

MPEG-4 Parts 12/14/15, MPEG-7, and MPEG-21

Shih-Hsuan Yang
CSIE Department, NTUT



Contents


MPEG-4 File Format

MPEG-7

MPEG-21



MPEG-4 File Format (ISO 14496-12/14/15)



MPEG-4 Part 12 (ISO base media file format) Overview

The ISO base media file format defines a general structure for timed multimedia files such as video and audio. It is used as the basis for other media file formats (e.g. container formats MP4 and 3GP).

The technically identical text is published as ISO/IEC 14496-12 for MPEG-4, and as ISO/IEC 15444-12 for JPEG 2000.

ISO base media file format is directly based on Apple's QuickTime container format and was generalized from MP4 file format.

MPEG-4 Part 12 (ISO base media file format) Overview

The first MP4 file format specification was created on the basis of the QuickTime format specification published in 2001. The MP4 file format known as "version 1" was published in 2001 as ISO/IEC 14496-1:2001, as revision of the MPEG-4 Part 1: Systems.

In 2003, the first version of MP4 file format was revised and replaced by MPEG-4 Part 14: MP4 file format (ISO/IEC 14496-14:2003), commonly known as MPEG-4 file format "version 2". The MP4 file format was generalized into the ISO Base Media File format (ISO/IEC 14496-12), which defines a general structure for time-based media files.



File Structure of ISO Base Media File Format

The file structure is object-oriented; a file can be decomposed into constituent objects very simply, and the structure of the objects inferred directly from their type.

ISO base media file format contains the timing, structure, and media information for timed sequences of media data, such as audio-visual presentations. The file structure is object-oriented. A file can be decomposed into constituent objects very simply, and the structure of the objects inferred directly from their type.



Terminologies and Core Concepts

In the file format, the overall presentation is called a **movie**. It is logically divided into **tracks**; each track represents a timed sequence of media (frames of video, for example). Within each track, each timed unit is called a **sample**; this might be a frame of video or audio. Samples are implicitly numbered in sequence. This specification uses the word sample to mean a timed frame or unit of data. Each track has one or more **sample descriptions**; each sample in the track is tied to a description by reference. The description defines how the sample may be decoded.

Media in ISO File Format

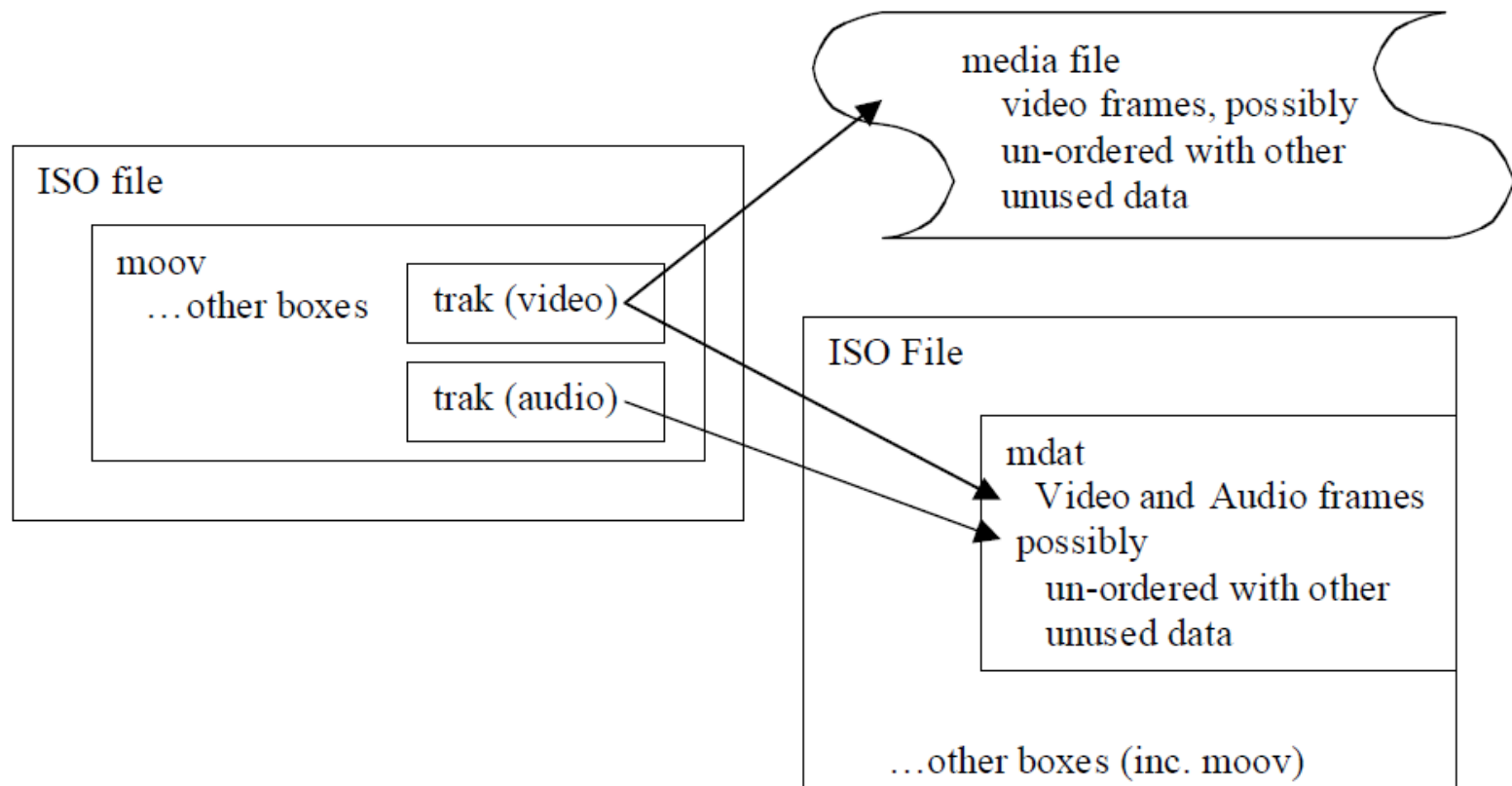


Figure 2 — Content Creation File



Terminologies and Core Concepts

Unlike many other multi-media file formats, this format, separates several concepts that are often linked. In particular: The physical structure of the file is not tied to the physical structures of the media itself. Neither the physical structure of the file, nor the layout of the media, is tied to the time ordering of the media. This means that there are file structures that describe the placement and timing of the media.

Terminologies and Core Concepts

All the data within a conforming file is encapsulated in **boxes** (called **atoms** in predecessors of this file format). All the metadata, including that defining the placement and timing of the media, is contained in structured boxes. The media data (frames of video, for example) is referred to by this metadata. Tracks can be of various kinds. **Video tracks** contain samples that are visual; **audio tracks** contain audio media. **Hint tracks** are rather different; they contain instructions for a streaming server in how to form packets for a streaming protocol, from the media tracks in a file. Hint tracks can be ignored when a file is read for local playback; they are only relevant to streaming.

Boxes of ISO Base Media File Format

Files conforming to the ISO base media file format are formed as a series of objects, called "boxes".

All data is contained in boxes and there is no other data within the file.

The 'box' is an object-oriented building block (Java-like syntax), defined by a unique type identifier and length.

The 'box' was called "atom" in some specifications (e.g. the first definition of MP4 file format).

The definition of a 'box' is somehow similar to a subroutine. That means, any syntax involving a 'box' will automatically have the fields size and type, as illustrated in the following example.

Box (Basic Definition)

```
aligned(8) class Box (unsigned int(32) boxtype,  
optional unsigned int(8)[16] extended_type) {  
    unsigned int(32) size;  
    unsigned int(32) type = boxtype;  
    if (size==1) {  
        unsigned int(64) largesize;  
    } else if (size==0) {  
        // box extends to end of file  
    }  
    if (boxtype=='uuid') {  
        unsigned int(8)[16] usertype = extended_type;  
    }  
}
```



Boxes of ISO Base Media File Format

Boxes start with a header which gives both size and type. The header permits compact or extended size (32 or 64 bits) and compact or extended types (32 bits or full UUIDs). The standard boxes all use compact types (32-bit) and most boxes will use the compact (32-bit) size.

The size is the entire size of the box, including the size and type header, fields, and all contained boxes. This facilitates general parsing of the file.

The fields in the objects are stored with the most significant byte first, commonly known as network byte order or big-endian format.

Box Example (FileTypeBox)

```
aligned(8) class FileTypeBox extends Box('ftyp') {  
    unsigned int(32) major_brand;  
    unsigned int(32) minor_version;  
    unsigned int(32) compatible_brands[]; // to end of the box  
}
```

// extends = inheritance in Java

// Meaning: call “box” subroutine with the parameter “ftyp”. Then, three 32-bit fields follow the box routine.

The above syntax is implemented as:

4-byte size (total number of bytes in the box, including these four bytes here) (= 20, 4+4+4+4+4.)

4-byte type (= 'ftyp', a FileTypeBox)

4-byte major_brand

4-byte minor_version

4-byte) compatible_brands[];

Box Example (FullBox)

```
aligned(8) class FullBox(unsigned int(32) boxtype,  
    unsigned int(8) v, bit(24) f)  
    extends Box(boxtype) {  
        unsigned int(8) version = v;  
        bit(24) flags = f;  
    }
```

The above syntax is implemented as:

4-byte size (total number of bytes in the box, including these four bytes here) (= 12, 4+4+1+3.)

4-byte type (= boxtype)

1-byte version (= v)

3-byte flags (= f)

Box Example (stsd)

```
aligned(8) class SampleDescriptionBox (unsigned int(32)
handler_type)
extends FullBox('stsd', 0, 0){
int i;
    unsigned int(32) entry_count;
    for (i = 1 ; i < entry_count ; i++){
        switch (handler_type){
            case 'soun': // for audio tracks
                AudioSampleEntry();
                break;
            case 'vide': // for video tracks
                VisualSampleEntry();
                break;
            case 'hint': // Hint track
                HintSampleEntry();
                break;
        }
    }
}
```


Parsing a stsd box

4 byte `size` = 16

4 byte `type` = 'stsd'

1 byte `version` = 0

3 byte `flags` = 0 0 0

4 byte `entry_count`

Note that `handler_type` should be obtained before parsing this box.

