

Information and Communication Technologies (ICT) Programme

**Project N°: FP7-ICT- 214063**

**SEA**



---

**Deliverable D3.2**

***MVC/SVC storage format***

---

**Author(s):** Karsten Grüneberg, Thomas Schierl (Fraunhofer HHI),  
Luca Celetto (ST Microelectronics)

**Status -Version:** Final

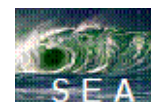
**Date:** 29 January 2009

**Distribution - Confidentiality:** Public after release

**Code:** SEA\_D3.2\_HHI\_FF\_20090129.doc

**Abstract:**

This document describes extensions to the MP4 File Format for the storage of SVC and MVC coded video sequences.



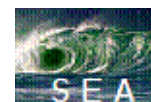
## Disclaimer

This document contains material, which is the copyright of certain SEA contractors, and may not be reproduced or copied without permission. All SEA consortium partners have agreed to the full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

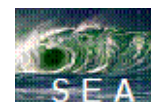
The SEA Consortium consists of the following companies:

No	Participant name	Participant short name	Country	Country
1	ST Microelectronics	STM	Co-ordinator	Italy
2	Synelixis Solutions	Synelixis	Contractor	Greece
3	Thomson Grass Valey	Thomson	Contractor	France
4	Philips Consumer Electronics	Philips	Contractor	Netherlands
5	Vodafone Panafon AEET	Vodafone	Contractor	Greece
6	Nomor Research	Nomor	Contractor	Germany
7	Fraunhofer HHI	HHI	Contractor	Germany
8	Politecnico di Torino	Polito	Contractor	Italy
9	Universidad Politécnica de Madrid	UPM	Contractor	Spain
10	University of California, Los Angeles	UCLA	Contractor	USA

The information in this document is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.



*This page has been intentionally left blank*



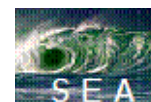
## Table of contents

<b>Abbreviations</b> .....	<b>5</b>
<b>Executive Summary</b> .....	<b>6</b>
<b>1. Introduction</b> .....	<b>7</b>
1.1. Placement of file format specifications in the SEA project.....	7
1.2. File format requirements.....	8
1.3. ISO Base Media File Format .....	8
1.4. Derived file formats.....	10
1.5. Hint track generation for media servers .....	11
<b>2. H.264/MPEG4-AVC specific file format extensions</b> .....	<b>14</b>
2.1. Basic extension for AVC.....	14
2.2. Extensions for SVC.....	15
2.2.1. Basic SVC features .....	15
2.2.2. Multi-track storage of SVC .....	16
2.2.3. Multiple tiers stored together in one SVC track .....	17
2.2.4. Signalling backward compatibility for legacy AVC readers .....	19
2.2.5. Fragmentation of SVC files .....	20
2.2.6. SVC layers stored in external files .....	20
2.3. Extensions for MVC .....	21
2.3.1. Basic MVC features.....	22
2.3.2. Multi-track storage of MVC .....	23
2.3.3. Multiple tiers stored together in one MVC track.....	24
2.3.4. Re-assembly of MVC Access Units .....	24
2.3.5. Signalling view properties in the MVC file format .....	26
2.3.6. MVC views stored in fragmented or external files.....	27
<b>3. Specific hint track generation</b> .....	<b>28</b>
3.1. Multi-session transmission.....	28
3.2. Media specific signalling .....	28
3.3. Example.....	29
<b>4. Conclusion</b> .....	<b>32</b>
<b>References</b> .....	<b>33</b>



## Abbreviations

<b>ADM</b>	<i>Adaptation Decision Module</i>
<b>AEM</b>	<i>Adaptation Execution Module</i>
<b>AU</b>	<i>Access Unit</i>
<b>AVC</b>	<i>Advanced Video Coding</i>
<b>CGS</b>	<i>Coarse Grain Scalability</i>
<b>CIF</b>	<i>Common Image Format (352 pixels x 288 lines)</i>
<b>DVB</b>	<i>Digital Video Broadcasting</i>
<b>DVB-H</b>	<i>Digital Video Broadcasting for Hand-helds</i>
<b>ETSI</b>	<i>European Telecommunications Standards Institute</i>
<b>IEC</b>	<i>International Electrotechnical Commission</i>
<b>IETF</b>	<i>Internet Engineering Task Force</i>
<b>IP</b>	<i>Internet Protocol</i>
<b>IPDC</b>	<i>IP DataCast</i>
<b>ISMA</b>	<i>Internet Streaming Media Alliance</i>
<b>ISO</b>	<i>International Organization for Standardization</i>
<b>GOP</b>	<i>Group Of Pictures</i>
<b>MANE</b>	<i>Media Aware Network Element</i>
<b>MGS</b>	<i>Medium Grain Scalability</i>
<b>MPEG</b>	<i>Motion Pictures Expert Group (ISO/IEC JTC1/SC29/WG11)</i>
<b>MST</b>	<i>Multi Session-Transmission</i>
<b>NAL</b>	<i>Network Abstraction Layer</i>
<b>QVGA</b>	<i>Quarter VGA (image format 320 x 240)</i>
<b>RFC</b>	<i>Request For Comments</i>
<b>RTP</b>	<i>Real Time Protocol</i>
<b>RTSP</b>	<i>Real Time Streaming Protocol</i>
<b>SAP</b>	<i>Session Announcement Protocol</i>
<b>SD</b>	<i>Standard Definition (TV format, either 720 x 576 or 720 x 480)</i>
<b>SDP</b>	<i>Session Description Protocol</i>
<b>SNR</b>	<i>Signal-to-Noise Ratio</i>
<b>SST</b>	<i>Single Session-Transmission</i>
<b>SVC</b>	<i>Scalable Video Coding</i>
<b>TS</b>	<i>Transport Stream</i>
<b>VGA</b>	<i>Video Graphics Array (image format 640 x 480)</i>
<b>4CIF</b>	<i>4 times CIF (704 pixels x 576 lines, comparable to SD)</i>
<b>720p</b>	<i>Image format with 1280 pixels by 720 lines, progressively scanned</i>
<b>Fps</b>	<i>frames per second (frame rate unit)</i>



## Executive Summary

Within Task3.2, Workpackage 3 of the SEA project has developed a storage format suitable for SVC as well as MVC coded video sequences. This has been accomplished by extensions to the well-established ISO Base Media File Format [19], aka "MP4 file format".

This document is structured as follows:

In order to make the text self-explanatory, an introduction on the MP4 file format (MPEG-4 Part 12) is given in subsection 1.3. Next, the way it can be extended is described in subsection 1.4. Subsection 1.5 refers to so-called Hint Tracks. Hint track generation is a very useful feature mainly used to support media streaming.

Section 2 deals with different derived file formats based on the ISO Base Media File Format. Its first subsection refers to the more general features added by MPEG-4 Part 15 [20] which are used for AVC. Extensions for SVC are described in subsection 2.2. These extensions have been added to the AVC File Format as Amendment 2 [21], which has been published in July 2008. Further extensions for MVC have been developed by WP3 in Task 3.2; these are described in subsection 2.3. All new MVC features have also been submitted to MPEG as proposals for another amendment of the AVC File Format (MPEG-4 Part 15).

Section 3 finally describes how hint tracks (cf. subsection 1.5) can be used in the context of SVC and MVC to enable very specific features in the context of layered media transmission.



# 1. Introduction

## 1.1. Placement of file format specifications in the SEA project

Towards the Future Internet age, SEA (SEAMless Content Delivery) aims to offer a new experience of seamless video delivery, maintaining the integrity and wherever applicable, adapting and enriching the quality of the media across the whole distribution chain. In more details, SEA aims to introduce novel services and new business models, by innovating over three key-content delivery pillars: a) Multi-layered/Multi-viewed content coding, b) Multi-source/multi-network streaming & adaptation and c) Content Protection and lightweight asset management.

This deliverable is related to both the first and second key content delivery pillar, namely the Multi-layered/Multi-viewed content coding and Multi-source/multi-network streaming & adaptation. Figure 1-1 shows some locations where media file storage is relevant in a simple streaming set-up. Within the SEA project, also P2P networks will be implemented which can also make use of featured media file storage. Besides, both server and client are not necessarily dedicated modules for only one direction of media processing. Instead, the client shown on the right side of Figure 1-1 may as well take the role of a server if the user decides to provide a media stream.

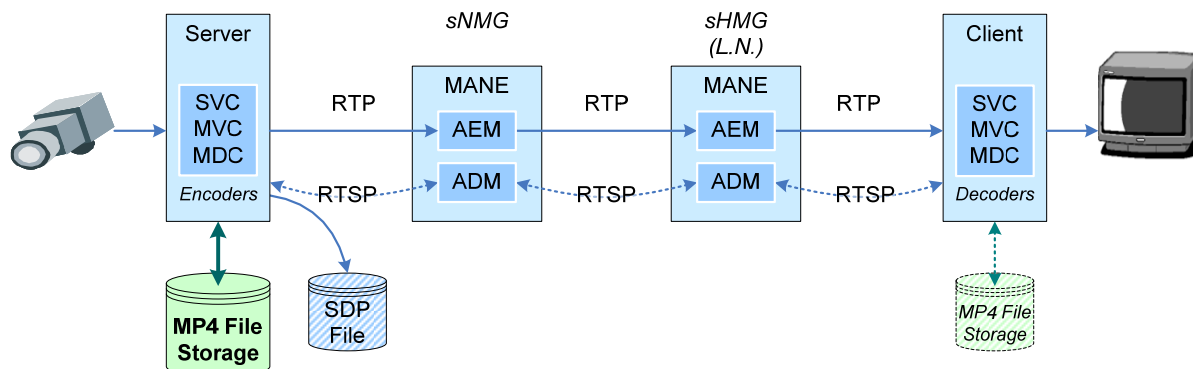


Figure 1-1 — Location of media file storage in SEA media delivery chain

Media file access at the server will be implemented as plug-in module in the SEA server as illustrated in Figure 1-2. Integration will be part of Deliverable D3.3 due M16.

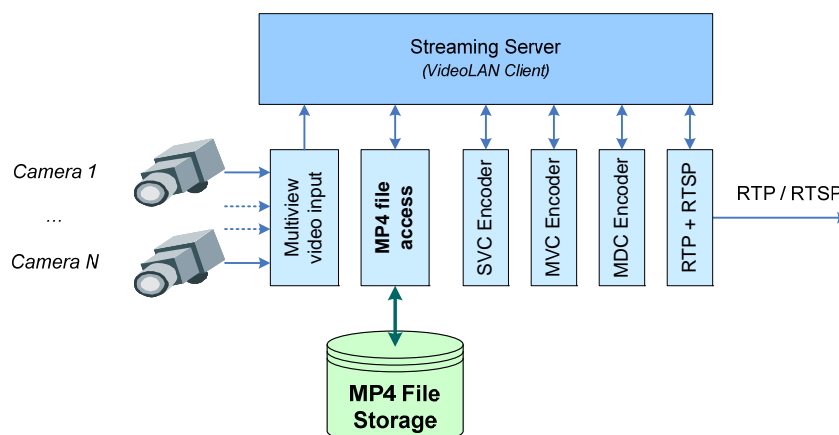
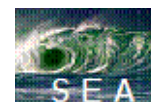


Figure 1-2 — File access plug-in module for SEA streaming server



## 1.2. File format requirements

Within the scope of the video compression standards for AVC, SVC and MVC, syntax and semantics of the bitstream are specified such that any compliant decoder generates exactly the same video data (hence the video representation is bit-true identical at all terminals). Additionally, a Network Abstraction Layer (NAL) is specified which defines the encapsulation both of the Video Coding Layer and additional information such as so-called Parameter Sets and Supplemental Enhancement Information (SEI) messages. The NAL byte stream consists of so-called NAL units, which are preceded by a header byte that identifies its type. The start of each NAL unit is identified by a start code, i.e., a special byte sequence 0x000001, which does not occur at other places within the byte stream by means of special start code prevention bytes.

Although it is possible to store the NAL unit sequence as a binary file, this does not meet many common requirements for media file storage. Typical requirements include the following features:

- storage of different media types (e.g., video and audio) in a single file;
- synchronized presentation of different media (video/audio, subtitles, etc.);
- standardized metadata for the description of the whole presentation and each of its parts;
- random access to a selected part of the presentation.

