader());
   unsigned int((lengthSizeMinusOne+1)*8)
      additional_bytes;
   i += lengthSizeMinusOne+1;
   while (i<AggregatorSize) {
      unsigned int((lengthSizeMinusOne+1)*8)
               NALUnitLength;
      unsigned int(NALUnitLength*8) NALUnit;
      i += NALUnitLength+lengthSizeMinusOne+1;
   };
}
```

B.2.3 Semantics

The value of the variable AggregatorSize is equal to the size of the aggregator NAL unit, and the function sizeof(X) returns the size of the field X in bytes.

`NALUnitHeader()`: the first four bytes of SVC and MVC VCL NAL units.

`nal_unit_type` shall be set to the aggregator NAL unit type (type 30).

For an aggregator including or referencing SVC NAL units, the following shall apply.

`forbidden_zero_bit` and `reserved_three_2bits` shall be set as specified in ISO/IEC 14496-10.

Other fields (`nal_ref_idc`, `idr_flag`, `priority_id`, `no_inter_layer_pred_flag`, `dependency_id`, `quality_id`, `temporal_id`, `use_ref_base_pic_flag`, `discardable_flag`, and `output_flag`) shall be set as specified in B.4.

For an aggregator including or referencing MVC NAL units, the following shall apply.

`forbidden_zero_bit` and `reserved_one_bit` shall be set as specified in ISO/IEC 14496-10.

Other fields (`nal_ref_idc`, `non_idr_flag`, `priority_id`, `view_id`, `temporal_id`, `anchor_pic_flag`, and `inter_view_flag`) shall be set as specified in B.5.

`additional_bytes`: The number of bytes following this aggregator NAL unit that should be considered as aggregated when this aggregator is referenced by an extractor with data_length equal to zero or Map sample group.

`NALUnitLength`: Specifies the size, in bytes, of the NAL unit following. The size of this field is specified with the `lengthSizeMinusOne` field.

`NALUnit`: a NAL unit as specified in ISO/IEC 14496-10, including the NAL unit header. The size of the NAL unit is specified by `NALUnitLength`.

B.3 Extractors

B.3.1 Definition

This subclause describes Extractors, which enable compact formation of tracks that extract, by reference, NAL unit data from other tracks.

An Aggregator may include or reference Extractors. An Extractor may reference Aggregators. When an extractor is processed by a file reader that requires it, the extractor is logically replaced by the bytes it references. Those bytes must not contain extractors; an extractor must not reference, directly or indirectly, another extractor.

> NOTE The track that is referenced may contain extractors even though the data that is referenced by the extractor must not.

An extractor contains an instruction to extract data from another track, which is linked to the track in which the extractor resides, by means of a track reference of type '`scal`'.

The bytes copied shall be one of the following:

a) One entire NAL unit; note that when an Aggregator is referenced, both the included and referenced bytes are copied
b) More than one entire NAL unit

In both cases the bytes extracted start with a valid length field and a NAL unit header.

The bytes are copied only from the single identified sample in the track referenced through the indicated '`scal`' track reference. The alignment is on decoding time, i.e. using the time-to-sample table only, followed by a counted offset in sample number. Extractors are a media-level concept and hence apply to the destination track before any edit list is considered. (However, one would normally expect that the edit lists in the two tracks would be identical).

B.3.2 Syntax

```
class aligned(8) Extractor () {
   NALUnitHeader();
   unsigned int(8) track_ref_index;
   signed   int(8) sample_offset;
   unsigned int((lengthSizeMinusOne+1)*8)
      data_offset;
   unsigned int((lengthSizeMinusOne+1)*8)
      data_length;
}
```

B.3.3 Semantics

`NALUnitHeader()`: the first four bytes of SVC and MVC VCL NAL units.

nal_unit_type shall be set to the extractor NAL unit type (type 31).

For an extractor referencing SVC NAL units, the following shall apply.

`forbidden_zero_bit` and `reserved_three_2bits` shall be set as specified in ISO/IEC 14496-10.

Other fields (`nal_ref_idc`, `idr_flag`, `priority_id`, `no_inter_layer_pred_flag`, `dependency_id`, `quality_id`, `temporal_id`, `use_ref_base_pic_flag`, `discardable_flag`, and `output_flag`) shall be set as specified in B.4.

For an extractor referencing MVC NAL units, the following shall apply.

`forbidden_zero_bit` and `reserved_one_bit` shall be set as specified in ISO/IEC 14496-10.

Other fields (`nal_ref_idc`, `non_idr_flag`, `priority_id`, `view_id`, `temporal_id`, `anchor_pic_flag`, and `inter_view_flag`) shall be set as specified in B.5.

track_ref_index specifies the index of the track reference of type 'scal' to use to find the track from which to extract data. The sample in that track from which data is extracted is temporally aligned or nearest preceding in the media decoding timeline, i.e. using the time-to-sample table only, adjusted by an offset specified by sample_offset with the sample containing the Extractor. The first track reference has the index value 1; the value 0 is reserved.

sample_offset gives the relative index of the sample in the linked track that shall be used as the source of information. Sample 0 (zero) is the sample with the same, or the closest preceding, decoding time compared to the decoding time of the sample containing the extractor; sample 1 (one) is the next sample, sample -1 (minus 1) is the previous sample, and so on.

data_offset: The offset of the first byte within the reference sample to copy. If the extraction starts with the first byte of data in that sample, the offset takes the value 0. The offset shall reference the beginning of a NAL unit length field.

data_length: The number of bytes to copy. If this field takes the value 0, then the entire single referenced NAL unit is copied (i.e. the length to copy is taken from the length field referenced by the data offset, augmented by the additional_bytes field in the case of Aggregators).

NOTE If the two tracks use different `lengthSizeMinusOne` values, then the extracted data will need re-formatting to conform to the destination track's length field size.

B.4 NAL unit header values for SVC

Both extractors and aggregators use NAL unit headers with the NAL unit header SVC extension. The NAL units extracted by an extractor or aggregated by an aggregator are all those NAL units that are referenced or included by recursively inspecting the contents of aggregator or extractor NAL units. The fields nal_ref_idc, idr_flag, priority_id, no_inter_layer_pred_flag, dependency_id, quality_id, temporal_id, use_ref_base_pic_flag, discardable_flag, and output_flag shall take the following values:

The fields `below` shall take the following values:

`nal_ref_idc` shall be set to the highest values of the fields, respectively, in all the extracted or aggregated NAL units.

`idr_flag` shall be set to the highest values of the fields, respectively, in all the extracted or aggregated NAL units.

`priority_id`, `temporal_id`, `dependency_id`, and `quality_id` shall be set to the lowest values of the fields, respectively, in all the extracted or aggregated NAL units.

`discardable_flag` shall be set to 1 if and only if all the extracted or aggregated NAL units have the `discardable_flag` set to 1, and set to 0 otherwise.

`output_flag` should be set to 1 if at least one of the aggregated or extracted NAL units has this flag set to 1, and otherwise set to 0.

`use_ref_base_pic_flag` shall be set to 1 if and only if at least one of the extracted or aggregated VCL NAL units have the `use_ref_base_pic_flag` set to 1, and set to 0 otherwise.

`no_inter_layer_pred_flag` shall be set to 1 if and only if all the extracted or aggregated VCL NAL units have the `no_inter_layer_pred_flag` set to 1, and set to 0 otherwise.
If the set of extracted or aggregated NAL units is empty, then each of these fields takes a value conformant with the mapped tier description.

NOTE Aggregators could group NAL units with different scalability information.

NOTE Aggregators could be used to group NAL units belonging to a level of scalability which may not be signalled by the NAL unit header SVC extension (e.g. NAL units belonging to a region of interest). The description of such Aggregators may be done with the tier description and the NAL unit map groups. In this case more than one Aggregator with the same scalability information may occur in one sample.

NOTE If multiple scalable tracks reference the same media data, then an aggregator should group NAL units with identical scalability information only. This ensures that the resulting pattern can be accessed by each of the tracks.

NOTE If no NAL unit of a particular layer exist in an access unit then an empty Aggregator (in which the length of the Aggregator includes only the header, and additional_bytes is zero) may exist.

B.5 NAL unit header values for MVC

Both Aggregators and Extractors use NAL unit headers with the NAL unit header extension. The NAL units extracted by an extractor or aggregated by an aggregator are all those NAL units that are referenced or included by recursively inspecting the contents of aggregator or extractor NAL units.

The fields `nal_ref_idc`, `non_idr_flag`, `priority_id`, `view_id`, `temporal_id`, `anchor_pic_flag`, and `inter_view_flag` shall take the following values:

`nal_ref_idc` shall be set to the highest values of the field in all the aggregated or extracted NAL units.

`non_idr_flag` shall be set to the lowest values of the field in all the aggregated or extracted NAL units.

`priority_id` and `temporal_id` shall be set to the lowest values of the fields, respectively, in all the aggregated or extracted NAL units.

`view_id` shall be set to the view_id value of the VCL NAL unit with the lowest view order index among all the aggregated or extracted VCL NAL units.

`anchor_pic_flag` and `inter_view_flag` shall be set to the highest value of the fields, respectively, in all the aggregated or extracted VCL NAL units.

Annex C
(normative)

SVC and MVC sample group definitions

C.1 Introduction

The following sample groups may be used in an SVC or MVC track to document the structure of the SVC or MVC stream and to ease obtaining information of subsets of the stream and extraction of any of the subsets.

If views from the same MVC bitstream are stored in multiple MVC tracks and one or more of these tracks contain multiple views, sample group entries and map groups can be used for these tracks containing multiple views.

There are a number of boxes, defined below, which may occur in the sample group description, namely the Scalable Group Entry for an SVC stream or the Multiview Group Entry for an MVC stream.

Each Scalable Group Entry or Multiview Group Entry documents a subset of the SVC stream or the MVC stream, respectively. Each of the subsets is associated with a tier and may contain one or more operating points. A grouping type of ‘scif’ or ‘mvif’ is used to define Scalable Group Entries or Multiview Group Entries, respectively.

For each tier, there may be more than one Scalable Group Entry or Multiview Group Entry in the SampleGroupDescriptionBox of grouping type ‘scif’ or ‘mvif’, respectively. Only one of those entries is the primary definition of the tier.

Though the Scalable and Multiview Group Entries are contained in the SampleGroupDescription box, the grouping is not a true *sample* grouping as each sample may be associated with more than one tier, as these groups are used to describe *sections* of the samples – the NAL units. As a result, it is possible that there may not be a SampleToGroup box of the grouping type 'scif' or ‘mvif’, unless it happens that a group does, in fact, describe a whole sample. Even if a SampleToGroup box of the grouping type 'scif' or ‘mvif’ is present, the information is not needed for extraction of NAL units of tiers; instead, the map groups must always document the ‘pattern’ of NAL units within the samples and provide the NAL-unit-to-tier mapping information that may be needed for extraction of NAL units.

A multiview group specifies an MVC operating point and is therefore associated with the target output views of the MVC operating point. The Multiview Group box, defined in F.8.3, is used to specify a multiview group. Many of the boxes used to characterize SVC and MVC tiers are also used to characterize MVC operating points and can therefore be contained in the Multiview Group box too.

C.2 Definition

C.2.1 Tier information box

C.2.1.1 Definition

Box Type: 'tiri'
Container: ScalableGroupEntry or MultiviewGroupEntry or MultiviewGroupBox
Mandatory: Yes
Quantity: Zero or One // depends on primary_definition

The tier information box provides information about the profile, level, frame size, discardability, and frame-rate of a covered bitstream subset. If the Tier Information box is included in a Scalable Group entry or a Multiview Group entry, the covered bitstream subset consists of the tier and tiers it depends upon. If the Tier Information box is included in a Multiview Group box, the covered bitstream subset consists of the target output views of the multiview group and all the views required for decoding the target output views.

C.2.1.2 Syntax

```
class TierInfoBox extends Box('tiri'){ //Mandatory Box
   unsigned int(16) tierID;
   unsigned int(8) profileIndication;
   unsigned int(8) profile_compatibility;
   unsigned int(8) levelIndication;
   unsigned int(8) reserved = 0;

   unsigned int(16) visualWidth;
   unsigned int(16) visualHeight;

   unsigned int(2) discardable;
   unsigned int(2) constantFrameRate;
   unsigned int(4) reserved = 0;
   unsigned int(16) frameRate;
}
```

C.2.1.3 Semantics

tierID gives the identifier of the tier, when the Tier Information box is included a Scalable Group entry or a Multiview Group entry. Otherwise, the semantics of tierID are unspecified, and in this case, tierID must be set to the reserved value 0.

profileIndication contains the profile_idc as defined in ISO/IEC 14496-10, when the parameter applies to the covered bitstream subset.

profile_compatibility is a byte defined exactly the same as the byte which occurs between the profile_idc and level_idc in a sequence parameter set or a subset sequence parameter set, as defined in ISO/IEC 14496-10 Annex G or Annex H, when the parameters apply to the covered bitstream subset.

levelIndication contains the level_idc as defined in ISO/IEC 14496-10, when the parameter applies to the covered bitstream subset. If the Tier Information Box is included in a Multiview Group Entry, levelIndication shall be valid when all the views of the covered bitstream subset are target output views. If the Tier Information Box is included in a Multiview Group Box, levelIndication shall be valid when the views specified by the respective multiview group are the target output views. If levelIndication is equal to 0 for an MVC stream, the level that applies to the covered bitstream subset and operating with all the views being target output views is unspecified.

The profile, profile compatibility flags and level indicated by the fields profileIndication, profile_compatibility, and levelIndication specifies an interoperability point with which the covered bitstream subset, and, for MVC, operating with the target output views as specified in the semantics of levelIndication, is compatible.

`visualWidth` gives the value of the width of the coded picture (of an SVC stream), coded sub-picture (of an SVC stream), or coded view component (of an MVC stream) in luma pixels of the representation of this tier in the stream or any view component of the covered bitstream subset. A coded sub-picture consists of a proper subset of coded slices of a coded picture. A tier may consist of only sub-pictures. In this case, the tier is referred to as a sub-picture tier. A sub-picture tier may represent a region-of-interest part of the region represented by the entire stream.

NOTE The tier representation of a sub-picture tier might not be a valid stream. One example is as follows. An AVC bitstream is encoded using two slice groups. The first slice group includes the macroblocks representing a region-of-interest and is coded without referring to slices in the other slice group for inter prediction over all the access units. The slices of the first slice group in each access unit then form a sub-picture and a sub-picture tier can be specified to include all the sub-pictures over all the access units.

`visualHeight` gives the value of the height of the coded picture (of an SVC stream), coded sub-picture (of an SVC stream), or coded view component (of an MVC stream) in luma pixels of the representation of this tier in the stream or any view component of the covered bitstream subset.

`discardable` takes one of the following values; the value 02 is reserved.

00 this tier does not contain NAL units with discardable_flag (for SVC) equal to 1 or inter_view_flag (for MVC) equal to 0.

01 this tier contains both NAL units with discardable_flag (for SVC) equal to 1 or inter_view_flag (for MVC) equal to 0 and discardable_flag (for SVC) equal to 0 or inter_view_flag (for MVC) equal to 1.

03 all NAL units in this tier are with discardable_flag (for SVC) equal to 1 or inter_view_flag (for MVC) equal to 0.

`constantFrameRate` specifies if the frame rate of this tier is constant. A value of 0 denotes a non-constant frame rate, a value of 1 denotes a constant frame rate and a value of 2 denotes that it is not clear whether the frame rate is constant. A value of 3 is reserved.

`frameRate` gives the frame rate when the bitstream corresponding to this tier and all the lower tiers that this tier depends on is decoded in frames per second rounded to the closest integer using the Round function specified in ISO/IEC 14496-10. If `constantFrameRate` has a value of 0 or 2 then `frameRate` gives the average frame rate. If `constantFrameRate` has a value of 1 then `frameRate` gives the constant frame rate. `frameRate` equal to 0 indicates an unspecified frame rate. For SVC streams, decoded frames, complementary field pairs and non-paired fields are regarded as frames when deriving the value of `frameRate`. For MVC streams, decoded view components of any single view only are regarded as frames when deriving the value of `frameRate`, regardless of the total number of the views, since all output views are required to have simultaneous view components.

C.2.2 Tier bit rate box

C.2.2.1 Definition

Box Type: ‘tibr’
Container: ScalableGroupEntry or MultiviewGroupEntry or MultiviewGroupBox
Mandatory: No
Quantity: Zero or One

When included in a Scalable Group entry or a Multiview Group entry, the tier bit rate box provides information about the bit rate values of a tier. Two sets of information are provided: for the tier representation, including all the tiers on which the current tier depends, and for the tier alone. Similarly, for each set of information, the following values are supplied:

a) for SVC streams, the lowest long-term average bit rate that this tier could deliver. Let maxDid be the greatest dependency_id for all NAL units of the tier, and minQid be the least quality_id for all the NAL units of the tier and having dependency_id equal to maxDid. The following NAL units of this tier are not considered in calculating this bit rate value: those having dependency_id equal to maxDid and quality_id greater than minQid. For MVC streams, the lowest long-term average bit rate that this tier could deliver is equal to the long-term average bit rate of the tier, when all NAL units of the tier are considered.

b) the long-term average bit rate of the tier; all NAL units of the tier are considered.

c) the maximum, or peak, bit rate of the tier; all NAL units of the tier are considered.

When included in a Multiview Group box, the tier bit rate box provides information about the bit rate values of the covered bitstream subset consisting of the target output views indicated by the multiview group and all the views required for decoding of the target output views. The maximum and long-term average bit rate for the covered bitstream subset are provided.

C.2.2.2 Syntax