Box Example (SampleEntry)

```
aligned(8) abstract class SampleEntry (unsigned int(32) format)
extends Box(format){
    const unsigned int(8)[6] reserved = 0;
    unsigned int(16) data_reference_index;
}

class AudioSampleEntry(codingname) extends SampleEntry (codingname){
    const unsigned int(32)[2] reserved = 0;
    template unsigned int(16) channelcount = 2;
    template unsigned int(16) samplesize = 16;
    unsigned int(16) pre_defined = 0;
    const unsigned int(16) reserved = 0;
    template unsigned int(32) samplerate = {timescale of media}<<16;
}
```

Parsing AudioSampleEntry

Therefore, following the entry count should be a series of boxes in the following form

4 byte `size` = 36

4 byte `type` = format (syntax from box)

6 byte: 0 0 0 0 0 0 (reserved from abstract class `SampleEntry`)

2 byte `data_ref_index` (syntax from abstract class)

8 byte: 0 0 0 0 0 0 0 0 (reserved from abstract class `SampleEntry`)

2 byte `channelcount` (# of channels)

2 byte: `samplesize` (in bits)

2 byte: 0 (pre-defined)

2 byte: 0 (reserved)

4 byte: `sample rate` (time scale, syntax from audio sample entry)

Box Structure

Table 1 — Box types, structure, and cross-reference

ftyp					*	4.3	file type and compatibility	
pdin						8.43	progressive download information	
moov					*	8.1	container for all the metadata	
	mvhd				*	8.3	movie header, overall declarations	
	trak				*	8.4	container for an individual track or stream	
		tkhd			*	8.5	track header, overall information about the track	
		tref				8.6	track reference container	
		edts				8.25	edit list container	
			elst			8.26	an edit list	
		mdia			*	8.7	container for the media information in a track	
			mdhd		*	8.8	media header, overall information about the media	
			hdlr		*	8.9	handler, declares the media (handler) type	
			minf		*	8.10	media information container	
				vmhd		8.11.2	video media header, overall information (video track only)	
				smhd		8.11.3	sound media header, overall information (sound track only)	
				hmhd		8.11.4	hint media header, overall information (hint track only)	
				nmhd		8.11.5	Null media header, overall information (some tracks only)	
				dinf	*	8.12	data information box, container	
					dref	*	8.13	data reference box, declares source(s) of media data in track
				stbl	*	8.14	sample table box, container for the time/space map	
					stsd	*	8.16	sample descriptions (codec types, initialization etc.)
					stts	*	8.15.2	(decoding) time-to-sample
					ctts		8.15.3	(composition) time to sample
					stsc	*	8.18	sample-to-chunk, partial data-offset information
					stsz		8.17.2	sample sizes (framing)
					stz2		8.17.3	compact sample sizes (framing)
					stco	*	8.19	chunk offset, partial data-offset information
					co64		8.19	64-bit chunk offset
					stss		8.20	sync sample table (random access points)
					stsh		8.21	shadow sync sample table
					padb		8.23	sample padding bits
					stdp		8.22	sample degradation priority
					sdtg		8.40.2	independent and disposable samples
					sbgp		8.40.3.2	sample-to-group
					sgpd		8.40.3.3	sample group description
					subs		8.42	sub-sample information
	mvex					8.29	movie extends box	
		mehd				8.30	movie extends header box	
		trex			*	8.31	track extends defaults	
	ipmc					8.45.4	IPMP Control Box	
moof						8.32	movie fragment	
	mfhd				*	8.33	movie fragment header	
	traf					8.34	track fragment	
		tfhd			*	8.35	track fragment header	
		trun				8.36	track fragment run	
		sdtg				8.40.2	independent and disposable samples	
		sbgp				8.40.3.2	sample-to-group	
		subs				8.42	sub-sample information	
mfra						8.37	movie fragment random access	
	tfra					8.38	track fragment random access	
	mfro				*	8.39	movie fragment random access offset	
mdat						8.2	media data container	

File Type Box (ftyp)

4.3.2 Syntax

```
aligned(8) class FileTypeBox
    extends Box('ftyp') {
    unsigned int(32)  major_brand;
    unsigned int(32)  minor_version;
    unsigned int(32)  compatible_brands[];    // to end of the box
}
```

4.3.3 Semantics

This box identifies the specifications to which this file complies.

Each brand is a printable four-character code, registered with ISO, that identifies a precise specification.

`major_brand` – is a brand identifier

`minor_version` – is an informative integer for the minor version of the major brand

`compatible_brands` – is a list, to the end of the box, of brands



File Type Box (ftyp)

In order to identify the specifications to which a file based on ISO base media file format complies, brands are used as identifiers in the file format. They are set in a box named File Type Box ('ftyp'), which must be placed in the beginning of the file. A brand might indicate the type of encoding used, how the data of each encoding is stored, constraints and extensions that are applied to the file, the compatibility, or the intended usage of the file. Brands are a printable four-character codes.

Movie Box (moov)

A container box whose sub-boxes define the metadata for a presentation ('moov').

Box Type: 'moov'
Container: File
Mandatory: Yes
Quantity: Exactly one

The metadata for a presentation is stored in the single Movie Box which occurs at the top-level of a file. Normally this box is close to the beginning or end of the file, though this is not required.

8.1.2 Syntax

```
aligned(8) class MovieBox extends Box('moov') {  
}
```

Media Data Box (mdat)

A container box which can hold the actual media data for a presentation.

Box Type: 'mdat'
Container: File
Mandatory: No
Quantity: Any number

This box contains the media data. In video tracks, this box would contain video frames. A presentation may contain zero or more Media Data Boxes. The actual media data follows the type field; its structure is described by the metadata (see particularly the sample table, subclause 8.14, and the item location box, subclause 8.44.3).

In large presentations, it may be desirable to have more data in this box than a 32-bit size would permit. In this case, the large variant of the size field, above in subclause 6.2, is used.

There may be any number of these boxes in the file (including zero, if all the media data is in other files). The metadata refers to media data by its absolute offset within the file (see subclause 8.19, the Chunk Offset Box); so Media Data Box headers and free space may easily be skipped, and files without any box structure may also be referenced and used.

8.2.2 Syntax

```
aligned(8) class MediaDataBox extends Box('mdat') {  
    bit(8) data[];  
}
```

8.2.3 Semantics

data is the contained media data



Physical Structure of the Media

The boxes that define the layout of the media data are found in the sample table. These include the data reference, the sample size table, the sample to chunk table, and the chunk offset table. Between them, these tables allow each sample in a track to be both located, and its size to be known.

The data references permit locating media within secondary media files. This allows a composition to be built from a 'library' of media in separate files, without actually copying the media into a single file. This greatly facilitates editing, for example.

The tables are compacted to save space. In addition, it is expected that the interleave will not be sample by sample, but that several samples for a single track will occur together, then a set of samples for another track, and so on. These sets of contiguous samples for one track are called **chunks**. Each chunk has an offset into its containing file (from the beginning of the file). Within the chunk, the samples are contiguously stored.



Streaming

The ISO base media file format supports streaming of media data over a network as well as local playback. A file that supports streaming includes information about the data units to stream. This information is placed in additional tracks of the file called "hint" tracks. Existing media can be easily made streamable for other specific protocols by the addition of an appropriate hint tracks. The media data itself need not be reformatted in any way. The streams sent by the servers under the direction of the hint tracks, need contain no trace of file-specific information. When the presentation is played back locally (not streamed), the hint tracks may be ignored. Hint tracks may be created by an authoring tool, or may be added to an existing file (presentation) by a hinting tool. In media authored for progressive download the moov atom, which contains the index of frames should precede the movie data mdat atom.

Streaming

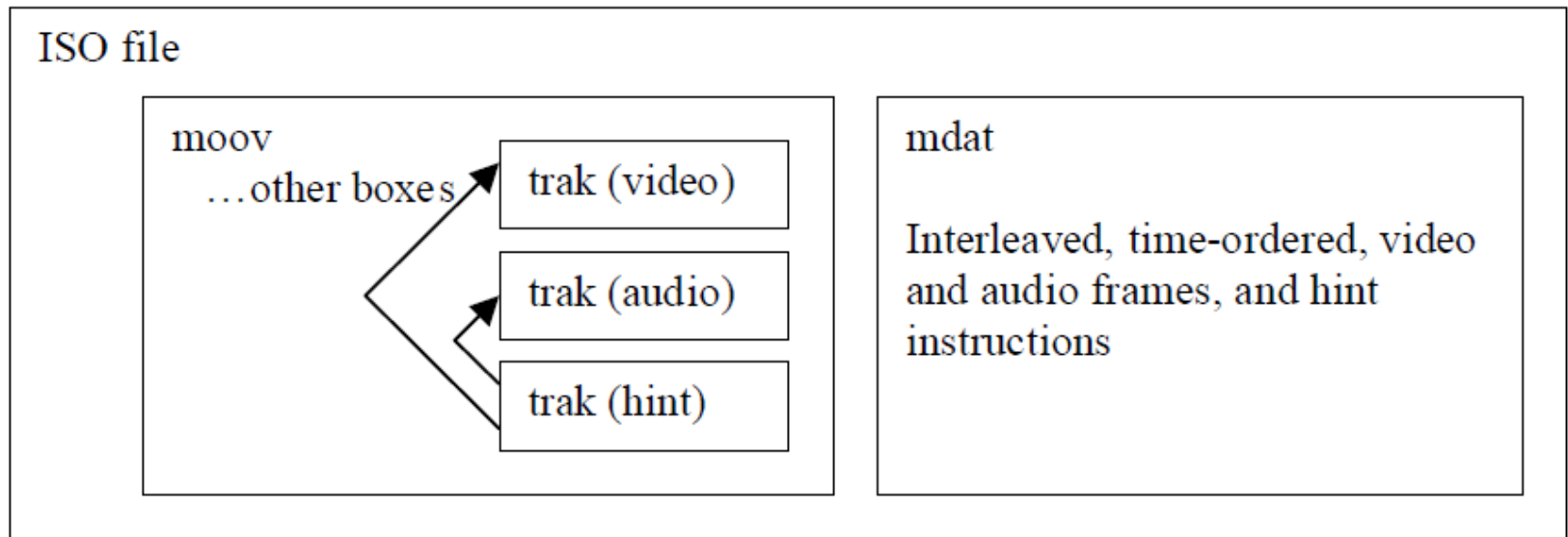


Figure 3 — Hinted Presentation for Streaming



MPEG-4 Part 14 Overview

MPEG-4 Part 14 (ISO/IEC 14496-14) defines MP4 as an instance of the ISO Media File format (ISO/IEC 14496-12), which is a multimedia container format standard specified as a part of MPEG-4. It is most-commonly used to store digital audio and digital video streams, especially those defined by MPEG, but also can be used to store other data such as subtitles and still images. Like most modern container formats, MPEG-4 Part 14 allows streaming over the Internet.

The official filename extension for MPEG-4 Part 14 files is .mp4, thus the container format is often referred to simply as MP4. Devices that play .mp4 files are referred to as MP4 players.



MPEG-4 Part 14 Overview

The official filename extension for MPEG-4 Part 14 files is .mp4, thus the container format is often referred to simply as MP4. Devices that play .mp4 files are referred to as MP4 players.