A number of different media file formats have been standardised which typically store the media data in a sort of container structure, interleaving different media data and adding metadata items. One such standard is the ISO Base Media File Format which will be described in more detail in the following section. This format is extensible in a way that all derived file formats use its basic structure and features, but can add some special features for a special purpose.

## 1.3. ISO Base Media File Format

The ISO Base Media File Format has been a spin-off from the development of the MPEG-4 standard which needed a very flexible structure amongst others for the storage of its object oriented scene descriptions. Its first version had been included in the systems part of the MPEG-4 standard, but later has been set as an independent part of the MPEG-4 standard family (MPEG-4 Part 12).

The ISO Base Media File Format – also known as MP4 File Format – has been based on the QuickTime file format owned by Apple. It specifies a file structure based on so-called boxes which are preceded by a length field and an identifier which is a four letter tag such as "ftyp" for the File Type box. Boxes can contain other boxes or box specific attributes. In an MP4 file, all data is encapsulated in boxes, so there is no data outside the box structures. A number of basic boxes and their relations are specified by the ISO Base Media File Format standard. There is a small set of box types which reside on file level, i.e., which are not contained within other boxes. Most of the boxes are child boxes which can exist in one or more specific box types. In its early versions, the standard specified two types of boxes: boxes which can contain a number of other boxes and boxes which contain a well-defined list of attributes. For the sake of extensibility, boxes can contain a version number, so that later versions of the same box type can contain additional or different attributes. At present, there are boxes that contain both other boxes and specific attributes.

Most of the metadata describing the media within an MP4 file are contained in the "Movie Box" and its child boxes (cf. Figure 1-3). The media data itself is contained in one or more Media Data Boxes. The abstract structure of a media file is given by so-called "tracks". Each different media type such as audio and video is stored in its own track, and the associated description is contained in a "Track Box" and its child boxes. Each track is structured as a sequence of so-called "samples". For each sample, its duration as well as location and size of the associated media data are recorded in tables which are child boxes of each track box (see the blue boxes in Figure 1-3).

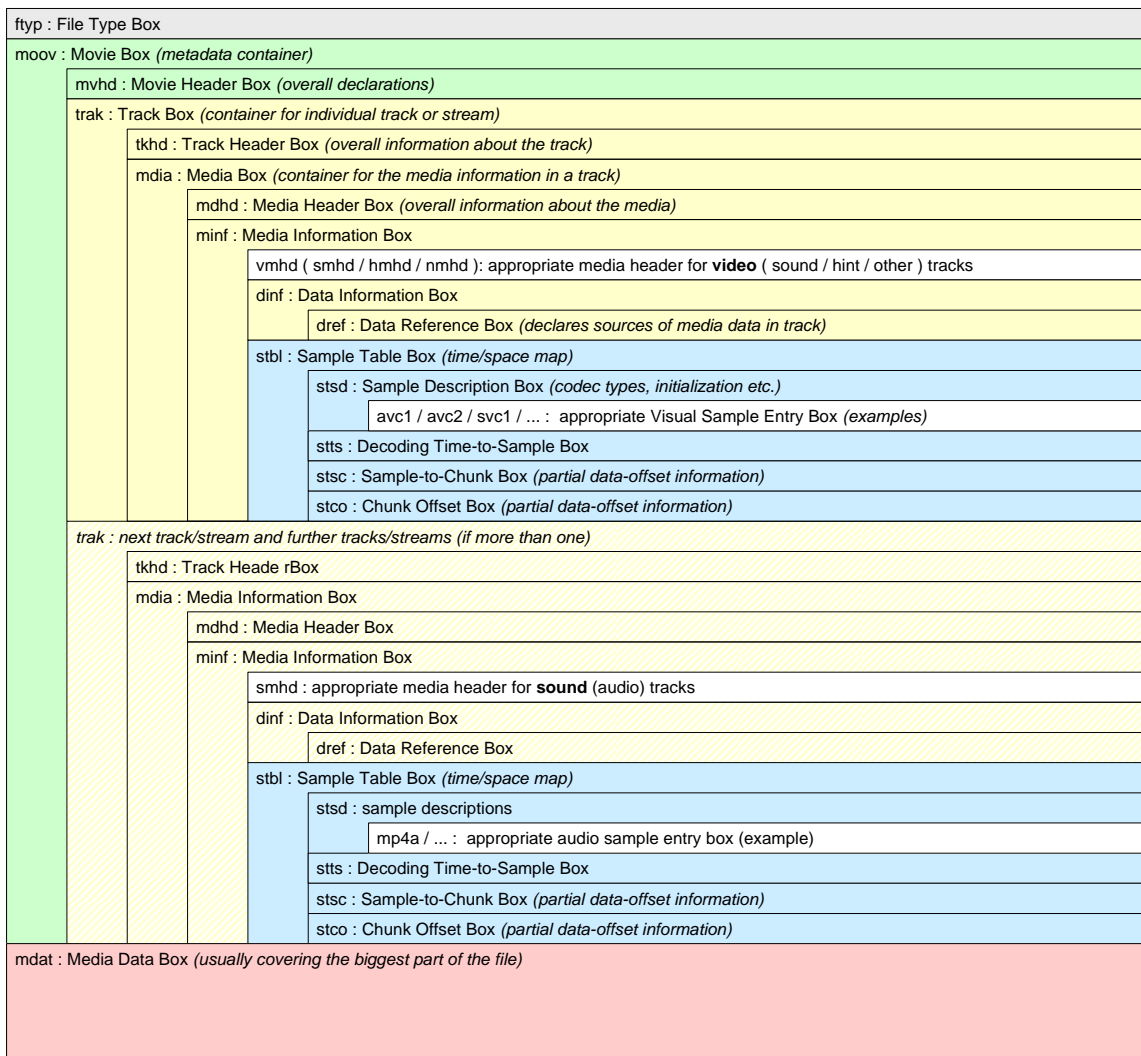


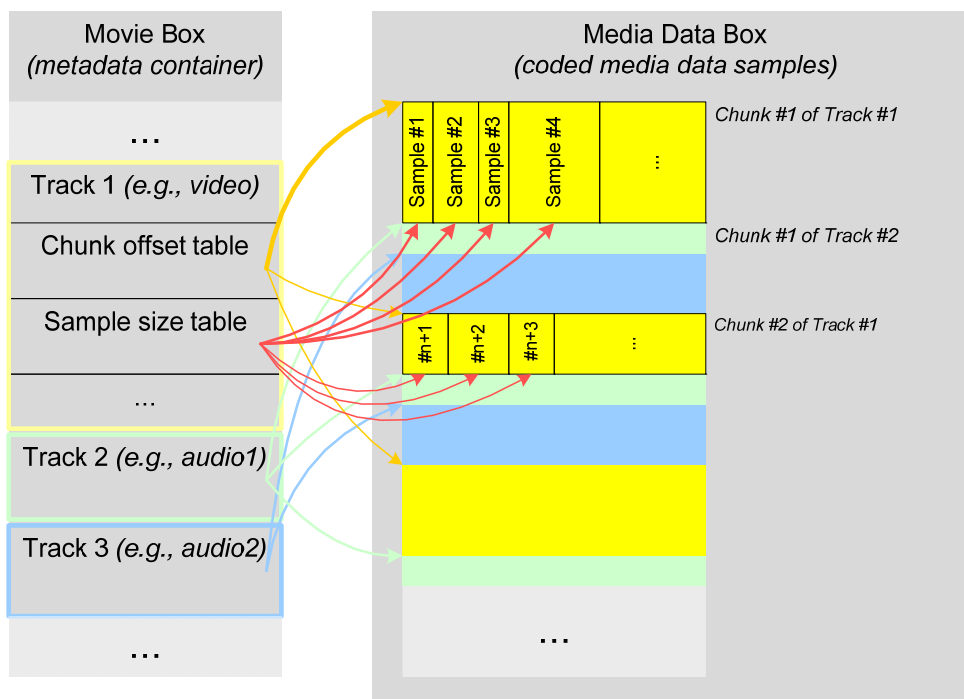
Figure 1-3 — Typical MP4 File Format Boxes for a file including a video and an audio track

During the creation of an MP4 file, media data of different tracks has to be written quasi-simultaneous to the same file, limiting buffering requirements. This is supported by the file format by a technique described in the following. For each track, a short segment of the media data which in the context of the file format is called a "chunk" has to be buffered. Each chunk consists of an integer number of subsequent samples. As soon as the maximum chunk size is reached, the chunk is appended to the media data box in the file. The ISO Base Media File Format standard does not limit the chunk size, nor does it impose any specific way how to define the number of samples to be stored continuously as a chunk. This could be based on memory size restrictions, but many implementations just write one chunk per second. Entries in the chunk offset box record the start address of each chunk as depicted in Figure 1-4. Most MP4 files are self-contained, which means that the media data is located within the same file as the metadata, but the standard also allows storing the chunks in one or more external data files referenced using the data reference box<sup>1</sup>. The association of samples to chunks is stored in a sample-to-chunk box which holds the number of samples per chunk for one or more chunks amongst other information. Syntax and semantics of this box result in a very compact description if the sample count is constant for all chunks, but does not prevent recording different sample counts for each chunk.

<sup>1</sup> A use case for this feature is described in section 2.2.6.

Random access to media data samples is supported by the sample size box which is also located within the metadata structure of each track (cf. Figure 1-4). In order to access a specific sample, a file reader determines the appropriate chunk and its start address by reading the boxes mentioned earlier. After that, it will find the start address of the sample by accumulating the sample sizes of all previous samples within the same chunk. Again, syntax and semantics of the sample size box allow for a very compact table if subsequent samples have the same size.

In addition to the basic information described so far, other information is stored in further metadata containers. Mandatory information includes the sample timing in terms of decoding time. If the presentation time differs from the decoding time, as it is the case for many video coding standards that use bi-directional prediction, an optional "composition time to sample box" is used to store the offset values. All timing information is based on time scale values which can be specified both for the whole file and for each track individually.



**Figure 1-4 — Locating media data of individual samples within an MP4 file**

In MP4 media files, metadata which is needed to access the encoded video data can be stored either in a compact so-called "Movie Box" as described above, or the file is structured in different fragments, each described by an individual "Movie Fragment Box". The latter will be explained in the context of SVC storage in subsection 2.2.5 below.

### 1.4. Derived file formats

When the ISO Base Media File Format was standardized, provisions had been made in order to keep it flexible and allow for extensions. New boxes can be specified which will be ignored by older implementations. Even the syntax of existing boxes can be changed if the box had been specified such that it includes a version number, but this option can easily break existing implementations as they might not be able to access mandatory information.

The main advantage of specifying a number of derived file formats rather than including new features in the ISO Base Media File Format is to save dedicated implementations from unnecessary burdens. A device capable of decoding only a certain set of audio-visual media does not need to understand the syntax of boxes that only make sense with other codecs. Thus, special requirements for video

(MPEG-4 Part 2) have been separated and standardized as MPEG-4 Part 14, and Part15 has been dedicated to MPEG-4 Part 10 (AVC), including extensions for SVC. As the emerging standard for MVC extends MPEG-4 Part 10, future file format extensions for MVC will consequently extend Part 15, too. Details on new features brought by Part 15 are described in Section 2 and its subsections.

### 1.5. Hint track generation for media servers

The ISO Base Media File Format not only specifies the syntax of a multitude of boxes for the storage of multiple media data structured by tracks, it also provides a very flexible technique to support the encapsulation of media data in pre-defined structures. RTP [4] packets are frequently used for the streaming of all kinds of media over the Internet, consequently an appropriate method to specify RTP encapsulation is include in MPEG-4 Part 12. The RTP payload format depends on the media type. In most cases, the RTP payload includes a block of media data, preceded by some header information which often includes a length field amongst other information. On example for mpeg4-generic payload (amongst others used for MPEG-4 Part 2 [3]) is illustrated in Figure 1-5.

