INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 N4668

March 2002

Source:	**WG11 (MPEG)**
Status:	**Final**
Title:	**MPEG-4 Overview - (V.21 – Jeju Version)**
Editor:	**Rob Koenen (rob.koenen@m4if.org)**

All comments, corrections, suggestions and additions to this document are welcome, and should be send to both the editor and the chairman of MPEG's Requirements Group: Fernando Pereira, fp@lx.it.pt

Overview of the MPEG-4 Standard

Executive Overview

MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group), the committee that also developed the Emmy Award winning standards known as MPEG-1 and MPEG-2. These standards made interactive video on CD-ROM, DVD and Digital Television possible. MPEG-4 is the result of another international effort involving hundreds of researchers and engineers from all over the world. MPEG-4, with formal as its ISO/IEC designation 'ISO/IEC 14496', was finalized in October 1998 and became an International Standard in the first months of 1999. The fully backward compatible extensions under the title of MPEG-4 Version 2 were frozen at the end of 1999, to acquire the formal International Standard Status early in 2000. Several extensions were added since and work on some specific work-items work is still in progress.

MPEG-4 builds on the proven success of three fields:

- Digital television;
- Interactive graphics applications (synthetic content);
- Interactive multimedia (World Wide Web, distribution of and access to content)

MPEG-4 provides the standardized technological elements enabling the integration of the production, distribution and content access paradigms of the three fields.

More information about MPEG-4 can be found at MPEG's home page (case sensitive): http://mpeg.telecomitalialab.com This web page contains links to a wealth of information about MPEG, including much about MPEG-4, many publicly available documents, several lists of 'Frequently Asked Questions' and links to other MPEG-4 web pages. The standard can be bought from ISO, send mail to sales@iso.ch. Notably, the complete software for MPEG-4 version 1 can be bought on a CD ROM, for 56 Swiss Francs. It can also be downloaded for free from ISO's website: www.iso.ch/ittf - look under publicly available standards and then for "14496-5". This software is free of copyright restrictions when used for implementing MPEG-4 compliant technology. (This does not mean that the software is fee of patents). As well, much information is available from the MPEG-4 Industry Forum, M4IF, http://www.m4if.org. See section 7, The MPEG-4 Industry Forum.

This document gives an overview of the MPEG-4 standard, explaining which pieces of technology it includes and what sort of applications are supported by this technology.

Table of Contents

1 Scope and features of the MPEG-4 standard

The MPEG-4 standard provides a set of technologies to satisfy the needs of authors, service providers and end users alike.

- For *authors*, MPEG-4 enables the production of content that has far greater reusability, has greater flexibility than is possible today with individual technologies such as digital television, animated graphics, World Wide Web (WWW) pages and their extensions. Also, it is now possible to better manage and protect content owner rights.
- For *network service providers* MPEG-4 offers transparent information, which can be interpreted and translated into the appropriate native signaling messages of each network with the help of relevant standards bodies. The foregoing, however, excludes Quality of Service considerations, for which MPEG-4 provides a generic QoS descriptor for different MPEG-4 media. The exact translations from the QoS parameters set for each media to the network QoS are beyond the scope of MPEG-4 and are left to network providers. Signaling of the MPEG-4 media QoS descriptors end-to-end enables transport optimization in heterogeneous networks.
- For *end users*, MPEG-4 brings higher levels of interaction with content, within the limits set by the author. It also brings multimedia to new networks, including those employing relatively low bitrate, and mobile ones. An MPEG-4 applications document exists on the MPEG Home page (www.cselt.it/mpeg), which describes many end user applications, including interactive multimedia broadcast and mobile communications.

For all parties involved, MPEG seeks to avoid a multitude of proprietary, non-interworking formats and players.

MPEG-4 achieves these goals by providing standardized ways to:

1. represent units of aural, visual or audiovisual content, called “media objects”. These media objects can be of natural or synthetic origin; this means they could be recorded with a camera or microphone, or generated with a computer;
2. describe the composition of these objects to create compound media objects that form audiovisual scenes;
3. multiplex and synchronize the data associated with media objects, so that they can be transported over network channels providing a QoS appropriate for the nature of the specific media objects; and
4. interact with the audiovisual scene generated at the receiver’s end.

The following sections illustrate the MPEG-4 functionalities described above, using the audiovisual scene depicted in Figure 1.

1.1 Coded representation of media objects

MPEG-4 audiovisual scenes are composed of several media objects, organized in a hierarchical fashion. At the leaves of the hierarchy, we find primitive media objects, such as:

- Still images (e.g. as a fixed background);
- Video objects (e.g. a talking person - without the background;
- Audio objects (e.g. the voice associated with that person, background music);

MPEG-4 standardizes a number of such primitive media objects, capable of representing both natural and synthetic content types, which can be either 2- or 3-dimensional. In addition to the

media objects mentioned above and shown in Figure 1, MPEG-4 defines the coded representation of objects such as:

- Text and graphics;
- Talking synthetic heads and associated text used to synthesize the speech and animate the head; animated bodies to go with the faces;
- Synthetic sound.

A media object in its coded form consists of descriptive elements that allow handling the object in an audiovisual scene as well as of associated streaming data, if needed. It is important to note that in its coded form, each media object can be represented independent of its surroundings or background.
The coded representation of media objects is as efficient as possible while taking into account the desired functionalities. Examples of such functionalities are error robustness, easy extraction and editing of an object, or having an object available in a scaleable form.

1.2 Composition of media objects

Figure 1 explains the way in which an audiovisual scene in MPEG-4 is described as composed of individual objects. The figure contains compound media objects that group primitive media objects together. Primitive media objects correspond to leaves in the descriptive tree while compound media objects encompass entire sub-trees. As an example: the visual object corresponding to the talking person and the corresponding voice are tied together to form a new compound media object, containing both the aural and visual components of that talking person.

Such grouping allows authors to construct complex scenes, and enables consumers to manipulate meaningful (sets of) objects.

More generally, MPEG-4 provides a standardized way to describe a scene, allowing for example to:

- Place media objects anywhere in a given coordinate system;
- Apply transforms to change the geometrical or acoustical appearance of a media object;
- Group primitive media objects in order to form compound media objects;
- Apply streamed data to media objects, in order to modify their attributes (e.g. a sound, a moving texture belonging to an object; animation parameters driving a synthetic face);
- Change, interactively, the user's viewing and listening points anywhere in the scene.

The scene description builds on several concepts from the Virtual Reality Modeling language (VRML) in terms of both its structure and the functionality of object composition nodes and extends it to fully enable the aforementioned features.

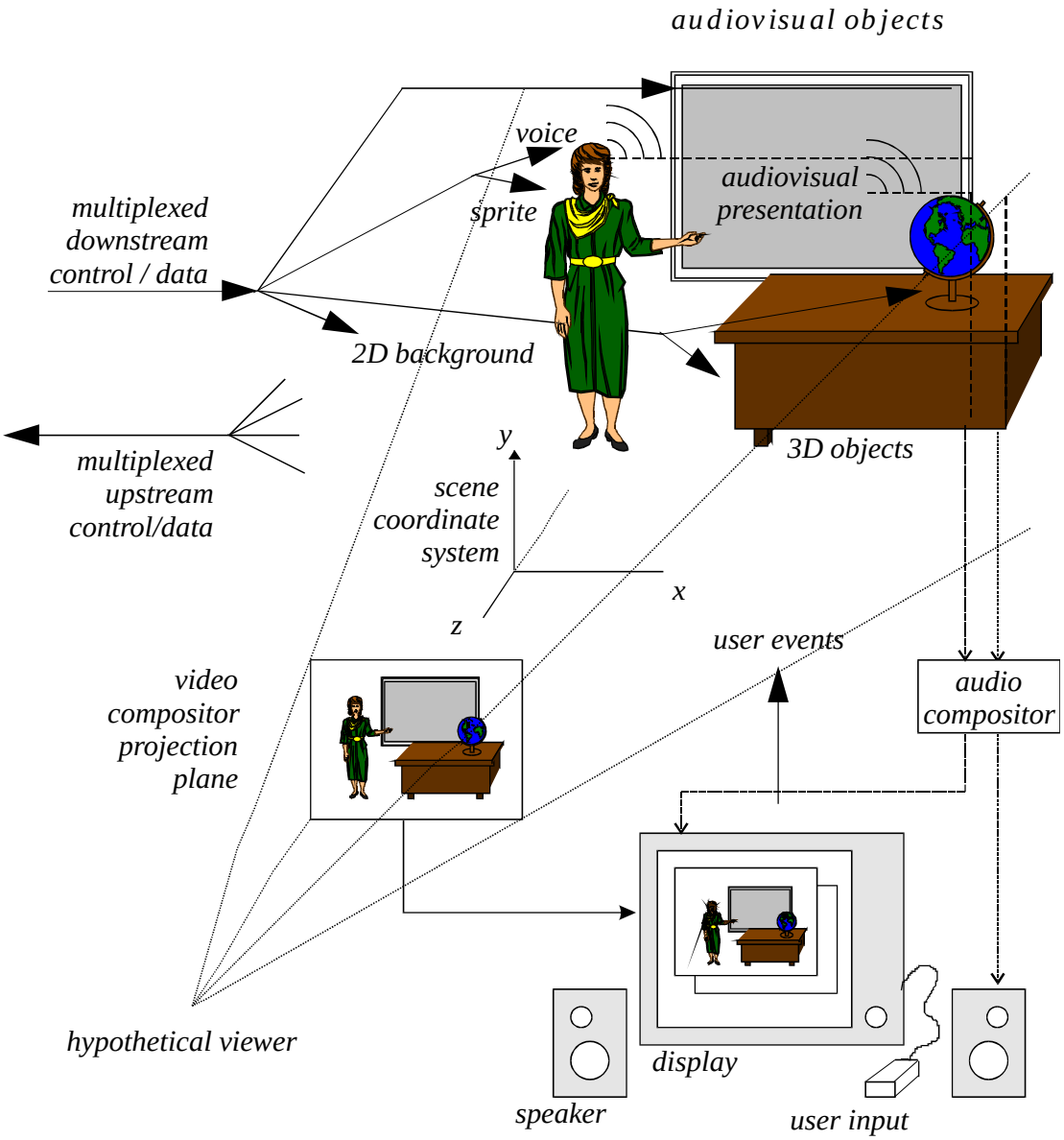


Figure 1 - an example of an MPEG-4 Scene

1.3 Description and synchronization of streaming data for media objects

Media objects may need streaming data, which is conveyed in one or more elementary streams. An object descriptor identifies all streams associated to one media object. This allows handling hierarchically encoded data as well as the association of meta-information about the content (called 'object content information') and the intellectual property rights associated with it. Each stream itself is characterized by a set of descriptors for configuration information, e.g., to determine the required decoder resources and the precision of encoded timing information. Furthermore the descriptors may carry hints to the Quality of Service (QoS) it requests for transmission (e.g., maximum bit rate, bit error rate, priority, etc.)

Synchronization of elementary streams is achieved through time stamping of individual access units within elementary streams. The synchronization layer manages the identification of such access units and the time stamping. Independent of the media type, this layer allows identification of the type of access unit (e.g., video or audio frames, scene description commands) in elementary streams, recovery of the media object's or scene description's time base, and it enables synchronization among them. The syntax of this layer is configurable in a large number of ways, allowing use in a broad spectrum of systems.

1.4 Delivery of streaming data

The synchronized delivery of streaming information from source to destination, exploiting different QoS as available from the network, is specified in terms of the synchronization layer and a delivery layer containing a two-layer multiplexer, as depicted in Figure 2.
The first multiplexing layer is managed according to the DMIF specification, part 6 of the MPEG-4 standard. (DMIF stands for Delivery Multimedia Integration Framework) This multiplex may be embodied by the MPEG-defined FlexMux tool, which allows grouping of Elementary Streams (ESs) with a low multiplexing overhead. Multiplexing at this layer may be used, for example, to group ES with similar QoS requirements, reduce the number of network connections or the end to end delay.

The "TransMux" (Transport Multiplexing) layer in Figure 2 models the layer that offers transport services matching the requested QoS. Only the interface to this layer is specified by MPEG-4 while the concrete mapping of the data packets and control signaling must be done in collaboration with the bodies that have jurisdiction over the respective transport protocol. Any suitable existing transport protocol stack such as (RTP)/UDP/IP, (AAL5)/ATM, or MPEG-2's Transport Stream over a suitable link layer may become a specific TransMux instance. The choice is left to the end user/service provider, and allows MPEG-4 to be used in a wide variety of operation environments.