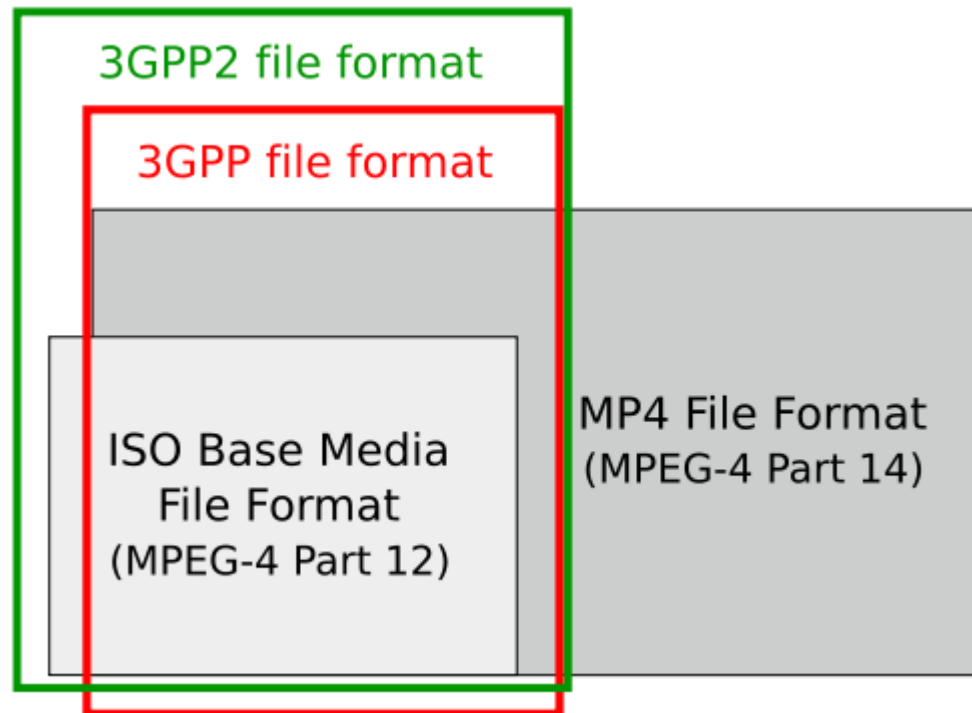
MPEG-4 Part 14 was based on Apple's QuickTime container format. Although MPEG-4 Part 14 still resembles the QuickTime format, it has been improved upon in many different ways.



MPEG-4 Part 14 Overview

The existence of two different file extensions for naming audio-only MP4 files has been a source of confusion among users and multimedia playback software. Since MPEG-4 is a container format, MP4 files may contain any number of audio, video, and even subtitle streams, making it impossible to guess what type of streams are present in an MP4 file based on its filename extension alone. In response, Apple Computer started using and popularizing the .m4a file extension. Software capable of audio/video playback should recognize files with either .m4a or .mp4 file extensions, as would be expected, as there are no file format differences between the two. Most software capable of creating MPEG-4 audio will allow the user to choose the filename extension of the created MPEG-4 files.

Relationship Among MP4 and 3GPP





File Identification

The brand 'mp41' is defined as identifying version 1 (ISO/IEC 14496-1:2001), and the brand 'mp42' identifies version 2 (ISO/IEC 14496-14); at least one of these brands shall appear in the compatible-brands list in the file-type box, in all files conforming to this specification.

The preferred file extension is '.mp4'.

The MIME types video/mp4, audio/mp4 are used as defined in the appropriate RFC.



MP4's Extensions to the Base Media Format

Object Descriptor Box (iods)

MP4 defines the following additional values for reference-type: dpnd, ipir, mpod, sync

Track Header Box

Handler Reference Types

MPEG-4 Media Header Boxes

Degradation Priority Values

Sample Description Boxes (stsd)

Box Types: 'mp4v', 'mp4a', 'mp4s'

Container: Sample Table Box ('stbl')

For visual streams, a VisualSampleEntry is used; for audio streams, an AudioSampleEntry. For all other MPEG-4 streams, a MpegSampleEntry is used. Hint tracks use an entry format specific to their protocol, with an appropriate name.

For all the MPEG-4 streams, the data field stores an ES_Descriptor with all its contents. Multiple entries in the table imply the occurrence of ES_DescriptorUpdate commands.

Sample Description Boxes

For hint tracks, the sample description contains appropriate declarative data for the protocol being used, and the format of the hint track. The definition of the sample description is specific to the streaming protocol. For visual streams, Annex K subclause 3.1 of the video specification requires that configuration information (e.g. the video sequence header) be carried in the decoder configuration structure, and not in stream. Since MP4 is a systems structure, it should be noted that that means that these headers (video object sequence, and so on) shall be in the ES_descriptor in the sample description, and not in the media samples themselves.

ESDBox

```
aligned(8) class ESDBox
    extends FullBox('esds', version = 0, 0) {
    ES_Descriptor ES;
}
    // Visual Streams

class MP4VisualSampleEntry() extends VisualSampleEntry ('mp4v'){
    ESDBox ES;
}
    // Audio Streams

class MP4AudioSampleEntry() extends AudioSampleEntry ('mp4a'){
    ESDBox ES;
}
    // all other Mpeg stream types
class MpegSampleEntry() extends SampleEntry ('mp4s'){
    ESDBox ES;
}

aligned(8) class SampleDescriptionBox (unsigned int(32) handler_type)
    extends FullBox('stsd', 0, 0){
    int i ;
    unsigned int(32) entry_count;
    for (i = 0 ; i < entry_count ; i++){
        switch (handler_type){
            case 'soun': // AudioStream
                AudioSampleEntry();
                break;
            case 'vide': // VisualStream
                VisualSampleEntry();
                break;
            case 'hint': // Hint track
                HintSampleEntbry();
                break;
            default :
                MpegSampleEntry();
                break;
        }
    }
}
```

Embedded Media Data

Almost any kind of data can be embedded in MPEG-4 Part 14 files through private streams; the widely-supported codecs and additional data streams are:

Video: MPEG-4 Part 10 (also known as H.264 and MPEG-4 AVC), MPEG-4 Part 2, MPEG-2, and MPEG-1.

Audio: AAC (also known as MPEG-2 Part 7), MP3 (also known as MPEG-1 Audio Layer 3), MPEG-4 Part 3, MP2 (also known as MPEG-1 Audio Layer 2), MPEG-1 Part 3, CELP (speech), TwinVQ (very low bitrates), SAOL (MIDI).

Subtitles: MPEG-4 Timed Text (also known as 3GPP Timed Text).



MPEG-4 Part 15: Advanced Video Coding (AVC) File Format

The Advanced Video Coding (AVC) standard 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. 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;
- inherit all the use cases and features of the ISO Base Media File Format on which MP4 and MJ2 are based.



MPEG-4 Part 15 Overview

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, which 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.



AVC's Extensions to the Base Media Format: File Identification

The brand 'avc1' shall be used to indicate that extensions conformant with this section are used in a file. The use of 'avc1' as a major-brand may be permitted by specifications; in that case, that specification defines the file extension and required behavior.

Independent and Disposable Samples Box

Box Types: 'sdtb'

Container: Sample Table Box ('stbl')

This optional table answers three questions about sample dependency:

Does this sample depend on others (is it an I-picture)?

Do no other samples depend on this one?

Does this sample contain multiple (redundant) encodings of the data at this time-instant (possibly with different dependencies)?



Sample Groups

A sample grouping is an assignment of each sample in a track to be a member of one sample group, based on a grouping criterion.

A sample group in a sample grouping is not limited to being contiguous samples and may contain non-adjacent samples. As there may be more than one sample grouping for the samples in a track, each sample grouping has a type field to indicate the type of grouping. For example, a file might contain two sample groupings for the same track: one based on an assignment of sample to layers and another to subsequences.



Random Access Recovery Points

In some coding systems it is possible to random access into a stream and achieve correct decoding after having decoded a number of samples. This is known as gradual decoding refresh. For example, in video, the encoder might encode intra-coded macroblocks in the stream, such that it knows that within a certain period the entire picture consists of pixels that are only dependent on intra-coded macroblocks supplied during that period.

Samples for which such gradual refresh is possible are marked by being a member of this group. The definition of the group allows the marking to occur at either the beginning of the period or the end. However, when used with a particular media type, the usage of this group may be restricted to marking only one end (i.e. restricted to only positive or negative roll values). A roll-group is defined as that group of samples having the same roll distance.



AVC 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 within the AVC file format. Two types of elementary streams are defined for this purpose:

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 message and access unit delimiter NAL units.

Parameter set elementary streams shall contain only sequence and picture parameter set NAL units.



AVC Elementary Stream Structure

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.

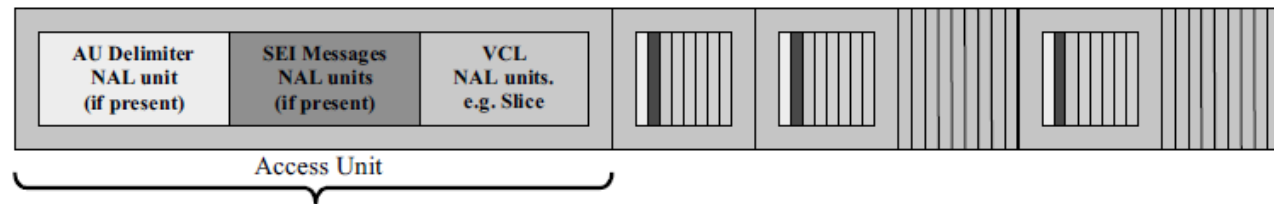
Video elementary stream and parameter set elementary stream: In this case, sequence and picture parameter set NAL units shall be transmitted only in the parameter set elementary stream and shall neither be present in the sample descriptions nor the AVC samples of the video elementary stream.

Types of NAL Units

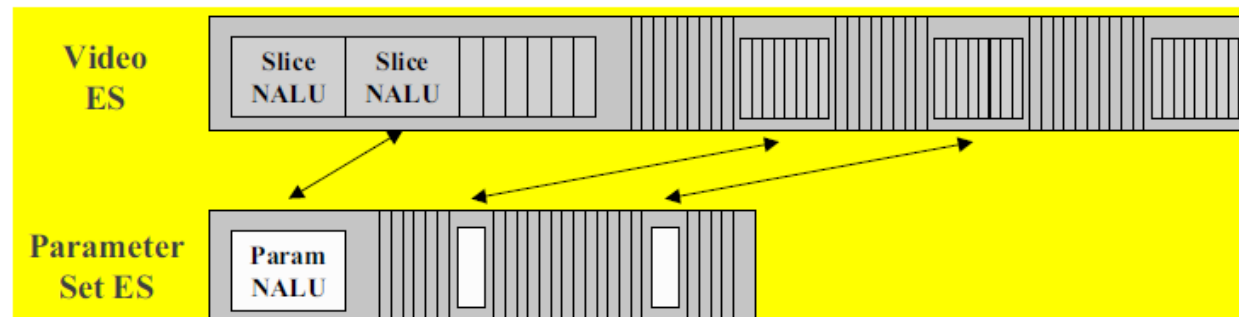
Table 1 — NAL unit types in elementary streams

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	No
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 – 23	Reserved	No	No
24 – 31	Unspecified	Not specified by this part of ISO/IEC 14496	Not specified by this part of ISO/IEC 14496

AVC Elementary Stream Structure



(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

Sample and Configuration Definition

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

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.

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

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.



The Canonical Stream Format

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

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.

AVC Sample Structure Definition

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.