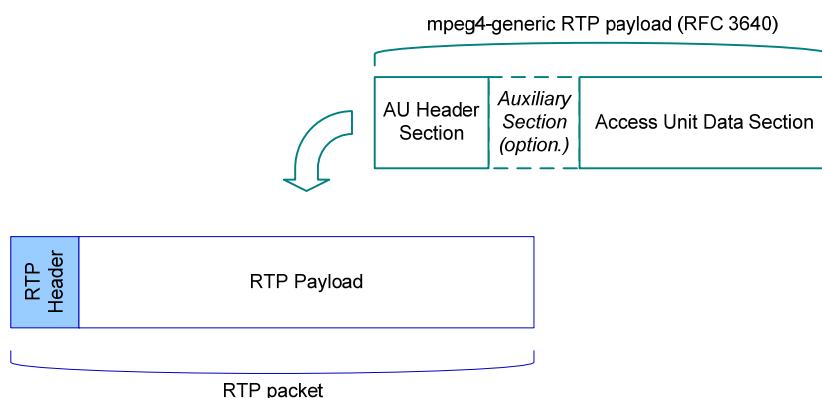


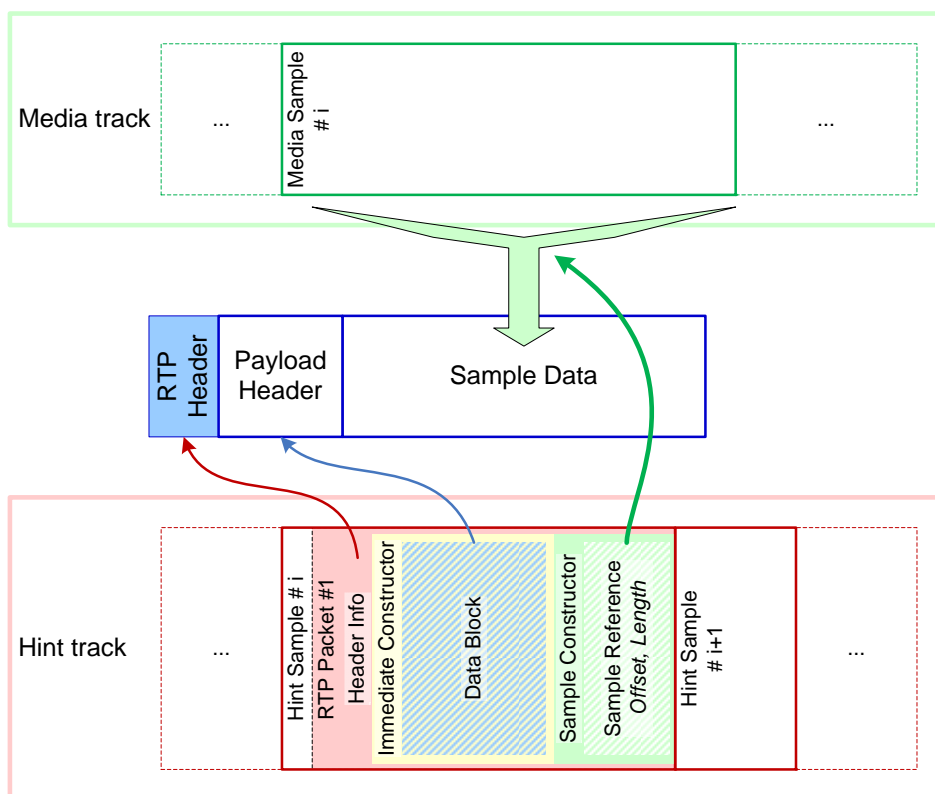
Figure 1-5 — Example for RTP encapsulation of media using the mpeg4-generic payload format

Implementation of a media server would be very painful if the server itself had to be aware of all possible RTP payload formats. Instead, the details how to do the RTP encapsulation can be stored in the same file as the media data itself using so-called hint tracks. Once a hint track is created for a certain media track by an application which is aware of the correct RTP payload format for that specific media, a streaming server will know how to packetize the data by reading the very simple syntax of these tracks.

As any other track, a hint track consists of a number of samples. A hint track sample contains the description how to construct one or more RTP packets. For this purpose, different syntax elements can be used which are called "constructors". These constructors are interpreted while they are read by a streaming server. Mainly, two types of constructors are used:

- (1) RTP Immediate Constructors contain a data array which is directly inserted into the RTP packet;
- (2) RTP Sample Constructors reference a certain portion or a whole sample which has to be fetched from a media track and inserted into the RTP packet.

As mentioned before, the application generating a hint track has to be aware of the correct RTP payload format syntax. Typically, header information is generated by the hint track creator and stored in an RTP Immediate Constructor. The data block can be copied from a specified media sample using the RTP Sample Constructor so that the media data itself will not be stored another time within the hint track. In order to provide enough flexibility for any possible payload format, the RTP Sample Constructor specifies both an offset and a length of the data block to be copied, see Figure 1-6.



**Figure 1-6 — Example for RTP packet construction using the hint track mechanism**

This can also be used to include different portions of the same media sample in different RTP packets as shown in Figure 1-7.

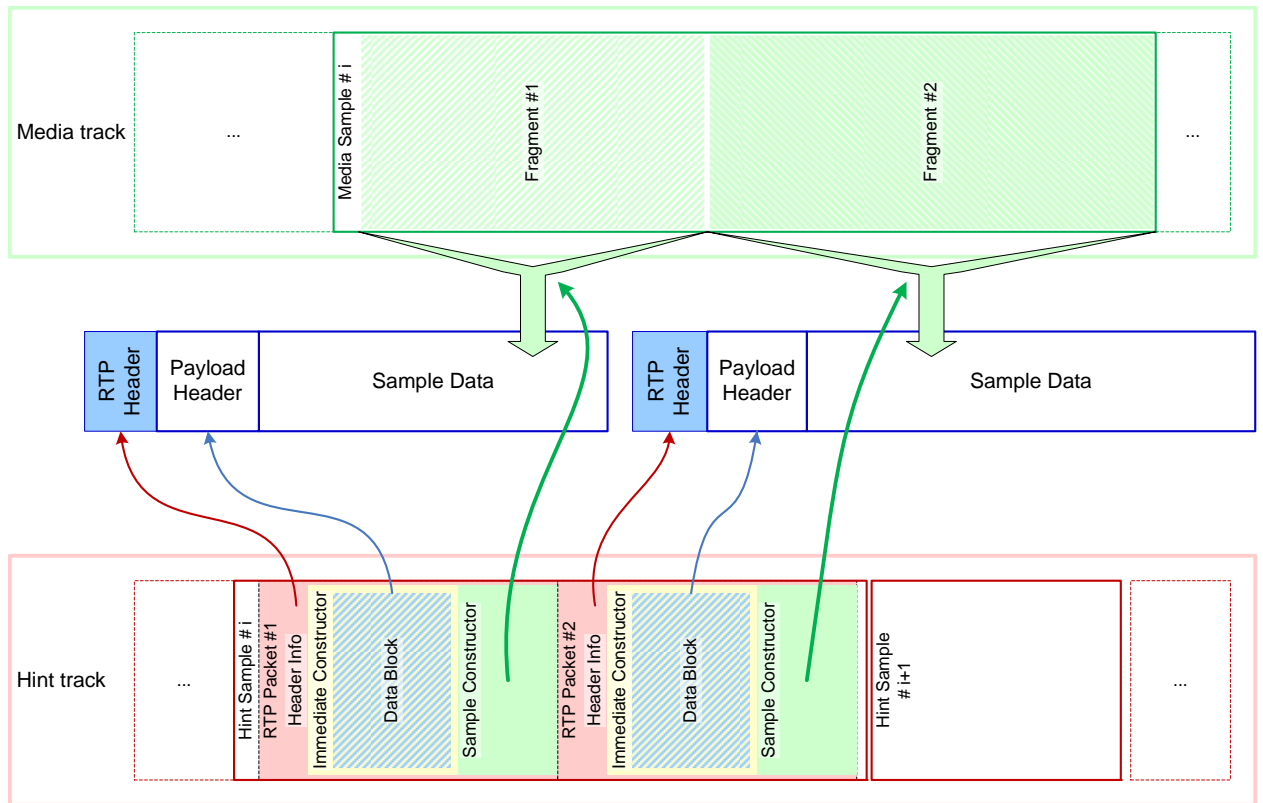


Figure 1-7 — Fragmented sample data sent in two RTP packets



## 2. H.264/MPEG4-AVC specific file format extensions

As already mentioned in subsection 1.4, a number of file formats have been derived from the ISO Base Media File Format using its extensibility features. In the next subsection we briefly describe the basic features of the AVC file format [20]. Awareness of the basic features of that file format will be crucial for the understanding of the following subsections which deal with extensions for SVC [21] and proposed MVC specific features.

### 2.1. Basic extension for AVC

MPEG-4 AVC has introduced the so-called Network Abstraction Layer (NAL) which encapsulates the video coding layer using a number of NAL unit types. This not only eases bitstream parsing but also defines a structure which is useful for the encapsulation of meaningful elements of the bitstream into any sort of data packets. Consequently, also the storage of AVC bitstreams in MP4 files according to MPEG-4 Part 15 is based on NAL units. For AVC media files, a sample consists of the NAL unit sequence in decoding order which belongs to the same Access Unit, i.e., all NAL units associated with the same picture within a video sequence. In order to ease parsing of the sample for the individual NAL units, the start code preceding each NAL unit in byte stream syntax according to [1] is replaced by a length field as shown in Figure 2-1 when the sample is stored in the media data box as described in subsection 1.3 above. If required, a file reader could easily replace that length field by a valid start code again. In many cases, file access and decoding are implemented within the same application which then can retrieve the bitstream from the sample data directly.

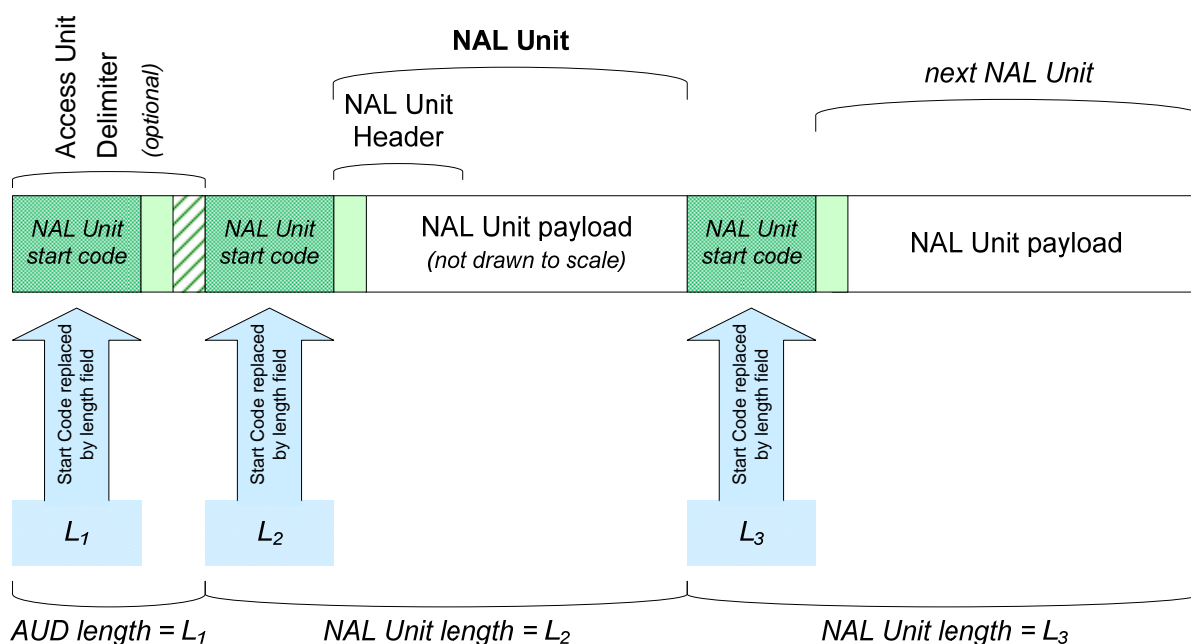


Figure 2-1 — AVC sample stored in the AVC File Format

Parameter sets carrying a lot of side information needed by the decoder are another feature of the AVC standard. Figure 2-2 shows an extract of the metadata boxes describing the samples within an AVC track (for the location of these boxes within file scope cf. Figure 1-3). Parameter sets are stored in the AVC Configuration record (shaded in red) as a part of the AVC sample entry. From that box a decoder can easily access this information before it starts decoding.