```
class TierBitRateBox extends Box('tibr'){
   unsigned int(32) baseBitRate;
   unsigned int(32) maxBitRate;
   unsigned int(32) avgBitRate;

   unsigned int(32) tierBaseBitRate;
   unsigned int(32) tierMaxBitRate;
   unsigned int(32) tierAvgBitRate;
}
```

C.2.2.3 Semantics

`baseBitRate` gives the lowest long-term average bit rate in bits/second of the stream made from this tier and the lower tiers this tier depends on over the entire stream.

For SVC streams, `baseBitRate` is derived as follows. Let maxDid be the greatest dependency_id for all NAL units of the tier, and minQid be the least quality_id for all NAL units of the tier and having dependency_id equal to maxDid. The NAL units that are taken into account when calculating this bit rate value are as follows: 1) all NAL units of the tier except for those having dependency_id equal to maxDid and quality_id greater than minQid; 2) all NAL units of the lower tiers the current tier depends on.

For MVC streams, `baseBitRate` shall be equal to `avgBitRate`.

`maxBitRate` gives the maximum bit rate in bits/second of the stream containing all NAL unit mapped to this tier and the lower tiers this tier depends on, over any window of one second. All NAL units in this tier and the lower tiers this tier depends on are taken into account.

`avgBitRate` gives the long-term average bit rate in bits/second of the stream containing all NAL unit mapped to this tier and the lower tiers this tier depends on, averaged over the entire stream. All NAL units in this tier and the lower tiers this tier depends on are taken into account.

`tierBaseBitRate` gives the lowest long-term average bit rate in bits/second of the stream made from only this tier over the entire stream. For SVC streams, the set of NAL units that are taken into account when calculating this bit rate value is the same as for baseBitRate but excluding all NAL units of the lower tiers this tier depends on. For MVC streams, `tierBaseBitRate` shall be equal to `tierAvgBitRate`.

`tierMaxBitRate` gives the maximum bit rate in bits/second that is provided by only this tier over any window of one second. All NAL units mapped to this tier are taken into account. All NAL units of the lower tiers this tier depends on are not considered.

`tierAvgBitRate` - gives the long-term average bit rate in bits/second that is provided by only this tier, averaged over the entire stream. All NAL units mapped to this tier are taken into account. All NAL units of the lower tiers this tier depends on are not considered.

C.2.3 Priority range

C.2.3.1 Definition

Box Type: 'svpr'
Container: ScalableGroupEntry or MultiviewGroupEntry
Mandatory: Yes
Quantity: Exactly One

NOTE – this box was previously called SVCPriorityRangeBox.

This box reports the minimum and maximum priority_id of the NAL units mapped to this tier.

C.2.3.2 Syntax

```
class PriorityRangeBox extends Box('svpr') {
   unsigned int(2) reserved1 = 0;
   unsigned int(6) min_priorityId;
   unsigned int(2) reserved2 = 0;
   unsigned int(6) max_priorityId;
}
```

C.2.3.3 Semantics

`min_priority_id, min_priority_id` take the minimum or maximum value of the priority_id syntax element that is present in the NAL unit header extension of the SVC or MVC NAL units mapped to the tier. For AVC streams this takes the value that is, or would be, in the prefix NAL unit.

C.2.4 SVC dependency range

C.2.4.1 Definition

Box Types: 'svdr'
Container: ScalableGroupEntry
Mandatory: Yes
Quantity: Exactly One

This box reports the minimum and maximum dependency_id of the NAL units mapped to this tier.

The field min_temporal_id reports the minimum value of temporal_id of the NAL units in the tier having dependency_id equal to min_dependency_id, Similarly the field min_quality_id reports the minimum quality_id of those NAL units. The fields max_temporal_id and max_quality_id similarly report on the maximum values of the respective fields in those NAL units having dependency_id equal to max_dependency_id.

C.2.4.2 Syntax

```
class SVCDependencyRangeBox extends Box('svop') {
   unsigned int(3) min_dependency_id;
   unsigned int(3) min_temporal_id;
   unsigned int(6) reserved = 0;
   unsigned int(4) min_quality_id;
   unsigned int(3) max_dependency_id;
   unsigned int(3) max_temporal_id;
   unsigned int(6) reserved = 0;
   unsigned int(4) max_quality_id;
}
```

C.2.4.3 Semantics

`min_dependency_id, max_dependency_id` take the minimum or maximum value of the `dependency_id` syntax element that is present in the scalable extension NAL unit header defined in the SVC video specification of the NAL units mapped to the tier. For AVC streams this takes the value that is, or would be, in the prefix NAL unit (note: this value is zero).

`min_temporal_id, max_temporal_id` take the minimum or value of the temporal_id syntax element that is present in the scalable extension NAL unit header defined in the SVC video specification of the NAL units mapped to the tier having dependency_id equal to `min_dependency_id` and `max_dependency_id` respectively . For AVC streams this takes the value that is, or would be, in the prefix NAL unit.

`min_quality_id, max_quality_id` take the minimum or value of the quality_id syntax element that is present in the scalable extension NAL unit header defined in the SVC video specification of the NAL units mapped to the tier having dependency_id equal to `min_dependency_id` and `max_dependency_id` respectively . For AVC streams this takes the value that is, or would be, in the prefix NAL unit.

C.2.5 Initial parameter sets box

C.2.5.1 Definition

Box Type: 'svip'
Container: ScalableGroupEntry or MultiviewGroupEntry
Mandatory: No
Quantity: Zero or One

The SVC initial parameter sets box documents which parameter sets are needed for decoding this tier and all the lower tiers it depends on.

C.2.5.2 Syntax

```
class InitialParameterSetBox extends Box ('svip') {
   unsigned int(8) sps_id_count;
   for (i=0; i< sps_id_count; i++)
      unsigned int(8) SPS_index;
   unsigned int(8) pps_id_count;
   for (i=0; i< pps_id_count; i++)
      unsigned int(8) PPS_index;
}
```

C.2.5.3 Semantics

`sps_id_count, pps_id_count` gives the number of entries in the following tables.

`SPS_index` specifies that the SPS or subset SPS with this index is needed for decoding this tier and all the lower tiers it depends on. These are 1-based indices into the arrays in the SVCDecoderConfigurationRecord or the MVCDecoderConfigurationRecord.

`PPS_index` specifies that the PPS with this index is needed for decoding this tier and all the lower tiers it depends on. These are 1-based indices into the arrays in the SVCDecoderConfigurationRecord or the MVCDecoderConfigurationRecord.

C.2.6 SVC rect region box

C.2.6.1 Definition

Box Type: 'rrgn'
Container: ScalableGroupEntry
Mandatory: No
Quantity: Zero or One

The SVC rect region box documents the geometry information of the region represented by the current tier relative to the region represented by another tier. When extended spatial scalability was used to encode in the current tier a cropped region of another tier, then the geometry information of the cropped region can be signaled by this box. This box can also be used to signal the geometry information of a region-of-interest (ROI) when the current tier is a sub-picture tier. This area can either be static for all samples or vary at sample-by-sample basis. Note that it is possible that independent sub-pictures do not depend on all the tiers with lower tierID. In this case dependencies can be given with the tier dependency box.

C.2.6.2 Syntax

```
class RectRegionBox extends Box('rrgn'){
   unsigned int(16) base_region_tierID;
   unsigned int(1) dynamic_rect;
   unsigned int(7) reserved = 0;
   if(dynamic_rect == 0) {
      unsigned int(16) horizontal_offset;
      unsigned int(16) vertical_offset;
      unsigned int(16) region_width;
      unsigned int(16) region_height;
   }
}
```

C.2.6.3 Semantics

`base_region_tierID` gives the tierId value of the tier wherein the represented region is used as the base region for derivation of the region represented by the current tier.

`dynamic_rect` equal to 1 indicates that the region represented by the current tier is a dynamically changing rectangular part of the base region. Otherwise the region represented by the current tier is a fixed rectangular part of the base region.

`horizontal_offset` and `vertical_offset` give respectively the horizontal and vertical offsets of the top-left pixel of the rectangular region represented by the tier, in relative to the top-left pixel of the base region, in luma samples of the base region.

`region_width` and `region_height` give respectively the width and height of the rectangular region represented by the tier, in luma samples of the base region.

C.2.7 Buffering information box

C.2.7.1 Definition

Box Type: 'buff'
Container: ScalableGroupEntry or MultiviewGroupEntry or MultiviewGroupBox
Mandatory: No
Quantity: Zero or One

The BufferingBox contains the buffer information of the covered bitstream subset. If the Buffering box is included in a Scalable Group entry or a Multiview Group entry, the covered bitstream subset consists of the tier and all tiers on which it depends. If the Buffering box is included in a Multiview Group box, the covered bitstream subset consists of the target output views of the multiview group and all the views required for decoding the target output views.

C.2.7.2 Syntax

```
class BufferingBox extends Box('buff'){
   unsigned int(16)     operating_point_count
   for (i = 0; i < operating_point_count; i++){
      unsigned int (32)    byte_rate
      unsigned int (32)    cpb_size
      unsigned int (32)    dpb_size
      unsigned int (32)    init_cpb_delay
      unsigned int (32)    init_dpb_delay
   }
}
```

C.2.7.3 Semantics

`operating_point_count` specifies the number of HRD operating points for the covered bitstream subset. Values of the HRD parameters are specified separately for each operating point. The value of operating_point_count shall be greater than 0.

`byte_rate` specifies the input byte rate (in bytes per second) to the coded picture buffer (CPB) of the HRD. The covered bitstream subset is constrained by the value of BitRate equal to byte_rate * 8 for NAL HRD parameters as specified in ISO/IEC 14496-10. For VCL HRD parameters, the value of BitRate is equal to byte_rate * 40 / 6. The value of byte_rate shall be greater than 0.

`cpb_size` specifies the required size of the coded picture buffer in bytes. The covered bitstream subset is constrained by the value of CpbSize equal to cpb_size * 8 for NAL HRD parameters as specified in ISO/IEC 14496-10. For VCL HRD parameters, the value of CpbSize is equal to cpb_size * 40 / 6.

At least one pair of values of byte_rate and cpb_size of the same operating point shall conform to the maximum bit rate and CPB size allowed by profile and level of the covered bitstream subset.

`dpb_size` specifies the required size of the decoded picture buffer (DPB), in unit of bytes. The covered bitstream subset is constrained by the value of max_dec_frame_buffering equal to Min(16, Floor(dpb_size) / (PicWidthMbs * FrameHeightInMbs * 256 * ChromaFormatFactor))) as specified in ISO/IEC 14496-10.

`init_cpb_delay` specifies the required delay between the time of arrival in the CPB of the first bit of the first access unit and the time of removal from the CPB of the first access unit. It is in units of a 90 kHz clock. The covered bitstream subset is constrained by the value of the nominal removal time of the first access unit from the CPB, $t_{r,n}(0)$, equal to init_cpb_delay as specified in ISO/IEC 14496-10.

`init_dpb_delay` specifies the required delay between the time of arrival in the DPB of the first decoded picture and the time of output from the DPB of the first decoded picture. It is in units of a 90 kHz clock. The covered bitstream subset is constrained by the value of dpb_output_delay for the first decoded picture in output order equal to init_dpb_delay as specified in ISO/IEC 14496-10 assuming that the clock tick variable, t_c, is equal to 1 / 90 000.

C.2.8 Tier dependency box

C.2.8.1 Definition

Box Type: 'ldep'
Container: ScalableGroupEntry or MultiviewGroupEntry
Mandatory: No for ScalableGroupEntry, Yes for MultiviewGroupEntry
Quantity: Zero or One

The TierDependencyBox identifies the tiers that the current tier is dependent on.

C.2.8.2 Syntax

```
class TierDependencyBox extends Box('ldep'){
   unsigned int(16) entry_count;
   for (i=0; i < entry_count; i++)
      unsigned int(16) dependencyTierId;
}
```

C.2.8.3 Semantics

`dependencyTierId` gives the tierId of a tier on which the current tier is directly or indirectly dependent. Tier A is directly dependent on tier B if there is at least one NAL unit in tier A using inter prediction, inter-layer prediction, or inter-view prediction from tier B. Tier A is indirectly dependent on tier B if tier A is not directly dependent on tier B while decoding of tier A requires the presence of tier B. The value of dependencyTierId shall be smaller than the tierId of the current tier. The decoding of the current tier requires the presence of the tier indicated by dependencyTierId. All dependencies up to the tier with the lowest tierId shall be given with the TierDependencyBox.

C.2.9 SVC region of interest box

C.2.9.1 Definition

Box Type: 'iroi'
Container: ScalableGroupEntry
Mandatory: No
Quantity: Zero or One

This box provides the geometry information of region-of-interest (ROI) divisions of the current tier, when the current tier is encoded to multiple (typically a large number of) independent rectangular ROIs.

NOTE This box is typically used for interactive ROI use cases, where the server can interactively transmit only the NAL units belonging to the ROIs requested by a client.

The assignment of NAL units to a ROI is done in a time parallel metadata track as specified in Annex D.

A ROI ID, denoted as roi_id, is specified for each ROI in a tier. If iroi_type is equal to 0, roi_id is equal to the index of a ROI in a ROI raster scan (see ISO/IEC 14496-10 for the definition of "raster scan" and the use of "macroblock raster scan") of the region represented by the tier starting with zero for the top-left ROI in the region. If iroi_type is equal to 1, roi_id is equal to the entry index i in the syntax of IroiInfoBox(). If iroi_type is equal to 2, roi_id is set to a number identifying the region of interest. In this case, the temporal metadata must contain statements mapping NAL units to roi_ids.

C.2.9.2 Syntax