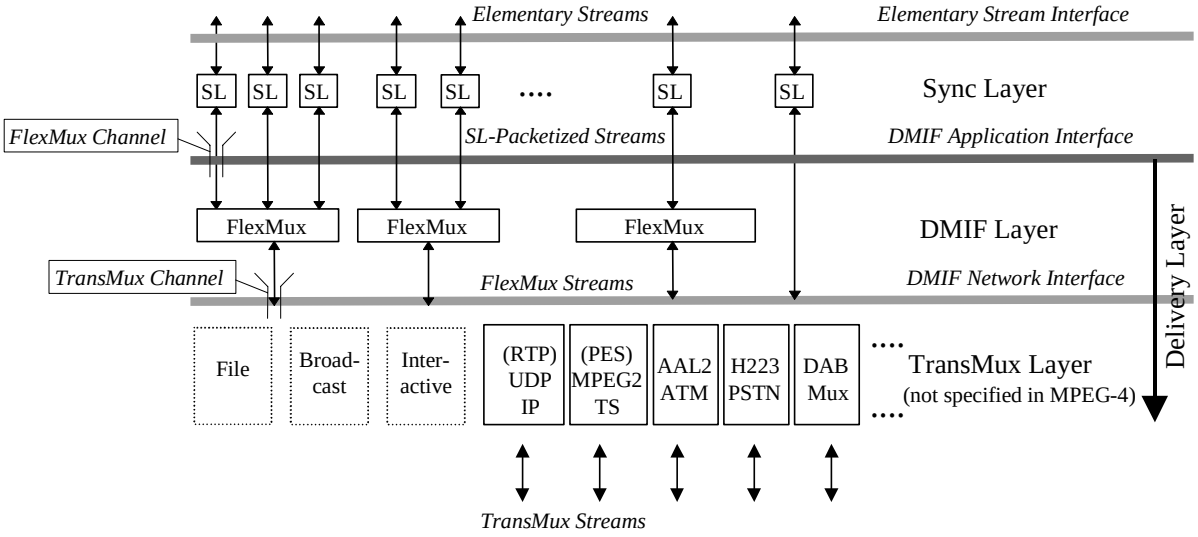


Figure 2 - The MPEG-4 System Layer Model

Use of the FlexMux multiplexing tool is optional and, as shown in Figure 2, this layer may be empty if the underlying TransMux instance provides all the required functionality. The synchronization layer, however, is always present.

With regard to Figure 2, it is possible to:

- Identify access units, transport timestamps and clock reference information and identify data loss.
- Optionally interleave data from different elementary streams into FlexMux streams
- Convey control information to:
- Indicate the required QoS for each elementary stream and FlexMux stream;
- Translate such QoS requirements into actual network resources;
- Associate elementary streams to media objects
- Convey the mapping of elementary streams to FlexMux and TransMux channels

Parts of the control functionalities are available only in conjunction with a transport control entity like the DMIF framework.

1.5 Interaction with media objects

In general, the user observes a scene that is composed following the design of the scene's author. Depending on the degree of freedom allowed by the author, however, the user has the possibility to interact with the scene. Operations a user may be allowed to perform include:

- Change the viewing/listening point of the scene, e.g. by navigation through a scene;
- Drag objects in the scene to a different position;
- Trigger a cascade of events by clicking on a specific object, e.g. starting or stopping a video stream;
- Select the desired language when multiple language tracks are available;

More complex kinds of behavior can also be triggered, e.g. a virtual phone rings, the user answers and a communication link is established.

1.6 Management and Identification of Intellectual Property

It is important to have the possibility to identify intellectual property in MPEG-4 media objects. Therefore, MPEG has worked with representatives of different creative industries in the definition of syntax and tools to support this. A full elaboration of the requirements for the identification of intellectual property can be found in 'Management and Protection of Intellectual Property in MPEG-4, which is publicly available from the MPEG home page.

MPEG-4 incorporates identification the intellectual property by storing unique identifiers, which are issued by international numbering systems (e.g. ISAN, ISRC, etc.1). These numbers can be applied to identify a current rights holder of a media object. Since not all content is identified by such a number, MPEG-4 Version 1 offers the possibility to identify intellectual property by a key-value pair (e.g.:»composer«/»John Smith«). Also, MPEG-4 offers a standardized interface that is integrated tightly into the Systems layer to people who want to use systems that control access to intellectual property. With this interface, proprietary control systems can be easily amalgamated with the standardized part of the decoder.

1 ISAN: International Audio-Visual Number, ISRC: International Standard Recording Code

2 Versions in MPEG-4

MPEG-4 Version 1 was approved by MPEG in December 1998; version 2 was frozen in December 1999. After these two major versions, more tools were added in subsequent amendments that could be qualified as versions, even though they are harder to recognize as such. Recognizing the versions is not too important, however; it is more important to distinguish Profiles. Existing tools and profiles from any version are never replaced in subsequent versions; technology is always added to MPEG-4 in the form of new profiles. Figure 3 below depicts the relationship between the versions. Version 2 is a backward compatible extension of Version 1, and version 3 is a backward compatible extension of Version 2 – and so on. The versions of all major parts of the MPEG-4 Standard (Systems, Audio, Video, DMIF) were synchronized; after that, the different parts took their own paths.

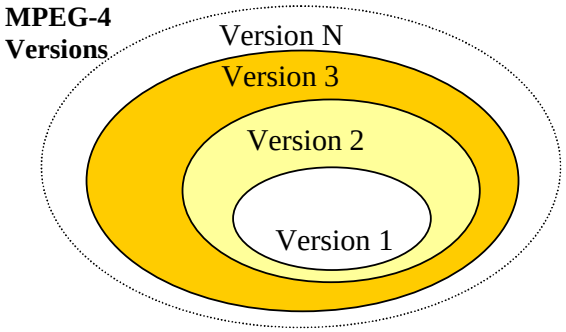


Figure 3 - relation between MPEG-4 Versions

The Systems layer of Version later versions is backward compatible with all earlier versions. In the area of Systems, Audio and Visual, new versions *add* Profiles, do not change existing ones. In fact, it is very important to note that existing systems will ***always*** remain compliant, because Profiles will ***never*** be changed in retrospect, and neither will the Systems Syntax, at least not in a backward-incompatible way.

3 Major Functionalities in MPEG-4

This section contains, in an itemized fashion, the major functionalities that the different parts of the MPEG-4 Standard offers in the finalized MPEG-4 Version 1. Description of the functionalities can be found in the following sections.

3.1 Transport

In principle, MPEG-4 does not define transport layers. In a number of cases, adaptation to a specific existing transport layer has been defined:

- Transport over MPEG-2 Transport Stream (this is an amendment to MPEG-2 Systems)
- Transport over IP (In cooperation with IETF, the Internet Engineering Task Force)

3.2 DMIF

DMIF, or Delivery Multimedia Integration Framework, is an interface between the application and the transport, that allows the MPEG-4 application developer to stop worrying about that transport. A single application can run on different transport layers when supported by the right DMIF instantiation.
MPEG-4 DMIF supports the following functionalities:

- A transparent MPEG-4 DMIF-application interface irrespective of whether the peer is a remote interactive peer, broadcast or local storage media.
- Control of the establishment of FlexMux channels
- Use of homogeneous networks between interactive peers: IP, ATM, mobile, PSTN, Narrowband ISDN.
- Support for mobile networks, developed together with ITU-T
- UserCommands with acknowledgment messages.
- Management of MPEG-4 Sync Layer information

3.3 Systems

As explained above, MPEG-4 defines a toolbox of advanced compression algorithms for audio and visual information. The data streams (Elementary Streams, ES) that result from the coding process can be transmitted or stored separately, and need to be composed so as to create the actual multimedia presentation at the receiver side.

The systems part of the MPEG-4 addresses the description of the relationship between the audio-visual components that constitute a scene. The relationship is described at two main levels.

- The Binary Format for Scenes (BIFS) describes the spatio-temporal arrangements of the objects in the scene. Viewers may have the possibility of interacting with the objects, e.g. by rearranging them on the scene or by changing their own point of view in a 3D virtual environment. The scene description provides a rich set of nodes for 2-D and 3-D composition operators and graphics primitives.
- At a lower level, Object Descriptors (ODs) define the relationship between the Elementary Streams pertinent to each object (e.g the audio and the video stream of a participant to a videoconference) ODs also provide additional information such as the URL needed to access the Elementary Steams, the characteristics of the decoders needed to parse them, intellectual property and others.

Other issues addressed by MPEG-4 Systems:

- A standard file format supports the exchange and authoring of MPEG-4 content
- Interactivity, including: client and server-based interaction; a general event model for triggering events or routing user actions; general event handling and routing between objects in the scene, upon user or scene triggered events.
- Java (MPEG-J) is used to be able to query to terminal and its environment support and there is also a Java application engine to code 'MPEGlets'.
- A tool for interleaving of multiple streams into a single stream, including timing information (FlexMux tool).
- A tool for storing MPEG-4 data in a file (the MPEG-4 File Format, 'MP4')
- Interfaces to various aspects of the terminal and networks, in the form of Java API's (MPEG-J)

- Transport layer independence. Mappings to relevant transport protocol stacks, like (RTP)/UDP/IP or MPEG-2 transport stream can be or are being defined jointly with the responsible standardization bodies.
- Text representation with international language support, font and font style selection, timing and synchronization.
- The initialization and continuous management of the receiving terminal's buffers.
- Timing identification, synchronization and recovery mechanisms.
- Datasets covering identification of Intellectual Property Rights relating to media objects.

3.4 Audio

MPEG-4 Audio facilitates a wide variety of applications which could range from intelligible speech to high quality multichannel audio, and from natural sounds to synthesized sounds. In particular, it supports the highly efficient representation of audio objects consisting of:

3.4.1 General Audio Signals

Support for coding general audio ranging from very low bitrates up to high quality is provided by transform coding techniques. With this functionality, a wide range of bitrates and bandwidths is covered. It starts at a bitrate of 6 kbit/s and a bandwidth below 4 kHz and extends to broadcast quality audio from mono up to multichannel. High quality can be achieved with low delays. Parametric Audio Coding allows sound manipulation at low speeds. Fine Granularity Scalability (or FGS, scalability resolution down to 1 kbit/s per channel)

3.4.2 Speech signals

Speech coding can be done using bitrates from 2 kbit/s up to 24 kbit/s using the speech coding tools. Lower bitrates, such as an average of 1.2 kbit/s, are also possible when variable rate coding is allowed. Low delay is possible for communications applications. When using the HVXC tools, speed and pitch can be modified under user control during playback. If the CELP tools are used, a change of the playback speed can be achieved by using and additional tool for effects processing.

3.4.3 Synthetic Audio

MPEG-4 Structured Audio is a language to describe 'instruments' (little programs that generate sound) and 'scores' (input that drives those objects). These objects are not necessarily musical instruments, they are in essence mathematical formulae, that could generate the sound of a piano, that of falling water – or something 'unheard' in nature.

3.4.4 Synthesized Speech

Scalable TTS coders bitrate range from 200 bit/s to 1.2 Kbit/s which allows a text, or a text with prosodic parameters (pitch contour, phoneme duration, and so on), as its inputs to generate intelligible synthetic speech.

3.5 Visual

The MPEG-4 Visual standard allows the hybrid coding of natural (pixel based) images and video together with synthetic (computer generated) scenes. This enables, for example, the virtual presence of videoconferencing participants. To this end, the Visual standard comprises tools and algorithms supporting the coding of natural (pixel based) still images and video sequences as well as tools to support the compression of synthetic 2-D and 3-D graphic geometry parameters (i.e. compression of wire grid parameters, synthetic text).

The subsections below give an itemized overview of functionalities that the tools and algorithms of in the MPEG-4 visual standard.

3.5.1 Formats Supported

The following formats and bitrates are be supported by MPEG-4 Visual :

- bitrates: typically between 5 kbit/s and more than 1 Gbit/s
- Formats: progressive as well as interlaced video
- Resolutions: typically from sub-QCIF to 'Studio' resolutions (4k x 4k pixels)

3.5.2 Compression Efficiency

- For all bit rates addressed, the algorithms are very efficient. This includes the compact coding of textures with a quality adjustable between "acceptable" for very high compression ratios up to "near lossless".
- Efficient compression of textures for texture mapping on 2-D and 3-D meshes.
- Random access of video to allow functionalities such as pause, fast forward and fast reverse of stored video.

3.5.3 Content-Based Functionalities

- *Content-based coding* of images and video allows separate decoding and reconstruction of arbitrarily shaped video objects.
- *Random access* of content in video sequences allows functionalities such as pause, fast forward and fast reverse of stored video objects.
- *Extended manipulation of content* in video sequences allows functionalities such as warping of synthetic or natural text, textures, image and video overlays on reconstructed video content. An example is the mapping of text in front of a moving video object where the text moves coherently with the object.