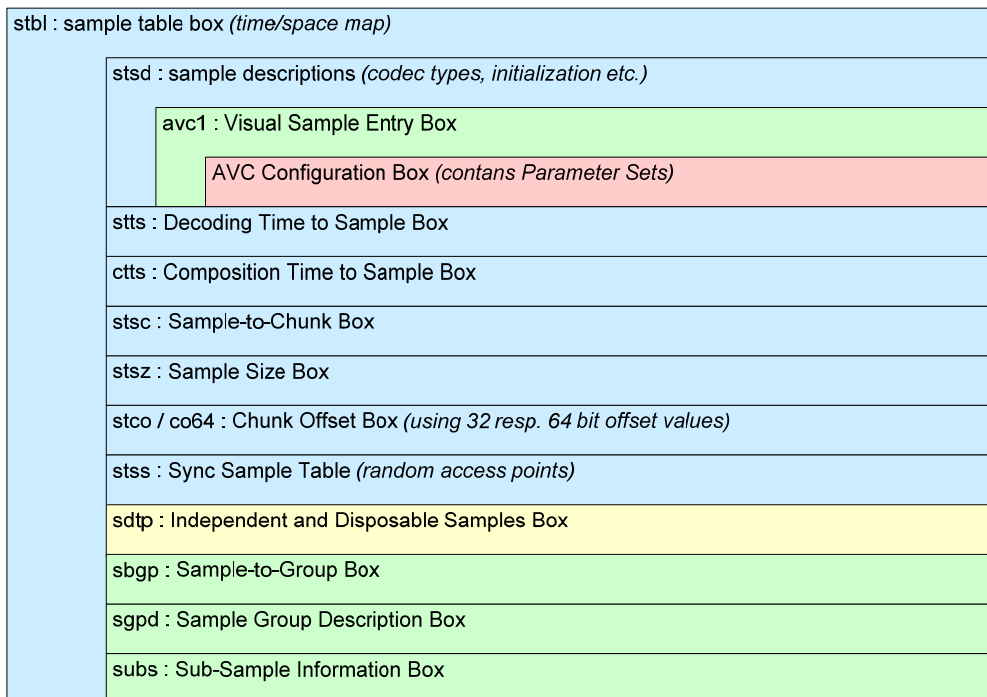


Figure 2-2 — Typical MP4 File Format Boxes

Finally, a sample-grouping mechanism using the three green boxes at the bottom of Figure 2-2 has been specified. Sample Groups can be used with plain AVC, but are more useful with SVC or MVC and will be mentioned in the appropriate chapters below.

## 2.2. Extensions for SVC

### 2.2.1. Basic SVC features

SVC bitstreams have similar properties as AVC bitstreams. In fact, a conformant AVC bitstream can be derived by extracting NAL units that belong to the base layer of an SVC bitstream and dropping all other NAL units. The association of SVC NAL units to a certain layer is given by dedicated bit fields in the NAL Unit Header SVC Extension as depicted in Figure 2-3.

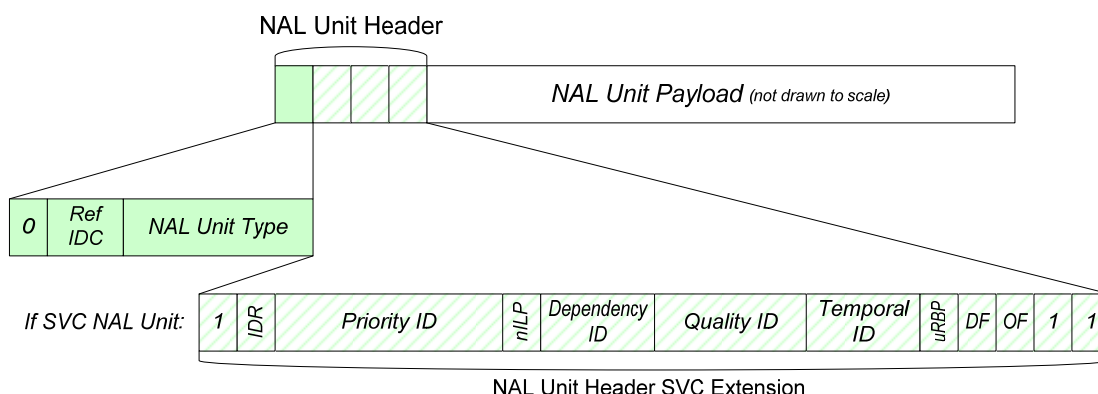


Figure 2-3 — Identification of SVC NAL units by the NAL Unit Header SVC Extension

AVC NAL units do not use this header extension directly, but special prefix NAL units have been specified in the SVC extensions (Annex G of [1]) which precede AVC NAL units carrying data of



the video coding layer (VCL NAL units). The bit fields in the NAL Unit Header SVC Extension of these Prefix NAL units are associated to the AVC NAL units. For the base layer, Dependency ID and Quality ID are zero. More information on SVC features are found in [2]

### 2.2.2. Multi-track storage of SVC

The file format extensions for SVC reflect the layered structure of the media data. Clearly, an SVC can be stored in one video track in the same way as described for AVC. A more specific storage mode is depicted in Figure 2-4. In that example, an SVC streams is split into 4 tiers, Tier 1 doubling the frame rate of the base layer, while both Tier 2 and Tier 3 operate at four times the base layer frame rate. Within the file format extensions for SVC, tiers are defined as entities containing the data of one or more scalable layers. An integer number of tiers can be stored as an individual track.

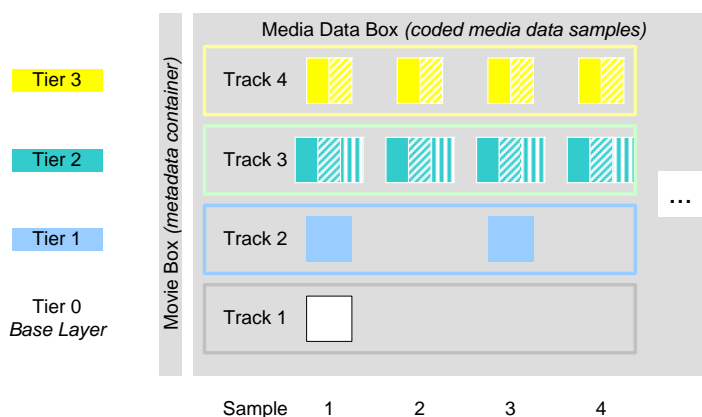


Figure 2-4: Distribution of SVC layers over several tracks

### Extractors

A special technique which allows inclusion of lower layer data needed for decoding in the tracks of an upper layer without duplicating the data has been specified in the SVC extensions. This is achieved by means of so-called extractors which follow the NAL unit syntax as shown in Figure 2-5. Extractors are distinguished from other NAL units by their type number being equal to 31 which had not been used by the MPEG4 Part 10 standard (cf. Table 7-1 in [1]). When a certain track is read, Extractors are de-referenced, i.e., the data referenced by its data fields are supplied instead of the Extractor itself.

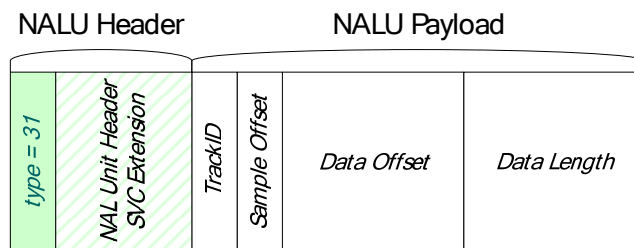


Figure 2-5: Extractor syntax

Extractors allow referencing one or more NAL units of a single track. Figure 2-6 gives a more detailed view on the same example as shown in Figure 2-4, this time explicitly showing the Extractor pseudo-NAL-units. In this example, Track 4 references data from all lower layers, using one Extractor per track. The third Extractor in Track 4 references three consecutive NAL units of Track 3 which is easily achieved by setting the Data Length field (cf. Figure 2-5) accordingly.



For the sake of compactness, Extractors do not contain the sample number of the sample they reference, but only an eight bit field for the sample offset. The base for this offset is found implicitly by evaluation of the decoding time of each sample, because it would not be trivial to find matching sample numbers between tracks operating at different frame rate. This leads to a specific issue regarding temporal scalability. A player can access one of the tracks in order to play the video at a certain operation point, i.e., at a certain combination of spatial resolution, frame rate and quality level. Track 2 contains the video at double the base layer frame rate; consequently, the duration of all its samples would be equal to half the duration of the base layer samples. Track 3 contains the video at four times the base layer frame rate, i.e., the duration of all its samples would be equal to one fourth of the duration of the base layer samples. Here, if samples from lower layers are referenced by Extractors, the original duration of the referenced samples differs from the duration of the referencing sample. Consequently, Extractors are not necessarily aligned with the samples they reference. Figure 2-6 shows that the decoding time of the second sample in track 2 (cross-hatched in blue) differs from the decoding time of the referencing Extractors.

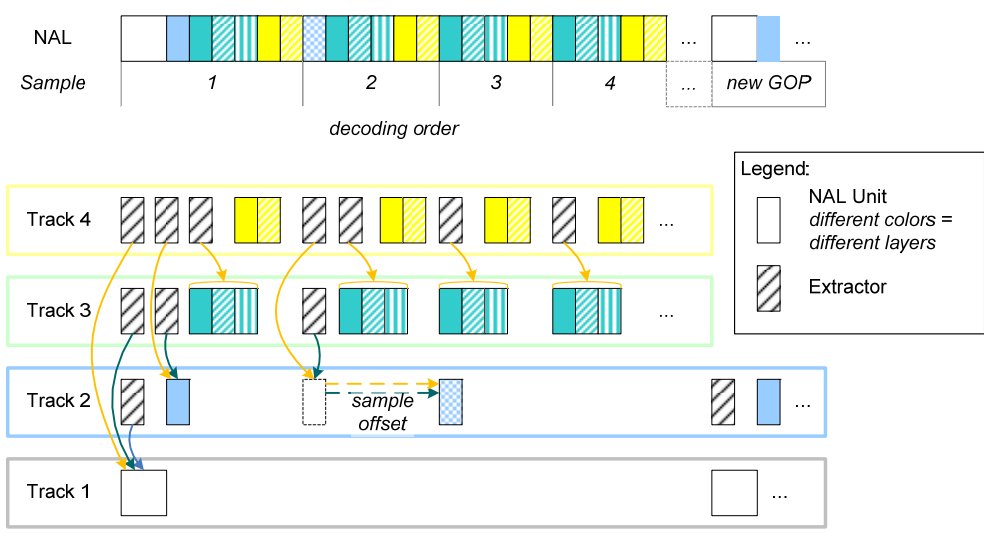


Figure 2-6: Multi-track SVC storage using extractors

**2.2.3. Multiple tiers stored together in one SVC track**

If a multitude of tiers is stored within a single track, Sample Groups can be used and special metadata boxes have been defined to support the extraction of NAL units from a sample for each tier individually. Figure 2-7 shows a sequence of samples, i.e., SVC Access Units stored inside the same track, which include NAL units that belong to different tiers. In this case, information on the sequence of NAL units within each of the samples can be stored in specific Scalable Group Entries specified in the SVC File Format standard [21] stored in the Sample Group Description Box (cf. Figure 2-3).