```
class IroiInfoBox extends Box('iroi'){
   unsigned int(2) iroi_type;
   unsigned int(6) reserved = 0;
   if(iroi_type == 0) {
      unsigned int(8) grid_roi_mb_width;
      unsigned int(8) grid_roi_mb_height;
   }
   else if(iroi_type == 1){
      unsigned int(24) num_roi;
      for(int i=1; i<= num_roi; i++) {
         unsigned int(32) top_left_mb;
         unsigned int(8) roi_mb_width;
         unsigned int(8) roi_mb_height;
      }
   }
}
```

C.2.9.3 Semantics

`iroi_type` indicates the types of region division for all the ROIs. The value 0 indicates that all the ROIs (except possibly the right-most ones and the bottom-most ones) are of identical width and height. The value 1 indicates that the geometry information for each ROI is separately signalled. The value 2 indicates that the geometry can not be given. Values greater than 2 are reserved.

`grid_roi_mb_width` and `grid_roi_mb_height` indicate the width and height, respectively, in units of macroblocks, of the ROIs. All the ROIs have identical width and height, with the following exceptions.

When (`PicWidthInMbs % grid_roi_mb_width`) is not equal to 0, the right-most ROIs have a width equal to `( PicWidthInMbs % grid_roi_mb_width )` macroblocks.

When (`PicHeightInMbs % grid_roi_mb_height`) is not equal to 0, the bottom-most ROIs have a height equal to (`PicHeightInMbs % grid_roi_mb_height`) macroblocks.

Where `PicWidthInMbs` and `PicHeightInMbs` are the visual width and height of a decoded picture of the tier representation in units of macroblocks, respectively, as specified in ISO/IEC 14496-10, (x % y) returns the remainder of x divided by y.

`num_roi` indicates the number of ROIs in a coded picture of the tier representation.

`top_left_mb` specifies the macroblock address of the first macroblock in raster scan order in the ROI of the current entry. The value of `top_left_mb` shall be equal to the syntax element first_mb_in_slice of the coded slices that belong to the current tier and that cover the top-left macroblock of the ROI of the current entry.

`roi_mb_width` and `roi_mb_height` indicate the width and height, respectively, in unit of macroblocks, of the ROI of the current entry.

C.2.10 SVC lightweight transcoding Box

C.2.10.1 Definition

Box Type: 'tran'
Container: ScalableGroupEntry
Mandatory: No
Quantity: Zero or One

The presence of the box indicates that the bitstream represented by this tier (and tiers it depends upon) can be transcoded from an SVC stream to an AVC stream as indicated, and that the transcoded bitstream can be given the indicated profile and level indicators, with the indicated bit rates. The information on the resulting profile, level, and bit rate may be given for either of the entropy coding systems, or both.

C.2.10.2 Syntax

```
class TranscodingInfoBox extends Box('tran'){
   unsigned int(4) reserved = 0;
   unsigned int(2) conversion_idc;
   unsigned int(1) cavlc_info_present_flag;
   unsigned int(1) cabac_info_present_flag;
   if(cavlc_info_present_flag){
      unsigned int(24) cavlc_profile_level_idc;
      unsigned int(32) cavlc_max_bitrate;
      unsigned int(32) cavlc_avg_bitrate;
   }
   if(cabac_info_present_flag){
      unsigned int(24) cabac_profile_level_idc;
      unsigned int(32) cabac_max_bitrate;
      unsigned int(32) cabac_avg_bitrate;
   }
}
```

C.2.10.3 Semantics

`conversion_idc` equal to 0, 1, or 2 indicates that the representation of the current tier can be translated to an AVC bit-stream as specified in the semantics of the scalability information SEI message in ISO/IEC 14496-10 Annex G. `conversion_idc` equal to 3 is reserved.

`cavlc_info_present_flag` specifies whether the transcoding information of the translated bitstream using the Context-based Adaptive Variable Length Coding (CAVLC) entropy coder (i.e. when the syntax element entropy_coding_mode_flag in the translated bitstream is equal to 0) as specified in ISO/IEC 14496-10 Annex G is present.

`cabac_info_present_flag` specifies whether the transcoding information of the translated bitstream using the Context-based Adaptive Binary Arithmetic Coding (CABAC) entropy coder (i.e. when entropy_coding_mode_flag in the translated bitstream is equal to 1) as specified in ISO/IEC 14496-10 Annex G is present.

`cavlc_profile_level_idc` is the exact copy of the three bytes comprised of profile_idc, constraint_set0_flag, constraint_set1_flag, constraint_set2_flag, constraint_set3_flag and level_idc, if these syntax elements were used to specify the profile and level compliancy of the transcoded bitstream using the CAVLC entropy coder.

`cavlc_max_bitrate` specifies the maximum bit rate in bits/second (units of 1000 bits/sec), over any window of one second, that is provided by the transcoded bitstream using the CAVLC entropy coder.

`cavlc_avg_bitrate` specifies the average bit rate in bits/second (units of 1000 bits/sec) that is provided by the transcoded bitstream using the CAVLC entropy coder.

`cabac_profile_level_idc` is the exact copy of the three bytes comprised of profile_idc, constraint_set0_flag, constraint_set1_flag, constraint_set2_flag, constraint_set3_flag and level_idc, if these syntax elements were used to specify the profile and level compliancy of the transcoded bitstream using the CABAC entropy coder.

`cabac_max_bitrate` specifies the maximum bit rate in bits/second (units of 1000 bits/sec), over any window of one second, that is provided by the transcoded bitstream using the CABAC entropy coder.

`cabac_avg_bitrate` specifies the average bit rate in bits/second (units of 1000 bits/sec) that is provided by the transcoded bitstream using the CABAC entropy coder.

C.2.11 Scalable and multiview group entries

C.2.11.1 Introduction

Each scalable or multiview group entry is associated with a groupID and a tierID. The tierID entries are ordered in terms of their dependency signalled by the value of tierID. A larger value of tierID indicates a higher tier. A value 0 indicates the lowest tier. Decoding of a tier is independent of any higher tier but may be dependent on lower tiers. Therefore, the lowest tier can be decoded independently, decoding of tier 1 may be dependent on tier 0, decoding of tier 2 may be dependent on tiers 0 and 1, and so on. A tier can include data from one or more layers or views in the video stream.

If two tiers are mutually independent in an SVC stream, then it is required that the tier that has the greater importance, in the view of the content creator, shall be the lower tier (i.e. have the smaller tierID).

NOTE For example, two tiers are mutually independent (though there may be some lower tiers that they both depend on). The first tier, if presented, has higher frame rate but lower individual picture quality, while the second tier, if presented, has lower frame rate but higher individual picture quality. If the file composer can identify that the first tier offers a better user experience for this content than the second tier, then the first tier is assigned a lower tierID value than the second tier.

There shall be exactly one primary definition for each tier. For each ScalableGroupEntry or MultiviewGroupEntry, when the field primary_groupID is equal to the field groupID, the group is the primary definition of this tier, and the following applies.

- TierInfoBox and PriorityRangeBox shall be present.

- For a certain tier, if any of the optional boxes is not present, then that information is not defined for that tier (there is no inheritance of tier information). If for a certain tier no TierDependencyBox is present then this tier may depend on all tiers with lower tierID.

- If the InitialParameterSetBox is present then the parameter sets needed for decoding this tier and all the lower tiers it depends on are indicated with this box. If this box is not present then it is not signalled whether all the parameter sets given by the SVCDecoderConfigurationRecord or MVCDecoderConfigurationRecord are needed. If parameter set streams are used, then the InitialParameterSetBox shall not be present.

- The values of tierIDs are not required to be contiguous.

Additionally, for each ScalableGroupEntry, when the field primary_groupID is equal to the field groupID, SVCDependencyRangeBox shall be present. Additionally, for each MultiviewGroupEntry, when the field primary_groupID is equal to the field groupID, ViewIdentifierBox shall be present.

For each specified tierID, there shall be at least one NAL unit that is associated with it. In other words, it is disallowed to specify tiers that are not used in the track.

Each NAL unit in the elementary stream is associated with a tierID value as follows. First, each sample is associated with a map of groupID values through the sample grouping of type "`scnm`" as specified subsequently. The "`scnm`" sample grouping therefore indicates the association between NAL units and groupID values within each sample. Values of groupID can then be associated with values of tierID using the sample group description box of type "`scif`" or "`mvif`". NAL units associated with a particular tierID value may require all or some of the NAL units associated with smaller tierID values for proper decoding operation, but will never require any NAL unit associated with a greater tierID value. (i.e., dependency will only exist in the direction of lower tiers).

A Server or Player can choose a subset of tierID values that will be needed for proper decoding operation based on the values of the description fields present within the entries (e.g., frame rate, etc) of the sample group description box of type "`scif`" or "`mvif`".

Since the ScalableGroupEntry and the MultiviewGroupEntry are of variable length and have no internal length field, the SampleGroupDescription Box which contains either of them must carry length information for its entries according to version 1 of the SampleGroupDescription box definition.

The data in a particular tier may be protected; this is indicated by the presence of a ProtectionSchemeInfoBox in the tier definition. If any tier is so protected then:

- If the base layer or base view (AVC) is protected, then the sample entry *must* also be transformed by changing its four-character code, and adding a ProtectionSchemeInfoBox, in the standard way.

- If *any* layer or view is protected in a track, a ProtectionSchemeInfoBox of some kind must be added to the sample entry (this is the 'warning' that some protection is in effect). The original format box in the ProtectionSchemeInfoBox is required but may not be needed as the four-character code in the SampleEntry might not have changed if, for example, the base layer is un-protected.

- Extractors may point to data in protected SVC streams; the byte references are to data 'on disc' (i.e. possibly protected). When protecting, if extractors are permitted by the scheme in use, and the protection changes data sizes, then extractors may need re-writing.

C.2.11.2 Scalable group entry

C.2.11.2.1 Definition

Group Type: 'scif'
Container: Sample Group Description Box ('sgpd')
Mandatory: No
Quantity: Zero or More

C.2.11.2.2 Syntax

```
class ScalableGroupEntry() extends VisualSampleGroupEntry ('scif') {
   unsigned int(8) groupID;
   unsigned int(8) primary_groupID;
   unsigned int(1) is_tier_IDR;
   unsigned int(1) noInterLayerPredFlag;
   unsigned int(1) useRefBasePicFlag;
   unsigned int(1) storeBaseRepFlag;
   unsigned int(1) is_tl_switching_point;
   unsigned int(3) reserved = 0;
   unsigned int(8) tl_switching_distance;

   if (groupID == primary_groupID)  // primary definition of tier
   {
      TierInfoBox();              // Mandatory
      SVCDependencyRangeBox();    // Mandatory
      PriorityRangeBox();         // Mandatory

      //Optional Boxes or fields may follow when defined later
      TierBitRateBox();               // optional
      RectRegionBox();                // optional
      BufferingBox();                 // optional
      TierDependencyBox();            // optional
      InitialParameterSetBox();       // optional
      IroiInfoBox();                  // optional
      ProtectionSchemeInfoBox();      // optional
      TranscodingInfoBox();           // optional
   }
}
```

C.2.11.2.3 Semantics

`groupID` gives the identifier of the group entry; groupIDs are arbitrary values but shall be unique.

`primary_groupID` specifies the group containing the primary definition of this tier. If this value is equal to the value of groupID then this group is the primary definition of this tier.

`is_tier_IDR` when set to 1, indicates that, for the members of this group, the coded pictures of the representation of the highest layer (i.e. the layer with the highest value of dependency_id as specified in ISO/IEC 14496-10 Annex G) are IDR pictures. A value of 0 indicates that, for the members of this group, the coded pictures of the representation of the highest layer are not IDR pictures.

`noInterLayerPredFlag` when set to 1, indicates that the members of this group are with no_inter_layer_pred_flag equal to 1 and coded without using inter layer prediction. A value of 0 indicates that the members of this group may have been coded using inter layer prediction.

`useRefBasePicFlag` when set to 1 indicates that the members of this group are with use_ref_base_pic_flag equal to 1 and using decoded base representations for inter prediction such that mismatch due to discarding of NAL units with quality_id greater than 0 is controlled. A value of 0 indicates that the members of this group may have any value of use_ref_base_pic_flag.

`storeBaseRepFlag` when set to 1 indicates that the members of this group are with `store_base_rep_flag` equal to 1 such that the corresponding decoded base representations are stored when the decoding operates at the current tier. A value of 0 indicates that the members of this group may have any value of store_base_rep_flag.

`is_tl_switching_point` when set to 1, indicates that, for the members of this group, those having the highest value of temporal_id as specified in ISO/IEC 14496-10 Annex G are temporal layer switching points. Let the highest value of temporal_id of the members of this group be tId, then the bitstream can be switched at any of the members having temporal_id equal to tId from the temporal layer with temporal_id equal to tId-1 to the temporal layer with temporal_id equal to tId, provided that the members with temporal_id equal to tId-1 indicated by tl_switching_distance have been processed (transmitted and decoded). `is_tl_switching_point` equal to 0 indicates that the members of this group having the highest value of temporal_id as specified in ISO/IEC 14496-10 Annex G may or may not be temporal layer switching points.

`tl_switching_distance` is used when is_tl_switching_point is 1. It indicates the number of samples of the temporal layer with temporal_id equal to tId-1 that must be decoded to ensure decodability of the stream at or above temporal layer tId from the switching point onward. The value 0 indicates a temporal switching point with no dependency on the lower temporal layer. This required distance for a particular sample may be reduced by a temporal layer switching distance statement in the time parallel metadata track for a specific sample.

C.2.11.3 Multiview group entry

C.2.11.3.1 Definition

Group Type: ‘mvif’
Container: Sample Group Description Box ('sgpd')
Mandatory: No
Quantity: Zero or More

C.2.11.3.2 Syntax

```
class MultiviewGroupEntry() extends VisualSampleGroupEntry ('mvif') {
   unsigned int(8) groupID;
   unsigned int(8) primary_groupID;
   unsigned int(4) reserved = 0;
   unsigned int(1) is_tl_switching_point;
   unsigned int(3) reserved = 0;
   unsigned int(8) tl_switching_distance;

   if (groupID == primary_groupID)  // primary definition of tier
   {
      ViewIdentifierBox();       // Mandatory
      TierInfoBox();             // Mandatory
      TierDependencyBox();       // Mandatory
      PriorityRangeBox();        // Mandatory

      //Optional Boxes or fields may follow when defined later
      TierBitRateBox();              // optional
      BufferingBox();                // optional
      InitialParameterSetBox();      // optional
      ProtectionSchemeInfoBox();     // optional
      ViewPriorityBox();             // optional
   }
}
```

C.2.11.3.3 Semantics

`groupID` gives the identifier of the group entry; groupIDs are arbitrary values but shall be unique.

`primary_groupID` specifies the group containing the primary definition of this tier. If this value is equal to the value of groupID then this group is the primary definition of this tier.

`is_tl_switching_point` when set to 1, indicates that, for the members of this group, those having the highest value of temporal_id as specified in ISO/IEC 14496-10 Annex H are temporal layer switching points. Let the highest value of temporal_id of the members of this group be tId, then the bitstream can be switched at any of the members having temporal_id equal to tId from the temporal layer with temporal_id equal to tId-1 to the temporal layer with temporal_id equal to tId, provided that the members with temporal_id equal to tId-1 indicated by tl_switching_distance have been processed (transmitted and decoded). `is_tl_switching_point` equal to 0 indicates that the members of this group having the highest value of temporal_id as specified in ISO/IEC 14496-10 Annex H may or may not be temporal layer switching points.

`tl_switching_distance` is used when is_tl_switching_point is 1. It indicates the number of samples of the temporal layer with temporal_id equal to tId-1 that must be decoded to ensure decodability of the stream at or above temporal layer tId from the switching point onward. The value 0 indicates a temporal switching point with no dependency on the lower temporal layer. This required distance for a particular sample may be reduced by a temporal layer switching distance statement in the time parallel metadata track for a specific sample.

C.3 Mapping NAL units to map groups and tiers

C.3.1 Introduction

In order to describe scalability or view hierarchy within an SVC or MVC access unit, two kinds of sample groups are used:

a) a group to describe sections of a sample. For each of the groups, a ScalableGroupEntry or a MultiviewGroupEntry exists which defines the group properties. Note that these describe tiers, not the entire stream, and therefore describe the NAL units belonging to one tier at any instant, not the entire AU. See C.1 and C.2.

b) a map group, that describes the mapping of each NAL unit inside an AU to a map group (of grouping_type ‘scnm’). For each different sequence of NAL units belonging to a particular map group, a ScalableNALUMapEntry exists. Within an AU a map group includes all NAL units of a tier.

Defining map groups requires that there is a limited number of map grouping patterns for all access units. If there is a varying number of NAL units in successive access units for a given tier, Aggregators can be used to make these varying structures consistent and to reduce the number of map groups required.

C.3.2 Map group definition

Group Type: ‘scnm’
Container: Sample Group Description Box ('sgpd')
Mandatory: No
Quantity: Zero or More

Each sample is associated with a group_description_index in the SampleToGroupBox with grouping_type ‘`scnm`’. A SampleGroupDescriptionBox with grouping_type ‘`scnm`’ contains a ScalableNALUMapEntry for each group_description_index.