3.5.4 Scalability of Textures, Images and Video

- *Complexity scalability in the encoder* allows encoders of different complexity to generate valid and meaningful bitstreams for a given texture, image or video.
- *Complexity scalability in the decoder* allows a given texture, image or video bitstream to be decoded by decoders of different levels of complexity. The reconstructed quality, in general, is related to the complexity of the decoder used. This may entail that less powerful decoders decode only a part of the bitstream.
- *Spatial scalability* allows decoders to decode a subset of the total bitstream generated by the encoder to reconstruct and display textures, images and video objects at reduced spatial resolution. A maximum of 11 levels of spatial scalability are supported in so-called 'fine-granularity scalability', for video as well as textures and still images.
- *Temporal scalability* allows decoders to decode a subset of the total bitstream generated by the encoder to reconstruct and display video at reduced temporal resolution. A maximum of three levels are supported.
- *Quality scalability* allows a bitstream to be parsed into a number of bitstream layers of different bitrate such that the combination of a subset of the layers can still be decoded into a meaningful signal. The bitstream parsing can occur either during transmission or in the decoder. The reconstructed quality, in general, is related to the number of layers used for decoding and reconstruction.
- *Fine Grain Scalability* – a combination of the above in fine grain steps, up to 11 steps

3.5.5 Shape and Alpha Channel Coding

- *Shape coding* assists the description and composition of conventional images and video as well as arbitrarily shaped video objects. Applications that benefit from binary shape maps with images are content-based image representations for image databases, interactive games, surveillance, and animation. There is an efficient technique to code binary shapes. A binary alpha map defines whether or not a pixel belongs to an object. It can be 'on' or 'off'.
- *'Gray Scale' or 'alpha' Shape Coding*
  An alpha plane defines the 'transparency' of an object, which is not necessarily uniform; it can vary over the object, so that, e.g., edges are more transparent (a technique called feathering). Multilevel alpha maps are frequently used to blend different layers of image sequences. Other applications that benefit from associated binary alpha maps with images are content-based image representations for image databases, interactive games, surveillance, and animation.

3.5.6 Robustness in Error Prone Environments

Error resilience allows accessing image and video over a wide range of storage and transmission media. This includes the useful operation of image and video compression algorithms in error-prone environments at low bit-rates (i.e., less than 64 Kbps). There are tools that address both the band-limited nature and error resiliency aspects of access over wireless networks.

3.5.7 Face and Body Animation

The 'Face and Body Animation' tools in the standard allow sending parameters that can define, calibrate and animate synthetic faces and bodies. These models themselves are not standardized by MPEG-4, only the parameters are, although there is a way to send, e.g., a well-defined face to a decoder.
The tools include:

- Definition and coding of face and body animation parameters (model independent):
  - Feature point positions and orientations to animate the face and body definition meshes
  - Visemes, or visual lip configurations equivalent to speech phonemes
- Definition and coding of face and body definition parameters (for model calibration):
  - 3-D feature point positions
  - 3-D head calibration meshes for animation
  - Personal characteristics
- Facial texture coding

3.5.8 Coding of 2-D Meshes with Implicit Structure

2D mesh coding includes:

- Mesh-based prediction and animated texture transfiguration
  - 2-D Delaunay or regular mesh formalism with motion tracking of animated objects
  - Motion prediction and suspended texture transmission with dynamic meshes.
- Geometry compression for motion vectors:
  - 2-D mesh compression with implicit structure & decoder reconstruction

3.5.9 Coding of 3-D Polygonal Meshes

MPEG-4 provides a suite of tools for coding 3-D polygonal meshes. Polygonal meshes are widely used as a generic representation of 3-D objects. The underlying technologies compress

the connectivity, geometry, and properties such as shading normals, colors and texture coordinates of 3-D polygonal meshes.

The Animation Framework eXtension (AFX, see further down) will provide more elaborate tools for 2D and 3D synthetic objects.

4 Extensions Underway

MPEG is currently working on a number of extensions:

4.1.1 New Video codec: "MPEG-4 Advanced Video Coding"

Work is ongoing on MPEG-4 part 10, 'Advanced Video Coding'. This codec is being developed jointly with ITU-T, in the so-called Joint Video Team (JVT). The JVT unites the standard world's video coding experts in a single group. The work currently underway is based on earlier work in ITU-T on 'H.26L'. H.26L and MPEG-4 part 10 will be the same. (H.26L will be renamed when it is done. The final name may be H.264, but that is not yet sure). MPEG-4 AVC/H.26L is slated to be ready by the end of 2002.

4.1.2 The Animation Framework eXtension, AFX

The Animation Framework extension (AFX – pronounced 'effects') provides an integrated toolbox for building attractive and powerful synthetic MPEG-4 environments. The framework defines a collection of interoperable tool categories that collaborate to produce a reusable architecture for interactive animated contents. In the context of AFX, a tool represents functionality such as a BIFS node, a synthetic stream, or an audio-visual stream.

AFX utilizes and enhances existing MPEG-4 tools, while keeping backward-compatibility, by offering:

- Higher-level descriptions of animations (e.g. inverse kinematics)
- Enhanced rendering (e.g. multi-texturing, procedural texturing)
- Compact representations (e.g. piecewise curve interpolators, subdivision surfaces)
- Low bitrate animations (e.g. using interpolator compression and dead-reckoning)
- Scalability based on terminal capabilities (e.g. parametric surfaces tessellation)
- Interactivity at user level, scene level, and client-server session level
- Compression of representations for static and dynamic tools

Compression of animated paths and animated models is required for improving the transmission and storage efficiency of representations for dynamic and static tools.

4.1.3 Audio extensions

There are two work items underway for improving audio coding efficiency even further.

a) Bandwidth extension

Bandwidth extension is a tool that gives a better quality perception over the existing audio signal, while keeping the existing signal backward compatible.
MPEG is investigating bandwidth extensions, and may standardize of one or both of:

1. General audio signals, to extend the capabilities currently provided by MPEG-4 general audio coders.

2. Speech signals, to extend the capabilities currently provided by MPEG-4 speech coders.

A single technology that addresses both of these signals is preferred. This technology shall be both forward and backward compatible with existing MPEG-4 technology. In other words, an MPEG-4 decoder can decode an enhanced stream and a new technology decoder can decode an MPEG-4 stream. There are two possible configurations for the enhanced stream: MPEG-4 AAC streams can carry the enhancement information in the DataStreamElement, while all MPEG-4 systems know the concept of elementary streams, which allow second Elementary Stream for a given audio object, containing the enhancement information.

b) Parametric coding

The MPEG-4 standard already provides a parametric coding scheme for coding of general audio signals for low bit-rates (HILN, "Harmonic Individual Lines and Noise"). The extension investigates parametric coding of general audio signals for the higher quality range, to extend the capabilities currently provided by HILN. Whenever possible this technology will build upon the existing MPEG-4 HILN technology.

5 Profiles in MPEG-4

MPEG-4 provides a large and rich set of tools for the coding of audio-visual objects. In order to allow effective implementations of the standard, subsets of the MPEG-4 Systems, Visual, and Audio tool sets have been identified, that can be used for specific applications. These subsets, called 'Profiles', limit the tool set a decoder has to implement. For each of these Profiles, one or more Levels have been set, restricting the computational complexity. The approach is similar to MPEG-2, where the most well known Profile/Level combination is 'Main Profile @ Main Level'. A Profile@Level combination allows:

- a codec builder to implement only the subset of the standard he needs, while maintaining interworking with other MPEG-4 devices built to the same combination, and
- checking whether MPEG-4 devices comply with the standard ('conformance testing').

Profiles exist for various types of media content (audio, visual, and graphics) and for scene descriptions. MPEG does not prescribe or advise combinations of these Profiles, but care has been taken that good matches exist between the different areas.

5.1 Visual Profiles

The visual part of the standard provides profiles for the coding of natural, synthetic, and synthetic/natural hybrid visual content. There are five profiles for natural video content:

1. The **Simple Visual Profile** provides efficient, error resilient coding of rectangular video objects, suitable for applications on mobile networks, such as PCS and IMT2000.
2. The **Simple Scalable Visual Profile** adds support for coding of temporal and spatial scalable objects to the Simple Visual Profile, It is useful for applications which provide services at more than one level of quality due to bit-rate or decoder resource limitations, such as Internet use and software decoding.
3. The **Core Visual Profile** adds support for coding of arbitrary-shaped and temporally scalable objects to the Simple Visual Profile. It is useful for applications such as those providing relatively simple content-interactivity (Internet multimedia applications).

4. The **Main Visual Profile** adds support for coding of interlaced, semi-transparent, and sprite objects to the Core Visual Profile. It is useful for interactive and entertainment-quality broadcast and DVD applications.
5. The **N-Bit Visual Profile** adds support for coding video objects having pixel-depths ranging from 4 to 12 bits to the Core Visual Profile. It is suitable for use in surveillance applications.

The profiles for synthetic and synthetic/natural hybrid visual content are:

6. The **Simple Facial Animation Visual Profile** provides a simple means to animate a face model, suitable for applications such as audio/video presentation for the hearing impaired.
7. The **Scalable Texture Visual Profile** provides spatial scalable coding of still image (texture) objects useful for applications needing multiple scalability levels, such as mapping texture onto objects in games, and high-resolution digital still cameras.
8. The **Basic Animated 2-D Texture Visual Profile** provides spatial scalability, SNR scalability, and mesh-based animation for still image (textures) objects and also simple face object animation.
9. The **Hybrid Visual Profile** combines the ability to decode arbitrary-shaped and temporally scalable natural video objects (as in the Core Visual Profile) with the ability to decode several synthetic and hybrid objects, including simple face and animated still image objects. It is suitable for various content-rich multimedia applications.

Version 2 adds the following Profiles for natural video:

10. The **Advanced Real-Time Simple Profile** (ARTS) provides advanced error resilient coding techniques of rectangular video objects using a back channel and improved temporal resolution stability with the low buffering delay. It is suitable for real time coding applications; such as the videophone, tele-conferencing and the remote observation.
11. The **Core Scalable Profile** adds support for coding of temporal and spatial scalable arbitrarily shaped objects to the Core Profile. The main functionality of this profile is object based SNR and spatial/temporal scalability for regions or objects of interest. It is useful for applications such as the Internet, mobile and broadcast.
12. The **Advanced Coding Efficiency (ACE) Profile** improves the coding efficiency for both rectangular and arbitrary shaped objects. It is suitable for applications such as mobile broadcast reception, the acquisition of image sequences (camcorders) and other applications where high coding efficiency is requested and small footprint is not the prime concern.

The Version 2 profiles for synthetic and synthetic/natural hybrid visual content are:

13. The **Advanced Scaleable Texture Profile** supports decoding of arbitrary-shaped texture and still images including scalable shape coding, wavelet tiling and error-resilience. It is useful for applications that require fast random access as well as multiple scalability levels and arbitrary-shaped coding of still objects. Examples are fast content-based still image browsing on the Internet, multimedia-enabled PDA's, and Internet-ready high-resolution digital still cameras.
14. The **Advanced Core Profile** combines the ability to decode arbitrary-shaped video objects (as in the Core Visual Profile) with the ability to decode arbitrary-shaped scalable still image objects (as in the Advanced Scaleable Texture Profile.) It is suitable for various content-rich multimedia applications such as interactive multimedia streaming over Internet.
15. The **Simple Face and Body Animation Profile** is a superset of the Simple Face Animation Profile, adding - obviously - body animation.

In subsequent Versions, the following Profiles were added:

16. The **Advanced Simple Profile** looks much like Simple in that it has only rectangular objects, but it has a few extra tools that make it more efficient: B-frames, ¼ pel motion compensation, extra quantization tables and global motion compensation.
17. The **Fine Granularity Scalability Profile** allows truncation of the enhancement layer bitstream at any bit position so that delivery quality can easily adapt to transmission and decoding circumstances. It can be used with *Simple* or *Advanced Simple* as a base layer.
18. The **Simple Studio Profile** is a profile with very high quality for usage in Studio editing applications. It only has I frames, but it does support arbitrary shape and in fact multiple alpha channels. Bitrates go up to almost 2 Gigabit per second.
19. The **Core Studio Profile** adds P frames to *Simple Studio*, making it more efficient but also requiring more complex implementations.

5.2 Audio Profiles

Four Audio Profiles have been defined in MPEG-4 V.1:

1. The **Speech Profile** provides HVXC, which is a very-low bit-rate parametric speech coder, a CELP narrowband/wideband speech coder, and a Text-To-Speech interface.
2. The **Synthesis Profile** provides score driven synthesis using SAOL and wavetables and a Text-to-Speech Interface to generate sound and speech at very low bitrates.
3. The **Scalable Profile**, a superset of the Speech Profile, is suitable for scalable coding of speech and music for networks, such as Internet and Narrow band Audio DIgital Broadcasting (NADIB). The bitrates range from 6 kbit/s and 24 kbit/s, with bandwidths between 3.5 and 9 kHz.
4. The **Main Profile** is a rich superset of all the other Profiles, containing tools for natural and synthetic Audio.

Another four Profiles were added in MPEG-4 V.2:

5. The **High Quality Audio Profile** contains the CELP speech coder and the Low Complexity AAC coder including Long Term Prediction. Scalable coding coding can be performed by the AAC Scalable object type. Optionally, the new error resilient (ER) bitstream syntax may be used.
6. The **Low Delay Audio Profile** contains the HVXC and CELP speech coders (optionally using the ER bitstream syntax), the low-delay AAC coder and the Text-to-Speech interface TTSI.
7. The **Natural Audio Profile** contains all natural audio coding tools available in MPEG-4, but not the synthetic ones.
8. The **Mobile Audio Internetworking Profile** (MAUI) contains the low-delay and scalable AAC object types including TwinVQ and BSAC. This profile is intended to extend communication applications using non-MPEG speech coding algorithms with high quality audio coding capabilities.