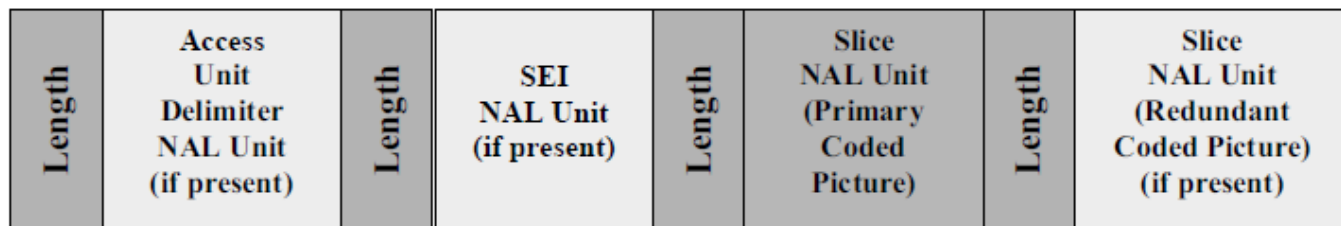


Figure 2 — The structure of an AVC sample



MPEG-7 “Multimedia Content Description Interface”

Evolution of Video Coding

Models	Coded Information	Examples
Pixels		PCM
Block	Transform coefficients or prediction error	Transform coding Predictive coding
Moving blocks	Motion vectors and prediction error	Block-based coding H.261/H.263, MPEG- 1/2
Moving objects	Shape, motion, and texture of objects	Object-based coding MPEG-4
A/V objects	Description	MPEG-7

Better modeling implies higher compression, more content accessibility, more complexity, and less error resilience.



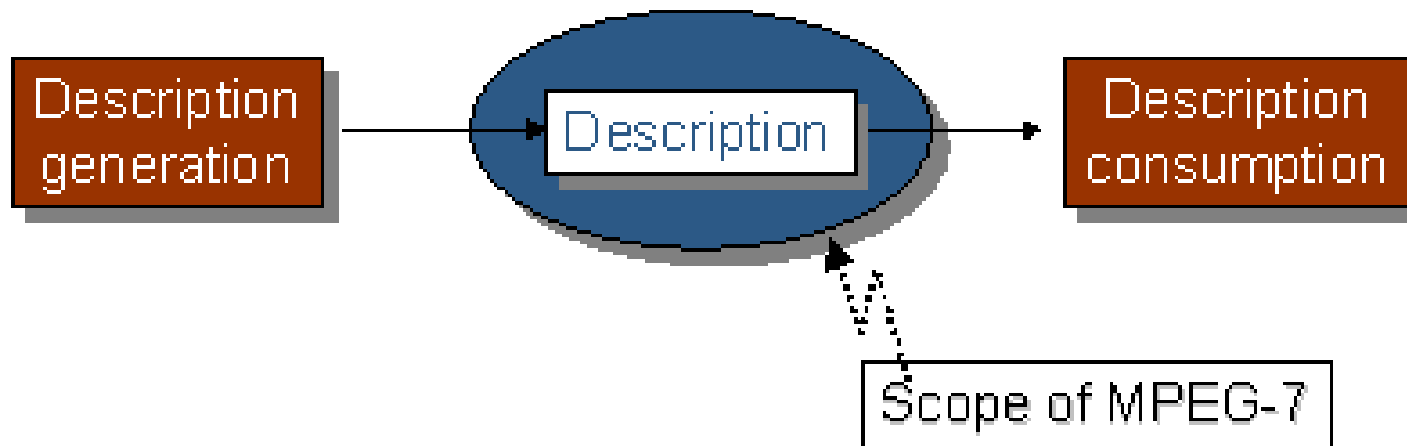
Overview of MPEG-7

MPEG-7 is very different from the previous MPEG standards that address the coded representations of audio-visual information (that's the reason that the numbers 5 and 6 are skipped). Instead, MPEG-7 focuses mainly on information about audio-visual information (metadata).

An incommensurable amount of audiovisual information is becoming available in digital form, in digital archives, on the World Wide Web, in broadcast data streams and in personal and professional databases, and this amount is growing. In spite of the fact that users have increasing access to these resources, identifying and managing them efficiently is becoming more difficult. MPEG-7 is a standard for describing the multimedia content data.

Overview of MPEG-7

MPEG-7 offers a comprehensive set of audiovisual Description Tools to create descriptions, which will form the basis for applications enabling the needed effective and efficient access (search, filtering and browsing) to multimedia content. Figure 1 shows a highly abstract block diagram of a possible MPEG-7 processing chain, to explain the scope of the MPEG-7 standard.





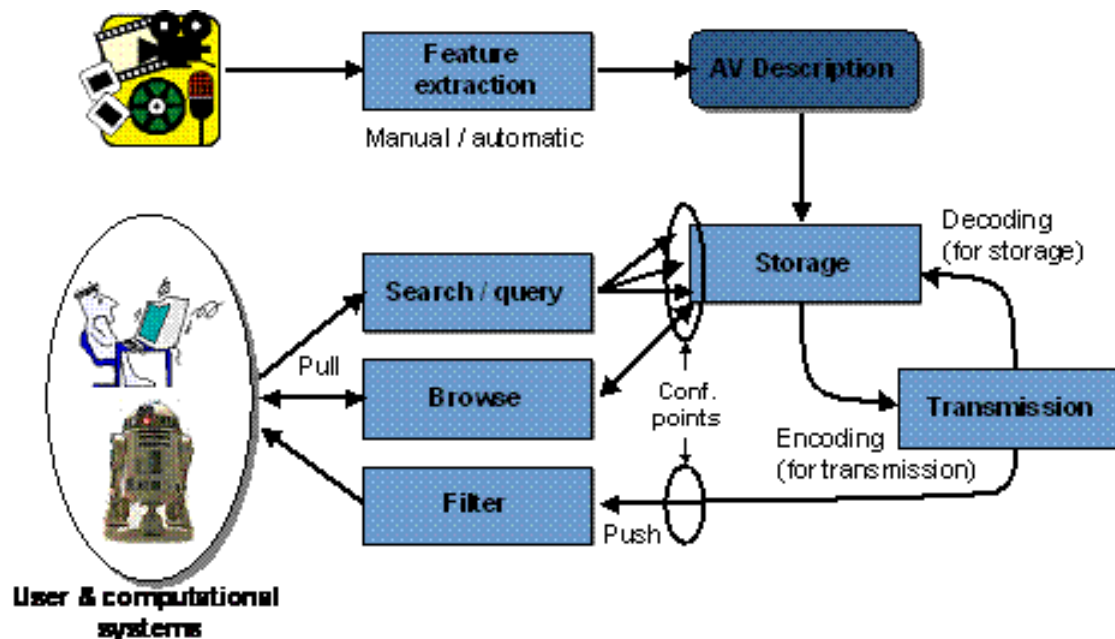
Overview of MPEG-7

Even though the MPEG-7 description does not depend on the (coded) representation of the material, MPEG-7 can exploit the advantages provided by MPEG-4 coded content. If the material is encoded using MPEG-4, it will be possible to attach descriptions to objects within the scene, such as audio and visual objects.

Because the descriptive features must be meaningful in the context of the application, they will be different for different user domains and different applications. To take the example of visual material: a lower abstraction level would be a description of e.g. shape, size, texture, color, movement (trajectory) and position; and for audio: key, mood, tempo, tempo changes, position in sound space.

Abstract Representation Using MPEG-7

The highest level would give semantic information: 'This is a scene with a barking brown dog on the left and a blue ball that falls down on the right, with the sound of passing cars in the background.' Many low-level features can be extracted in fully automatic ways, whereas high level features need (much) more human interaction.





Overview of MPEG-7

MPEG-7 data may be physically located with the associated AV material, in the same data stream or on the same storage system, but the descriptions could also live somewhere else on the globe.

MPEG-7 Alliance website, <http://www.mpeg-industry.com/>



Main Elements of the MPEG-7's Standard

Descriptors (D), that define the syntax and the semantics of each feature (metadata element).

Description Schemes (DS), that specify the structure and semantics of the relationships between their components, which may be both Descriptors and Description Schemes.

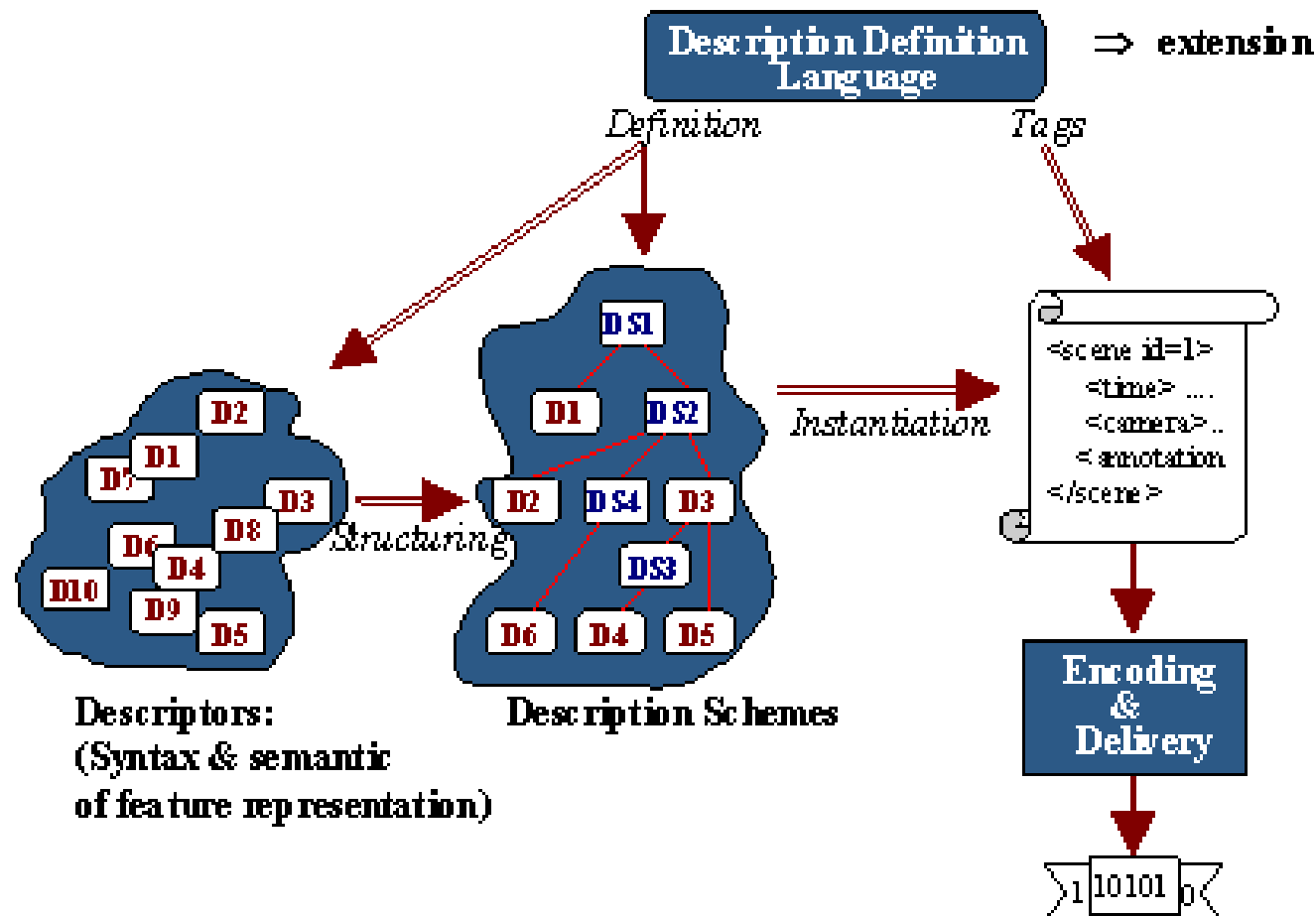
A Description Definition Language (DDL) to define the syntax of the MPEG-7 Description Tools and to allow the creation of new Description Schemes and, possibly, Descriptors and to allow the extension and modification of existing Description Schemes.