```
class ScalableNALUMapEntry() extends VisualSampleGroupEntry ('scnm') {
   unsigned int(8) reserved = 0;
   unsigned int(8) NALU_count;
   for (i=1; i<= NALU_count; i++)
      unsigned int(8) groupID;
   }
}
```

Each sample belonging to a given map group has exactly `NALU_count` NAL units in it (possibly by using aggregators to group together NAL units of the same layer or view). Each of those NAL units maps to the corresponding scalable or multiview group as described by the `groupID`.

NOTE 1 An arbitrarily chosen groupID is used here, rather than the more obvious scalable or multiview group index from the sample group description box, so that if scalable groups are deleted or re-ordered these operations can be detected and handled. Note also that there may be one or more scalable or multiview groups in a given tier.

NOTE 2 If movie fragments are used, new maps cannot be introduced in the fragments (only the association of the new samples to pre-existing maps). In this case, care should be taken to introduce, in the movie box, all the maps that may be needed.

C.4 Decode re-timing groups

C.4.1 Introduction

Group Type: 'dtrt'
Container: Sample Group Description Box ('sgpd')
Mandatory: No
Quantity: Zero or More

When temporal layers are discarded, re-timing the decoding times of some or all samples may be needed to ensure that the stream complies with all buffer and HRD requirements. Also re-timing may improve the transmission and decoding process. Composition times are not affected. If the stream is 'thinned' to tierID X, and there is a re-timing sample grouping for tierID Y, where Y is the largest such tierID less than X that contains re-timing sample grouping, then the adjusted decode time is the time from the time-to-sample table (the original decode time), plus the given re-timing: newDTS = oldDTS + delta. The CTS is given as usual by the composition time to sample table: CTS = oldDTS + compositionOffset, which is CTS = newDTS - delta + compositionOffset.

This re-timing is given as sample groups, which are associated with samples by using the normal sample-to-group structures. Each group provides a set of re-timing deltas and their associated tierIDs. The group definition must be ordered by increasing tierID.

C.4.2 Syntax

```
class DecodeRetimingEntry() extends VisualSampleGroupEntry ('dtrt') {
   unsigned int(8) tierCount;
   for (i=1; i<=tierCount; i++) {
      unsigned int(16) tierID;
      signed int(16) delta;
   }
}
```

C.4.3 Semantics

`tierID` gives the ID of a tier that maps to a temporal level; the tiers with equal or greater tierID, up to the next tierID in this group, use the given decode time delta

`delta` provides an adjustment for the decode time

C.5 View Priority Sample Grouping

C.5.1 Definition

View Priority sample grouping is used to label views with priorities based on content. The higher the content priority, the more interesting or important the view is for the viewer (audience). Note that the 'structural' priority id used in the NAL unit header extension has another meaning and indicates dependencies on other views due to encoding constraints rather than the importance of the view itself, and that though coding dependency imposes constraints on priority_id values, priority_id values do not necessarily indicate coding dependencies.

Content priority id can help a player or viewer selecting interesting views and can also be used as additional information when pruning views from a file. In the latter case, content priority indicates where pruning is least harmful when several views have similar structural priorities due to encoding constraints.

Either version 0 or version 1 of the Sample to Group Box may be used with the View Priority sample grouping. If version 0 of the Sample to Group Box is used and the MVC View Priority Assignment URI box is present in the sample entry, the used priority assignment method is indicated by the first URI entry of the MVC View Priority Assignment URI box. If version 1 of the Sample to Group Box is used and the MVC View Priority Assignment URI box is present in the sample entry, `grouping_type_parameter` is a 1-based index to the MVC View Priority Assignment URI box. If `grouping_type_parameter` points to a non-existing item in the MVC View Priority Assignment URI box, or version 1 of the Sample to Group Box is used and the MVC View Priority Assignment URI box is not present in the sample entry, `grouping_type_parameter` has no defined semantics but the same priority assignment method should be used consistently for a particular value of `grouping_type_parameter`.

NOTE Sub-bitstreams extracted according to content_priority_id only might not form a conforming bitstream; for example, non-output views might have low content priority but be needed for decoding some output views.

C.5.2 Syntax

```
class ViewPriorityBox extends Box ('vipr') {
   for (i=0; i++) {     // To end of box
      unsigned int(6)   reserved = 0;
      unsigned int(10)  view_id;
      unsigned int(32)  content_priority_id;
   }
}

class ViewPriorityEntry() extends VisualSampleGroupEntry ('vipr')
{
   ViewPriorityBox();
}
```

C.5.3 Semantics

`view_id` identifies the view. See the View Identifier box.

`content_priority_id` indicates real-world view priority based on content, i.e., not related to encoding structure. A view that has a lower value than another view has a higher priority than that view.

Annex D
(normative)

Temporal metadata support

D.1 Introduction

A timed metadata track, as defined in ISO/IEC 14496-12, may be used to provide temporal information about the associated video track.

This metadata is stored in metadata tracks. These tracks have a handler type 'meta' and are linked to the media track using a track reference of type 'cdsc' as specified in ISO/IEC 14496-12. The metadata is stored in samples, the decoding time of which is equal to the media samples it describes. Composition offsets are permitted but not required in timed metadata tracks, but, if used, the composition timing must match the composition timing of the associated media track.

The metadata is structured using conceptual *statements*. Each statement has a one-byte type field – indicating what it is asserting, and a size, which is the length of its payload in bytes, *not* including the size and type fields. The length of the size field depends on the type field.

There are two important 'structuring' statements, groupOfStatements and sequenceOfStatements.

The statement groupOfStatements allows several statements to be made about one thing, by grouping them. A groupOfStatements contains a set of statements all of which are asserted about the thing described.

The statement type sequenceOfStatements may be used in the description of the entire sample or of a NAL unit in the media stream that is an Aggregator or Extractor, to describe its sequence of NAL units. A sequenceOfStatements contains a set of statements, which are in one-to-one correspondence with the sequence of contained objects in that which is described.

Each metadata sample is a collection (a group or sequence) of one or more statements about the temporally aligned media sample. Each of the statements in the collection may have a default type from the sample entry, or have an explicit single type in each statement. Similarly, the default length may be indicated in the sample entry, or be inline in each sample. The overall sample is a collection of N statements. The sample entry provides the statement type of each sample (group or sequence), and (optionally) the default type and length values of the statements in the sample. It can also supply a statement which is true of every sample described by this metadata (an 'overall' statement).

There is a set of pre-defined statement types defined in this International Standard, and there is explicit provision for extension statements by other bodies.

There are statement types reserved to ISO, and other statement types reserved for dynamic assignment. Dynamic assignment consists of a table in the Sample Entry of the metadata track, containing pairs mapping a local statement ID to URIs.

The allocation of different categories of statement types is as follows.

0	no metadata (empty statement), reserved to ISO
1-95	short, reserved to ISO
96-191	short, user extension
192-223	long, reserved to ISO
224-255	long, user extension

The URIs are used in the same sense as namespace identifiers in XML; they are not guaranteed to be de-referenceable. If URLs are used, they should contain a month indication in the form yyyymm, indicating that this use of the domain in the URL was authorized by the owner of that domain as of that month. An example may be:

2 maps to http://www.example.com/200610/gateway-types#quality-model

For the purposes of this metadata, a prefix NAL unit and its associated AVC NAL unit are considered as one NAL unit, and the prefix NAL unit provides the NAL unit header values except for the NAL unit type, which is taken from the associated AVC NAL unit.

An aggregator NAL unit may be described, and if it is described by a sequence, the elements in the sequence correspond one-one to the NAL units aggregated by both inclusion and reference. Similarly, an extractor may be described, and if it is described by a sequence, the elements in the sequence correspond one-one to the NAL units in the extracted data.

In all sequences, SEI NAL units are treated as any other NAL unit. If they need to be skipped in sequences, an empty statement or a NAL header statement can be used.

D.2 Connection to the video media data

A metadata sample may store a data structure for each NAL unit in the media data stream. The metadata sample must be temporally aligned to the media data sample (in decoding time).

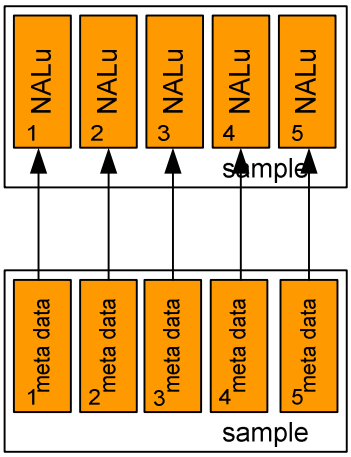


Figure D.1 —Connection between media data and metadata

NOTE As illustrated in Figure D.1 synchronization between the NAL units in the media data stream and the corresponding meta information is done by using the same structure in both streams (e.g. by counting NAL units and metadata statements).

D.3 SVC meta data sample entry

D.3.1 Definition

The SVC Metadata Sample Entry extends the Metadata Sample Entry and includes a configuration box which defines some default values for the samples and statements. It can also supply a statement which is valid for every sample described by this metadata sample entry, and a mapping of user extension statements to URIs.

The following example sample entry field values demonstrate different default possibilities:

sample_statement_type = groupOfStatements, default_statement_type=0, default_statement_length=0
— this is the 'normal' case, a group of statements about the entire sample

sample_statement_type = sequenceOfStatements, default_statement_type=0, default_statement_length=0
— where no statement needs to be made about the overall sample

sample_statement_type = sequenceOfStatements, default_statement_type=NALHeaderStatement, default_statement_length=N
— compact samples consisting merely of NAL headers of length N, for each aligned NAL unit

If priority override statements are used, then a priority assignment box may be present, providing the names of the methods referenced. When the box occurs here, the method_count may be greater than 1.

D.3.2 Syntax

```
class statement (default_type, default_length) {
   int st_type, field_size = 0;     // local variables, not fields
   int st_length = 0, body_len = 0; // local variables, not fields

   if (default_type != 0) st_type = default_type;
   else {
         unsigned int(8)   statement_type;
         st_type    = statement_type;
         st_length  = 1;
   }

   if (default_length != 0) body_len = default_length;
   else {
      if   (st_type >= 192) field_size = 4;
      else if (st_type > 0) field_size = 1;
      else  { body_len = 0; field_size = 0; }
      if (field_size > 0) {
         unsigned int(8*field_size) statement_length;
         body_len = statement_length;
         st_length += field_size;
      }
   }

   unsigned int(8) statement_body[body_len];
   st_length += body_len;
}

class SVCMetadataSampleConfigBox extends FullBox('svmC')
{
   int i;      // local variable, not a field
   unsigned int(8) sample_statement_type;   /* normally group, or seq */
   unsigned int(8) default_statement_type;
   unsigned int(8) default_statement_length;
   unsigned int(8) entry_count;
   for (i=1; i<=entry_count; i++) {
      unsigned int(8) statement_type;    // from the user extension ranges
      string statement_namespace;
   }
   statement(0,0) overall_statement; /* NB may be the empty statement, type==0 */
}
```

```
class SVCPriorityLayerInfoBox extends Box('qlif'){
   unsigned int(8) pr_layer_num;
   for(j=0; j< pr_layer_num; j++){
      unsigned int(8) pr_layer;
      unsigned int(24) profile_level_idc;
      unsigned int(32) max_bitrate;
      unsigned int(32) avg_bitrate;
   }
}

class SVCMetadataSampleEntry () extends MetadataSampleEntry('svcM')
{
   SVCMetadataSampleConfigBox config;
   SVCPriorityAssignmentBox   methods;      // optional
   SVCPriorityLayerInfoBox    priorities;   // optional
}

class MetaDataSample {
   int totalLength;     // local variable, not a field
   for (totalLength = 0; totalLength<sample_size; ) {
      statement(default_statement_type, default_statement_length) the_statement;
      totalLength += the_statement.st_length;
   }
}
```

D.3.3 Semantics

`statement_type` - an integer identifying a statement type defined in this International Standard, or dynamically defined in the corresponding sample entry of this track. In the SVCMetaDataSampleEntry this entry also defines the namespace mapping for this statement type of a dynamic statement. When used to define a mapping, the statement_type must have a value taken from the ranges reserved for user extensions.

`statement_length` - the length in bytes of the statement body, not including the statement_type and statement_length fields.

`statement_namespace` - gives a valid URI, in null-terminated UTF-8 characters; if a URL, it should contain a month in the form yyyymm, defined or approved by the owner of the domain name in that URL as of the month indicated.

`statement_body` - the contents of the statement, as defined by the statement type

`sample_statement_type` - describes whether the collection of statements in each sample associated with this sample entry is either a group (each describing the whole sample) or a sequence (each mapped to a NAL unit of the sample), and therefore normally takes the value groupOfStatements or sequenceOfStatements.

`default_statement_type` - in the case where all the first-level statements in all the associated samples have the same type, that type can be supplied here and then not be present in each sample; this would normally only be used when the sample_statement_type is sequenceOfStatements. If no default is needed, then 0 should be supplied in this field.

`default_statement_length` - in the case where all statements in all the associated samples have the same length, that length can be supplied here and then not be present in each sample. If no default is needed, then 0 should be supplied in this field.

`pr_layer_num` specifies the number of the priority layer.

`pr_layer` specifies the identifier of the priority layer. Priority layer identifiers are unique across the stream that is mapped to this metadata stream.

`profile_level_idc` specifies the profile and level compliancy of the bitstream of the priority layer identified by pr_layer. profile_level_idc is the exact copy of the three bytes comprised of profile_idc, constraint_set0_flag, constraint_set1_flag, constraint_set2_flag, constraint_set3_flag and level_idc, if

these syntax elements were used to specify the profile and level compliancy of the bitstream of the priority layer.

`max_bitrate` specifies the maximum bit rate, in units of 1000 bits per second, of the bitstream of the priority layer identified by pr_layer in any one-second time window.

`avg_bitrate` specifies the average bit rate, in units of 1000 bits per second, of the bitstream of the priority layer identified by pr_layer.

D.4 Helper Functions

```
function next_NALu (){
      // the next statement is made about the next NAL unit
}
```

D.5 Statement Types

The following statement types, and their required contents, are defined.

0 emptyStatement: when nothing needs to be said about the thing described (allows skipping of an item in a sequence)

192 groupOfStatements: the contents of the statement are exactly a set of statements, all of which apply to the described sample or NAL unit

```
class groupOfStatementsBody(size) {
   int i=0;
   do {
      statement(0,0) the_statement;
      i+=the_statement.st_length;
   } while (i < size);
}
```

193 sequenceOfStatements: the contents of the statement are exactly a set of statements, which describe, in one-to-one correspondence, the contents of a sample, extractor or aggregator (except note that the inlinesequenceOfStatements corresponds to one or more NAL units)

```
class sequenceOfStatementsBody(size) {
   int i=0;
   do {
      statement(0,0) the_statement;
      i+=the_statement.st_length;
      next_NALu();
   } while (i < size);
}
```

194 sequenceOfFixedStatements: same semantics as for sequenceOfStatements, except that the contents of the statement are a one-byte statement type, followed by a one-byte length indication; then follows a number of statements of that single type, each of the same length. The number of statements following is given by (statement_length-2)/fixedLength. The fixedLength shall be greater than zero.