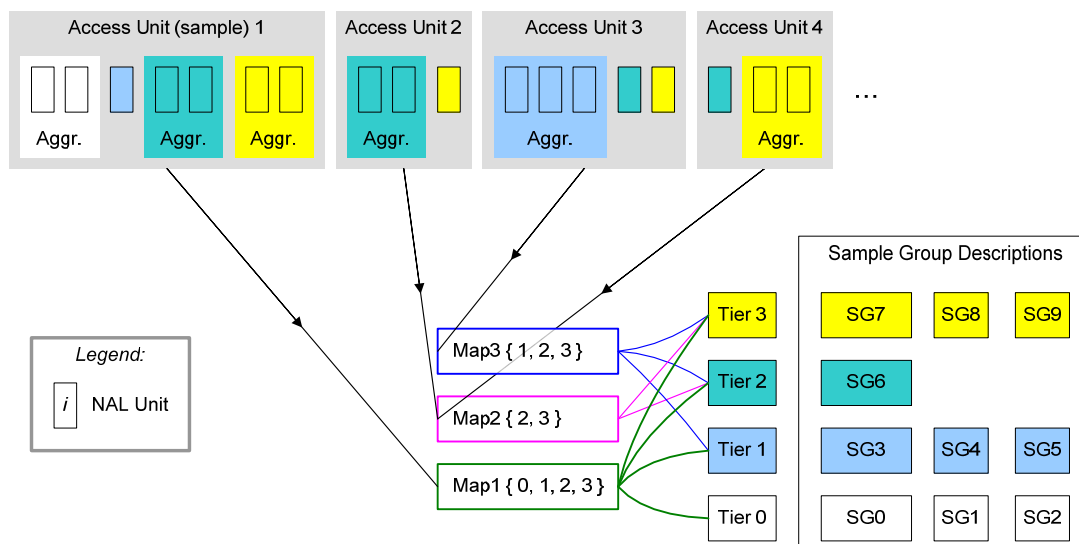


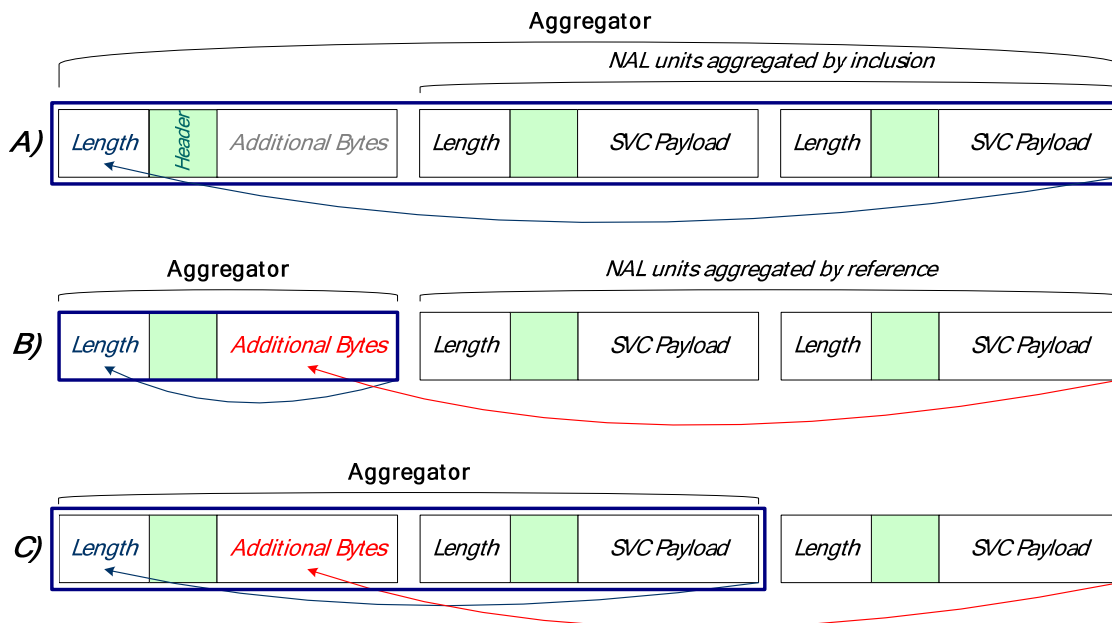
Figure 2-7: Sample grouping for the storage of multiple tiers in the same track

In order to keep this information compact, double indirection is used. Each sample is associated with a Scalable NAL Unit Map Entry (see the outlined boxes in the centre of Figure 2-7) which describes the sequence of tiers inside each sample. In Figure 2-7 we can notice that Access Unit 2 and Access Unit 4 maps on the same Scalable NAL Unit Map Entry, the magenta one. The properties of each tier are listed in Scalable Group Entries<sup>2</sup> (see the coloured blocks in the Sample Group Description Box at the right side of Figure 2-7).

### Aggregators

Additionally, a special technique has been specified in order to both reduce the number of Scalable NAL Unit Map Entries and keep them as short as possible. If a sample contains more than one NAL unit which belongs to the same tier, these NAL units can be virtually concatenated using so-called Aggregators as illustrated at the top of Figure 2-7. Shaded blocks labelled "Aggr." contain such a sub-sequence of NAL units belonging to the same tier. The syntax element used for this purpose is a pseudo-NAL-unit with the type field equal to 30. Aggregators can be used in different ways as shown in Figure 2-8. The greenish blocks stand for NAL unit headers, of which the leftmost in each row has a type field equal 30. Note that the length field preceding each header is the NAL unit length (cf. Figure 2-1). The top row of Figure 2-8 (labelled A) shows an Aggregator which contains two SVC NAL units. This is achieved by the length field which is set to the total length of Aggregator plus included NAL units. The middle row (B) shows an Aggregator which aggregates NAL units by reference. This is achieved by the Additional Bytes field in the payload of the Aggregator which is set to the total length of the following NAL units to be aggregated. Finally, the bottom row (C) shows the combination of both mechanisms.

<sup>2</sup> Each Scalable Group Entry documents a subset of the SVC stream. Each of the subsets is associated with a tier and may contain one or more operating points. These entries are defined using a grouping type of 'scif'.



**Figure 2-8: Aggregation of NAL units**

A file reader which is aware of the SVC File Format extensions will handle included NAL units and NAL units aggregated by reference just the same. In contrast to this, AVC file readers which do not understand Aggregators (e.g., older implementations or MP4 file interfaces of legacy AVC decoders) will not recognise the Aggregator and handle it as an unknown NAL unit type. This leads to different behaviour for the three depicted situations:

- In case A, either the file reader will skip the Aggregator with the two included SVC NAL units, or it will forward all NAL units to the AVC decoder. In the latter case, an AVC decoder which is only capable of decoding stream conforming to AVC profiles as specified in Annex A of [1] will skip the whole ensemble.
- In case B, file readers or legacy decoders will only skip the Aggregator (box drawn with a blue outline in Figure 2-8) and ignore its payload, so no "additional bytes" are affected.
- In case C, file readers or legacy decoders will skip the Aggregator with the included SVC NAL unit (box drawn with a blue outline in Figure 2-8) and ignore its payload, but the second NAL unit is not affected.

#### **2.2.4. Signalling backward compatibility for legacy AVC readers**

Backward compatibility for legacy AVC readers had been one of the main requirements of all SVC extensions. Thus, features mentioned above need not be supported in order to retrieve the AVC compatible base layer of an SVC stream. Still, an AVC reader needs a clue which of the tracks it can read and interpret correctly; consequently, different Visual Sample Entries have been specified. An MP4 file reader will read the Sample Entry (cf. Figure 1-3) of all tracks contained in an MP4 file in order to decide which of the tracks it can use or which is appropriate for a certain action requested by the user. As any other box of the ISO Base Media File Format, Sample Entries are identified by a 32 bit field containing a human readable 4-letter tag. For SVC files, the Visual Sample Entry Box type in each track has one of the following values: "avc1", "avc2" or "svc1".

avc1: Tracks with a Visual Sample Entry of type "avc1" can be read by a legacy device. Support for SVC features such as Extractors or Aggregators is not required. An AVC decoder will find the base layer NAL units and ignore SVC NAL units, if any. If this track only contains the base layer, the Sample Entry contains the AVC Configuration Record only (cf. section 2.1). If the track also contains SVC NAL units, the sample entry



additionally contains an SVC Configuration Record providing the Parameter Sets necessary to decode higher layers of the SVC bitstream.

- avc2: Tracks with a Visual Sample Entry of type "avc2" can be read by devices that support Aggregators and Extractors. These tracks contain an AVC base layer. As for an "avc1" Sample Entry, the AVC Configuration Record is always present, while an SVC Configuration Record is only present, if the track contains SVC data other than the base layer.
- svc1: Tracks with a Visual Sample Entry of type "svc1" do not contain an AVC base layer; they can be read by devices that support Aggregators and Extractors.

### 2.2.5. Fragmentation of SVC files

In order to access an MP4 file, the Movie Box must be read completely. If a file is downloaded over an IP Datacast system, this can be an issue if the file size is large or if a source appends data to a file while it is already being forwarded. In such a context, special SVC file format features can be activated in order to allow for both backward compatibility and fast channel switching. A different storage mode is illustrated in Figure 2-9. Here, a certain portion of the file is accompanied by its own metadata. Note that the first portion is structured in the same way as in a non-fragmented file for compatibility reasons as legacy devices might not be aware of movie fragments. File readers compliant to the DVB File Format [22], which is a derived standard of the ISO Base Media File Format [19], are mandated to support fragmentation. Fragmented files are advantageous in the context of media download as the reader can start accessing the file as soon as the first portion has been received. In fact, it might start shortly after the first metadata block and some sample data are received, if the incoming rate of sample data is higher than the output rate.

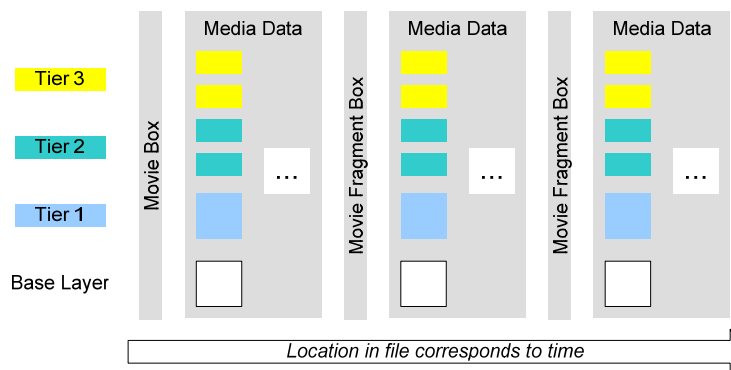
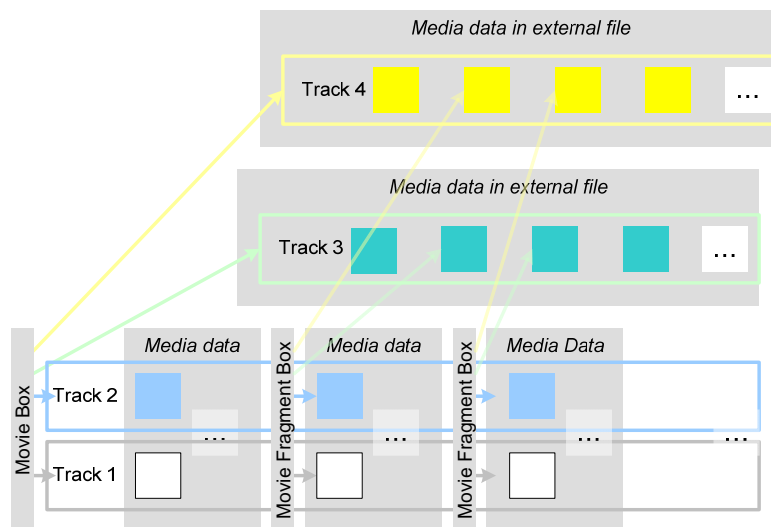


Figure 2-9: Multiple fragments stored in an MP4 media file

### 2.2.6. SVC layers stored in external files

Still the storage modes illustrated above will not optimize the download experience for a receiver that can only use a subset of the tiers sent for that video. This can be the case if a file server offers the same contents to devices with different capability. If additional layers are stored in the same file, the total download size would be larger than the amount of data actually used by the receiver. Here, another storage mode can be used beneficially. The MP4 file standard allows storing portions of the media data in different files, keeping all metadata in one file (which in the following will be referred to as "main" file). This situation is illustrated in Figure 2-10 where Tracks 1 and 2 are stored in the "main" file with all metadata, while Track 3 and Track 4 are stored individually in external files. Referring to the same association of tiers to tracks as in the previous example, this storage concept allows accessing a subset of tiers, in this example the lowest two tiers, without handling the additional data stored in the external files.



**Figure 2-10: Multiple fragments stored in multiple files, all metadata stored in "main" MP4 file**

The file can be accessed, if the Movie Box has been read completely, which can be an issue if the file size is large or if a source appends data to a file while it is already being forwarded. Consequently, it can be advantageous to use fragmentation together with the storage of parts of the media in external files.

In addition to the use case described above, the storage of different tiers in different files can be used to support storage erosion. A user might wish to store a certain video sequence with high quality and later decide to drop the high quality part keeping just the lower layers. An application might be a security system with limited storage capacity. In the example shown in Figure 2-10 above, this "erosion" could be done in two steps, first deleting the external file containing Track 4 (resp. Tier 3), then deleting Track 3 (resp. Tier 2), replacing the obsolete metadata in the main MP4 file by so-called "free" boxes. This can avoid rewriting a large amount of media data.

### 2.3. Extensions for MVC

Within the SEA project, we will go one step beyond the transmission of classical 2D video. As a first move towards future systems which allow the user to freely choose a viewpoint of a visual scene or provide a 3D depth impression, SEA implementations will allow transmission of multiple views of the same scene to the user. A possible solution for this would be to encode multiple video signals independently using an AVC codec. However, specific multi-view video coding (MVC) algorithms in a system as shown in Figure 2-11 give significantly better results [28].