Main Elements of the MPEG-7's Standard

System tools, to support binary coded representation for efficient storage and transmission, transmission mechanisms (both for textual and binary formats), multiplexing of descriptions, synchronization of descriptions with content, management and protection of intellectual property in MPEG-7 descriptions, etc.

MPEG-7 Main Elements






MPEG-7 Parts

Part 1: Systems - the binary format for encoding MPEG-7 descriptions and the terminal architecture.

Part 2: Description Definition Language - the language for defining the syntax of the MPEG-7 Description Tools and for defining new Description Schemes. The DDL is based on XML Schema Language. But because XML Schema Language has not been designed specifically for audiovisual content description, there are certain MPEG-7 extensions that have been added.

Part 3: Visual – the Description Tools dealing with (only) Visual descriptions.

Part 4: Audio – the Description Tools dealing with (only) Audio descriptions.



MPEG-7 Parts

Part 5: Multimedia Description Schemes (MDS) - the Description Tools dealing with generic features and multimedia descriptions.

Part 6: Reference Software - a software implementation of relevant parts of the MPEG-7 Standard with normative status.

Part 7: Conformance Testing - guidelines and procedures for testing conformance of MPEG-7 implementations.

Part 8: Extraction and Use of MPEG-7 Descriptions

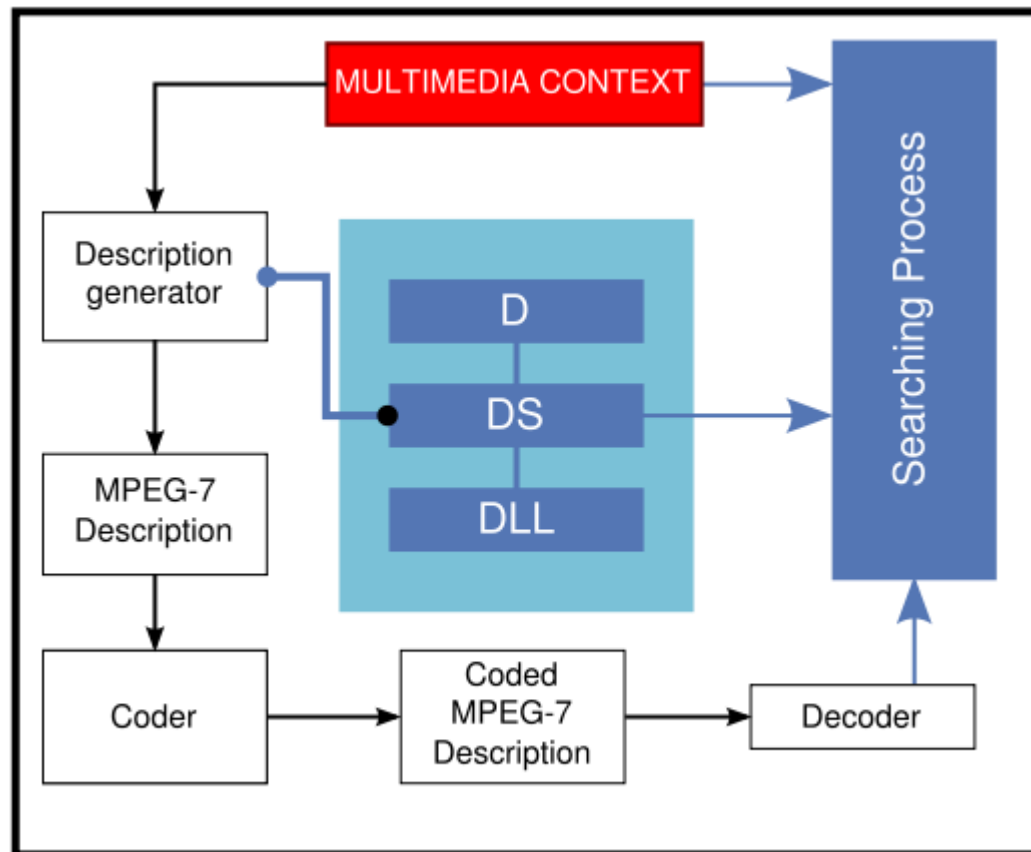
Part 9: Profiles

Part 10: Schema Definition

Part 11: MPEG-7 Profile Schemas

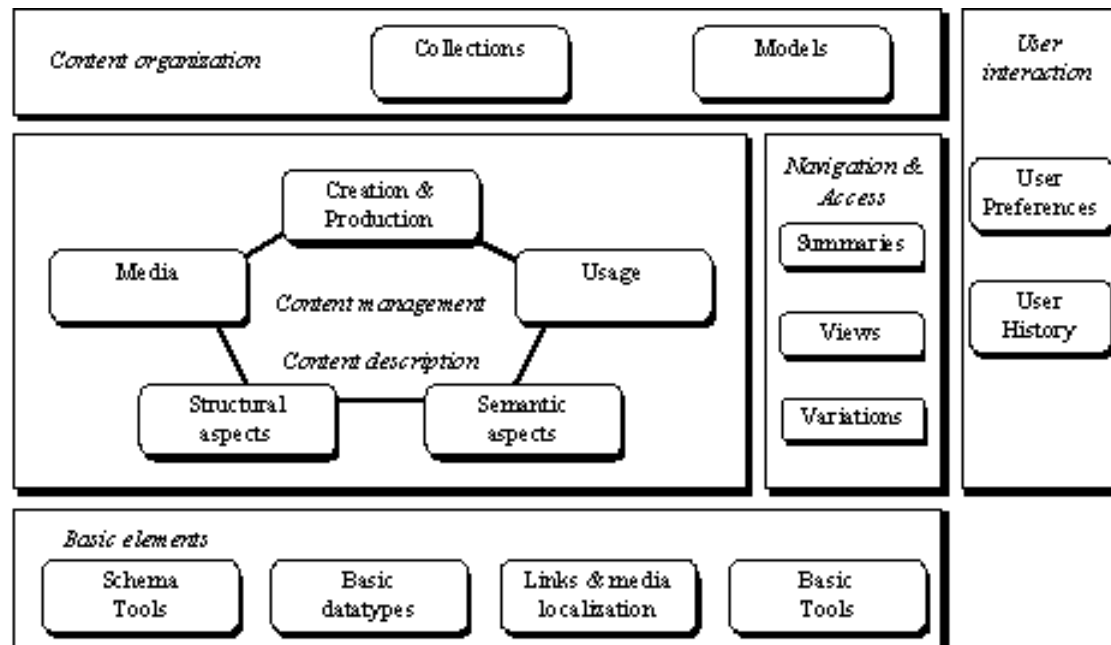
Part 12: Query Format

Relation Between Different Tools and Elaboration Process of MPEG-7

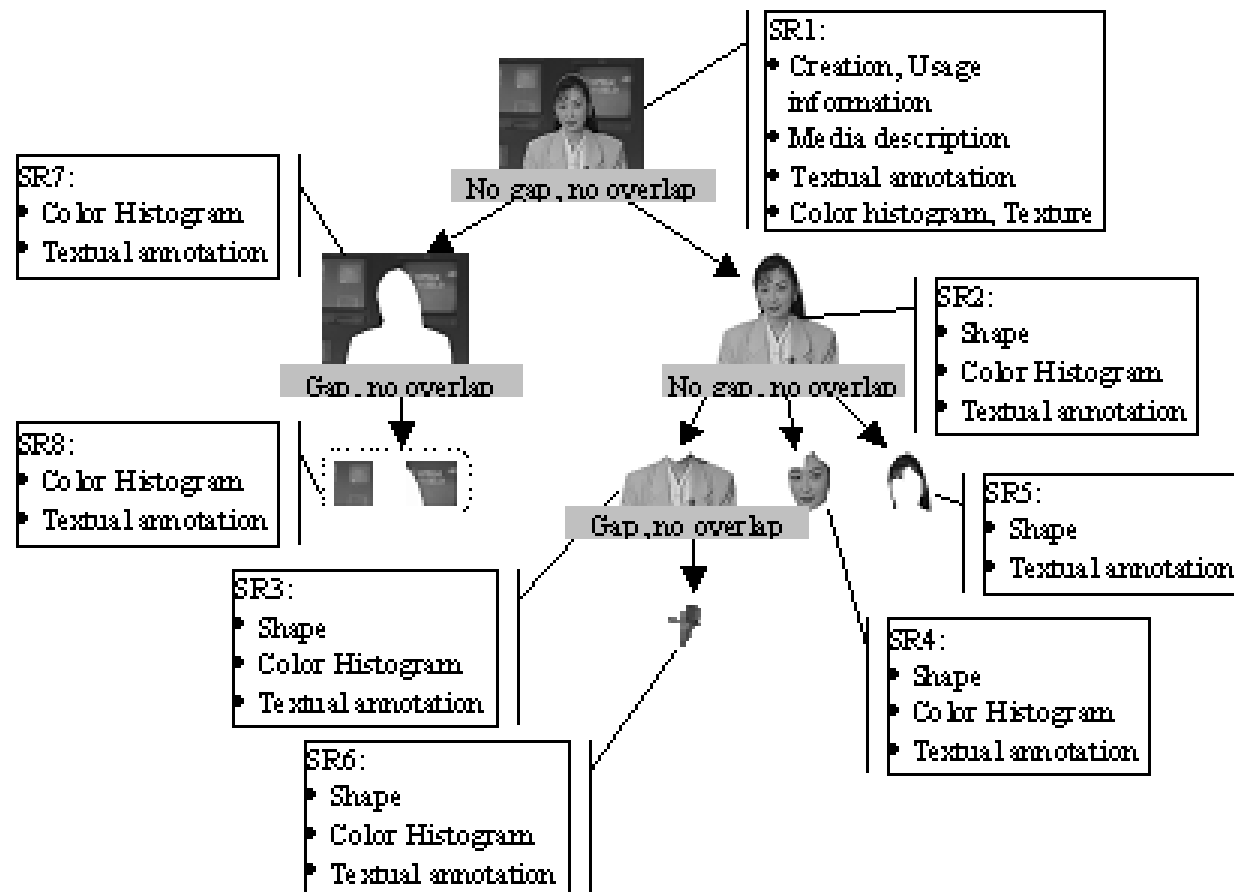


MPEG-7 Multimedia Description Schemes

The figure below provides an overview of the organization of MPEG-7 Multimedia DSs into the following areas: Basic Elements, Content Description, Content Management, Content Organization, Navigation and Access, and User Interaction.



Examples of Image description with Still Regions





MPEG-7 Visual Overview

MPEG-7 Visual Description Tools included in the standard consist of basic structures and Descriptors that cover the following basic visual features:

- Color

- Texture

- Shape

- Motion

- Localization

- Face recognition.