```
class sequenceOfFixedStatementsBody(size) {
   int i=2;
   unsigned int(8) fixedType;
   unsigned int(8) fixedLength;
   do {
      statement(fixedType, fixedLength) the_statement;
      i+=fixedLength;
      next_NALu();
   } while (i < size);
}
```

195 inlineGroupOfStatements; this structure can be used to describe a consecutive set of items (NAL units). This structure starts with a one-byte count of the number of items described, and then describes all these items together with a number of statements (like a groupOfStatements). If individual statements about each item are also desired, a sequenceOfStatements may be included in the inlineGroupOfStatements to describe the items individually. In this case the included sequenceOfStatements shall describe as many items as specified by the value of count.

```
class inlineGroupOfStatementsBody(size) {
   int i=1;
   unsigned int(8) count;
   do {
      statement(0,0) the_statement;
      i+=the_statement.st_length;
   } while (i < size);
   for (j=0; j<count; j++)
      next_NALu();
}
```

1 itemLengthStatement: 1, 2, or 4 bytes of payload containing the length of the corresponding item (sample, group, or NAL unit)

2 aggregatorStatement: indicates that the item described is an Aggregator. If contained in a groupOfStatements about the Aggregator, the aggregatorStatement shall be the first statement in the groupOfStatements. The groupOfStatements may contain additional statements about the whole aggregation and a sequenceOfStatements to individually describe the NAL units aggregated. An aggregatorStatement contains no body.

3 extractorStatement: indicates that the item described is an Extractor. If contained in a groupOfStatements about the Extractor, the extractorStatement shall be the first statement in the groupOfStatements. The groupOfStatements may contain additional statements about all NAL units referenced and a sequenceOfStatements to individually describe the NAL units referenced. An extractorStatement contains no body.

4 overridePriorityStatement:

```
class overridePriorityStatementBody(size) {
   unsigned int(8) priority_assignment_method_index;
   bit(2) reserved = 0;
   bit(6) priority_id;
}
```

Contains a value for priority_id which may replace the priority_id value in the NAL unit header of the corresponding NAL unit. The field priority_assignment_method_index identifies the method used to calculate the priorities, as a 1-based index into the priority assignment URI box in the metadata sample entry (if any). There may be more than one of these statements for a given NAL unit; if there are several, they must differ in the value of the priority_assignment_method_index. If a stream is logically or physically re-labelled with priority_id values from these statements, then the same method must be used for all NAL units. If, for a given NAL unit, the priority_id value desired for a given method is the same as the value in the bitstream, then this statement may be omitted for that method for that NAL unit.

5 priorityRangeStatement: This is used when multiple NAL units are described. This contains two bytes, each containing a priority value in their lower 6 bits. The first is the lowest P value and the second the highest in the matching NAL units.

```
class priorityRangeStatementBody(size) {
   bit(2) reserved = 0;
   bit(6) min_priority_id;
   bit(2) reserved = 0;
   bit(6) max_priority_id;
}
```

6 DTQrangeStatement: This is used when multiple NAL units are described. The fields are defined exactly as for the SVCDependencyRangeBox.

```
class DTQRangeStatementBody(size) {
   bit(3) min_dependency_id;
   bit(3) min_temporal_id;
   bit(6) reserved = 0;
   bit(4) min_quality_id;
   bit(3) max_dependency_id;
   bit(3) max_temporal_id;
   bit(6) reserved = 0;
   bit(4) max_quality_id;
}
```

7 ROIindicationStatement: applies to a set of NAL units and gives IROI information.

```
class iroiStatementBody() {
   unsigned int(16) tierID;
   unsigned int(24) roi_id;
}
```

`tierID` specifies the tier the `roi_id` is defined in.

`roi_id` gives the ID of the ROI to which the NAL units pertaining to this statement belong.

8 scalabilityInfoStatement. This statement contains the header information from the matching NAL unit. The syntax is below, and the fields are as defined in ISO/IEC 14496-10 Annex G for NAL unit headers with NAL unit header SVC extension. The first byte is taken from the matching NAL unit, and the remaining bytes also if it is an SVC VCL NAL unit. If it is an AVC NAL unit, the remaining bytes are taken from the prefix NAL unit, if any, or else filled with zeroes.

```
class scalabilityInfoStatementBody() {
   bit (1)  forbidden_zero_bit;
   bit (2)  nal_ref_idc;
   bit (5)  nal_unit_type;
   bit (1)  reserved_zero_one_bit;
   bit (1)  idr_flag;
   bit (6)  priority_id;
   bit (1)  no_inter_layer_pred_flag;
   bit (3)  dependency_id;
   bit (4)  quality_id;
   bit (3)  temporal_id;
   bit (1)  use_ref_base_pic_flag;
   bit (1)  discardable_flag;
   bit (1)  output_flag;
   bit (2)  reserved_three_2bits;
}
```

9 temporalLayerSwitchingDistanceStatement. This statement provides a smaller value than that supplied in the group for tl_switching_distance for the tier, for switching points where the maximum value indicated in the tier is not needed.

```
class TLSwitchingDistanceStatementBody() {
   unsigned int(8) groupID;
   unsigned int(8) alt_tl_switching_distance;
}
```

`alt_tl_switching_distance` specifies a smaller value than tl_switching_distance in the group for the tier that applies to the target samples of the current statement.

10 priorityLayerStatement. This statement provides the priority layer to which the corresponding NAL unit(s) are mapped. Decoding can be performed at consistent quality by selecting the NAL units that are at, or below, a given priority layer. The body is a single 8-bit integer, the priority layer identifier. The characteristics of the priority layer are given in the optional SVCPriorityLayerInfoBox in the metadata sample entry.

```
class PriorityLayerStatementBody() {
   unsigned int(8) priorityLayer;
}
```

`prorityLayer` specifies the identifier of the priority layer the NAL unit(s) are mapped to. These identifiers shall be unique across the stream that is mapped to this metadata stream, i.e., a same value shall not be reused by NAL units with different dependency_id values.

Annex E
(normative)

File format toolsets

E.1 Introduction

This Annex defines what constitutes tools, for the purposes of branding files containing AVC or SVC content. A specific brand may require some or all of the tools indicated here. A brand should be chosen that indicates the full level of support required, including any requirements on other specifications (e.g. support for aspects of the ISO base media file format specification, ISO/IEC 14496-12).

E.2 SVC Toolsets

For all these toolsets the implementation of the SVC specific definitions from Annex A are required.

The following toolsets are defined:

- *SVCExtractor*: this toolset includes Extractors (Annex B).
- *SVCAggregator:* this toolset includes Aggregators (Annex B)).
- *SVCTiers*: this toolset includes map/group/tier implementation (Annex C).
- *SVCTimedMetaData*: this toolset includes the techniques from Annex D.

NOTE The SVCTiers and the SVCTimedMetaData toolsets define descriptive tools; if the file reader does not need this information, these toolsets need not be implemented as the video data can be processed without them. Extractors and Aggregators, however, are in-stream structures and must be implemented under some circumstances to yield the correct video stream. A brand requiring SVCTiers or SVCTimedMetaData might also need to require SVCAggregator.

E.3 MVC Toolsets

For all these toolsets the implementation of the MVC specific definitions from Annex F are required.

The following toolsets are defined:

- *MVCExtractor*: this toolset includes Extractors (Annex B).
- *MVCAggregator:* this toolset includes Aggregators (Annex B).
- *MVCTiers*: this toolset includes map/group/tier implementation (Annex C).
- *MVCTimedMetaData*: this toolset includes the techniques from Annex D.

NOTE The MVCTiers toolset defines descriptive tools; if the file reader does not need this information, this toolset need not be implemented as the video data can be processed without it. Extractors and Aggregators, however, are in-stream structures and must be implemented under some circumstances to yield the correct video stream. A brand requiring MVCTiers might also need to require MVCAggregator.

Annex F
(normative)

MVC elementary stream and sample definitions

F.1 Introduction

This Annex specifies the storage format of MVC data. It extends the definitions of the storage format of AVC in clause 5.

The support for MVC includes a number of tools, and there are various 'models' of how they might be used. In particular, an MVC stream can be placed in tracks in a number of ways, among which are the following:

1. all the views in one track, labelled with sample groups;
2. each view in its own track, labelled in the sample entries;
3. a hybrid, one track containing all views, and one or more single-view tracks each containing a view that can be independently coded;
4. the expected operating points each in a track (e.g. the AVC base, a stereo pair, a multiview scene).

The MVC file format allows storage of one or more views into a track, similarly to the support for SVC in Annex A. Storage of multiple views per track can be used, e.g., when a content provider wants to provide a multiview bitstream that is not intended for subsetting or when the bitstream has been created for a few pre-defined sets of output views (such as 1, 2, 5, or 9 views) where tracks can be created accordingly. If more than one view is stored in a track and there are several tracks (more than one) representing the MVC bitstream, the use of the sample grouping mechanism is recommended. The sample grouping mechanism is used to define tiers identifying the views present in the track and to extract required NAL units for certain operation points conveniently. The sample grouping mechanism is usually used with aggregator NAL units to form regular NAL unit patterns within samples. Thus, SVC-like sample grouping, aggregators, and view definitions for sample groups are specified for MVC.

The Multiview Information box ('mvci') is specified to indicate information that applies to more than one view, such as the target output views in one or more Multiview Group boxes. Characteristics (such as camera parameters) of the respective bitstream subset can also be indicated within the Multiview Group box using the Multiview Relation Attributes box ('mvra'), which is similar to the Track Selection box.

A player should have means to determine which views are preferred for displaying, and select one or more tracks that provide the data for the desired operating point, preferring a track that is specific to that operating point over tracks that also contain other data. The display characteristics of players may differ; for example, the number of simultaneously displayed views and the optimal angle between views can be different. In order to guide a player for selection of output views, alternative groups of output views and the common and differentiating characteristics between them can be indicated with the Multiview Group Relation box ('swtc'), which also includes the Multiview Relation Attributes box ('mvra').

When an MVC bitstream is represented by multiple tracks and a player uses an operating point that contains data in multiple tracks, the player must reconstruct MVC access units before passing them to the MVC decoder. An MVC operating point may be explicitly represented by a track, i.e., an access unit is reconstructed simply by resolving all extractor and aggregator NAL units of a sample. If the number of operating points is large, it may be space-consuming and impractical to create a track for each operating point. In such a case, MVC access units are reconstructed as specified in F.7.2. The MVC Decoder Configuration record contains a field indicating whether the associated samples use explicit or implicit access unit reconstruction (see the explicit_au_track field).

F.2 Terms and Definitions

For the purpose of storage of MVC elementary streams, the following terms and definitions apply. The definitions in ISO/IEC 14496-10 (including Annexes G and H) also apply.

The terms and definitions in A.2 also apply here, with the exception of the following, that are redefined for the purpose of storage of MVC elementary streams.

F.2.1
MVC VCL NAL unit
NAL units with type 20, and NAL units with type 14 when the immediately following NAL units are AVC VCL NAL units. MVC VCL NAL units do not affect the decoding process of a legacy AVC decoder.

F.2.2
operating point
subset of a bitstream associated with a set of target output views

NOTE 1 The bitstream subset of an MVC operating point represents a particular set of target output views at a particular temporal resolution, and consists of all the data needed to decode this particular bitstream subset.

NOTE 2 An operating point is referred to as an operation point in Annex H of ISO/IEC 14496-10. The term operating point is used in this Annex because it has also been used in other parts of this International Standard.

F.2.3
tier
set of operating points within a track, providing information about the operating points and instructions on how to access the corresponding bitstream portions (using maps and groups)

NOTE A tier represents a particular set of temporal subsets of a particular set of views.

F.2.4
virtual base view
AVC compatible representation of an independently coded non-base view

NOTE The virtual base view of an independently coded non-base view is created according to the process specified in H.8.5.5 of ISO/IEC 14496-10. Samples containing data units of an independently coded non-base view and samples of the virtual base view are aligned by decoding times.

F.3 Overview of MVC Storage

The storage of MVC streams can be supported by a number of structures, including information in the sample entry, the media information box, and sample groups. The following table provides an overview of the structures provided, their names, and a brief description of their functions.

NOTE – each group of rows starting with an entry in the left column (e.g. ‘minf’, ‘?vc?’) document a containment structure within that container; the higher level containment is not shown.

Table F.1 — Box, sample entry and group structures for MVC streams

				Box Name	Brief Description
minf				Media Information Box	
	mvci			Multiview Information Box	
		mvcg		Multiview Group Box	Specifies a multiview group for the views of the multiview video stream that are output
			buff	Buffering Information Box	Contains the buffering information of the bitstream subset specified by the multiview group
			mvra	Multiview Relation Attribute Box	Indicates the relation of the tracks or tiers of the respective multiview group with each other (when contained in a Multiview Group box)
			tibr	Tier Bit Rate Box	Provides information about the bit rate values of the bitstream subset specified by the multiview group
			tiri	Tier Information Box	Provides information about the profile, level, frame size, discardability, and frame-rate of the bitstream subset specified by the multiview group
			vwdi	Multiview Scene Information Box	Indicates the maximum disparity in a scene with multiple views
		swtc		Multiview Group Relation Box	Specifies a set of multiview groups from which one multiview group is decoded and played at any time
			mvra	Multiview Relation Attribute Box	Indicates the relation of the multiview groups with each other (when contained in a Multiview Group Relation box)
?vc?				Sample Entry	(Note: various codes are used for sample entries)
	ecam			Extrinsic Camera Parameters Box	Contains camera parameters that define the location and orientation of the camera reference frame with respect to a known world reference frame
	icam			Intrinsic Camera Parameters Box	Contains camera parameters that link the pixel coordinates of an image point with the corresponding coordinates in the camera reference frame
	vwid			View Identifier Box	Indicates the views included in the track (when included in a sample entry)
	mvcP			MVC View Priority Assignment Box	Provides a URI containing a unique name of the method used to assign content_priority_id values for the View Priority sample grouping
	mvcC			MVC Configuration Box	
sgpd				Sample Group Description Box	
	mvif			Multiview Group Entry	Contains the following boxes
		buff		Buffering Information Box	Contains the buffer information of the tier
		ldep		Tier Dependency Box	Identifies the tiers that the current tier is dependent on
		svip		Initial Parameter Sets Box	Contains parameter sets needed for decoding this tier and all the tiers it depends on
		svpr		Priority Range Box	Reports the minimum and maximum priority_id of the NAL units mapped to this tier
		tibr		Tier Bit Rate Box	Provides information about the bit rate values of a tier
		tiri		Tier Information Box	Provides information about the profile, level, frame size, discardability, and frame-rate of a tier
		vipr		View Priority Box	Labels views with priorities based on content
		vwid		View Identifier Box	Indicates the views included in the tier (when included in a Multiview Group entry,)
	dtrt			Decode Re-timing Group Entry	Provides adjusted decoding times when high temporal layers are discarded
	scnm			Sample Map Group Entry	Provides the mapping of NAL units to multiview groups for all samples in the track

The structures within a sample entry provide information for the decoding or use of the samples (video information) that are associated with that sample entry. Sample groups provide time-varying information about the track as a whole, assisting (for example) with the extraction of subsets of the media within a track. Information in the Multiview Information Box (appearing in the media information box) can span several tracks and is descriptive of collections of tracks, even though the Multiview Information Box resides in the track containing the base view of the stream.

F.4 MVC Track Structure

MVC streams are stored in accordance with 5.1, with the following definition of an MVC video elementary stream:

- **MVC Video Elementary Streams** shall contain all video coding related NAL units (i.e. those NAL units containing video data or signalling video structure) and may contain non-video coding related NAL units such as SEI messages and access unit delimiter NAL units. Also Aggregators (see B.2) or Extractors (see B.3) may be present. Aggregators and Extractors shall be processed as defined in this International Standard (e.g. shall not directly be placed in the output buffer while accessing the file). Other NAL units that are not expressly prohibited may be present, and if they are unrecognized they should be ignored (e.g. not placed in the output buffer while accessing the file).

MVC streams may also be stored using associated parameter set streams, when needed.

For MVC streams, Table 1 is amended by Table F.2.

Table F.2 — NAL Unit Types in MVC and AVC Streams

Value of nal_unit_type	Description	AVC video elementary stream	MVC video elementary stream	Parameter set elementary stream
14	Prefix NAL unit prefix_nal_unit_rbsp( )	Not specified	Yes	No
15	Subset sequence parameter set subset_seq_parameter_set_rbsp( )	Not specified	Yes	Yes
20	Coded slice extension slice_layer_extension_rbsp( )	Not specified	Yes	No
24 – 29	Not specified	Not specified	Not specified	Not specified
30	Aggregator	Not specified	Yes	No
31	Extractor	Not specified	Yes	No

There may be AVC VCL NAL units, MVC VCL NAL units and other NAL units, i.e. non-VCL NAL units, present in an MVC video elementary stream. Additionally, there may be Aggregator or Extractor NAL units present in an MVC video elementary stream.

An AVC VCL NAL unit in an MVC video elementary stream conforming to one or more profiles specified in Annex H of ISO/IEC 14496-10 shall be immediately preceded by a prefix NAL unit. In this file format an AVC VCL NAL unit and the immediately preceding prefix NAL unit are logically seen as one NAL unit.

F.5 Use of the plain AVC File Format

The MVC file format is an extension of the plain AVC file format defined in this International Standard.

5.3.12 is defined for use with plain AVC streams. Its use with MVC streams is deprecated.

F.6 Sample and configuration definition

F.6.1 Introduction

MVC Sample: An MVC sample consists of one or more view components as defined in Annex H of ISO/IEC 14496-10 and the associated non-VCL NAL units.

F.6.2 Canonical Order and Restriction

F.6.2.1 Restrictions

The following restrictions apply to MVC data in addition to the requirements in 5.2.2.

- **MVC coded slice NAL units** (Coded slice extension): All MVC coded slice NAL units for a single instant in time shall be contained in the sample whose composition time is that of the picture represented by the access unit. An MVC sample shall contain at least one AVC or MVC VCL NAL unit.

- **Prefix NAL units**: Each prefix NAL unit is placed immediately before the corresponding AVC VCL NAL unit.

- **Aggregators/Extractors**: The order of all NAL units included in an Aggregator or referenced by an Extractor is exactly the decoding order as if these NAL units were present in a sample not containing aggregators or extractors. After processing the Aggregator or the Extractor, all NAL units must be in valid decoding order as specified in ISO/IEC 14496-10.

F.6.2.2 Decoder Configuration Record

When the decoder configuration record defined in 5.2.4.1 is used for a stream that can be interpreted as either an MVC or AVC stream, the AVC decoder configuration record shall reflect the properties of the AVC compatible base view, e.g. it shall contain only parameter sets needed for decoding the AVC base view.

A parameter set stream may be used with MVC streams, as with AVC streams. In that case, parameter sets shall not be included in the decoder configuration record.

Sequence parameter sets, including subset sequence parameter sets, are numbered in order of storage from 1 to `numOfSequenceParameterSets` or `numOfPictureParameterSets`, respectively. Sequence and picture parameter sets stored in this record in a file may be referenced using this 1-based index by the `InitialParameterSetBox`.

The `MVCDecoderConfigurationRecord` is structurally identical to an `AVCDecoderConfigurationRecord`. However, the reserved bits preceding and succeeding the `lengthSizeMinusOne` field are re-defined. The syntax is as follows:

```
aligned(8) class MVCDecoderConfigurationRecord {
   unsigned int(8) configurationVersion = 1;
   unsigned int(8) AVCProfileIndication;
   unsigned int(8) profile_compatibility;
   unsigned int(8) AVCLevelIndication;
   bit(1) complete_representation;
   bit(1) explicit_au_track;
   bit(4) reserved = '1111'b;
   unsigned int(2) lengthSizeMinusOne;
   bit(1) reserved = '0'b;
   unsigned int(7) numOfSequenceParameterSets;
   for (i=0; i< numOfSequenceParameterSets; i++) {
      unsigned int(16) sequenceParameterSetLength ;
      bit(8*sequenceParameterSetLength) sequenceParameterSetNALUnit;
   }
   unsigned int(8) numOfPictureParameterSets;
   for (i=0; i< numOfPictureParameterSets; i++) {
      unsigned int(16) pictureParameterSetLength;
      bit(8*pictureParameterSetLength) pictureParameterSetNALUnit;
   }
}
```

The semantics of the fields `AVCProfileIndication`, `profile_compatibility`, and `AVCLevelIndication` differ from the `AVCDecoderConfigurationRecord` as follows:

The fields `AVCProfileIndication`, `AVCLevelIndication` carry the profile and level indications, respectively, indicating the profile and level for the bitstream represented by this track, i.e., the bitstream that contains all the views of this track and the views required for decoding of this track and wherein all the views in this track are the target output views. If `AVCLevelIndication` is equal to 0, the level that applies to the bitstream defined above operating with all the views of this track being the target output views is unspecified. `AVCProfileIndication`, `profile_compatibility`, and `AVCLevelIndication`, if non-zero, must have values such that a conforming MVC decoder is able to decode bitstreams conforming to the profile, level and profile compatibility flags indicated in any of the sequence parameter sets or subset sequence parameter sets contained in this record.

The semantics of other fields are as follows, or, if not present in the following, are as defined for an `AVCDecoderConfigurationRecord`:

`complete_representation` is set on a minimal set of tracks that contain a portion of the original encoded stream, as defined in F.7.1. Other tracks may be removed from the file without loss of any portion of the original encoded bitstream, and, once the set of tracks has been reduced to only those in the complete subset, any further removal of a track removes a portion of the encoded information.

`explicit_au_track` is set on a track that is 'complete'; it is not necessary to determine the view dependencies, nor calculate whether views not present in this track must be found from other tracks. However, subject to the rules for the sample entry types, extractors may be present and need to be followed to gather all the NAL units needed.

`numOfSequenceParameterSets` indicates the number of SPSs and subset SPSs that are used for decoding the MVC elementary stream.

`SequenceParameterSetLength` indicates the length in bytes of the SPS or subset SPS NAL unit.

`SequenceParameterSetNALUnit` contains a SPS or subset SPS NAL unit. SPSs shall occur in order of ascending parameter set identifier with gaps being allowed. Subset SPSs shall occur in order of ascending parameter set identifier with gaps being allowed. Any SPS shall occur before all the subset SPSs, if any.

F.6.3 Sync Sample (IDR)

A sync sample identifies the presence of an IDR access unit of the MVC bitstream for any sample entry that includes an MVC configuration record.

F.6.4 Shadow sync

A shadow sync box shall not be used for video data described by any MVC sample entry. Its use for AVC is deprecated.

F.6.5 Independent and disposable samples box

If it is used in a track which is both AVC and MVC compatible, then care should be taken that the statements are true no matter what valid subset of the MVC data (possibly only the AVC data) is used. The 'unknown' values (value 0 of the fields sample-depends-on, sample-is-depended-on, and sample-has-redundancy) may be needed if the information varies.

F.6.6 Random access recovery points

For video data described by a sample entry of type 'avc1' or 'avc2', the random access recovery sample group identifies random access recovery points for both an AVC decoder, and an MVC decoder (if any) operating on the entire bitstream.

For video data described by an MVC sample entry type, the random access recovery sample group identifies random access recovery in the entire MVC bitstream.

F.6.7 Hinting

Care should be taken when aggregators or extractors are in use and the track is hinted. These structures are defined only for use in the file format and should not be transmitted. In particular, a hint track that points at an extractor in a video track would cause the extractor itself to be transmitted (which is probably both incorrect and not the desired behaviour), not the data the extractor references. Hint tracks should normally directly reference NAL units specified in ISO/IEC 14496-10.

F.7 Derivation from the ISO base media file format

F.7.1 MVC track structure

A multi-view video stream is represented by one or more video tracks in a file. Each track represents one or more views of the stream.

There is a minimal set of one or more tracks that, when taken together, contain the complete set of encoded information. All these tracks shall have the flag "`complete_representation`" set in all their sample entries. This group of tracks that form the complete encoded information are called the "complete subset".

The track that has the flag "`complete_representation`" set and contains NAL units of the base view with temporal_id equal to 0 shall be nominated as the 'base view track'. All the other tracks that are part of the same stream shall be linked to this base track by means of a track reference of type '`sbas`' (view base). The complete encoded information can be retained when the tracks included in the "complete subset" are retained; all other tracks shall be subsets, copies or re-orderings of the complete subset.

All the tracks sharing the same base view track must share the same timescale.

If a view represented by a track uses another view represented by another track as an inter-view prediction reference, a track reference of type '`scal`' shall be included in the track referring to the source track for inter-view prediction.

If edits are applied to tracks that contain view components of an MVC bitstream, edit lists shall be consistent over all tracks affected by the edits.

> NOTE If a track containing a part of an MVC bitstream is removed from a file, care should be taken to remove also those tracks that contain `'scal'` and `'sbas'` track references to the removed track and references to the multiview groups that include the removed track.

F.7.2 Reconstruction of an access unit

In order to reconstruct an access unit from samples of one or more MVC tracks, the target output views may need to be determined first, by examining the Multiview Group box (F.8.3) and the Multiview Group Relation box (F.8.4). The `explicit_au_track` flag states that this track is a complete operating point; nonetheless, the track should be examined to determine which views delivered by this track are the output views.

If the target output views are not exactly represented by any track marked with `explicit_au_track` equal to 1 in the MVC decoder configuration record, access units are reconstructed as follows.

The views that are required for decoding the determined target output views can be concluded from reference view identifiers included in the View Identifier box, the '`scal`' track references, or Tier Dependency boxes.

If several tracks contain data for the access unit, the alignment of respective samples in tracks is performed on decoding time, i.e. using the time-to-sample table only without considering edit lists.

An access unit is reconstructed from the respective samples in the required tracks and tiers by arranging their NAL units in an order conforming to ISO/IEC 14496-10. The following order provides an outline of the procedure to form a conforming access unit:

- All parameter set NAL units (from the associated parameter set tracks and from the associated elementary stream tracks).
- All SEI NAL units (from the associated parameter set tracks and from the associated elementary stream tracks).
- View components in ascending order of view order index value. NAL units within a view component are in their appearance order within the sample.

F.7.3 Sample Entry

F.7.3.1 Boxes for Sample Entry

F.7.3.1.1 Intrinsic Camera Parameters Box

F.7.3.1.1.1 Definition

Box Type: `'icam'`
Container: Sample Entry (`'avc1', 'avc2', 'mvc1', 'mvc2'`)
Mandatory: No
Quantity: Zero or more

This subclause specifies intrinsic camera parameters that link the pixel coordinates of an image point with the corresponding coordinates in the camera reference frame. A specification of focal length and parameters related to the geometric distortion due to camera optics is given in Annex H of ISO/IEC 14496-10.

F.7.3.1.1.2 Syntax

```
class IntrinsicCameraParametersBox extends FullBox ('icam', version=0, flags) {
   unsigned int(6)   reserved=0;
   unsigned int(10)  ref_view_id;
   unsigned int(32)  prec_focal_length;
   unsigned int(32)  prec_principal_point;
   unsigned int(32)  prec_skew_factor;
   unsigned int(8)   exponent_focal_length_x;
   signed   int(64)  mantissa_focal_length_x;
   unsigned int(8)   exponent_focal_length_y;
   signed   int(64)  mantissa_focal_length_y;
   unsigned int(8)   exponent_principal_point_x;
   signed   int(64)  mantissa_principal_point_x;
   unsigned int(8)   exponent_principal_point_y;
   signed   int(64)  mantissa_principal_point_y;
   unsigned int(8)   exponent_skew_factor;
   signed   int(64)  mantissa_skew_factor;
}
```

F.7.3.1.1.3 Semantics

`reserved` this field shall be equal to zero

`ref_view_id` indicates the view_id identifying a view for which intrinsic camera parameters are indicated in this Intrinsic Camera Parameters Box

`prec_focal_length` specifies the exponent of the maximum allowable truncation error for focal_length_x and focal_length_y as given by $2^{-\text{prec\_focal\_length}}$. The value of `prec_focal_length` shall be in the range of 0 to 31, inclusive.

`prec_principal_point` specifies the exponent of the maximum allowable truncation error for principal_point_x and principal_point_y as given by $2^{-\text{prec\_principal\_point}}$. The value of `prec_principal_point` shall be in the range of 0 to 31, inclusive.

`prec_skew_factor` specifies the exponent of the maximum allowable truncation error for skew factor as given by $2^{-\text{prec\_skew\_factor}}$. The value of `prec_skew_factor` shall be in the range of 0 to 31, inclusive.

`exponent_focal_length_x` specifies the exponent part of the focal length in the horizontal direction. The value of `exponent_focal_length_x` shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified focal length.

`mantissa_focal_length_x` specifies the mantissa part of the focal length of the i-th camera in the horizontal direction.

`exponent_focal_length_y` specifies the exponent part of the focal length in the vertical direction. The value of `exponent_focal_length_y` shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified focal length.

`mantissa_focal_length_y` specifies the mantissa part of the focal length in the vertical direction.

`mantissa_principal_point_x` specifies the mantissa part of the principal point in the horizontal direction.

`exponent_principal_point_y` specifies the exponent part of the principal point in the vertical direction. The value of `exponent_principal_point_y` shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified principal point.

`mantissa_principal_point_y` specifies the mantissa part of the principal point in the vertical direction.

`exponent_skew_factor` specifies the exponent part of the skew factor. The value of `exponent_skew_factor` shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified skew factor.

`mantissa_skew_factor` specifies the mantissa part of the skew factor.

The intrinsic matrix A for the camera associated to the view indicated by `ref_view_id` is represented as follows:

$$\begin{bmatrix} \text{focalLengthX} & \text{skewFactor} & \text{principalPointX} \\ 0 & \text{focalLengthY} & \text{principalPointY} \\ 0 & 0 & 1 \end{bmatrix}$$

Each component of the intrinsic matrix is obtained from the variables specified in Table F.3 as the variable x computed as follows.

– If $0 < e < 63$, $x = 2^{e-31} * (1 + n \div 2v)$, with $v = \max(0, e + p - 31)$ [Eq. F-1]

– If e is equal to 0, $x = 2^{-(30+v)} * n$, with $v = \max(0, p - 30)$ [Eq. F-2]

Table F.3 – Association between camera parameter variables and syntax elements

x	e	n	p
focalLengthX	exponent_focal_length_x	mantissa_focal_length_x	prec_focal_length
focalLengthY	exponent_focal_length_y	mantissa_focal_length_y	prec_focal_length
principalPointX	exponent_principal_point_x	mantissa_principal_point_x	prec_principal_point
principalPointY	exponent_principal_point_y	mantissa_principal_point_y	prec_principal_point
skewFactor	exponent_skew_factor	mantissa_skew_factor	prec_skew_factor

F.7.3.1.2 Extrinsic Camera Parameters Box

F.7.3.1.2.1 Definition

Box Type: `'ecam'`
Container: Sample Entry (`'avc1'`, `'avc2'`, `'mvc1'`, `'mvc2'`)
Mandatory: No
Quantity: Zero or more

This subclause specifies extrinsic camera parameters that define the location and orientation of the camera reference frame with respect to a known world reference frame. A specification of extrinsic camera parameters including translation vector and rotation matrix is given in Annex H of ISO/IEC 14496-10.

The extrinsic camera parameters are specified according to a right-handed coordinate system, where the upper left corner of the image is the origin, i.e., the (0, 0) coordinate, with the other corners of the image having non-negative coordinates. With these specifications, a 3-dimensional world point, wP=[x y z] is mapped to a 2-dimensional camera point, cP = [u v 1], according to:

$$s * cP = A * R^{-1} * (wP – T)$$

where A denotes the intrinsic camera parameter matrix which can be indicated by an intrinsic camera parameters box (see F.7.3.1.1), R^{-1} denotes the inverse of the rotation matrix R, T denotes the translation vector, and s (a scalar value) is an arbitrary scale factor chosen to make the third coordinate of cP equal to 1. The elements of A, R, T are determined according the syntax elements signalled in this box and as specified below.

F.7.3.1.2.2 Syntax