MVC can be regarded as a first step in the direction of efficient compression of multi-view video data. Further studies exceeding the scope of this project by far may lead to advanced 3D-TV systems with free-viewpoint functionality. A next step could be an advanced data representation consisting of multiview video plus scene depth (MVD) suiting systems for video presentation on multiscopic 3D displays [29].

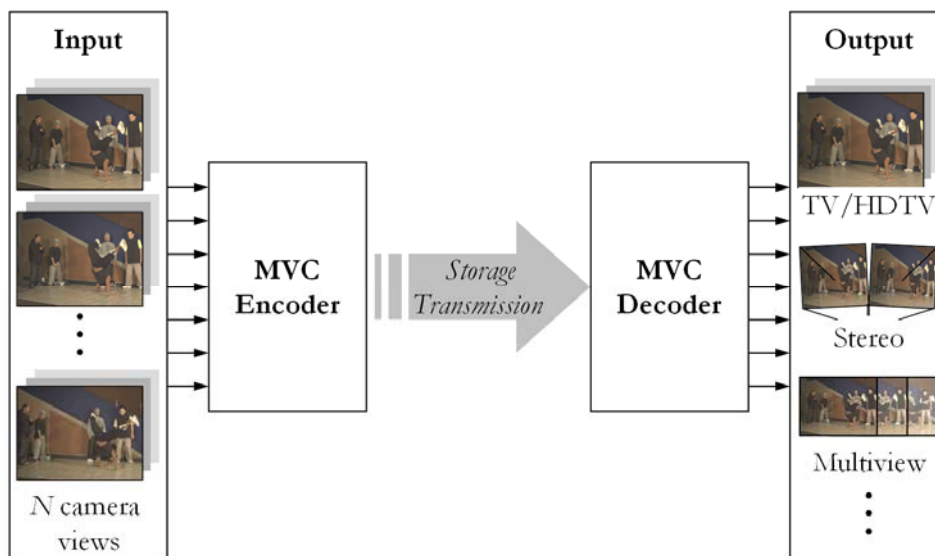


Figure 2-11: MVC overall system structure

### 2.3.1. Basic MVC features

MVC standardisation has reached the Final Draft Amendment (FDAM) [30] ballot stage which is the last step before the final Amendment can be published. Thus, we do not expect major changes of its basic features which are described in the following as a background for MVC specific file format extensions. Regarding some remaining issues, the SEA project will continue contributing to the MVC File Format and its standardisation and it may be necessary to upgrade the implementation of the MVC File Format to the final standard during the second year of the project.

MVC bitstreams are quite similar to SVC bitstreams; in fact MVC extensions have been limited to the high-level syntax. Instead of layers that represent different quality of the same content, different views of the same scenery are coded and represented in a common bitstream. Analogous to the base layer for SVC, a conformant AVC bitstream can be derived by extracting NAL units that belong to the base view of an MVC bitstream and dropping all other NAL units. The association of MVC NAL units to a certain layer is given by dedicated bit fields in the NAL Unit Header Extension as depicted in Figure 2-3. Note that the NAL Unit Header Extension will be inserted in the same way as for SVC, but its syntax and semantics differ between SVC and MVC.

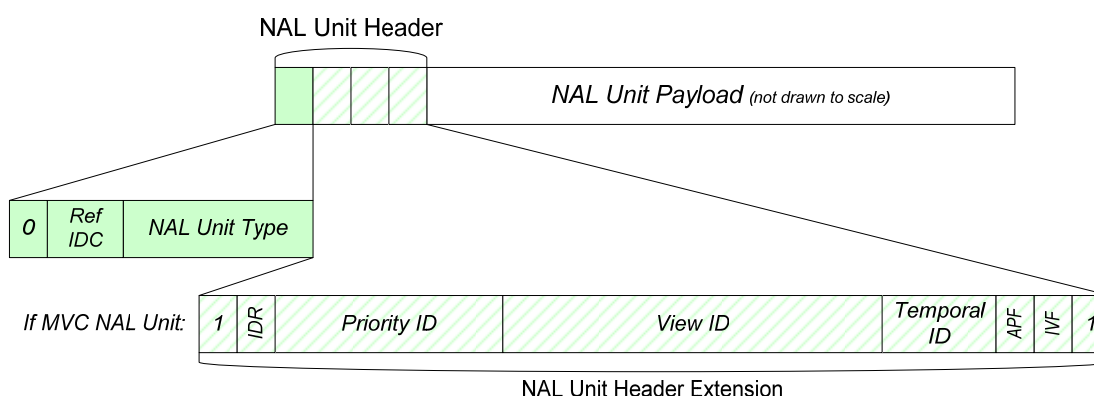


Figure 2-12 — Identification of MVC NAL units by the NAL Unit Header Extension

As for SVC, prefix NAL units have been specified in the MVC extensions (cf. Annex H of [30]) which precede AVC NAL units carrying data of the video coding layer (VCL NAL units). The bit



fields in the NAL Unit Header Extension of these Prefix NAL units are associated to the AVC NAL units. The coded representation of a view in a single access unit is called a View Component.

In contrast to SVC layers, MVC views are identified by arbitrary View ID numbers which do not imply any order or a specific dependency between different numbers. Instead, the decoding order of different views has to be derived from the position of each View IDs in a list which is included in the Sequence Parameter Set for an MVC stream. The index into this ordered list of View IDs is called View Order Index. An example for the relation between different views is shown in Figure 2-13. Here inter-view prediction is used for so-called anchor pictures. All other frames, temporally located between the anchor frames, use intra-view prediction. Clearly, the MVC standard also allows inter-view prediction for all frames, but it is up to the content creator how to find an optimum balance between complexity and efficiency of the prediction structure.

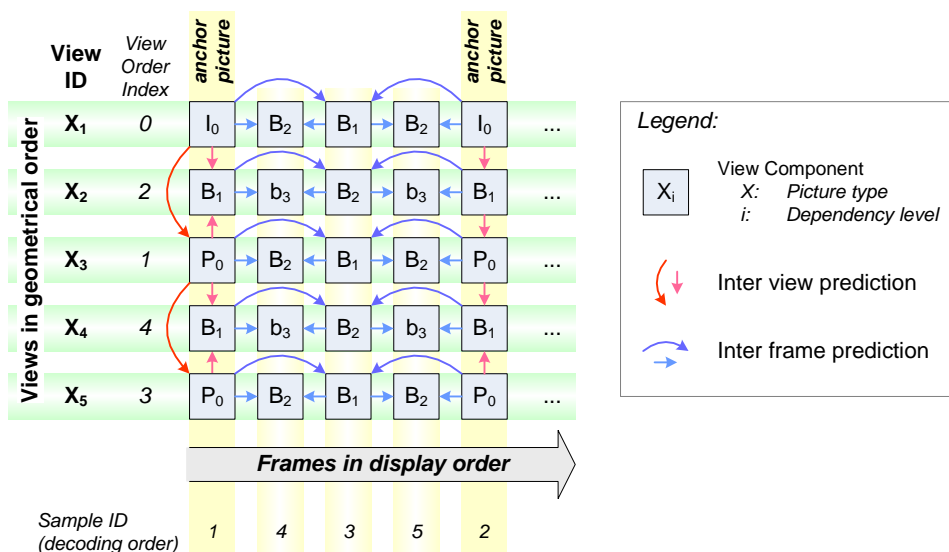


Figure 2-13 — Decoding dependencies between MVC view components (example)

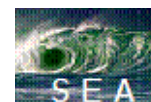
According to the inter-view prediction structure, decoding dependencies arise between different views. This implies a certain decoding order of the different MVC View Components. On the one hand, View Component B can only depend on another View Component A if A precedes B in decoding order given by the View Order Index. On the other hand, this does not necessarily mean that each View Component depends on all preceding View Components of the same Access Unit.

Besides, neither camera position or orientation nor the geometrical relation between different views can be inferred from View ID or View Order Index. For this purpose, both intrinsic and extrinsic camera parameters can be included in the bitstream using a Multiview Acquisition Information SEI message.

### 2.3.2. Multi-track storage of MVC

The first approach proposed to MPEG by the SEA project was to represent a multi-view video stream by two or more video tracks in a file. Each track would then represent a single view of the stream.

In this mode, the base view of the MVC bitstream is stored in one track. This track is nominated as the ‘base view track’ and it is a plain AVC track which can be decoded by a legacy decoder. All other tracks that are part of the same multi-view stream are linked to this base track by means of a track reference of type ‘vbas’ (view base). Additionally, a track reference of type ‘mvpr’ shall be inserted in the ‘tref’ box of a certain track *n* for each other track that contains samples from another view on which track *n* depends.



All the tracks in the group sharing the same view base must share the same timescale, and all samples have to be time-aligned. The sample entry format for a track is suitable for the stream represented by the track (e.g. unused parameter sets need not be present etc.).

### 2.3.3. Multiple tiers stored together in one MVC track

In the second SEA proposal regarding MVC File Format [24], we proposed to also allow storing multiple views within the same track. This may reduce the implementation effort and improve performances for recording devices. Implementations will benefit especially in the presence of buffer limitations, as a recording application needs to reserve one buffer per track for the chunks. In contrast to that, file readers does not necessarily need to buffer the whole chunk for each track, though readers still need to manage reading all chunks in parallel.

A mixed track structure is shown in Figure 2-14 for the example shown in Figure 2-13. Note that the views are re-ordered both according to their view order index, reflecting inter-view dependencies, and according to their intra-view decoding order. Here, the base view with view ID  $X_1$  is stored as a single view in the first track (shaded in green) which allows for optimum backward compatible file access by a legacy AVC reader. The next two views ( $X_3, X_2$ ) which depend on the base view only are stored together in a second track. In contrast to that, the remaining two views ( $X_5, X_4$ ) are stored individually each in its own track. Only MVC-aware devices would interpret the MVC tracks (shaded in light blue) correctly.

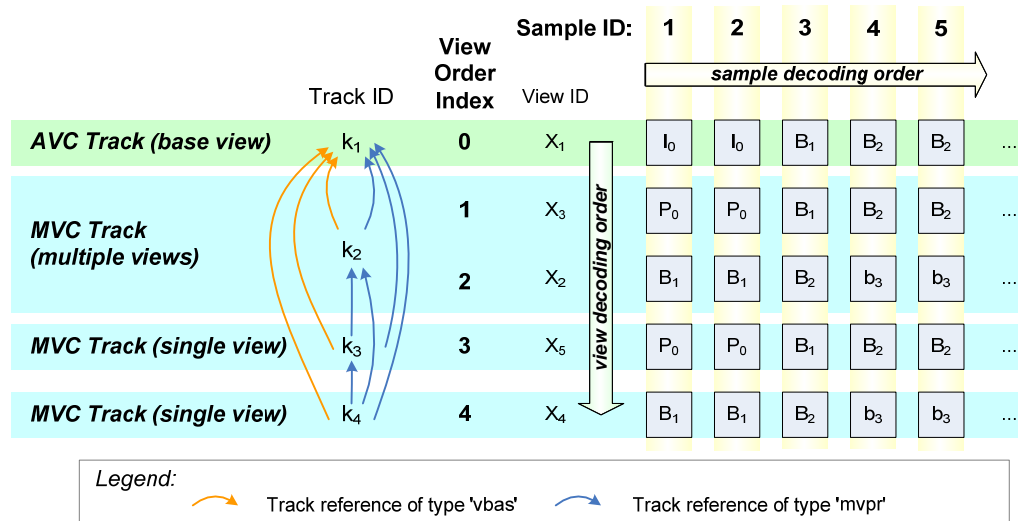


Figure 2-14 — MVC view distributed over several tracks (example)

For full functionality, a file reader may need to separate view components which are stored together as one sample, such as the view components in track  $k_2$  in Figure 2-14. This is supported by the same technique using Sample Group Definitions as described for SVC in subsection 2.2.3 above.

### 2.3.4. Re-assembly of MVC Access Units

For MVC, the re-assembly of each Access Unit, i.e., the concatenation of NAL units in decoding order, has also been tackled by the second SEA proposal. As mentioned in 2.3.1 above, views are identified by a *view\_id* field in the NAL unit header extension which is a 10-bit field. The decoding order of different view components is specified through the View Order Index. We proposed to specify a View Identifier Box as shown in Table 2-1 which holds all information needed to find and reorder samples that belong to one or more views stored in the track in which this box is contained.