Each category consists of elementary and sophisticated Descriptors.



Color Descriptors

There are seven Color Descriptors:

- Color space,
- Color Quantization,
- Dominant Colors,
- Scalable Color,
- Color Layout,
- Color-Structure,
- GoF/GoP Color.

Color Descriptors

Color Space. The following color spaces are supported:

R,G,B

Y,Cr,Cb

H,S,V

HMMD

Linear transformation matrix with reference to R, G, B

Monochrome

Color Quantization

This descriptor defines a uniform quantization of a color space. The number of bins which the quantizer produces is configurable. For a meaningful application in the context of MPEG-7, this descriptor has to be combined with dominant color descriptors, e.g. to express the meaning of the values of dominant colors.



Color descriptors

Dominant Color(s)

This color descriptor is most suitable for representing local (object or image region) features where a small number of colors are enough to characterize the color information. The percentage of each quantized color in the region is calculated correspondingly. A spatial coherency on the entire descriptor is also defined, and is used in similarity retrieval.

Scalable Color

The Scalable Color Descriptor is a Color Histogram in HSV Color Space, which is encoded by a Haar transform. Its binary representation is scalable in terms of bin numbers and bit representation accuracy over a broad range of data rates. Retrieval accuracy increases with the number of bits used in the representation.



Color Descriptors

Color Layout

This descriptor effectively represents the spatial distribution of color of visual signals in a very compact form. It also provides very friendly user interface using hand-written sketch queries since this descriptor captures the layout information of color feature. The sketch queries are not supported in other color descriptors.

Color-Structure Descriptor

This descriptor is a color feature descriptor that captures both color content (similar to a color histogram) and information about the structure of this content. The extraction method embeds color structure information into the descriptor by taking into account all colors in a structuring element of 8x8 pixels that slides over the image, instead of considering each pixel separately. Color values are represented in the double-coned HMMD color.



Color Descriptors

GoF/GoP Color

The Group of Frames/Group of Pictures color descriptor extends the ScalableColor descriptor that is defined for a still image to color description of a video segment or a collection of still images. Additional two bits allow to define how the color histogram was calculated, before the Haar transform is applied to it: by average, median or intersection. The same similarity/distance measures that are used to compare scalable color descriptions can be employed to compare GoF/GoP color Descriptors.

Texture Descriptors

Homogenous Texture Descriptors

An image can be considered as a mosaic of homogeneous textures so that these texture features associated with the regions can be used to index the image data. Examples of queries that could be supported in this context could include “Retrieve all Land-Satellite images of Santa Barbara which have less than 20% cloud cover”. The Homogeneous Texture Descriptor provides a quantitative representation using 62 numbers (quantified to 8 bits each) that is useful for similarity retrieval. The extraction is done as follows; the image is first filtered with a bank of orientation and scale-tuned filters (modeled using Gabor functions) using Gabor filters. The first and the second moments of the energy in the frequency domain in the corresponding sub-bands are then used as the components of the texture descriptor.



Texture Descriptors

Texture Browsing

This descriptor is useful for representing homogeneous texture for browsing type applications, and requires only 12 bits (maximum). It provides a perceptual characterization of texture, similar to a human characterization, in terms of regularity, coarseness and directionality. The computation of this descriptor proceeds similarly as the Homogeneous Texture Descriptor.

Edge Histogram

This descriptor represents the spatial distribution of five types of edges, namely four directional edges and one non-directional edge. Since edges play an important role for image perception, it can retrieve images with similar semantic meaning.

Shape Descriptors

Region Shape

The shape of an object may consist of either a single region or a set of regions as well as some holes in the object. Since the Region Shape descriptor makes use of all pixels constituting the shape within a frame, it can describe any shapes. A black pixel within the object corresponds to 1 in an image, while white background corresponds to 0.



Examples of various shapes



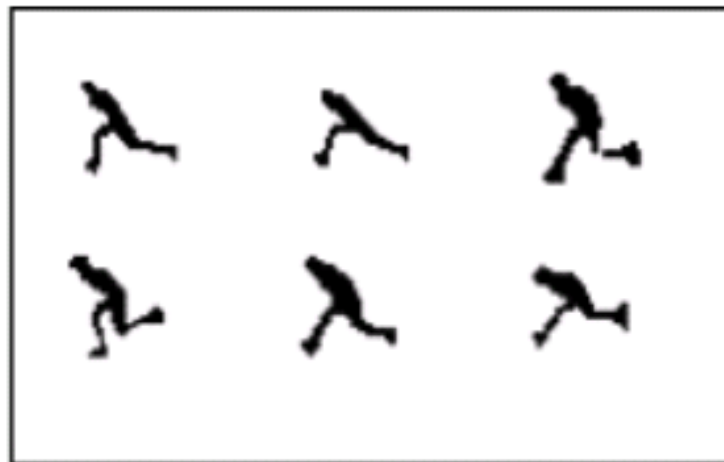
Shape Descriptors

Contour Shape

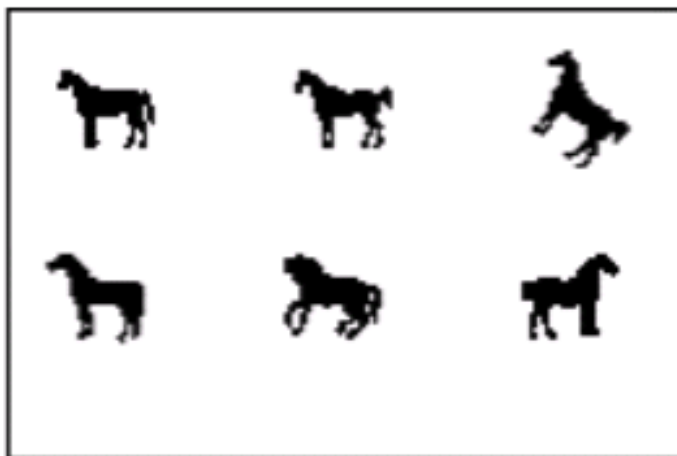
The Contour Shape descriptor captures characteristic shape features of an object or region based on its contour. It uses so-called Curvature Scale-Space (CSS) representation, which captures perceptually meaningful features of the shape. Each frame in the Figure contains very similar images according to CSS, based on the actual retrieval results from the MPEG-7 shape database.



**(a) shape generalization properties
(perceptual similarity among
different shapes)**



**(b) robustness to non-rigid
motion (man running)**



**(c) robustness to partial
occlusion (tails or legs
of the horses)**



Shape Descriptors

Shape 3D

Most of the time, 3D information is represented as polygonal meshes. MPEG-4, within the SNHC subgroup, considered this issue and developed technologies for efficient 3D mesh model coding. The 3D Shape Descriptor described in detail provides an intrinsic shape description of 3D mesh models. It exploits some local attributes of the 3D surface.

Motion Descriptors

Camera Motion

This descriptor characterizes 3-D camera motion parameters. It is based on 3-D camera motion parameter information, which can be automatically extracted or generated by capture devices. The camera motion descriptor supports the following well-known basic camera operations (see Figure 8).

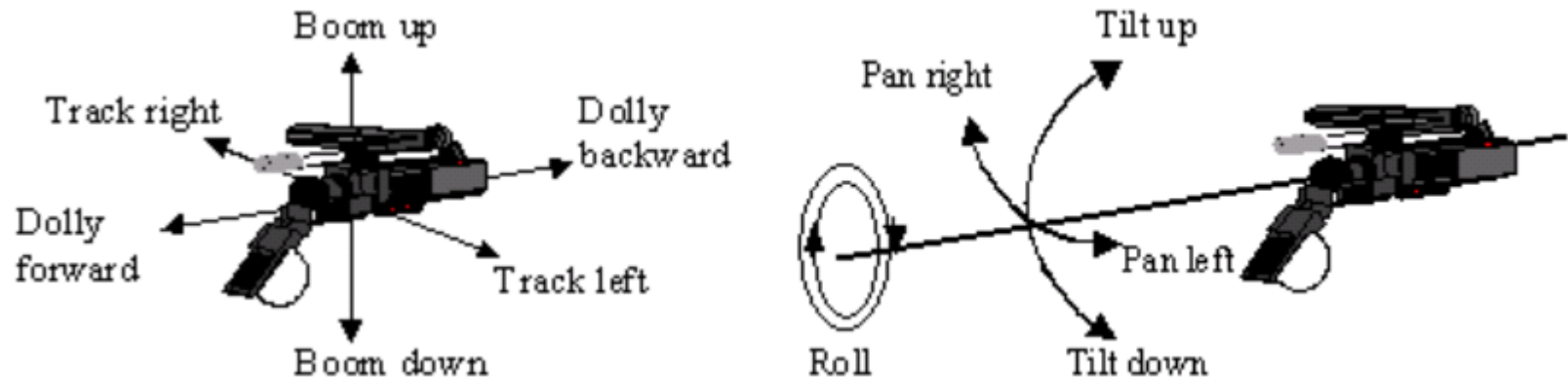


Figure 8: (a) Camera track, boom, and dolly motion modes, (b) Camera pan, tilt and roll motion modes.



Motion Descriptors

Motion Trajectory

The motion trajectory of an object is a simple, high-level feature, defined as the localization, in time and space, of one representative point of this object. In given contexts with a priori knowledge, trajectory can enable many functionalities. In surveillance, alarms can be triggered if some object has a trajectory identified as dangerous (e.g. passing through a forbidden area, being unusually quick, etc.). In sports, specific actions (e.g. tennis rallies taking place at the net) can be recognized.



Motion Descriptors

Parametric Motion

Parametric motion models have been extensively used within various related image processing and analysis areas, including motion-based segmentation and estimation, global motion estimation, mosaicing and object tracking. Parametric motion models have been already used in MPEG-4, for global motion estimation and compensation and sprite generation. Specifically, affine models include translations, rotations, scaling and combination of them, planar perspective models make possible to take into account global deformations associated with perspective projections and quadratic models makes it possible to describe more complex movements.



Motion Descriptors

Motion Activity

A human watching a video or animation sequence perceives it as being a slow sequence, fast paced sequence, action sequence etc. The activity descriptor captures this intuitive notion of 'intensity of action' or 'pace of action' in a video segment. Examples of high 'activity' include scenes such as 'goal scoring in a soccer match', 'scoring in a basketball game', 'a high speed car chase' etc. On the other hand, scenes such as 'news reader shot', 'an interview scene', 'a still shot' etc. are perceived as low action shots.

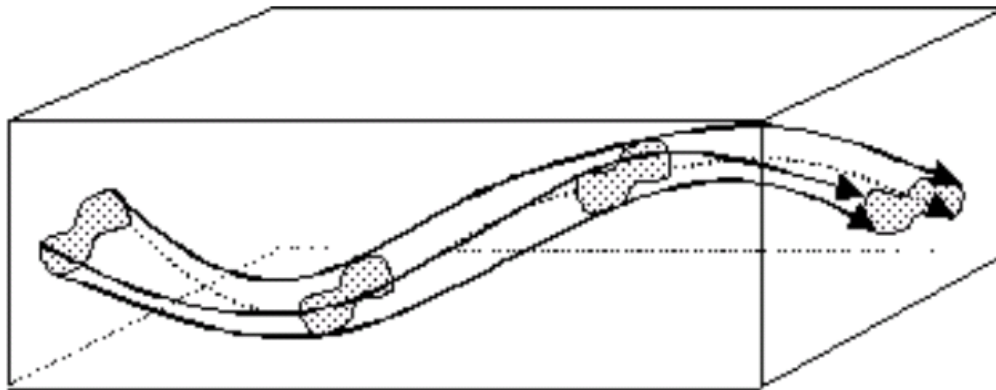
Localization

Region Locator

This descriptor enables localization of regions within images or frames by specifying them with a brief and scalable representation of a Box or a Polygon.