5.3 Graphics Profiles

Graphics Profiles define which graphical and textual elements can be used in a scene. These profiles are defined in the Systems part of the standard:

1. **Simple 2-D Graphics Profile** The Simple 2-D Graphics profile provides for only those graphics elements of the BIFS tool that are necessary to place one or more visual objects in a scene.

2. **Complete 2-D Graphics Profile** The Complete 2-D Graphics profile provides two-dimensional graphics functionalities and supports features such as arbitrary two-dimensional graphics and text, possibly in conjunction with visual objects.
3. **Complete Graphics Profile** The Complete Graphics profile provides advanced graphical elements such as elevation grids and extrusions and allows creating content with sophisticated lighting. The Complete Graphics profile enables applications such as complex virtual worlds that exhibit a high degree of realism.
4. The **3D Audio Graphics Profile** sounds like a contradictory in terms, but really isn't. This profile does not propose visual rendering, but graphics tools are provided to define the acoustical properties of the scene (geometry, acoustics absorption, diffusion, transparency of the material). This profile is used for applications that do environmental spatialization of audio signals. (See Section 12.1.7)

5.3.1 Profiles under Definition or Consideration

The following profiles were under development at the time of writing this Overview; their inclusion in the standard was highly likely, but not guaranteed.

5. The *Simple 2D+Text* profile looks like simple 2D, adding the BIFS nodes to display text which can be colored or transparent. Like simple 2D, this is a useful profile for low-complexity audiovisual devices.
6. The *Core 2D* Profile supports fairly simple 2D graphics and text. Meant for set tops and similar devices, it can do such things as picture-in-picture, video warping for animated advertisements, logos, and so on.
7. The *Advanced 2D* profile contains tools for advanced 2D graphics. Using it, one can implement cartoons, games, advanced graphical user interfaces, and complex, streamed graphics animations.
8. The *X3D Core* profile is the only 3D profile that is likely to be added to MPEG-4. It is compatible with Web3D's X3D core profile under development [Web3D], and it gives a rich environment for games, virtual worlds and other 3D applications.

5.4 Scene Graph Profiles

Scene Graph Profiles (or Scene Description Profiles), defined in the Systems part of the standard, allow audiovisual scenes with audio-only, 2-dimensional, 3-dimensional or mixed 2-D/3-D content.

1. The *Audio* Scene Graph Profile provides for a set of BIFS scene graph elements for usage in audio only applications. The Audio Scene Graph profile supports applications like broadcast radio.
2. The *Simple 2-D* Scene Graph Profile provides for only those BIFS scene graph elements necessary to place one or more audio-visual objects in a scene. The Simple 2-D Scene Graph profile allows presentation of audio-visual content with potential update of the complete scene but no interaction capabilities. The Simple 2-D Scene Graph profile supports applications like broadcast television.
3. The *Complete 2-D* Scene Graph Profile provides for all the 2-D scene description elements of the BIFS tool. It supports features such as 2-D transformations and alpha blending. The Complete 2-D Scene Graph profile enables 2-D applications that require extensive and customized interactivity.
4. The *Complete* Scene Graph profile provides the complete set of scene graph elements of the BIFS tool. The Complete Scene Graph profile enables applications like dynamic virtual 3-D world and games.
5. The *3D Audio* Scene Graph Profile provides the tools three-dimensional sound positioning in relation either with acoustic parameters of the scene or its perceptual attributes. The user

can interact with the scene by changing the position of the sound source, by changing the room effect or moving the listening point. This Profile is intended for usage in audio-only applications.

5.4.1 Profiles under definition

At the time of writing, the following profiles were likely to be defined:

6. The *basic 2D* profile provides basic 2D composition for very simple scenes with only audio and visual elements. Only basic 2D composition and audio and video nodes interfaces are included. These nodes are required to put an audio or a video object in the scene.
7. The *Core 2D* profile has tools for creating scenes with visual and audio objects using basic 2D composition. Included are quantization tools, local animation and interaction, 2D texturing, Scene tree updates, and the inclusion of subscenes through weblinks. Also included are interactive service tools (ServerCommand, MediaControl, and MediaSensor), to be used in video-on-demand services.
8. The *Advanced 2D* profile forms a full superset of the basic 2D and core 2D profiles. It adds scripting, the PROTO tool, BIF-Anim for streamed animation, local interaction and local 2D composition as well as advanced audio.
9. The *Main 2D* profile adds the FlexTime model to Core 2D, as well as Layer2D and WorldInfo nodes and all input sensors. This profile was designed to be an interoperability point with tSMIL (see [SMIL]). It provides a very rich set of tools for highly interactive applications on, e.g., the World Wide Web. This name might still change.
10. The *X3D core* profile was designed to be a common interworking point with the Web3D specifications [Web3D] and the MPEG-4 standard. The same profile is will be in a Web3D specification. It includes the nodes for an implementation of 3D applications on a low-footprint engine, reckoning with the limitations of software renderers.

5.5 MPEG-J Profiles

Two MPEG-J Profiles exist: Personal and Main:

1. *Personal* - a lightweight package for personal devices.
   The personal profile addresses a range of constrained devices including mobile and portable devices. Examples of such devices are cell video phones, PDAs, personal gaming devices. This profile includes the following packages of MPEG-J APIs:
   1. Network
   2. Scene
   3. Resource

2. *Main* - includes all the MPEG-J API's.
   The Main profile addresses a range of consumer devices including entertainment devices. Examples of such devices are set top boxes, computer based multimedia systems etc. It is a superset of the Personal profile. Apart from the packages in the Personal profile, this profile includes the following packages of the MPEG-J APIs:
   4. Decoder
   5. Decoder Functionality
   6. Section Filter and Service Information

5.6 Object Descriptor Profile

The Object Descriptor Profile includes the following tools:

- Object Descriptor (OD) tool
- Sync Layer (SL) tool
- Object Content Information (OCI) tool
- Intellectual Property Management and Protection (IPMP) tool

Currently, only one profile is defined that includes all these tools. The main reason for defining this profile is not subsetting the tools, but rather defining levels for them. This applies especially to the Sync Layer tool, as MPEG-4 allows multiple time bases to exist. In the context of Levels for this Profile, restrictions can be defined, e.g. to allow only a single time base.

6 Verification Testing: checking MPEG's performance

MPEG carries out verification tests to check whether the standard delivers what it promises. The test results can be found on MPEG's home page,
http://www.cselt.it/mpeg/quality_tests.htm
The main results are described below; more verification tests are planned.

6.1 Video

A number of MPEG-4's capabilities have been formally evaluated using subjective tests. Coding efficiency, although not the only MPEG-4 functionality, is an important selling point of MPEG-4, and one that has been tested more thoroughly. Also error robustness has been put to rigorous tests. Furthermore, scalability tests were done and for one specific profile the temporal resolution stability was examined. Many of these tests address a specific profile.

6.1.1 Coding Efficiency Tests

a) Low and Medium Bit rates (version 1)

In this Low and Medium Bitrates Test, frame-based sequences were examined, with MPEG-1 as a reference. (MPEG-2 would be identical for the progressive sequences used, except that MPEG-1 is a bit more efficient as it uses less overhead for header information). The test uses typical test sequences for CIF and QCIF resolutions, encoded with the same rate control for both MPEG-1 and MPEG-4 to compare the coding algorithms without the impact of different rate control schemes. The test was performed for low bit rates starting at 40 kbps to medium bit rate up to 768 kbps.

The tests of the Coding Efficiency functionality show a clear superiority of MPEG-4 toward MPEG-1 at both the low and medium bit rate coding conditions whatever the criticality of the scene. The human subjects have consistently chose MPEG-4 as statistically significantly superior by one point difference for a full scale of five points.

b) Content Based Coding (version 1)

The verification tests for Content Based Coding compare the visual quality of object-based versus frame-based coding. The major objective was to ensure that object-based coding can be supported without impacting the visual quality. Test content was chosen to cover a wide variety of simulation conditions, including video segments with various types of motions and encoding complexities. Additionally, test conditions were established to cover low bit rates ranging from 256kb/s to 384kb/s, as well as high bit-rates ranging from 512kb/s to 1.15Mb/s. The results of the tests clearly demonstrated that object-based functionality is provided by MPEG-4 with no overhead or loss in terms of visual quality, when compared to frame-based

coding. There is no statistically significant difference among any object-based case and the relevant frame-based ones. Hence the conclusion: MPEG-4 is able to provide content-based functionality without introducing any loss in terms of visual quality.

c) Advanced Coding Efficiency (ACE) Profile (version 2)

The formal verification tests on Advanced Coding Efficiency (ACE) Profile were performed to check whether three new Version 2 tools, as included the MPEG-4 Visual Version 2 ACE Profile (Global Motion Compensation, Quarter Pel Motion Compensation and Shape-adaptive DCT) enhance the coding efficiency compared with MPEG-4 Visual Version 1. The tests explored the performance of the ACE Profile and the MPEG-4 Visual Version 1 Main Profile in the object-based low bit rate case, the frame-based low bit rate case and the frame-based high bit rate case. The results obtained show a clear superiority of the ACE Profile compared with the Main Profile; more in detail:

- For the object based case, the quality provided by the ACE Profile at 256 kb/s is equal to the quality provided by Main Profile at 384 kb/s.
- For the frame based at low bit rate case, the quality provided by the ACE Profile at 128 kb/s and 256 kb/s is equal to the quality provided by Main Profile at 256 kb/s and 384 kb/s respectively.
- For the frame based at high bit rate case, the quality provided by the ACE Profile at 768 kb/s is equal to the quality provided by Main Profile at 1024 kb/s.

When interpreting these results, it must be noted that the MPEG-4 Main Profile is already more efficient than MPEG-1 and MPEG-2.

6.1.2 Error Robustness Tests

a) Simple Profile (version 1)

The performance of error resilient video in the MPEG-4 Simple Profile was evaluated in subjective tests simulating MPEG-4 video carried in a realistic multiplex and over ditto radio channels, at bitrates between 32 kbit/s and 384 kbit/s. The test used a simulation of the residual errors after channel coding at bit error rates up to 10^{-3}, and the average length of the burst errors was about 10ms. The test methodology was based on a continuous quality evaluation over a period of three minutes. In such a test, subjects constantly score the degradation they experience.

The results show that the average video quality achieved on the mobile channel is high, that the impact of errors is effectively kept local by the tools in MPEG-4 video, and that the video quality recovers quickly at the end of periods of error. These excellent results were achieved with very low overheads, less than those typically associated with the GOP structure used in MPEG-1 and MPEG-2 video.

b) Advanced Real-Time Simple (ARTS) Profile (version 2)

The performance of error resilient video in MPEG-4 ARTS Profile was checked in subjective tests similar to those mentioned in the previous section, at bitrates between 32 kbit/s and 128 kbit/s. In this case, the residual errors after channel coding was up to 10^{-3}, and the average length of the burst errors was about 10 ms (called “critical”) or 1 ms (called “very critical” - this one is more critical because the same amount of errors is more spread over the bitstream than in the “critical” case).

The results show a clear superiority of the ARTS Profile over the Simple Profile for both the error cases (“critical” and “very critical”). More in detail the ARTS Profile outperforms Simple Profile in the recovery time from transmission errors. Furthermore ARTS Profile in the “critical” error condition provides results that for most of the test time are close to a complete transparency, while Simple Profile is still severely affected by errors. These excellent results

were achieved with very low overheads and very fast error recovery provided the NEWPRED, and under low delay conditions.

6.1.3 Temporal Resolution Stability Test

a) Advanced Real-Time Simple (ARTS) Profile (version 2)

This test explored the performance of a video codec using the Dynamic Resolution Conversion technique that adapts the resolution to the video content and to circumstances in real-time. Active scene content was coded at 64 kb/s, 96 kb/s and 128 kb/s datarates. The results show that at 64 kbit/s, it outperforms the already effective Simple Profile operating at 96 kbit/s, and at 96 kb/s, the visual quality is equally to that of the Simple profile at 128 kbit/s. (The Simple profile already compares well to other, existing systems.)

6.1.4 Scalability Tests

a) Simple Scalable Profile (version 1)

The scalability test for the Simple Scalable Profile was designed to verify that the quality provided by Temporal Scalability tool in Simple Scalable Profile compared to the quality provided by Single Layer coding in Simple Profile, and to the quality provided by Simulcast coding in Simple Profile.

In this test, 5 sequences with 4 combinations of bitrates were used:

a) 24 kbps for base layer and 40 kbps for enhancement layer.
b) 32 kbps for both layers.
c) 64 kbps for the base layer and 64 kbps for the enhancement layer.
d) 128 kbps for both layers.