A file reader would open an MP4 file and gather all sample descriptions found in the boxes `moov→trak[i]→mdia→minf→stbl→stsd`, going over all tracks  $i$ . It selects the track which contains the base view, indicated by an 'avc1' sample entry, and all tracks that reference this track by a track



reference moov→trak[i]→tref of type 'vbas'. Configuration information is retrieved from the AvcConfigurationBox and all MvcConfigurationBoxes, all SPSs and PPSs therein are aggregated, removing duplicates (if any).

The  $j^{\text{th}}$  MVC Access Unit is reassembled from the  $j^{\text{th}}$  sample of all selected tracks, each consisting of one or more view components, including related non-VCL NAL units. The first view component is taken from the AVC track, and the other view components are ordered according to the View Order Index found in the View Identifier Box located in the box path moov→trak[i]→mdia→minf→stbl→stds→mvc1→vwid.

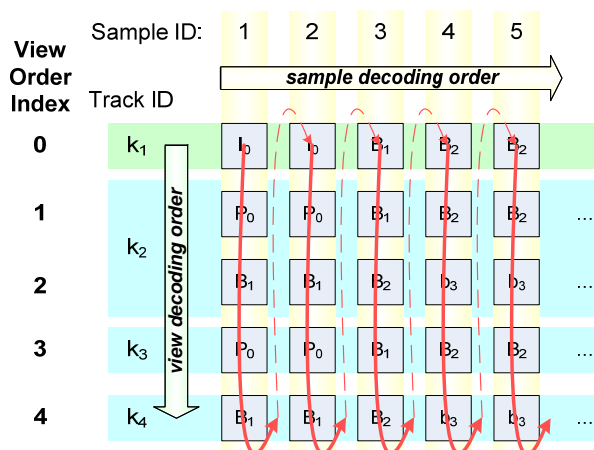


Figure 2-15 — Re-assembly of MVC Access Units when distributed over several tracks

If necessary, NAL units are re-ordered so that a block of Parameter Sets (if any) comes first, followed by another block of non-VCL NAL units (if any), followed by the VCL NAL units according to the decoding order given by the View Order Index.



**Table 2-1 — Syntax of the View Identifier Box located in the Sample Entry**

```
class ViewIdentifierBox extends FullBox ('vwid', version=0, flags)
{
    unsigned int(2)  reserved1 = 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)  reserved2 = 0;
        unsigned int(10) view_id;
        unsigned int(6)  reserved3 = 0;
        unsigned int(10) view_order_index;
        unsigned int(6)  reserved4 = 0;
        unsigned int(10) num_ref_views[i];

        for (j = 0; j < num_ref_views; j++) {
            unsigned int(6)  reserved5 = 0;
            unsigned int(10) view_id[i][j];
        }
    }
}
```

### 2.3.5. Signalling view properties in the MVC file format

#### Signalling backward compatibility for legacy AVC readers

Backward compatibility for legacy AVC readers had been one of the main requirements of the MVC extension. For MVC files, the Visual Sample Entry Box type in each track has one of the following values: "avc1", "avc2" or "mvc1".

- avc1: Tracks with a Visual Sample Entry of type "avc1" can be read by a legacy device. Support for features such as Extractors or Aggregators is not required. An AVC decoder will find the base layer NAL units and ignore MVC NAL units, if any. If this track only contains the base layer, the Sample Entry contains the AVC Configuration Record only (cf. section 2.1). If the track also contains MVC NAL units, the sample entry additionally contains an MVC Configuration Record providing the Parameter Sets necessary to decode the views other than the base view which are included in that track.
- avc2: Tracks with a Visual Sample Entry of type "avc2" contain an AVC base layer and will be read by devices that support Aggregators and Extractors. Currently, Extractors are not considered to be as useful in the context of MVC as they are in the context of SVC. As for an "avc1" Sample Entry, the AVC Configuration Record is always present, while an MVC Configuration Record is only present, if the track contains MVC data other than the base layer.
- mvc1: Tracks with a Visual Sample Entry of type "mvc1" do not contain an AVC base layer but a dependant view.

For example, track  $k_1$  in Figure 2-14 would use the "avc1" sample entry, while all other tracks of that example would use an "mvc1" sample entry.

#### Signalling acquisition properties of MVC views

The MVC standard does not limit the physical configuration the capturing devices in any way. Neither does the View ID impose any meaningful associating of the views. The only option to signal physical parameters of the camera set-up is given by Supplemental Enhancement Information (SEI) messages, namely the Multiview Acquisition Information SEI message. We proposed to store the related information in appropriate boxes within the Sample Entry. The Intrinsic Camera Parameters



Box stores parameters related to a single capturing device (e.g., its focal length) while the Extrinsic Camera Parameters Box stores its location and orientation in World Coordinates.

**Signalling association of MVC views**

Camera parameters have to be very precise in order to support high-sophisticated devices doing inter-view synthesis or other processing which operates on different view at the same time. Thus, intrinsic and extrinsic camera parameters can be rather complex to handle for a simple device which is only capable of presenting one or more separate views to the user. Consequently, the author of a bitstream will need a means for signalling which set of views should be used by a display or how the views are related to each other. The most useful information would be if two views can be presented as a stereoscopic pair by an appropriate device, and if so, an indication of left and right view - with respect to the left and right eye of a spectator - would be needed. This has been proposed in a way so that for more than two views, a certain view could be both the left view of one pair and the right view of another pair, e.g., the middle view shown in Figure 2-16. Clearly, three stereo pairs could be used as listed in Table 2-2, provided the camera parameters fulfil the requirements of the display as well as the visual needs of a spectators.

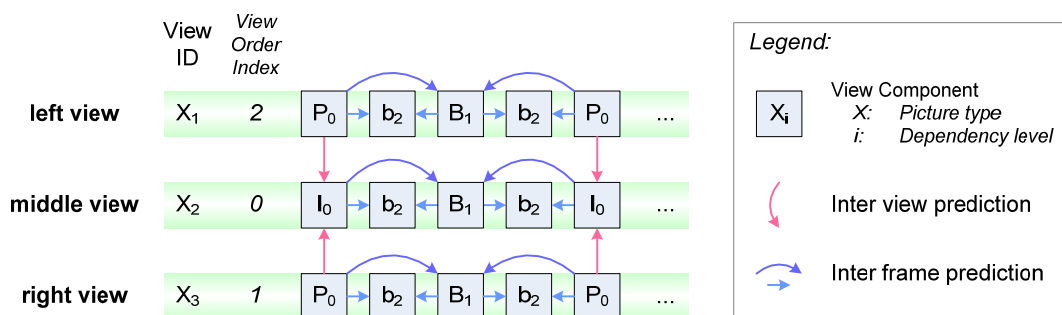


Figure 2-16 — MVC configuration with three aligned views

Table 2-2 — Different possible stereo pairs for the example shown in Figure 2-16

	View ID		
	Association #1	Association #2	Association #3
left view	X <sub>1</sub>	X <sub>2</sub>	X <sub>1</sub>
right view	X <sub>2</sub>	X <sub>3</sub>	X <sub>3</sub>

**2.3.6. MVC views stored in fragmented or external files**

As for SVC, also MVC data can be stored either in fragmented files or offloaded to external files linked to the main file by Data Reference Box entries or both fragmented and spread over different files, if this is advantageous for a certain use case.



### 3. Specific hint track generation

As described in subsection 1.5, hint tracks can be used to prepare MP4 files for RTP streaming by any media server which can be completely unaware of the specific media type(s) contained in that file. Thus, MP4 files can also be used to support novel forms of layered transmission which are still in the last stage of the standardisation process. More details on the standardisation activities of the SEA project are reported in Deliverable D7.4.1 which has also been issued M12.

#### 3.1. Multi-session transmission

A most interesting feature in the context of the SEA project will be the transmission of layered content, namely SVC and MVC streams, in a multi-session transmission mode. This has been considered by the IETF and is included in the latest draft on RTP payload format for SVC video [5]. Stream adaptation by so-called Media Aware Network Elements will be based on multi-session transmission, i.e., each tier or a certain set of tiers is sent in its own RTP session as shown in Figure 3-1. In this example, an empty packet is included in a higher layer (e.g., in  $S_3$ ) whenever an Access Unit exists in a lower layer but not in the higher layer. This is done for the sake of correct re-assembly. As an alternative, packets could be numbered across all layers which belong to the same bitstream.

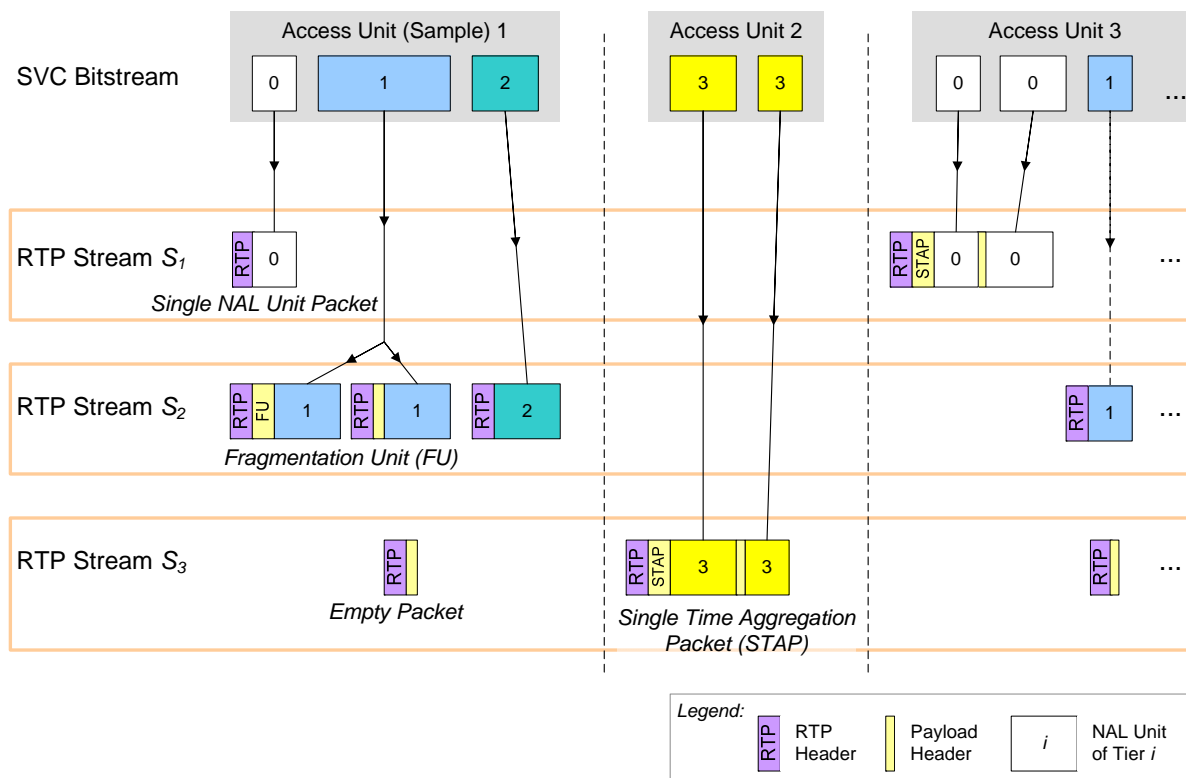
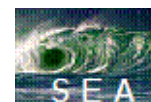


Figure 3-1: Multi-session transport of SVC

#### 3.2. Media specific signalling

Furthermore, appropriate SDP (Session Description Protocol) signalling [10] can be included in specific MP4 file format boxes. A text block stored either in a so-called Track Hint Information Box for a track or in a Movie Hint Information Box for the whole movie will be included in the SDP



information sent from the media server to the client. In a point-to-point streaming scenario, the SDP information is typically conveyed during the session setup using the Real Time Streaming Protocol (RTSP) [8]. Along with the RTP Payload Format for SVC, some media specific parameters are specified in [5] and will be standardized soon. Besides, dependencies between different layers can be signalled according to a proposed draft on the signalling of media decoding dependency in Session Description Protocol (SDP) [13]. More details on signalling and transport will be reported in Deliverable D3.3 which is due M16.

### **3.3. Example**

Figure 3-2 shows an example file which has been generated with a tool implemented within WP3 of the SEA project; the visualization of its contents is also based on the File Format library by WP3. On the left, the MP4 file format metadata structure is shown.