```
class ExtrinsicCameraParametersBox extends FullBox ('ecam', version=0, flags) {
   unsigned int(6)   reserved=0;
   unsigned int(10)  ref_view_id;
   unsigned int(8)   prec_rotation_param;
   unsigned int(8)   prec_translation_param;
   for (j=1; j<=3; j++) { /* row */
      for (k=1; k<=3; k++) { /* column */
         unsigned int(8)   exponent_r[j][k];
         signed   int(64)  mantissa_r [j][k];
      }
      unsigned int(8)   exponent_t[j];
      signed   int(64)  mantissa_t[j];
   }
}
```

F.7.3.1.2.3 Semantics

`reserved` this field shall be equal to zero

`ref_view_id` indicates the view_id identifying a view for which intrinsic camera parameters are indicated in this Intrinsic Camera Parameters Box

`prec_rotation_param` specifies the exponent of the maximum allowable truncation error for r[j][k] as given by $2^{-\text{prec\_rotation\_param}}$. The value of `prec_rotation_param` shall be in the range of 0 to 31, inclusive.

`prec_translation_param` specifies the exponent of the maximum allowable truncation error for t[j] as given by $2^{-\text{prec\_translation\_param}}$. The value of `prec_ translation_param` shall be in the range of 0 to 31, inclusive.

`exponent_r[j][k]` specifies the exponent part of (j, k) component of the rotation matrix. The value of `exponent_r[j][k]` shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified rotation matrix.

`mantissa_r[j][k]` specifies the mantissa part of (j, k) component of the rotation matrix.

`exponent_t[j]` specifies the exponent part of the j-th component of the translation vector. The value of `exponent_t[j]` shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified translation vector.

`mantissa_t[j]` specifies the mantissa part of the j-th component of the translation vector.

The rotation matrix R is represented as follows:

$$\begin{bmatrix} rE[0][0] & rE[0][1] & rE[0][2] \\ rE[1][0] & rE[1][1] & rE[1][2] \\ rE[2][0] & rE[2][1] & rE[2][2] \end{bmatrix}$$

The translation vector T is represented as follows:

$$\begin{bmatrix} tE[0] \\ tE[1] \\ tE[2] \end{bmatrix}$$

Each component of the rotation matrix and the translation vector is obtained from the variables specified in Table F-4 as the variable x computed as follows.

– If $0 < e < 63$, $x = 2^{e-31} * (1 + n \div 2^{v})$, with $v = \max(0, e + p - 31)$ [Eq. F-3]

– If e is equal to 0, $x = 2^{-(30+v)} * n$, with $v = \max(0, p - 30)$ [Eq. F-4]

Table F.4 – Association between camera parameter variables and syntax elements

x	e	n	p
rE[j][k]	exponent_r[j][k]	mantissa_r[j][k]	prec_rotation_param
tE[j]	exponent_t[j]	mantissa_t[j]	prec_translation_param

F.7.3.1.3 View Identifier Box

F.7.3.1.3.1 Definition

Box Type: 'vwid'
Container: Sample Entry ('avc1', 'avc2', 'mvc1', 'mvc2') or MultiviewGroupEntry
Mandatory: Yes (for sample entries and the primary group definition in Multiview Group entries)
Quantity: Exactly one (for sample entries and the primary group definition in Multiview Group entries)
Zero for non-primary group definitions in Multiview Group entries

When included in a sample entry, this box indicates the views included in the track. When included in a Multiview Group entry, this box indicates the views included in the respective tier. This box also indicates the view order index for each listed view. Additionally, the box includes the minimum and maximum values of temporal_id included in the track or tier when the View Identifier box is included in a sample entry or Multiview Group entry, respectively. Moreover, the box indicates the referenced views required for decoding the views included in the track or tier.

F.7.3.1.3.2 Syntax

```
class ViewIdentifierBox extends FullBox ('vwid', version=0, flags)
{
   unsigned int(2) reserved6 = 0;
   unsigned int(3) min_temporal_id;
   unsigned int(3) max_temporal_id;
   unsigned int(16)  num_views;
   for (i=0; i<num_views; i++) {
      unsigned int(6) reserved1 = 0;
      unsigned int(10) view_id[i];
      unsigned int(6) reserved2 = 0;
      unsigned int(10) view_order_index;
      unsigned int(4)   reserved3 = 0;
      unsigned int(2)   base_view_type;
      unsigned int(10)  num_ref_views;
      for (j = 0; j < num_ref_views; j++) {
         unsigned int(6)   reserved5 = 0;
         unsigned int(10)  ref_view_id[i][j];
      }
   }
}
```

F.7.3.1.3.3 Semantics

`min_temporal_id, max_temporal_id` take the minimum and maximum value, respectively, of the temporal_id syntax element that is present in the NAL unit header extension of the NAL units mapped to the track or tier when the View Identifier box is included in a sample entry or Multiview Group entry, respectively. For AVC streams this takes the value that is, or would be, in the prefix NAL unit.

`num_views`, when the View Identifer box is present in a sample entry, indicates the number of views included in the track. When the View Identifier box is present in a Multiview Group entry, num_views indicates the number of views included in the respective tier.

`view_id[i]` indicates the value of the view_id syntax element in the NAL unit header extension of a view included in the track or tier when the View Identifier box is included in a sample entry or Multiview Group entry, respectively.

`view_order_index` indicates the value of the VOIdx variable, as specified in Annex H of ISO/IEC 14496-10, for a view included in the track or tier when the View Identifier box is included in a sample entry or Multiview Group entry, respectively.

`base_view_type` indicates whether the view is a base view (virtual or not). It takes the following values:

0 indicates that the view is neither a base view nor virtual base view.

1 shall be used to label the non-virtual base view of the MVC bitstream.

2 is a reserved value and shall not be used.

3 indicates that the view with view_id[i] is a virtual base view. The respective independently coded non-base view with view_id[i] resides in another track. When `base_view_type` is equal to 3, the subsequent num_ref_views shall be equal to 0.

`num_ref_views` indicates the number of views which are referenced by the view with view_id[i].

`ref_view_id[i][j]` indicates the view identifier of the j-th view that may be directly or indirectly referenced by the view with view_id[i], i.e., that may be required for decoding of the view with view_id[i]. If a view is required for decoding the view with view_id[i], it shall be listed as one of ref_view_id[i][j]. When the View Identifier box is included in a sample entry, it is recommended to indicate the referenced views for both anchor and non-anchor access units in the same sample entry.

F.7.3.2 Sample Entry Definition

Sample Entry Types: `'avc1', 'avc2', 'mvc1', 'mvc2'`
Container: Sample Description Box (`'stbl'`)
Mandatory: Either the avc1, avc2, mvc1 or mvc2 box is mandatory
Quantity: One or more sample entries may be present

If an MVC elementary stream contains a usable AVC compatible base view, then an AVC visual sample entry ('avc1' or 'avc2') shall be used. Here, the entry shall contain initially an AVC Configuration Box, possibly followed by an MVC Configuration Box as defined below. The AVC Configuration Box documents the Profile, Level and Parameter Set information pertaining to the AVC compatible base view as defined by the `AVCDecoderConfigurationRecord`. The MVC Configuration Box documents the Profile, Level and Parameter Set information pertaining to the entire stream containing the non-base views as defined by the `MVCDecoderConfigurationRecord`, stored in the MVCConfigurationBox.

For all sample entries 'avc1' and 'avc2', the width and height fields in the sample entry document the AVC base layer. For an 'mvc' sample entry ('mvc1' or 'mvc2'), the width and height document the resolution achieved by decoding any single view of the entire stream.

If the MVC elementary stream does not contain a usable AVC base view, then an MVC visual sample entry ('mvc1' or 'mvc2') shall be used. The MVC visual sample entry shall contain an MVC Configuration Box, as defined below. This includes an `MVCDecoderConfigurationRecord`, as defined in this International Standard.

The `lengthSizeMinusOne` field in the MVC and AVC configurations in any given sample entry shall have the same value.

A priority assignment URI provides the name (in the URI space) of a method used to assign priority_id values. When it occurs in an AVC or MVC sample entry, exactly one URI shall be present, that documents the priority_id assignments in the stream. The URI is treated here as a name only; it should be de-referenceable, though this is not required. File readers may be able to recognize some methods and thereby know what stream extraction operations based on priority_id would do.

The requirements for the sample entry types 'avc1' and 'avc2' as documented in A.6.3.1.1 also apply here.

Any of the boxes MVCConfigurationBox, ViewScalabilityInfoSEIBox, IntrinsicCameraParametersBox, and ExtrinsicCameraParametersBox may be present in an 'avc1' or 'avc2' sample entry. In these cases, the AVCMVCSampleEntry or AVC2MVCSampleEntry definition below applies, respectively. The ScalabilityInformationSEIBox is defined in A.6.3.1.2.

`Compressorname` in the base class `VisualSampleEntry` indicates the name of the compressor used, with the value `"\012MVC Coding"` being recommended (\012 is 10, the length of the string "MVC coding" in bytes).

The parameter sets required to decode a NAL unit that is present in the sample data of a video stream, either directly or by reference from an Extractor, shall be present in the decoder configuration of that video stream or in the associated parameter set stream (if used).

The following table shows for a video track all the possible uses of sample entries when an MVC elementary stream is stored in one or more tracks, configurations, and the MVC tools (excluding timed metadata, which is always used in another track).

Table F.5 — Use of sample entries for AVC and MVC tracks

sample entry name	with configuration records	Meaning
'avc1'	AVC and MVC Configurations	An MVC track with both AVC and MVC NAL units; Aggregators and Extractors may be present; Aggregators shall not contain but may reference AVC NAL units; Tier grouping may be present.
'avc2'	AVC and MVC Configurations	An MVC track with both AVC NAL units and MVC NAL units; Extractors may be present and used to reference both AVC and MVC NAL units; Aggregators may be present to contain and reference both AVC and MVC NAL units; Tier grouping may be present.
'mvc1'	MVC Configuration	An MVC track without a usable AVC base layer; Aggregators may be present to contain and reference both AVC and MVC NAL units; Tier grouping may be present.
'mvc2'	MVC Configuration	An MVC track without a usable AVC base layer; Extractors may be present and used to reference MVC NAL units; Aggregators may be present to contain and reference MVC NAL units; Tier grouping may be present.

The sample entry `mvc-type` in the following is one of `{mvc1, mvc2}`.

F.7.3.3 Syntax

```
class MVCConfigurationBox extends Box('mvcC') {
   MVCDecoderConfigurationRecord() MVCConfig;
}

class ViewScalabilityInformationSEIBox extends Box('vsib', size)
{
   unsigned int(8*size-64) viewscalinfosei;
}
```

```
class AVCMVCSampleEntry() extends AVCSampleEntry ('avc1'){
   AVCConfigurationBox  avcconfig;   // mandatory
   ViewScalabilityInformationSEIBox scalability;  // optional
   ViewIdentifierBox    view_identifiers;   // optional
   MVCConfigurationBox  mvcconfig;   // optional
   MVCViewPriorityAssignmentBox  view_priority_method;   // optional
   IntrinsicCameraParametersBox  intrinsic_camera_params;// optional
   ExtrinsicCameraParametersBox  extrinsic_camera_params;// optional
}

class AVC2MVCSampleEntry() extends AVC2SampleEntry ('avc2'){
   AVCConfigurationBox  avcconfig;   // mandatory
   ViewScalabilityInformationSEIBox scalability;  // optional
   ViewIdentifierBox    view_identifiers;   // optional
   MVCConfigurationBox  mvcconfig;   // optional
   MPEG4BitRateBox      bitrate;     // optional
   MPEG4ExtensionDescriptorsBox descr;  // optional
   MVCViewPriorityAssignmentBox  view_priority_method;   // optional
   IntrinsicCameraParametersBox  intrinsic_camera_params;// optional
   ExtrinsicCameraParametersBox  extrinsic_camera_params // optional
}

// Use this if the track is NOT AVC compatible
class MVCSampleEntry() extends VisualSampleEntry (mvc-type){
   MVCConfigurationBox  mvcconfig;      // mandatory
   ViewScalabilityInformationSEIBox scalability;  // optional
   ViewIdentifierBox view_identifiers;      // mandatory
   MPEG4BitRateBox bitrate;             // optional
   MPEG4ExtensionDescriptorsBox descr;      // optional
   MVCViewPriorityAssignmentBox  view_priority_method;   // optional
   IntrinsicCameraParametersBox  intrinsic_camera_params;// optional
   ExtrinsicCameraParametersBox  extrinsic_camera_params // optional
}
```

F.7.3.4 Semantics

`viewscalinfosei` contains an SEI NAL unit containing only a view scalability information SEI message as specified in ISO/IEC 14496-10 Annex H. The 'size' field of the container box ViewScalabilityInformationSEIBox shall not be equal to 0 or 1.

F.8 MVC specific information boxes

F.8.1 Introduction

The following boxes specify information that relate to more than one output view of an MVC elementary stream. As any subset of views of an MVC elementary stream can be chosen for output, the information carried in these boxes is not necessarily specific to any track and thus contained separately. The information can be specified for different groups of output views.

F.8.2 Multiview Information Box

F.8.2.1 Definition

Box Type: `mvci'
Container: Media Information Box (`minf')
Mandatory: No
Quantity: Zero or one

Located in the Media Information Box of the base view track indicated by the 'sbas' track reference, this box contains Multiview Group boxes, and Multiview Group Relation boxes.

F.8.2.2 Syntax

```
aligned(8) class MultiviewInformationBox
   extends FullBox('mvci', version = 0, flags) {
}
```

F.8.3 Multiview Group Box

F.8.3.1 Definition

Box Type: 'mvcg'
Container: Multiview Information box (`'mvci'`)
Mandatory: No
Quantity: Zero or more

This box specifies a multiview group for the views of the multiview video stream that are output. Target output views can be indicated on the basis of track_id, tier_id, or view_id. When the views included in a track match an operating point, it is recommended to use track_id (i.e., entry_type equal to 0) within the Multiview Group box. When multiview sample grouping is in use, and tiers cover more than one view or some tiers contain a temporal subset of the bitstream, it is recommended to use tier_id (i.e., entry_type equal to 1) within the Multiview Group box. Otherwise, it is recommended to use one of the view_id based indications (i.e., entry_type equal to 2 or 3).

Each view in a track or tier that is included in this box is a target output view. If a track or tier included in this box contains multiple views, all the contained views are target output views.

Decoding of the output views may require decoding of other views that are not target output views. The views that are required for decoding but are not target output views can be concluded from reference view identifiers included in the View Identifier box, the 'scal' track references, or from the Tier Dependency box.

If the box contains a `track_id` or `tier_id` that is not present or refers to a view_id of a view that is not present, the respective view should be considered removed and the multiview group should be ignored.

F.8.3.2 Syntax