The formal verification tests showed that in all the given conditions, the Temporal Scalability coding in Simple Scalable Profile exhibits the same or slightly lower quality than can be achieved using Single layer coding in Simple Profile. Furthermore it is evident that the Temporal Scalability coding in Simple Scalable Profile provides better quality than the simulcast coding in Simple Profile for that condition. (Simulcast entails simultaneously broadcasting or streaming at multiple bitrates.)

b) Core Profile (version 1)

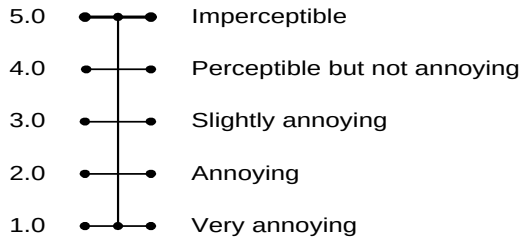
The verification test was designed to evaluate the performance of MPEG-4 video Temporal Scalability tool in the Core Profile.

The test was performed using the "Single Stimulus" method. The subjects were to evaluate how annoying the impairments of compressed sequences were, with and without use of temporal scalability. The test was conducted using a total of 45 subjects in two different laboratories and the results showed that the quality of sequences encoded using MPEG-4 temporal scalability tools are comparable to the quality of sequences encoded without temporal scalability. Furthermore it is evident that the Temporal Scalability tool in Core Profile provides better quality than the simulcast coding in Core Profile for that condition.

6.2 Audio

MPEG-4 audio technology is composed of many coding tools. Verification tests have focused on small sets of coding tools that are appropriate in one application arena, and hence can be effectively compared. Since compression is a critical capability in MPEG, the verification tests have for the most part compared coding tools operating at similar bit rates. The results of these tests will be presented progressing from higher bit rate to lower bit rates. The exception to this is the error robustness tools, whose performance will be noted at the end of this section.

The primary purpose of verification tests is to report the subjective quality of a coding tool operating at a specified bit rate. The audio tests concerned with very high subjective audio quality report this on the ITU-R BS.1116 subjective *impairment scale* while for lower bitrates the ITU-R BS.1284 *quality scale* was employed. Both are continuous 5-point scales with subjective anchors as shown here:



BS.1116 Impairment Scale

5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

BS.1284 Quality Scale

Please note that it is not meaningful to compare grades across those scales.

The performance of the various MPEG-4 coding tools are summarized in the following table. To better enable the evaluation of MPEG-4 technology, several coders from MPEG-1/2 and the ITU-T were included in the tests and their evaluation has also been included in the table. In the table results from the same test are delimited by heavy lines. These results can be directly compared. Results taken from different tests should not be compared, but nevertheless give an indication of the expected quality of a coding tool operating at a specific bit rate.

Coding tool	Number of channels	Total bit rate	Grading scale type	Typical subjective quality
AAC	5	320 kb/s	impairment	4.6
1995 Backward Compatible MPEG-2 Layer II	5	640 kb/s	impairment	4.6
AAC	2	128 kb/s	impairment	4.8
AAC	2	96 kb/s	impairment	4.4
MPEG-1 Layer II	2	192 kb/s	impairment	4.3
MPEG-1 Layer III	2	128 kb/s	impairment	4.1
AAC	1	24 kb/s	quality	4.2
Scalable: CELP base and AAC enhancement	1	6 kb/s base, 18 kb/s enh.	quality	3.7
Scalable: Twin VQ base and AAC enhancement	1	6 kb/s base, 18 kb/s enh.	quality	3.6
AAC	1	18 kb/s	quality	3.2
G.723	1	6.3 kb/s	quality	2.8
Wideband CELP	1	18.2 kb/s	quality	2.3
BSAC	2	96 kb/s	quality	4.4
BSAC	2	80 kb/s	quality	3.7
BSAC	2	64 kb/s	quality	3.0
AAC – LD (20 ms one-	1	64 kb/s	quality	4.4

way delay)				
G.722	1	64 kb/s	quality	4.2
AAC – LD (30 ms one-way delay)	1	32 kb/s	quality	3.4
Narrowband CELP	1	6 kb/s	quality	2.5
Twin VQ	1	6 kb/s	quality	1.8
HILN	1	16 kb/s	quality	2.8
HILN	1	6 kb/s	quality	1.8

Coding tools were tested under circumstances that assessed their strengths. The salient features of the MPEG-4 audio coding tools are briefly noted here.

When coding 5-channel material at 64 kb/s/channel (320 kbit/s) Advanced Audio Coding (AAC) Main Profile was judged to have "indistinguishable quality" (relative to the original) according to the EBU definition. When coding 2-channel material at 128 kbps both AAC Main Profile and AAC Low Complexity Profile were judged to have "indistinguishable quality" (relative to the original) according to the EBU definition.

The two scaleable coders, CELP base with AAC enhancement, and TwinVQ base with AAC enhancement both performed better than an AAC "multicast" operating at the enhancement layer bitrate, but not as good as an AAC coder operating at the total bitrate.

The wideband CELP coding tool showed excellent performance for speech-only signals. (The verification test result shown is for both speech and music signals.)

Bit Slice Arithmetic Coding (BSAC) provides a very fine step bitrate scalability. At the top of the scalability range it has no penalty relative to single-rate AAC, however at the bottom of the scale it has a slight penalty relative to single-rate AAC.

Relative to normal AAC, Low Delay AAC (AAC LD) provides equivalent subjective quality, but with very low on-way delay and only a slight increase in bit rate.

Narrowband CELP, TwinVQ and Harmonic Individual Lines and Noise (HILN) all have the ability to provide very high signal compression.

The Error Robustness (ER) tools provide equivalently good error robustness over a wide range of channel error conditions, and does so with only a modest overhead in bit rate. Verification test results suggest that the ER tools used with an audio coding system provide performance in error-prone channels that is "nearly as good" as the same coding system operating over a clear channel.

7 The MPEG-4 Industry Forum

The MPEG-4 Industry Forum is a not-for-profit organization with the following goal: *To further the adoption of the MPEG-4 Standard, by establishing MPEG-4 as an accepted and widely used standard among application developers, service providers, content creators and end users.*

The following is a non-exhaustive excerpt from M4IF's Statutes about the way of operation:

- The purpose of M4IF shall be pursued by: promoting MPEG-4, making available information on MPEG-4, making available MPEG-4 tools or giving information on where to obtain these, creating a single point for information about MPEG-4, creating industrial focus around the usage of MPEG-4
- The goals are realized through the open international collaboration of all interested parties, on reasonable terms applied uniformly and openly. M4IF will contribute the results of its activities to appropriate formal standards bodies if applicable.
- The business of M4IF shall not be conducted for the financial profits of its Members but for their mutual benefits.
- Any corporation and individual firm, partnership, governmental body or international organization supporting the purpose of M4IF may apply for Membership.
- Members are not bound to implement or use specific technology standards, or recommendations by virtue of participation in M4IF.
- There is no licensing requirement attached to M4IF membership, and M4IF will not set any licensing terms for MPEG-4 technology. (There can, however, be studies about what licensing models would suite MPEG-4's operational environments)
- The membership fee is currently US$ 3,000 per year, and US$ 300 for not-for-profit organizations.

M4IF has its homepage: http://www.m4if.org
Interested parties can visit that page for details, and they can also sign up to public mail lists (an information list and a technical list). The site also has all the information about formally joining the Forum.

The Forum currently has more than 100 members and some 1,500 people subscribed to its mail lists, with a very broad, worldwide representation from the following industries: Consumer Electronics, Computer, Telecommunications, Research Institutions . Also, some of the members are 'business users' of MPEG-4. Among the list of current participants, there are many large and small companies that develop or deploy MPEG-4 technology. Membership goes beyond the MPEG constituency, partly because some of the smaller companies find it hard to comply with the requirements for participation in MPEG (which vary from one country to another) and also because some companies just don't need to be involved in the standardization phase.

The activities of M4IF generally start where MPEG stops. This includes issues that MPEG cannot deal with, e.g. because of ISO rules, such as discussing licensing issues. This following is a list of M4IF's current activities:

- Promoting the standard, and serving as a single point of information on MPEG-4 technology, products and services;
- Initiating discussions leading to the potential establishment of patent pools outside of M4IF, that should grant a license to an unlimited number of applicants throughout the world under reasonable terms and conditions that are demonstrably free of any unfair competition;
  this work includes studying licensing models for downloadable software decoders, such as internet players;
- Organization of MPEG-4 exhibitions and tutorials. There have been 2 successful instances of the Workshop and Exhibition on MPEG-4; the third will be held in June 2002;
- Interoperability testing, leading to certification of MPEG-4 products.

M4IF anticipates holding some 3 physical meetings per year, with a slightly higher frequency in the current start-up phase. About a hundred people from all over the world have attended these meetings. The focus has been on initiating the patent pools, but it will now shift towards interoperability tests. See the home page for meeting details.

8 Licensing of patents necessary to implement MPEG-4

Disclaimer: the author and editor of this Section has done his best to adequately reflect the situation around licensing of MPEG-4, but he is not a lawyer and the explanation below may not be fully correct. Although he has been very careful in writing this all down, neither the author nor MPEG can be held responsible for the consequence of any errors in this text. Please consult a lawyer if you want to understand exactly how this works.

8.1 Roles in Licensing MPEG-4

Licensing MPEG-4 is an important issue, and also one that is cause for much confusion. First, on should understand the roles of the different organizations involved in getting MPEG-4 deployed. Below is a short clarification of the role of some of the main players.

ISO/IEC MPEG is the group that makes MPEG standards. MPEG does not (and cannot, under ISO rules) deal with patents and licensing, other than requiring companies whose technologies are adopted into the standard to sign a statement that they will license their patents on Reasonable and Non-Discriminatory Terms (also called RAND terms) to all parties that wish to create a standards-compliant device, (hardware or software) or create a standards-compliant bitstream. 'Non-Discriminatory' means that the patent needs to be licensed to al parties that wish to implement MPEG-4 on the same terms. 'Reasonable' is not further defined anywhere.

In developing MPEG-4 part 10 / H.26L, the ITU/MPEG Joint Video Team is now attempting to establish a royalty-free baseline coder, and to this end it also asks of proposers to submit a statement that specifies whether they would want to make available any necessary patents on a royalty-free basis, if (and only if) others are prepared to license under the same terms.

The MPEG-4 Industry Forum, M4IF has in its statutes that it shall not license patents or determine licensing fees, but has nonetheless played an important role in driving the availability of licenses for the patents needed to implement MPEG-4. M4IF can discuss issues pertaining to licensing, and has acted as a catalyst, in the very literal sense, in getting patent pools going. In the years 1999 and 2000, M4IF adopted a series of resolutions recommending on ways that a joint license ('patent pool') might be established; it also mentioned names of parties/people that could play a role in this process. The process was detailed for the Systems, Visual and Audio parts of the standard, and involved an independent evaluator and a neutral administrator in all cases. Even though it will not actively pursue these, M4IF encourages alternative patent pools to be created, the more the better, and if possible even competing ones. (Competition is good, also in licensing, for the same reasons as why technology competition is good). It should be noted that no-one is forced to do business with any patent pool; one can also go straight to all the individual licensors (e.g., at least 18 in MPEG-4 Visual) as the licenses are always non-exclusive. However, doing so is cumbersome and there is a risk that negotiating 18 individual licenses turn out more costly than doing business with a one-stop joint licensing scheme.

M4If has recommended that licensors create patent joint licensing schemes for specific profiles, as profiles are the interoperability points of MPEG-4, and only when one implements a profile there is a standards-compliant implementation. Also, M4IF held a poll among its members to determine for which profile there was the most interest.

There are the 'patent pools' (joint licensing schemes) and their administrators. It is the licensors that determine who will be their licensing agent(s), and they alone. Other parties (including ISO/IEC MPEG, M4IF or the licensees) have no say in this choice. MPEG LA is an example of a licensing administrator, licensing MPEG-2 Video and MPEG-2 Systems. MPEG LA also announced licensing for MPEG-4 Visual. Dolby licenses MPEG-2 AAC, and has announced a joint licensing scheme for some of the patents needed to implement MPEG-4 AAC. Thomson licenses MP3 (MPEG-1 Layer III Audio). Patent holders determine the fees; MPEG LA, Dolby, Thomson (and there are others) collect on behalf of the patent owners and distribute the proceeds.

Patent pools usually (always?) allow new patents to enter the pool when they are found essential. Sometimes this is because patents were submitted later, sometimes because they issued only at a later date. A license pool is never (at least in the case of MPEG standards, and as far as the author knows) 'closed' after a certain date. In MPEG-2, many patents were added after the start of licensing.

It should be clear – yet cannot be stressed enough – that neither M4IF not MPEG receives even one cent of the collected royalties, nor does do they want to. M4IF has among its members licensors, licensees and entities that are neither of those. Collectively, the members have an interest in fair and reasonable licensing, because the standard will fail without it.

8.2 Licensing Situation

This section contains an overview of the current situation with respect to MPEG-4 licensing. See for last-minute information http://www.m4if.org/patents/index.php

8.2.1 Systems

M4IF issued a recommendation in September 2000. An independent evaluator looked at all submissions, and a number of patents form 6 companies were found essential. A press release was issued by MPEGLA on behalf of these patent holders in November 2001. Nothing was heard since; the wait is for more news. The hope is that more news will follow once licensing for Visual has been finalized.