Spatio-Temporal Locator

This describes spatio-temporal regions in a video sequence, such as moving object regions, and provides localization functionality.





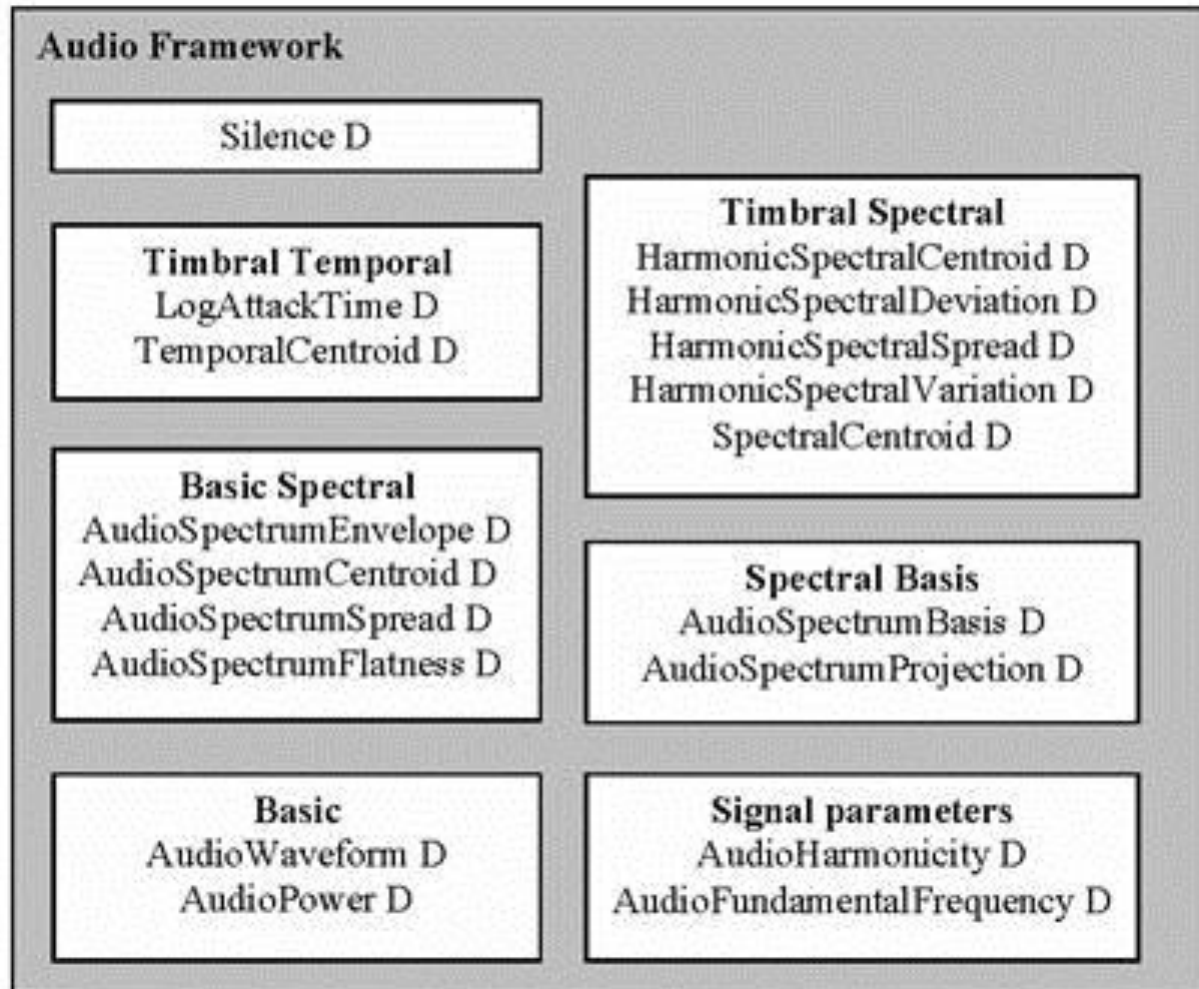
Other Visual Descriptors

Face Recognition

The FaceRecognition descriptor can be used to retrieve face images which match a query face image. The descriptor represents the projection of a face vector onto a set of basis vectors which span the space of possible face vectors.

The FaceRecognition feature set is extracted from a normalized face image. This normalized face image contains 56 lines with 46 intensity values in each line. The FaceRecogniton feature set is then calculated by projecting the one dimensional face vector onto the space defined by a set of basis vectors.

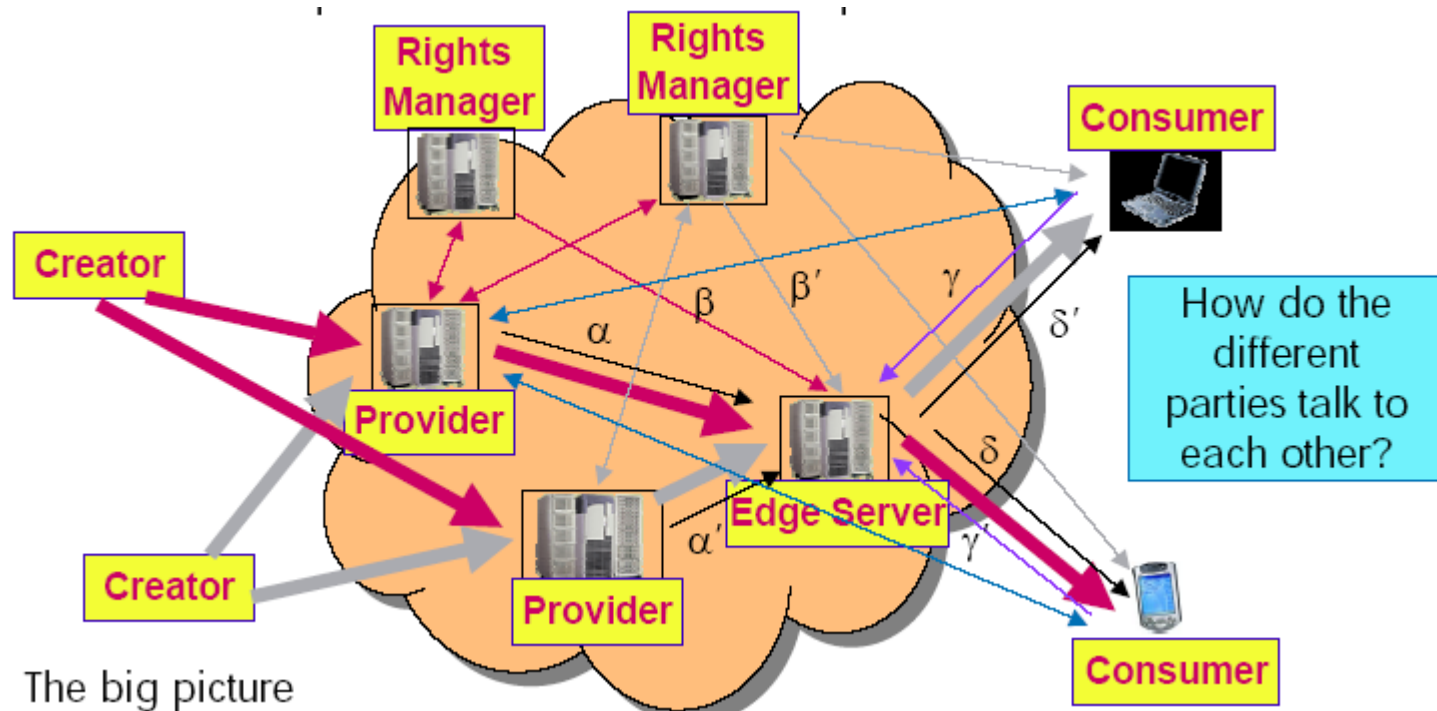
MPEG-7 Audio Framework





MPEG-21

MPEG-21 Overview





MPEG-21 Overview

MPEG-21 aims at defining a normative *open framework for multimedia delivery and consumption* for use by all the players in the delivery and consumption chain. This open framework will provide content creators, producers, distributors and service providers with equal opportunities in the MPEG-21 enabled open market. This will also be to the benefit of the content consumer providing them access to a large variety of content in an interoperable manner.



MPEG-21 Overview

Multimedia technology currently provides content creators and consumers with a myriad of coding, access and distribution possibilities. Access to information and services from almost anywhere at anytime can be provided with ubiquitous terminals and networks. However, no complete solutions exist that allow different communities, each with their own models, rules, procedures, interests and content formats, to interact efficiently using this complex infrastructure. Developing a common multimedia framework will facilitate co-operation between these sectors and support a more efficient implementation and integration of the different models, rules, procedures, interests and content formats. The vision for MPEG-21 is to define a multimedia framework *to enable transparent and augmented use of multimedia resources across a wide range of networks and devices* used by different communities.



MPEG-21 Overview

MPEG-21 is based on two essential concepts: the definition of a fundamental unit of distribution and transaction (the Digital Item) and the concept of Users interacting with Digital Items. The Digital Items can be considered the “what” of the Multimedia Framework (e.g., a video collection, a music album) and the Users can be considered the “who” of the Multimedia Framework.

The goal of MPEG-21 can thus be rephrased to: defining the technology needed to support Users to exchange, access, consume, trade and otherwise manipulate Digital Items in an efficient, transparent and interoperable way.