```
aligned(8) class MultiviewGroupBox extends FullBox('mvcg', version = 0, flags) {
   unsigned int(32) multiview_group_id;
   unsigned int(16) num_entries;
   unsigned int(8) entry_type;
   for(i=0; i<num_entries; i++) {
      unsigned int(8) entry_type;
      if (entry_type == 0)
         unsigned int(32) track_id;
      else if (entry_type == 1) {
         unsigned int(32) track_id;
         unsigned int(16) tier_id;
      }
      else if (entry_type == 2) {
         unsigned int(6) reserved1 = 0;
         unsigned int(10) output_view_id;
      }
      else if (entry_type == 3) {
         unsigned int(6) reserved2 = 0;
         unsigned int(10) start_view_id;
         unsigned int(16) view_count;
      }
   }
   TierInfoBox subset_stream_info;        // optional
   MultiviewRelationAttributeBox relation_attributes; // optional
   TierBitRateBox subset_stream_bit_rate; // optional
   BufferingBox subset_stream_buffering;  // optional
   MultiviewSceneInfoBox multiview_scene_info;        // optional
}
```

F.8.3.3 Semantics

`multiview_group_id` provides a unique identifier for the multiview group within the file.

`num_entries` is the number of tracks and tiers included in this multiview group.

`entry_type` specifies how the target output views are indicated. The following values of entry_type are specified:

0 – all the views included in an indicated track are target output views

1 – the view(s) of an indicated tier within an indicated track are target output views

2 – the views with view_id equal to output_view_id are target output views

3 – the views having view_id within the range of start_view_id to (start_view_id + view_count – 1), inclusive, are target output views

`track_id` indicates a track containing target output views.

`tier_id` indicates a tier within a track where all views within the tier are target output views.

`output_view_id` indicates a view_id of a target output view.

`start_view_id` indicates the first view_id in a range of contiguous values of view_id all being target output views.

`view_count` indicates the number of contiguous values of view_id all being target output views.

`track_id` indicates a track.

`tier_id` indicates a tier within a track.

`subset_stream_info` indicates the characteristics of the bitstream subset containing the indicated output views and the views they depend on.

`relation_attributes` indicate the relations between output views. If 'ecam' is used as a common attribute, all the output views are associated with extrinsic camera parameters indicating that the cameras have identical rotation and constant spacing. If 'ecam' is used as a differentiating attribute, at least one output view is associated with extrinsic camera parameters with different rotation from the others or the output views are associated with extrinsic camera parameters not having a constant spacing.

`subset_stream_bit_rate` indicates the bit rate statistics of the bitstream subset containing the indicated output views and the views they depend on. The values of `tierBaseBitRate`, `tierMaxBitRate`, and `tierAvgBitRate` within the TierBitRateBox are unspecified.

`subset_stream_buffering` indicates the HRD parameters that apply to the bitstream subset containing the indicated output views and the views they depend on and operating with the indicated target output views.

`multiview_scene_info` contains the maximum disparity in units of integer pixel resolution between any spatially adjacent output views in any access unit.

F.8.4 Multiview Group Relation Box

F.8.4.1 Definition

Box Type: `'swtc'`
Container: Multiview Information box (`'mvci'`)
Mandatory: No
Quantity: Zero or more

This box specifies a set of multiview groups from which one multiview group is decoded and played at any time. The given relation attributes specify which features are common in all associated multiview groups and which factors make the multiview groups differ from each other. The relation attributes can be used to select a suitable set of multiview groups for playback, e.g., based on the number of output views. The differentiating attributes can be used to select which multiview group within the set is suitable for the player, e.g., based on the required level for decoding.

F.8.4.2 Syntax

```
aligned(8) class MultiviewGroupRelationBox() extends FullBox('swtc', version = 0,
flags) {
   unsigned int(32) num_entries;
   for (i=0; i<num_entries; i++)
      unsigned int(32) multiview_group_id;
   MultiviewRelationAttributeBox relation_attributes;
}
```

F.8.4.3 Semantics

`num_entries` indicates the number of associated multiview groups.

`multiview_group_id` is the identifier of an associated multiview group.

`relation_attributes` indicate the relations between the associated multiview groups.

F.8.5 Multiview Relation Attribute Box

F.8.5.1 Definition

Box Type: `mvra'
Container: MultiviewGroupBox or MultiviewGroupRelationBox
Mandatory: No in MultiviewGroupBox, Yes in MultiviewGroupRelationBox
Quantity: Zero or One in MultiviewGroupBox
One in MultiviewGroupRelationBox

When the Multiview Relation Attribute box is contained in a Multiview Group box, it indicates the relation of the output views of the respective multiview group with each other. When the Multiview Relation Attribute box is contained in a Multiview Group Relation box, it indicates the relation of the multiview groups with each other.

The Multiview Relation Attribute box contains common and differentiating attributes. When the Multiview Relation Attribute box is included in a Multiview Group box, a common attribute indicates a characteristic that is common for each one of the target output views of the multiview group and a differentiating attribute indicates a characteristic which differs in at least one of one of the target output views of the multiview group. When Multiview Relation Attribute box is included in a Multiview Group Relation box, a common attribute indicates a characteristic that is common for the indicated multiview groups or for the respective target output views in each one of the indicated multiview groups, whereas a differentiating attribute indicates a characteristic that differs in at least one of the indicated multiview groups or at least one of the respective target output views in the indicated multiview groups.

A common attribute is associated with an additional parameter, which carries the value of the common attribute. The syntax and semantics of the additional parameter depend on the attribute in question.

For example, a file writer can create a Multiview Group for each stereo pair suitable for display from a multiview bitstream. Furthermore, a file writer can create a Multiview Group Relation box listing all the multiview groups for stereo pair output and including a Multiview Relation Attribute box with common attributes number of views (equal to 2) and in-line camera arrangement. A file reader can study the Multiview Group Relation box to find the options for stereo pair output and choose one multiview group for processing. Note that the presence of views in a group does not necessarily imply they are all suggested as output views at any given time – the terminal may choose which views to output, and it is not limited by the group information.

F.8.5.2 Syntax

```
aligned(8) class MultiviewRelationAttributeBox
   extends FullBox('mvra', version = 0, flags) {
   unsigned int(16) reserved1 = 0;
   unsigned int(16) num_common_attributes;
   for (i=0; i<num_common_attributes; i++) {
      unsigned int(32) common_attribute;
      bit(32) common_value;
   }
   unsigned int(16) reserved2 = 0;
   unsigned int(16) num_differentiating_attributes;
   for (i=0; i<num_differentiating_attributes; i++)
      unsigned int(32) differentiating_attribute;
}
```

F.8.5.3 Semantics

`common_attribute` and `differentiating_attribute` are selected from the list below. Attributes that can be used as a differentiating attribute are associated with a distinguishing pointer to the field or information.

`common_value` specifies the value for the common attribute. Its syntax and semantics depend on the common attribute and are specified in the table below.

Name	*Attribute*	*Pointer and semantics*	*common_value syntax and semantics*
Profile	`'prfl'`	This attribute shall not be included in the Multiview Group box.	`unsigned int(24) reserved = 0;` `unsigned int(8) profileIndication;`
		When included in the Multiview Group Relation box, the attribute refers to the profile required for decoding the bitstream subset corresponding to the multiview group. The attribute points to the profileIndication field of the subset_stream_info element of the Multiview Group box.	profileIndication is the profile sufficient for decoding the bitstream subset corresponding to all indicated multiview groups.
Level	`'levl'`	This attribute shall not be included in the Multiview Group box.	`unsigned int(24) reserved = 0;` `unsigned int(8) levelIndication;`
		When included in the Multiview Group Relation box, the attribute refers to the level required for decoding the bitstream subset corresponding to the multiview group. The attribute points to the levelIndication field of the subset_stream_info element of the Multiview Group box.	profileIndication is the level sufficient for decoding the bitstream subset corresponding to all indicated multiview groups, or 0 if the level is unspecified.
Bitrate	`'bitr'`	This attribute shall not be included in the Multiview Group box.	`unsigned int(32) bitrate;`
		When included in the Multiview Group Relation box, the attribute refers to the total size of bitstream subset required for decoding of the multiview group divided by the duration in the track header box. The attribute points to the avgBitRate field of the subset_stream_bit_rate element of the Multiview Group box, if present, or a value that would be contained in the avgBitRate field of the subset_stream_bit_rate element of the Multiview Group box, if it were present.	bitrate indicates the average bit rate in bits per second of the bitstream subset required for decoding the multiview group. The bitrate may be rounded up.

Frame rate	'frar'	This attribute shall not be included in the Multiview Group box.
		When included in the Multiview Group Relation box, the attribute refers the number of samples in the track divided by duration in the track header box.
			`unsigned int(16) integer_part;`
			`unsigned int(16) reserved = 0;`
			integer_part shall be equal to the output rate of decoded access units in second rounded to the closest integer using the Round function specified in ISO/IEC 14496-10.

Number of output views	'nvws'	Number of target output views indicated in the Multiview Group Box ('mvcg') If this attribute is included in the Multiview Group box, it shall be a common attribute and merely documents the number of output views in the respective multiview group.	`unsigned int(32) num_views;` num_views indicates the number of views in the multiview group.
Intrinsic camera parameters	'icam'	The intrinsic camera parameters are stored in 'avc1', 'avc2', 'mvc1', or 'mvc2' Sample Entry (in Sample Description box of media track). If this attribute is included in the Multiview Group box and used as a common attribute, the intrinsic camera parameters of the target output views are identical. If this attribute is included in the Multiview Group box and used as a differentiating attribute, the intrinsic camera parameters of the target output views differ at least partly. If this attribute is included in the Multiview Group Relation box and used as a common attribute, the number of target output views in all indicated multiview groups shall be the same and the intrinsic camera parameters of the respective target output views in all indicated multiview groups are identical. If this attribute is included in the Multiview Group Relation box and used as a differentiating attribute, the intrinsic camera parameters of the respective target output views differ at least partly.	Unspecified.
Extrinsic camera parameters	'ecam'	The extrinsic camera parameters are stored in 'avc1', 'avc2', 'mvc1', or 'mvc2' Sample Entry (in Sample Description box of media track). If this attribute is included in the Multiview Group box and used as a common attribute, the rotation of the cameras for all the target output views is the same and, if the cameras are arranged in linear, elliptical, or rectangular arrangement, the distance of adjacent cameras is the same. If this attribute is included in the Multiview Group box and used as a differentiating attribute, the rotation or the distance of adjacent cameras in linear, elliptical, or rectangular arrangement differs. If the attribute is included in the Multiview Group Relation box and used as a common attribute, the relative extrinsic camera parameters target output views in all indicated multiview groups are identical. That is, the distance of cameras relative to each other and	Unspecified.

		their rotation matches in the indicated multiview groups. If the attribute is included in the Multiview Group Relation box and used as a differentiating attribute, the relative extrinsic camera parameters of respective target output views differ at least partly.	

Inline view array	'ilvi'	If used as a common attribute, the associated cameras are located on a straight line.
		When included in a Multiview Group box, the attribute shall be a common attribute.
			`unsigned int(28) reserved = 0;`
			`unsigned int(2) horizontal_order;`
			`unsigned int(2) vertical_order;`
			`horizontal_order` indicates the horizontal order of the views:
			0: the views are in the same horizontal location
			1: the views are ordered left-to-right
			2: the views are ordered right-to-left
			3: the order of the views is undefined, or left and right are not well-defined.
			`vertical_order` indicates the vertical order of the views:
			0: the views are in the same vertical location
			1: the views are ordered bottom-to-top
			2: the views are ordered top-to-bottom
			3: the order of the views is undefined, or top and bottom are not well-defined.

Rectangular view array	'rtvi'	If used as a common attribute, the associated cameras form a rectangular shape and are regularly spaced along the orthogonal coordinate axes.
		When included in a Multiview Group box, the attribute shall be a common attribute.
			`unsigned int(16) row_view_count;`
			`unsigned int(16) col_view_count;`
			`row_view_count` specifies the number of rows in the rectangular array.
			`col_view_count` specifies the number of columns in the rectangular array.
			The views are indicated in raster scan order in the Multiview Group box.

Planar view array	'plvi'	If used as a common attribute, the associated cameras are located on a plane, but may be irregularly spaced. When included in a Multiview Group box, the attribute shall be a common attribute.	Unspecified.

Elliptical view array	'elvi'	If used as a common attribute, the associated cameras are located on the arc of an ellipse.
		When included in a Multiview Group box, the attribute shall be a common attribute.
			`unsigned int(28) reserved = 0;`
			`unsigned int(2) horizontal_order;`
			`unsigned int(2) vertical_order;`
			The semantics are identical to those for the Inline view array.

Spherical view array	'spvi'	If used as a common attribute, the associated cameras are located on the surface of a sphere. When included in a Multiview Group box, the attribute shall be a common attribute.	Unspecified.

Stereo view array	'stvi'	If used as a common attribute, the associated cameras are a pair of views suitable for stereo viewing.
		When included in a Multiview Group box, the attribute shall be a common attribute.
			`unsigned int(6) reserved1 = 0;`
			`unsigned int(10) left_view_id;`
			`unsigned int(6) reserved2 = 0;`
			`unsigned int(10) right_view_id;`

Geometry	'geom'	If used as a differentiating attribute, indicates that the views or groups of views belong to different view arrangements (e.g. inline, planar, etc.)	Unspecified.

F.8.6 Multiview Scene Info Box

F.8.6.1 Definition

Box Type: 'vwdi'
Container: Multiview Group box ('mvcg')
Mandatory: No
Quantity: Zero or one

An optional Multiview Scene Info Box includes the maximum disparity between the adjacent views of the respective multiview group. This information can be used for processing the multiview video prior to rendering on a 3D display.

NOTE A Multiview Scene Information SEI message, as specified in MVC H.12.1.5, can indicate the maximum disparity between any adjacent views in the bitstream. Thus, the Multiview Scene Info Box represents similar information as carried in the Multiview Scene Information SEI message but is limited to a certain set of views rather than concerns all the views in the bitstream.

The Multiview Scene Info Box shall not be present for multiview groups associated with cameras that do not form a one-dimensional arrangement, such as a line or an arc of an ellipse.

F.8.6.2 Syntax

```
class MultiviewSceneInfoBox extends Box ('vwdi')
{
   unsigned int(8)   max_disparity;
}
```

F.8.6.3 Semantics

`max_disparity` specifies the maximum disparity in units of integer luma samples between the spatially adjacent view components (within an access unit) in this multiview group. This information can be used for processing the multiview video prior to rendering on a 3D display.

F.8.7 MVC View priority Assignment Box

F.8.7.1 Definition

Box Type: `'mvcP'`
Container: Sample Entry (`'avc1'`, `'avc2'`, `'mvc1'`, `'mvc2'`)
Mandatory: No
Quantity: Zero or one

A priority assignment URI provides the name (in the URI space) of a method used to assign content priority values in the View Priority sample grouping. The URI is treated here as a name only; it should be de-referenceable, though this is not required. File readers may be able to recognize some methods and thereby know what stream extraction or selection of output views based on particular content priority values would do.

F.8.7.2 Syntax

```
class MVCViewPriorityAssignmentBox extends Box('mvcP')
{
   unsigned int(8)   method_count;
   string PriorityAssignmentURI[method_count];
}
```

F.8.7.3 Semantics

`method_count` provides a count of the number of following URIs.

`PriorityAssignmentURI` provides a unique name of the method used to assign `content_priority_id` values in View Priority sample groupings. In the case of absence of this box, the priority assignment method is unknown.

Annex G
(Informative)

Patent Statements

The International Organization for Standardization and the International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this part of ISO/IEC 14496 may involve the use of patents.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patents right are registered with ISO and IEC. Information may be obtained from the companies listed below.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 14496 may be the subject of patent rights other than those identified in this annex. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Company
TDVision Systems, Inc.

ICS 35.040

Price based on 89 pages