8.2.2 Visual

The first announcement of a plan that should lead to a joint license was done on 13 April. The licensing activity was then hoped to start in October 2000. This schedule has been delayed considerably, but at least 18 holders of essential patents in Simple and Core profile were identified, and provisional licensing terms were announced on 31 January 2002. They have caused much commotion; many parties believed that these are not fit for several of MPEG-4's application areas. At the time of writing of this document (April 2002) the market was waiting for adaptations.

In the meantime, a call for essential patents for Advanced Simple, Fine Granularity Scalability and Simple Scalable was also issued. No statements about the result of this call have surfaced yet.

8.2.3 Audio

A press release announcing plans for a joint licensing scheme was announced in March 2001, seeking to encourage joint licensing schemes to be established for the Profiles Speech, Mobile Audio Internetworking and High Quality Audio. Progress was reported on 19 October 2001, and since then the group has been silent. In April 2002 Dolby announced that it would start licensing patents from 5 companies necessary to implement MPEG-4 AAC. While the terms were met with enthusiasm by the market, it is likely that more license holders exist for MPEG-4 AAC and MPEG-4 AAC does not constitute a profile in itself; rather, it is part of High Quality Audio and MAUI Profiles.

9 Deployment of MPEG-4

MPEG-4 is deployed in a number of environments. Full MPEG-4 Systems, with MPEG-4 Visual, Audio and Systems technology can be bought today from a number of providers, see http://www.m4if.org/products/ There are many providers for MPEG-4 Visual and MPEG-4 Audio codecs, and for, e.g., MPEG-4 Face Animation.

MPEG-4 Video and the MPEG-4 file format can be found in the 3GPP and 3GPP2 specifications for mobile multimedia terminals. A very widespread application of MPEG-4 Visual is the DivX coder, see www.divxnetwords.com

This implementation of MPEG-4 Visual is combined with MP3 (MPEG-1 Layer III) Audio, and it looks like the file format is headed to MP4 compliance. There is some confusion about whether this is really a compliant MPEG-4 implementation. The authors of the software have joined the interoperability work in M4IF, and recently announced support for MPEG-4 Advanced Simple Profile.

Perhaps the first MPEG-4 deployment was the ‘Eggy’, which was sold by NTT DoCoMo. It was a small retrieval terminal for MPEG-4 visual content. NTT DoCoMo has also released the FOMA, a multimedia 3G cell phone which uses MPEG-4 Visual.

The Internet Streaming Media Alliance (ISMA), founded in by Apple, Cisco, IBM, Kasenna, Philips and Sun, has specified a fully end-to-end interoperable system for internet streaming, including MPEG-4 Visual, MPEG-4 Audio and the transport of RTP as described above.

Microsoft’s Windows Media software contains an MPEG-4 encoder and decoder. This is an implementation of MPEG-4 Simple Profile, the simplest and lowest cost Visual Profile in MPEG-4. While Microsoft indicates that this implementation is fully compliant, it is ‘packaged’ in a Microsoft’s proprietary way.

Real Networks supports MPEG-4 through a certified plug-in from Envivio. Real has announced native MPEG-4 support for an unspecified date.

Apple’s Quicktime version 6, which includes MPEG-4 support, has been demonstrated at e.g. NAB 2002. Apple has stated to not release QT6 until the licensing situation is resolved to Apple’s satisfaction.

MPEG members are kindly requested to send more deployment information to the editor of this document.

10 Detailed technical description of MPEG-4 DMIF and Systems

This Section contains a detailed overview of the MPEG-4 Standard. It first described the entire system, and then describes all parts of the system in subsections.

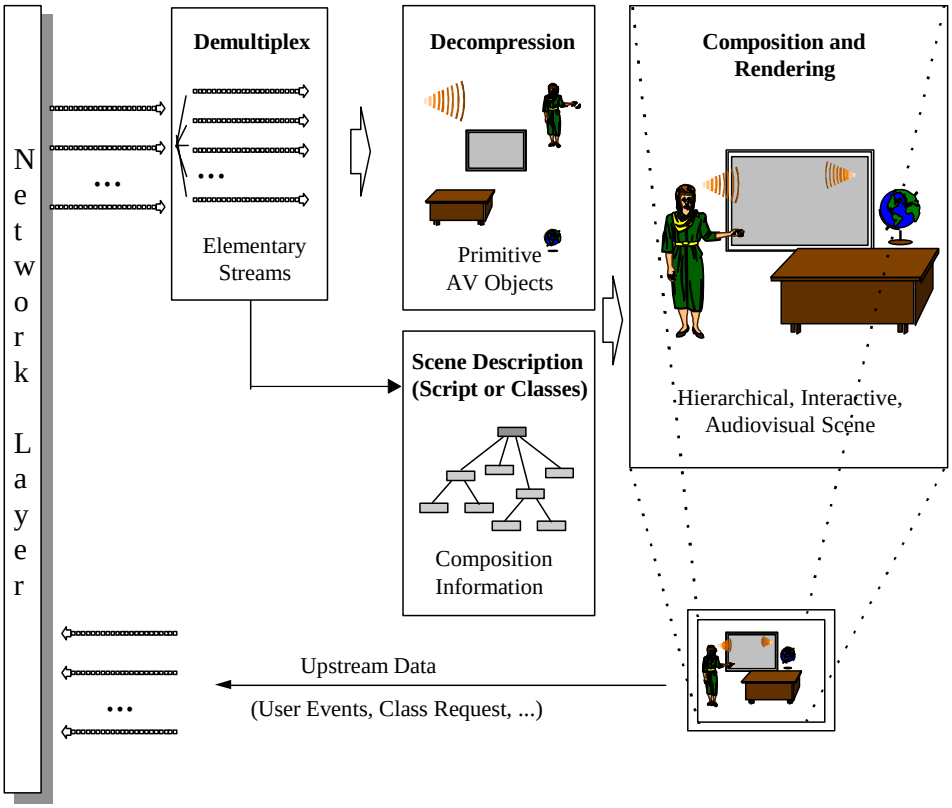


Figure 4- Major components of an MPEG-4 terminal (receiver side)

Figure 4 shows how streams coming from the network (or a storage device), as TransMux Streams, are demultiplexed into FlexMux Streams and passed to appropriate FlexMux demultiplexers that retrieve Elementary Streams. How this works is described in Section 10.3. The Elementary Streams (ESs) are parsed and passed to the appropriate decoders. Decoding recovers the data in an AV object from its encoded form and performs the necessary operations to reconstruct the original AV object ready for rendering on the appropriate device. Audio and visual objects are represented in their coded form, which is described in sections 12 and 10.11 respectively. The reconstructed AV object is made available to the composition layer for potential use during scene rendering. Decoded AVOs, along with scene description information, are used to compose the scene as described by the author. Scene description and Composition are explained in Section 10.6. The user can, to the extent allowed by the author, interact with the scene which is eventually rendered and presented. Section 10.6.1 describes this interaction. Sections 10.8 and 10.11 discuss the 'Intellectual Property Management and Protection' and Object Content Information respectively. Sections 8.9 and 8.10 describe Version 2 additions: the File Format and MPEG-J.

10.1 Transport of MPEG-4

MPEG-4 is transport-agnostic, and designed that way on purpose. This means that MPEG-4 content can be carried over many different transport layers, and move form one transport to the other. There are some cases, however, where MPEG did some work relating to the transport of MPEG-4 content.

10.1.1 MPEG-4 on MPEG-2

An often-asked question is 'when will MPEG-4 replace MPEG-2'? Then answer is invariably: "certainly not anytime soon, as there are tens of billions of dollars invested in MPEG-2, and MPEG is the last committee to want to make these investments useless." Rather, MPEG has taken care that MPEG-2 investments can be leveraged when deploying MPEG-4 technology. Notably, MPEG has taken care that MPEG-4 content can be transported over an MPEG-2 transport stream (MPEG-2 TS). This is done by amending the MPEG-2 Systems standard, 13818-1. This amendment is part of ISO/IEC 13818-1:2000, the 200 edition of MPEG-2 Systems.

10.1.2 MPEG-4 over IP

The specifications on the carriage of MPEG-4 contents over IP networks are developed jointly with IETF AVT working group. These include a framework and several RTP payload format specifications. The framework is standardized as both a part 8 of MPEG-4, i.e. ISO/IEC 14496-8 and informative RFC in IETF. RTP payload format specifications are only standardized as a standard track RFC in IETF.
Framework is an umbrella specification for the carriage and operation of MPEG-4 sessions over IP-based protocols, including RTP, RTSP, and HTTP, among others. It provides a framework for the carriage of MPEG-4 contents over IP networks and guidelines for designing payload format specifications for the detailed mapping of MPEG-4 content into several IP-based protocols. To assure compatibility between different RTP payload formats, framework defines a conformance point as illustrated in the Figure 1. To conform this framework all the payload formats shall provide normative mapping functions to reconstruct logical MPEG-4 SL packets. Framework also defines the standard MIME types associated with MPEG-4 contents.

Several RTP payload formats are developed under this framework including generic payload format and FlexMux payload format. Generic RTP payload format specify a homogeneous carriage of various MPEG-4 streams. It defines a simple but efficient mapping between logical MPEG-4 SL packets and RTP packets. It also supports concatenation of multiple SL packets into one RTP packets to minimize overheads. FlexMux payload format specifies a carriage of FlexMux packetized streams via RTP packets. It includes a payload formats to convey FlexMux descriptors to dynamically signal the configuration of FlexMux.

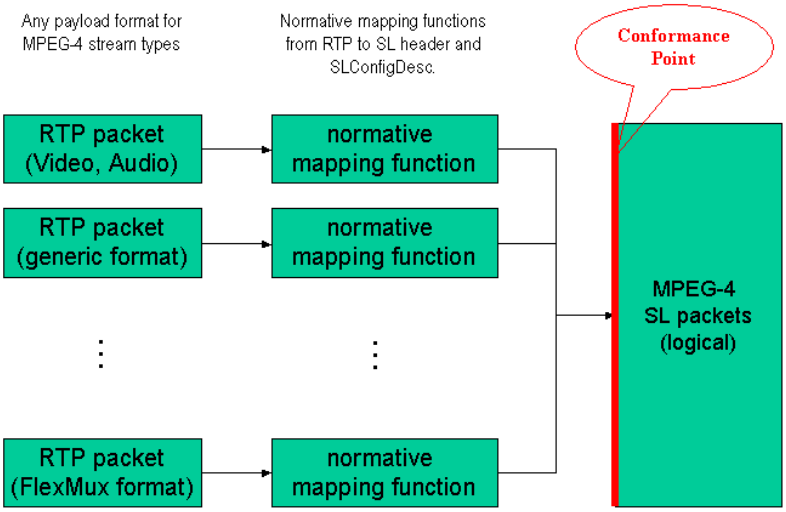


Figure 5. RTP packet to logical SL packet mapping

10.2 DMIF

DMIF (Delivery Multimedia Integration Framework) is a session protocol for the management of multimedia streaming over generic delivery technologies. In principle it is similar to FTP. The only (essential!) difference is that FTP returns data, DMIF returns pointers to where to get (streamed) data

When FTP is run, the very first action it performs is the setup of a session with the remote side. Later, files are selected and FTP sends a request to download them, the FTP peer will return the files in a separate connection.

Similarly, when DMIF is run, the very first action it performs is the setup of a session with the remote side. Later, streams are selected and DMIF sends a request to stream them, the DMIF peer will return the pointers to the connections where the streams will be streamed, and then also establishes the connection themselves.

Compared to FTP, DMIF is both a framework and a protocol. The functionality provided by DMIF is expressed by an interface called DMIF-Application Interface (DAI), and translated into protocol messages. These protocol messages may differ based on the network on which they operate.

The Quality of Service is also considered in the DMIF design, and the DAI allows the DMIF user to specify the requirements for the desired stream. It is then up to the DMIF implementation to make sure that the requirements are fulfilled. The DMIF specification provides hints on how to perform such tasks on a few network types, such as the Internet.

The DAI is also used for accessing broadcast material and local files, this means that a single, uniform interface is defined to access multimedia contents on a multitude of delivery technologies.

As a consequence, it is appropriate to state that the integration framework of DMIF covers three major technologies, interactive network technology, broadcast technology and the disk technology; this is shown in the Figure 6 below.