Key Elements in MPEG-21

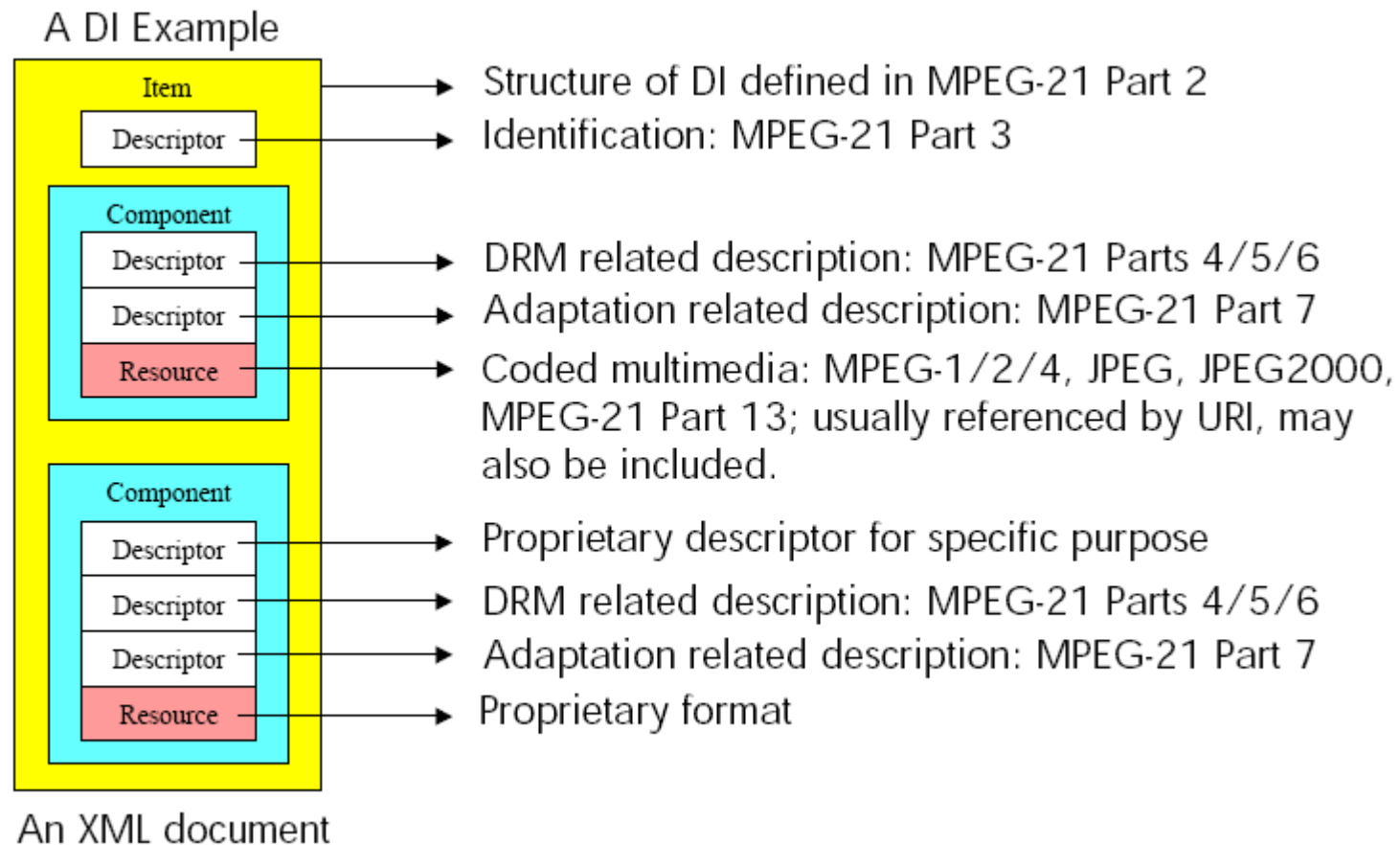
Digital Item Declaration (a uniform and flexible abstraction and interoperable schema for declaring Digital Items).

The Digital Item (DI) is a fundamental unit of transaction between Users.

DI is a structured packaging of resources and/or various kinds of metadata for supporting various functionalities.

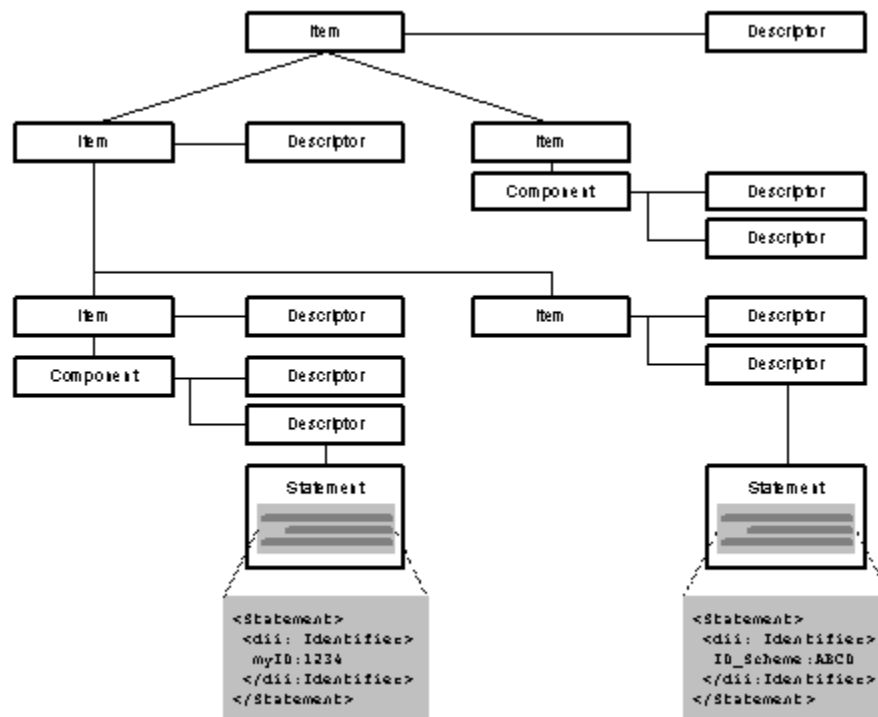
DI is an XML document that includes various kinds of metadata (descriptions) whose schemas are defined in other parts.

A DI Example



Key Elements in MPEG-21

Digital Item Identification and Description (a framework for identification and description of any entity regardless of its nature, type or granularity).



**Relationship between
Digital Item Declaration
and Digital Item
Identification**



Key Elements in MPEG-21

Content Handling and Usage (provide interfaces and protocols that enable creation, manipulation, search, access, storage, delivery, and (re)use of content across the content distribution and consumption value chain).

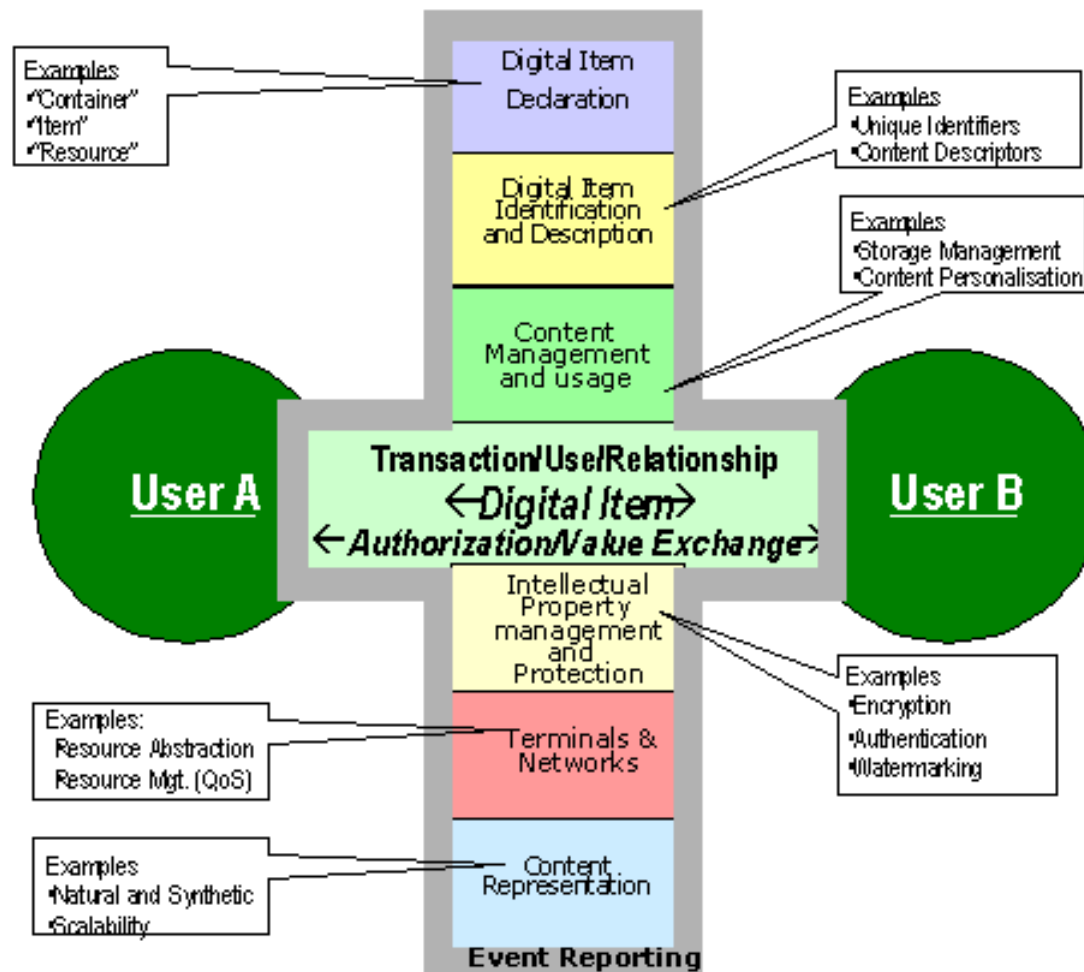
Intellectual Property Management and Protection (the means to enable content to be persistently and reliably managed and protected across a wide range of networks and devices).


Terminals and Networks (the ability to provide interoperable and transparent access to content across networks and terminals).

Content Representation (how the media resources are represented).

Event Reporting (the metrics and interfaces that enable Users to understand precisely the performance of all reportable events within the framework).

Key Elements in MPEG-21






MPEG-21 Parts

Parts 1-3: DI

1. Vision, Technologies and Strategy
2. Digital Item Declaration
3. Digital Item Identification and Description

Parts 4-6: Digital Rights Management (DRM)

4. Intellectual Property Management and Protection Components
5. Rights Expression Language
6. Rights Data Dictionary



MPEG-21 Parts

- 7. Digital Item Adaptation and Session Mobility
- 8. Reference Software
- 9. File Formats (.m21 or .mp21)
- 10. Digital Item Processing and C++ bindings
- 11. Evaluation Tools for Persistent Association
- 15. Event Reporting
- 17. Fragment Identification for MPEG Resources
- 18. Digital Item Streaming



MPEG-A,B,C,...

MPEG-A (ISO/IEC 23000) Multimedia Application Formats

MPEG-B (ISO/IEC 23001) MPEG Systems Technologies

MPEG-C (ISO/IEC 23002) MPEG Video Technologies

MPEG-D (ISO/IEC 23003) MPEG Audio Technologies

MPEG-E (ISO/IEC 23004) Multimedia Middleware (M3W)

MPEG-H High Efficiency Video Coding (HEVC)

MPEG-M (ISO/IEC 23006) MPEG Extensible Middleware (MXM)

MPEG-U (ISO/IEC 23007) Rich-Media User Interface

MPEG-V (ISO/IEC 23005) Media Context and Control

References

ISO MPEG specifications

<http://www.chiariglione.org/mpeg>

Wiki, <http://en.wikipedia.org/wiki/>

Multimedia Systems, Standards, and Networks, edited by Atul Puri and Tsuhan Chen, Marcel Dekker, Inc., 2000.

H.264 and MPEG-4 Video Compression by Iain E.G. Richardson, Wiley, 2003.

Special Issue on MPEG-7, *IEEE Trans. Circuits Syst. Video Technol.*, June 2001.

J. Bormans, J. Gelissen, and A. Perkis, “MPRG-21: the 21st century multimedia framework,” *IEEE Signal Process. Mag.*, pp. 53-62, March 2003.