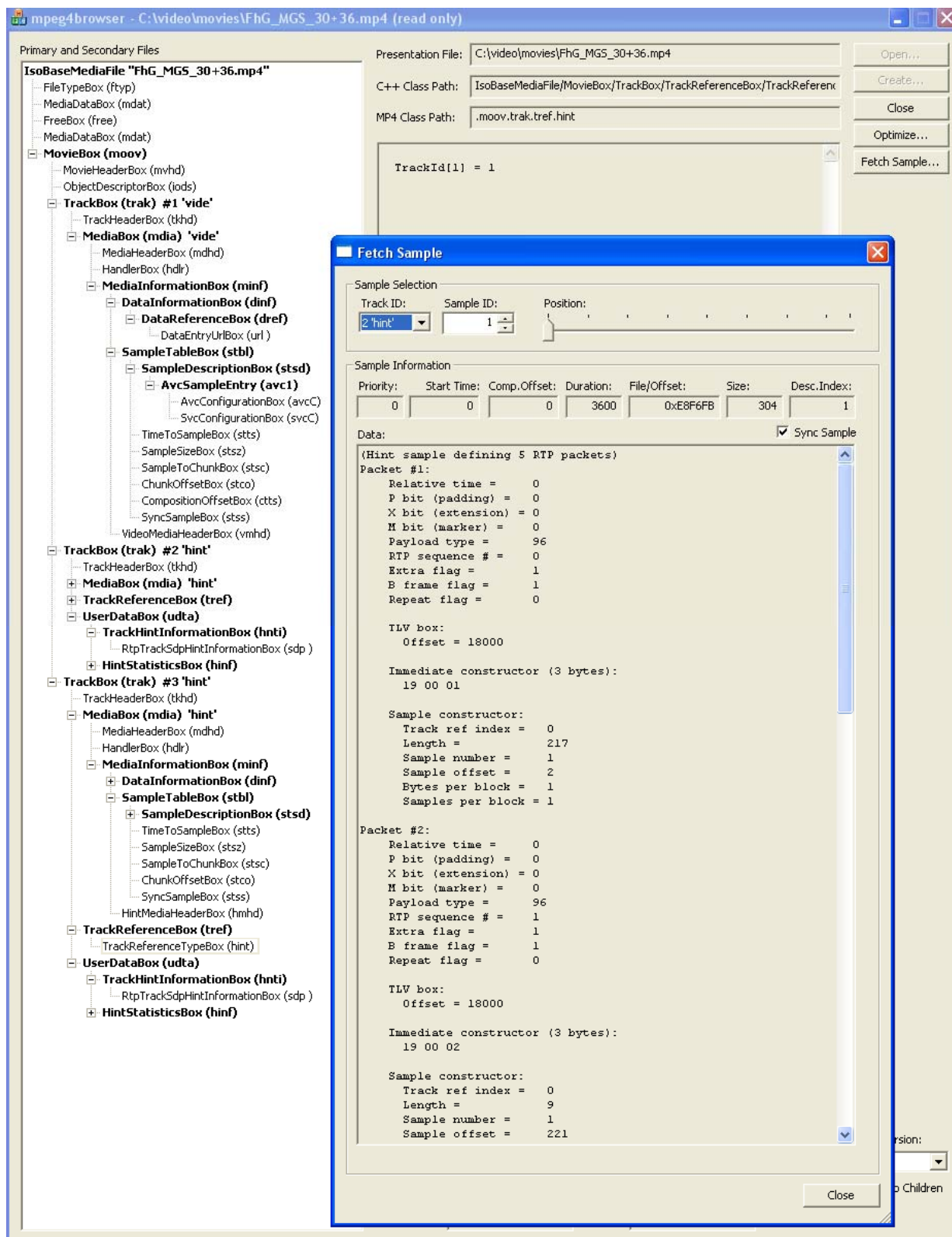


Figure 3-2: Example MP4 file prepared for multi-session transport of SVC

As can be seen, the file contains one media track with an "avc1" Sample Entry (cf. subsection 2.2.4). Additionally, two hint tracks are shown. Track 2 is the hint track for the AVC base layer, while track 3 is the hint track for the enhancement layer. The dialog window on top shows the contents of the first sample of track 2. Note that the hint sample does not contain complete RTP packets, but a



description for a media server how to construct these packets at the time a client request certain media content. The RTP header data will be generated based on the properties stated in the lines directly below the line containing the text "Packet #1". Data contained in the Immediate Constructors will be inserted directly to the RTP packet. Media data will be copied from the media track according to the values given by the Sample Constructors. Note that the media track is not referenced by its number but by the index into a table given in the. In this example, the Track Reference Type Box of type "hint" contains only one referenced track; in this case, a track ref index equal to zero can be used to reference the "one and only" entry. The media data copied to the first RTP packet are 217 bytes starting from the third byte of the first sample in the media track.



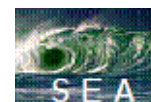
## 4. Conclusion

Within the SEA project, extensions to the AVC File Format (MPEG-4 Part 15) have been developed for SVC and MVC support. While SVC extension have been standardised as Amendment 2 to the AVC File Format in July 2008 [21], the proposed extensions for MVC have been included in a new Proposed Draft Amendment (PDAM) to the same standard. These new extensions enable numerous use cases both for SVC and MVC contents. Not only can different SVC layers or MVC views be stored in an MP4 file using a variety of track structures, but these files also enable streaming of the media in various configurations, using existing servers which are agnostic to media specific features such as coding dependencies between different layers of an SVC representation or different views of an MVC video sequence.



## References

- [1] ITU-T Recommendation H.264 (11/2007), Advanced video coding for generic audiovisual services | ISO/IEC 14496-10:2008, Information technology – Coding of audio-visual objects – Part 10: Advanced video coding
- [2] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the scalable video coding extension of the H.264/AVC standard”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, Sep. 2007, pp. 1103-1120
- [3] ISO/IEC 14496-2:2004/Amd.1:2004, Information technology – Coding of audio-visual objects – Part 2: Visual, Amendment 1: Error resilient simple scalable profile
- [4] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", IETF STD 0064, RFC 3550, July 2003, <http://tools.ietf.org/html/rfc3550>
- [5] S. Wenger, Y.-K. Wang, T. Schierl, A. Eleftheriadis, "RTP payload format for SVC video", work in progress, Internet Engineering Task Force (IETF), Audio Video Transport Group (avt), December 2008, <http://tools.ietf.org/html/draft-ietf-avt-rtp-svc-16>
- [6] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, D. Singer, "RTP Payload Format for H.264 Video", Internet Engineering Task Force (IETF), Audio Video Transport Group (avt), November 2008, <http://tools.ietf.org/html/rfc3984>
- [7] Y.-K. Wang, R. Even, T. Kristensen, "RTP Payload Format for H.264 Video", work in progress, Internet Engineering Task Force (IETF), Audio Video Transport Group (avt), November 2008, <http://tools.ietf.org/html/draft-ietf-avt-rtp-rfc3984bis-01>
- [8] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming Protocol (RTSP)", IETF RFC 2326, April 1998, <http://tools.ietf.org/html/rfc2326>
- [9] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, M. Stiemerling "Real Time Streaming Protocol 2.0 (RTSP)", work in progress, Internet Engineering Task Force (IETF), Multiparty Multimedia Session Control (mmusic) , 3 Nov 2008, <http://tools.ietf.org/html/draft-ietf-mmusic-rfc2326bis-19>
- [10] M. Handly, V. Jacobson and C. Perkins, "SDP: Session Description Protocol", IETF RFC 4566, July 2006, <http://tools.ietf.org/html/rfc4566>
- [11] H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", IETF RFC 3261, June 2002, <http://tools.ietf.org/html/rfc3261>
- [12] T. Schierl, J. Lennox, "Multi-Session and Multi-Source Transmission in the Real-Time Transport Protocol (RTP)", work in progress, Internet Engineering Task Force (IETF), Audio Video Transport Group (avt), October 27, 2008, <http://tools.ietf.org/html/draft-schierl-avt-rtp-multi-session-transmission-00>
- [13] T. Schierl, S. Wenger, "Signaling media decoding dependency in Session Description Protocol (SDP)", work in progress, Internet Engineering Task Force (IETF), Multiparty Multimedia Session Control (mmusic) , 21 Oct 2008, <http://tools.ietf.org/html/draft-ietf-mmusic-decoding-dependency-05>
- [14] Stephan Wenger, Ye-Kui Wang, and Thomas Schierl, "Transport and Signaling of SVC in IP Networks", *IEEE Transactions on Circuits and Systems for Video Technology*, Special Issue on Scalable Video Coding, vol. 17, no. 9, pp. 1164-1173, September 2007



- [15] ETSI EN 301 192 Digital Video Broadcasting (DVB); DVB specification for data broadcasting, April 2008
- [16] ISO/IEC 13818-1: Information technology – Generic coding of moving pictures and associated audio information; Systems
- [17] ISO/IEC 13818-1:2007/AMD3 – Transport of SVC video over ITU-T Rec H.222.0 | ISO/IEC 13818-1, to be published Feb 2009
- [18] J. van der Meer, D. Mackie, V. Swaminathan, D. Singer, P. Gentric, RTP Payload Format for Transport of MPEG-4 Elementary Streams, November 2003, IETF RFC 3640, <http://tools.ietf.org/html/rfc3640>
- [19] Information technology — Coding of audio-visual objects — Part 12: ISO base media file format, ISO/IEC 14496-12:2008 (3rd edition)
- [20] Information technology — Coding of audio-visual objects — Part 15: Advanced Video Coding (AVC) file format, ISO/IEC 14496-15:2004 (E)
- [21] Information technology — Coding of audio-visual objects — Part 15: Advanced Video Coding (AVC) file format, AMENDMENT 2: File format support for Scalable Video Coding, ISO/IEC 14496-15:2004/AMD2:2008(E)
- [22] Digital Video Broadcast (DVB), File Format Specification for the Storage and Playback of DVB Services, work in progress, doc TM-FF0020
- [23] K. Grüneberg, T. Schierl, "On MVC File Format", 84th MPEG Meeting - ISO/IEC JTC1/SC29/WG11, Archamps, France, MPEG84/M15356, April 2008
- [24] K. Grüneberg, T. Schierl, "On MVC File Format", 85th MPEG Meeting - ISO/IEC JTC1/SC29/WG11, Hanover, Germany, MPEG85/M15600, July 2008
- [25] K. Grüneberg, T. Schierl, "On MVC File Format", 86th MPEG Meeting - ISO/IEC JTC1/SC29/WG11, Busan, Korea, MPEG86/M15874, October 2008
- [26] K. Grüneberg, T. Schierl, D. Singer, "Working Draft 1.0 for the Amendment 3 (MVC File Format) to 14496-15 (2004) (AVC File Format)", MPEG Meeting - ISO/IEC JTC1/SC29/WG11, Archamps, France, MPEG85/N9827, April 2008.
- [27] K. Grüneberg, T. Schierl, M. Hannuksela, Y.-K. Wang, Y. Chen, D. Singer, "Working Draft 2.0 for the Amendment 3 (MVC File Format) to 14496-15 (2004) (AVC File Format)", MPEG Meeting - ISO/IEC JTC1/SC29/WG11, Hanover, Germany, MPEG85/N10062, July 2008.
- [28] Philipp Merkle, Aljoscha Smolic, Karsten Müller, and Thomas Wiegand, "Efficient Prediction Structures for Multiview Video Coding", *IEEE Transactions on Circuits and Systems for Video Technology*, Special Issue on Multi-view Video Coding and 3DTV, vol. 17, no. 11, November 2007, pp. 1461-1473
- [29] Aljoscha Smolic, Karsten Müller, Kristina Dix, Philipp Merkle, Peter Kauff, and Thomas Wiegand, "Intermediate View Interpolation Based on Multiview Video Plus Depth for Advanced 3D Video Systems", *Proceedings of 15th International Conference on Image Processing (ICIP 2008)*, San Diego, CA, USA, October 2008.
- [30] Information technology — Coding of audio-visual objects — Part 10: Advanced Video Coding, AMENDMENT 1: Multiview Video Coding ISO/IEC 14496-10:2008/FDAM 1:2008(E), ISO/IEC JTC1 doc ref SC 29 N 9783 (FDAM), to be published March 2009