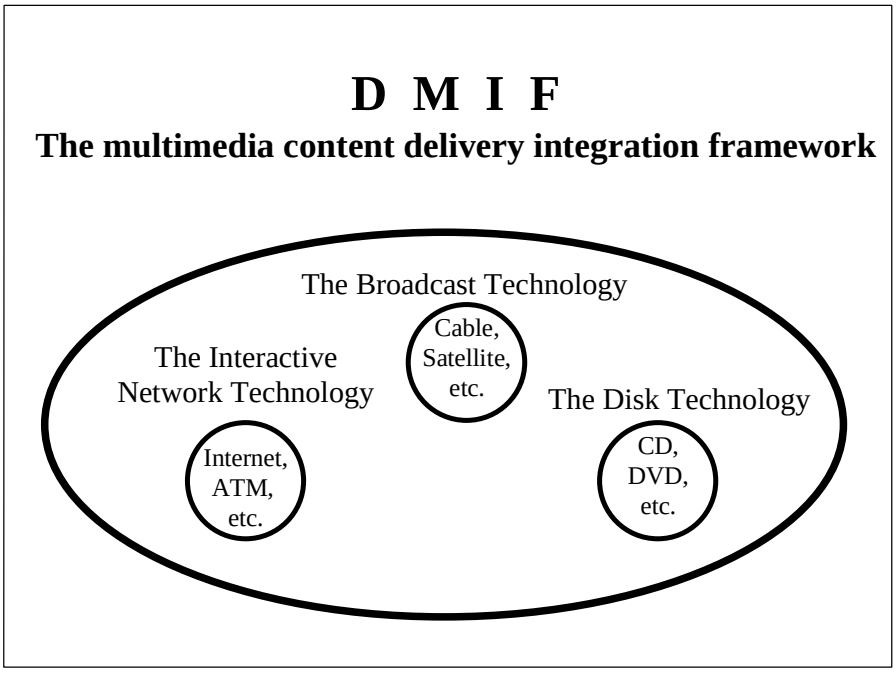


Figure 6 - DMIF addresses the delivery integration of three major technologies

The DMIF architecture is such that applications that rely on DMIF for communication do not have to be concerned with the underlying communication method. The implementation of DMIF takes care of the delivery technology details presenting a simple interface to the application.

Figure 6 represents the above concept. An application accesses data through the DMIF-Application Interface, irrespective of whether such data comes from a broadcast source, from local storage or from a remote server. In all scenarios the Local Application only interacts through a uniform interface (DAI). Different DMIF instances will then translate the Local Application requests into specific messages to be delivered to the Remote Application, taking care of the peculiarities of the involved delivery technology. Similarly, data entering the terminal (from remote servers, broadcast networks or local files) is uniformly delivered to the Local Application through the DAI.

Different, specialized DMIF instances are indirectly invoked by the Application to manage the various specific delivery technologies, this is however transparent to the Application, that only interacts with a single “DMIF filter”. This filter is in charge of directing the particular DAI primitive to the right instance. DMIF does not specify this mechanism, just assumes it is implemented. This is further emphasized by the shaded boxes in the figure, whose aim is to clarify what are the borders of a DMIF implementation, while the DMIF communication architecture defines a number of modules, actual DMIF implementations only need to preserve their appearance at those borders.

Conceptually, a “real” remote application accessed through a network e.g., IP- or ATM-based, is no different than an emulated remote producer application getting content from a broadcast source or from a disk. In the former case, however, the messages exchanged between the two entities have to be normatively defined to ensure interoperability (these are the DMIF Signaling messages). In the latter case, on the other hand, the interfaces between the two DMIF

peers and the emulated Remote Application are internal to a single implementation and need not be considered in this specification. Note that for the broadcast and local storage scenarios, the figure shows a chain of “Local DMIF”, “Remote DMIF (emulated)” and “Remote Application (emulated)”. This chain only represents a conceptual model and need not be reflected in actual implementations (it is shown in the figure totally internal to a shaded box).

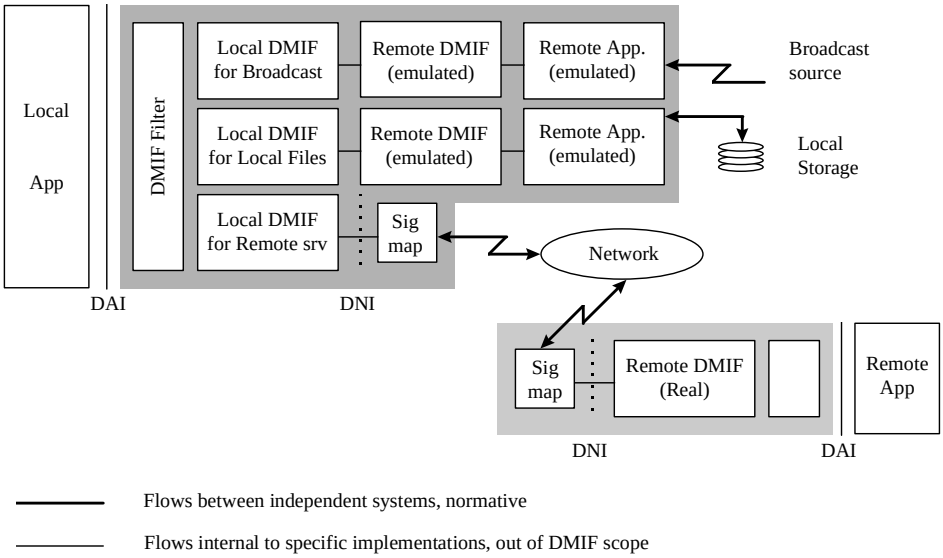


Figure 7 - DMIF communication architecture

When considering the Broadcast and Local Storage scenarios, it is assumed that the (emulated) Remote Application has knowledge on how the data is delivered/stored. This implies knowledge of the kind of application it is dealing with. In the case of MPEG-4, this actually means knowledge of concepts like Elementary Stream ID, First Object Descriptor, ServiceName. Thus, while the DMIF Layer is conceptually unaware of the application it is providing support to, in the particular case of DMIF instances for Broadcast and Local Storage this assumption is not completely true due to the presence of the (emulated) Remote Application (which, from the Local Application perspective, is still part of the DMIF Layer).

It is worth noting that since the (emulated) Remote Application has knowledge on how the data is delivered/stored, the specification of how data is delivered/stored is crucial for such a DMIF implementation, which is thus “MPEG-4 systems aware”.

When considering the Remote Interactive scenario instead, the DMIF Layer is totally application-unaware. An additional interface -the DMIF-Network Interface (DNI)- is introduced to emphasize what kind of information DMIF peers need to exchange; an additional module (”Signaling mapping” in the figure) takes care of mapping the DNI primitives into signaling messages used on the specific Network. Note that DNI primitives are only specified for information purposes, and a DNI interface need not be present in an actual implementation, Figure 7 also clearly represents the DNI as internal to the shaded box. Instead, the syntax of the messages flowing in the Network is fully specified for each specific network supported.

DMIF allows the concurrent presence of one or more DMIF instances, each one targeted for a particular delivery technology, in order to support in the same terminal multiple delivery technologies and even multiple scenarios (broadcast, local storage, remote interactive).

Multiple delivery technologies may be activated by the same application, that could therefore seamlessly manage data sent by broadcast networks, local file systems and remote interactive peers

10.2.1 The DMIF Computational Model

When an application requests the activation of a service, it uses the Service primitives of the DAI, and creates a service session; the DMIF implementation then contacts its corresponding peer (that conceptually can be either a remote peer, or a local emulated peer) and creates a network session with it. Network sessions have network-wide significance, service sessions have instead local meaning. The association between them is maintained by the DMIF Layer. In the case of Broadcast and Local Storage scenarios, the way the network session is created and then managed is out of the scope of this specification. In the case of a remote interactive scenario instead, DMIF uses the native signalling mechanism for that network to create and then manage the network session e.g., ATM signalling. The application peers then use this session to create connections which are used to transport application data e.g., MPEG-4 Elementary Streams.

When an application needs a Channel, it uses the Channel primitives of the DAI, DMIF translates these requests into connection requests which are specific to the particular network implementation. In the case of Broadcast and Local Storage scenarios, the way the connections are created and then managed is out of the scope of this specification. In the case of a networked scenario instead, DMIF uses the native signalling mechanism for that network to create those connections. The application then uses these connections to deliver the service.

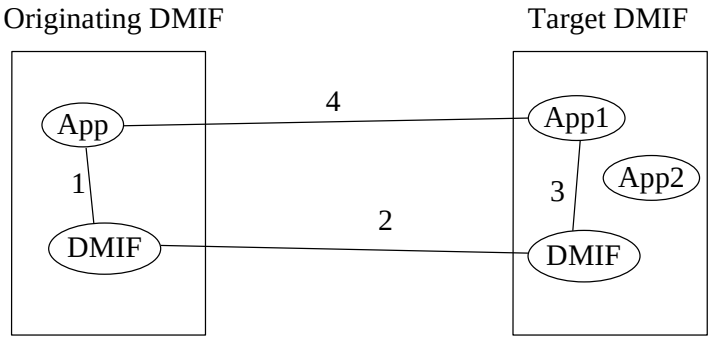


Figure 8 — DMIF Computational Model

Figure 8 provides a high level view of a service activation and of the beginning of data exchange; the high level walk-through consists of the following steps:

The Originating Application request the activation of a service to its local DMIF Layer -- a communication path between the Originating Application and its local DMIF peer is established in the control plane (1)

The Originating DMIF peer establishes a network session with the Target DMIF peer -- a communication path between the Originating DMIF peer and the Target DMIF Peer is established in the control plane (2)

The Target DMIF peer identifies the Target Application and forwards the service activation request -- a communication path between the Target DMIF peer and the Target Application is established in the control plane (3)

The peer Applications create channels (requests flowing through communication paths 1, 2 and 3). The resulting channels in the user plane (4) will carry the actual data exchanged by the Applications.

DMIF is involved in all four steps above.
The DMIF Layer automatically determines whether a particular service is supposed to be provided by a remote server on a particular network e.g., IP based, or ATM based, by a broadcast network, or resides in a local storage device: the selection is based on the peer address information provided by the Application as part of a URL passed to the DAI.

DMIF Features:

- Support for all kinds of networks, including mobile networks

  In conjunction with ITU-T, the H.245 specification has been extended (H.245v6) to include support to MPEG-4 Systems; the DMIF specification provides the appropriate walkthrough and mapping to H.245 signals. Mobile terminals can now use MPEG-4 Systems features such as BIFS and OD streams, although with some limitation (the MPEG-4 presentation is uniquely selected by the target peer)

- Qos Monitoring
  DMIF knows the concept of monitoring the Quality of Service actually delivered by a network. The DMIF-Application Interface has been extended accordingly. The model allows for three different modes of QoS monitoring: continuous monitoring, specific queries, and QoS violation notification.

- UserCommands with ACK
  The DMIF model allows peer applications to exchange user messages of any kind (included stream control messages). DMIF supports acknowledgment messages.

- Management of MPEG-4 Sync Layer information
  The DMIF model allows applications to exchange application-specific data with the DMIF layer. This addition was introduced to enable, within the model, the exchange of Sync Layer Protocol Data Units as a combination of pure media data (PDU) and logical Sync Layer information. The model acknowledges that within the existing transport stacks there are features that overlap with the MPEG-4 Systems Sync Layer. This is the case of RTP and MPEG-2 PES (Packetized Elementary Steams) as well as MP4 atoms in the file format: in all such cases the obvious implementation of a DMIF instance is to map the Sync Layer information extracted from those structures, as well as from a true SL-PDU, into a uniform logical representation of the Sync Layer Packet Header. As a consequence, the appropriate parameters have been introduced at the DAI, taking care, as usual, to make their semantic independent of both transport stack and application.

- DAI syntax in C language
  DMIF includes an informative annex that gives a C/C++ syntax for the DMIF Application Interface, as a recommended API syntax.

10.3 Demultiplexing, synchronization and description of streaming data

Individual Elementary Streams have to be retrieved on the delivery layer from incoming data from some network connection or a storage device. Each network connection or file is homogeneously considered a TransMux Channel in the MPEG-4 system model. The demultiplexing is partially or completely done by layers outside the scope of MPEG-4, depending on the application. The only multiplexing tool defined by MPEG-4 is the FlexMux tool that may optionally be used for low delay, low overhead multiplexing and for saving network connection resources.

For the purpose of integrating MPEG-4 in system environments, the DMIF Application Interface is the reference point at which elementary streams can be accessed as sync layer-packetized streams. The DMIF Network Interface specifies how either SL(Sync Layer)-packetized streams —no FlexMux used— or FlexMux Streams are to be retrieved from the TransMux Layer. This is the interface to the transport functionalities not defined by MPEG. The data part of the interfaces is considered here, while the control part is dealt with by DMIF.

In the same way that MPEG-1 and MPEG-2 describe the behavior of an idealized decoding device along with the bitstream syntax and semantics, MPEG-4 defines a System Decoder Model. This allows the precise definition of the terminal's operation without making unnecessary assumptions about implementation details. This is essential in order to give implementers the freedom to design real MPEG-4 terminals and decoding devices in a variety of ways. These devices range from television receivers, which have no ability to communicate with the sender, to computers that are fully enabled with bi-directional communication. Some devices will receive MPEG-4 streams over isochronous networks, while others will use non-isochronous means (e.g., the Internet) to exchange MPEG-4 information. The System Decoder Model provides a common model on which all implementations of MPEG-4 terminals can be based.

The specification of a buffer and timing model is essential to encoding devices which may not know ahead of time what the terminal device is or how it will receive the encoded stream. Though the MPEG-4 specification will enable the encoding device to inform the decoding device of resource requirements, it may not be possible, as indicated earlier, for that device to respond to the sender. It is also possible that an MPEG-4 session is received simultaneously by widely different devices; it will, however, be properly rendered according to the capability of each device.

10.3.1 Demultiplexing

Demultiplexing occurs on the delivery layer that is modeled as consisting of a TransMux layer and a DMIF layer. The retrieval of incoming data streams from network connections or storage media consists of two tasks. First, the channels must be located and opened. This requires a transport control entity that manages, among others, the tables that associate transport channels to specific elementary streams. Stream map tables link each stream to a ChannelAssociationTag that serves as a handle to the channel that carries this stream. Resolving ChannelAssociationTags to the actual transport channel as well as the management of the sessions and channels is addressed by the DMIF part of the MPEG-4 standard.

Second, the incoming streams must be properly demultiplexed to recover SL-packetized streams from downstream channels (incoming at the receiving terminal) to be passed on to the synchronization layer. In interactive applications, a corresponding multiplexing stage will multiplex upstream data in upstream channels (outgoing from the receiving terminal).

The generic term 'TransMux Layer' is used to abstract any underlying multiplex functionality – existing or future – that is suitable to transport MPEG-4 data streams. Note that this layer is not defined in the context of MPEG-4. Examples are MPEG-2 Transport Stream, H.223, ATM AAL 2, IP/UDP. The TransMux Layer is assumed to provide protection and multiplexing functionality, indicating that this layer is responsible for offering a specific QoS. Protection functionality includes error protection and error detection tools suitable for the given network or storage medium.

In any concrete application scenario one or more specific TransMux Instances will be used. Each TransMux demultiplexer gives access to TransMux Channels. The requirements on the data interface to access a TransMux Channel are the same for all TransMux Instances. They include the need for reliable error detection, delivery, if possible, of erroneous data with a suitable error indication and framing of the payload, which may consist of either SL-packetized streams or FlexMux streams. These requirements are summarized in an informative way in the TransMux Interface, in the Systems part of the MPEG-4 Standard. An adaptation of SL-packetized streams must be specified to each transport protocol stack of interest according to these requirements and in conjunction with the standardization body that has the proper jurisdiction. This is happening for RTP and mobile channels at the moment.

The FlexMux tool is specified by MPEG to optionally provide a flexible, low overhead, low delay method for interleaving data whenever this is not sufficiently supported by the underlying protocol stack. It is especially useful when the packet size or overhead of the underlying TransMux instance is large, so that a waste of bandwidth or number of network connections would result otherwise. The FlexMux tool is not itself robust to errors and can either be used on TransMux Channels with a high QoS or to bundle Elementary Streams that are equally error tolerant. The FlexMux requires reliable error detection and sufficient framing of FlexMux packets (for random access and error recovery) from the underlying layer. These requirements are also reflected in the data primitives of the DMIF Application Interface, which defines the data access to individual transport channels. The FlexMux demultiplexer retrieves SL-packetized streams from FlexMux Streams.

10.3.2 Synchronization and description of elementary streams

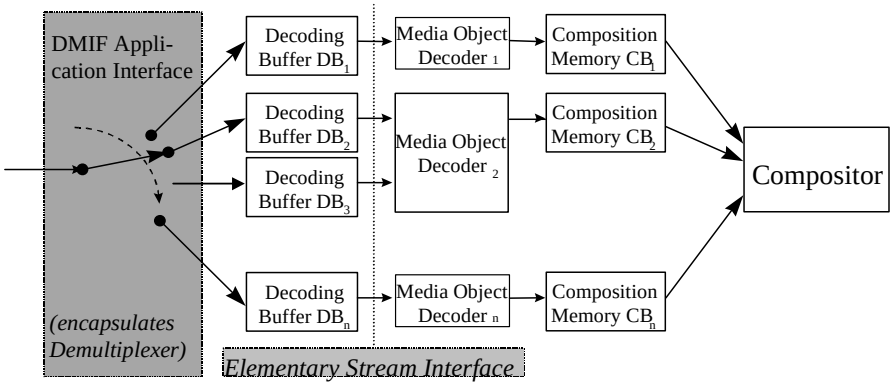


Figure 9 - Buffer architecture of the System Decoder Model

The sync layer has a minimum set of tools for consistency checking, padding, to convey time base information and to carry time stamped access units of an elementary stream. Each packet consists of one access unit or a fragment of an access unit. These time stamped access units

form the only semantic structure of elementary streams that is visible on this layer. Time stamps are used to convey the nominal decoding and composition time for an access unit. The sync layer requires reliable error detection and framing of each individual packet from the underlying layer, which can be accomplished, e.g., by using the FlexMux. How data can be accessed by the compression layer is summarized in the informative Elementary Stream Interface, which can be found in the Systems part of the MPEG-4 Standard. The sync layer retrieves elementary streams from SL-packetized streams.

To be able to relate elementary streams to media objects within a scene, object descriptors are used. Object Descriptors convey information about the number and properties of elementary streams that are associated to particular media objects. Object descriptors are themselves conveyed in one or more elementary streams, since it is possible to add and discard streams (and objects) during the course of an MPEG-4 session. Such updates are time stamped in order to guarantee synchronization. The object descriptor streams can be considered as a description of the streaming resources for a presentation. Similarly, the scene description is also conveyed as an elementary stream, allowing to modify the spatio-temporal layout of the presentation over time.

10.3.3 Buffer Management

To predict how the decoder will behave when it decodes the various elementary data streams that form an MPEG-4 session, the Systems Decoder Model enables the encoder to specify and monitor the minimum buffer resources that are needed to decode a session. The required buffer resources are conveyed to the decoder within object descriptors during the setup of the MPEG-4 session, so that the decoder can decide whether it is capable of handling this session.

By managing the finite amount of buffer space the model allows a sender, for example, to transfer non real-time data ahead of time, if sufficient space is available at the receiver side to store it. The pre-stored data can then be accessed when needed, allowing at that time real-time information to use a larger amount of the channel's capacity if so desired.

10.3.4 Time Identification

For real-time operation, a timing model is assumed in which the end-to-end delay from the signal output from an encoder to the signal input to a decoder is constant. Furthermore, the transmitted data streams must contain implicit or explicit timing information. There are two types of timing information. The first is used to convey the speed of the encoder clock, or time base, to the decoder. The second, consisting of time stamps attached to portions of the encoded AV data, contains the desired decoding time for access units or composition and expiration time for composition units. This information is conveyed in SL-packet headers generated in the sync layer. With this timing information, the inter-picture interval and audio sample rate can be adjusted at the decoder to match the encoder's inter-picture interval and audio sample rate for synchronized operation.

Different media objects may have been encoded by encoders with different time bases, with the accompanying slightly different speed. It is always possible to map these time bases to the time base of the receiving terminal. In this case, however, no real implementation of a receiving terminal can avoid the occasional repetition or drop of AV data, due to temporal aliasing (the relative reduction or extension of their time scale).

Although systems operation without any timing information is allowed, defining a buffering model is not possible for this case.

10.4 Advanced Synchronization (FlexTime) Model

The FlexTime model (Advanced Synchronization Model) augments the traditional MPEG-4 timing model
to permit synchronization of multiple streams and objects, such as video, audio, text, graphics, or even programs, that may originate from multiple sources.

The traditional MPEG-4 timing model has been designed primarily for “push” broadcast applications where temporal synchronization among access units is achieved via “hard” timestamps and reference clocks. While this mechanism provides accurate intra-stream synchronization, it falls short of providing inter-stream synchronization for streams coming from different sources (and possibly with different reference clocks) as is the case in most Internet applications and in emerging more sophisticated broadcast applications.

The FlexTime model allows the content author to specify simple temporal relationships among selected MPEG-4 objects, such as "CoStart," "CoEnd," and "Meet." The content author can also specify flexibility constraints for MPEG-4 objects, as if the objects were on stretchable springs. This allows synchronization among multiple objects according to the specified temporal relationships, while stretching or shrinking them if necessary within specified bounds.

The most immediate and beneficial impact of this new functionality will be on emerging Internet applications and enhanced broadcast applications where multiple sources need to be synchronized at the consumer device.

10.4.1 Flexible duration

In an environment of unreliable delivery it may very well happen that the delivery of a certain Elementary Stream, or portions of the stream, can be delayed beyond its required playback start time.

To be less sensitive to stream delivery delay, the FlexTime model is based upon a so-called "spring" metaphor. A spring comes with a set of 3 constants: the minimum length beyond which it won’t shrink, the maximum length beyond which it will break, and the optimal length at which it may rest comfortably.

Following this spring model, Elementary Streams, or stream segments, are viewed temporally as springs, each with the corresponding 3 spring constants. The optimal spring length (the stream playback duration) can be viewed as a hint to aid a receiver to choose a particular duration when more than one value is possible. Note, that while stretching or shrinking the duration of a continuous media such as video implies respectively slowing down or speeding up playback, when an Elementary Stream consists of a still image, shrinking or stretching is merely holding the display shorter or longer.

10.4.2 Relative start and end time

Two or more Elementary Streams or stream segments can be synchronized with respect to one another, by defining that they either start at the same time (“CoStart”), end at the same time (“CoEnd”), or the end time of one coincides with the start time of another (“Meet”).

It is important to note that there are two classes of MPEG-4 objects. The timing and rendering of an MPEG-4 object that uses an Elementary stream, such as video, is not determined by the stream alone, but also by the corresponding BIFS nodes and their timing. Whereas the timing and rending of an MPEG-4 object that does not use a stream, such as text or rectangle, is determined only by the corresponding BIFS nodes and their timing.

The FlexTime model allows the content author to express synchronization among MPEG-4 objects with streams or stream segments, by assigning temporal relationships among them.

The temporal relationships (or relative timestamps) can be seen as "functional" timestamps, and their functional timestamps are resolved at playback time. Thus, a FlexTime player can:

- Compensate for various network delays by supporting a timed wait for the initial arrival of the stream, before the player starts rendering/playing the node associated with it.
- Compensate for various network jitters by supporting a timed wait for the arrival of the stream segment.
- Synchronize multiple media/BIFS nodes with some of the media stream of unknown length or uncontrolled arrival time.
- Synchronize BIFS updates (e.g. events such as field updates) among multiple nodes/streams with some of the streams of unknown length or uncontrolled arrival time.
- Slow down or speed up the rendering/playback speed of portions of streams to re-adjust out-of-sync situations caused by unknown length, uncontrolled arrival time or variation.

10.4.3 The FlexTime support in MPEG-4

The FlexTime model is supported in MPEG-4 by two nodes : TemporalTransform and TemporalGroup nodes, and a descriptor : SegmentDescriptor. The TemporalTransform node specifies temporal properties of an MPEG-4 object that needs to be synchronized (or flexed). The TemporalGroup node specifies temporal relationships among the objects that are represented by the TemporalTransform nodes, and the SegmentDescriptor identifies portions of a stream that can be synchronized.

1.1.1.1 The TemporalTransform node

The **TemporalTransform** supports synchronization of nodes within the scene to a media stream, or segment thereof, and supports flexible transformation to scene time. This grouping node can flexibly support the slowing down, speeding up, freezing or shifting of the scene time for rendering of nodes contained within. Its **children** field may contain a list of nodes of the type SF3Dnode, and the node can effect the slowing down, speeding up, freezing or shifting the time base of the compositor when it renders the child nodes that are transformed by this

node. In addition, this node has a **url** field that may reference an elementary stream or a segment thereof and in this case, the node affects the time base of the referenced stream.

a) The TemporalGroup node

The **TemporalGroup** node specifies the temporal relationship between a given number of **TemporalTransforms** to align in time both nodes, and media streams with nodes, in the scene graph. Temporal adjustment of media to meet the constraint and flexibility being done in the sync layer. **TemporalGroup** can examine the temporal properties of its children and when all the children are ready and the temporal constraint met can allow its children to play.

b) The SegmentDescriptor

An array of **SegmentDescriptor**s are added as an item in the ES_Descriptor. A **SegmentDescriptor** identifies and labels segments of a stream, so that the specific stream segments can be referenced by their url fields in the **TemporalTansform** node.

10.4.4 The Execution Model

TemporaI decoding and clock adjustment of media streams according to timestamps is a function of the sync layer. The FlexTime model requires a small change to the MPEG-4 buffer model in terms of media delivery and decoding. Decoding may be delayed on the client, beyond the standard decoding time, by an amount determined by the flexibility expressed in the relationships.

The buffer model for flextime can thus be specified as follows: “At any time from the instant of time corresponding to its DTS up to a time limit specified by Flextime, and AU is instantaneously decoded and removed from the decoding buffer.” As such the exact time of removal of the AU from the decoding buffer can vary and should not be assumed to be removed ahead of the worst case time (maximum delay for the media stream). By using the worst case time rather than the fixed DTS the decoding buffer can be managed otherwise as prescribed by MPEG-4.

To support media stream flexing the OD/ESD delete command execution shall be delayed by the same amount (or less) that the media stream it refers to has been delayed. If the OD/ESD delete is received after the object has ended then the command can be executed immediately.

10.5 Syntax Description

MPEG-4 defines a *syntactic description language* to describe the exact binary syntax for bitstreams carrying media objects and for bitstreams with scene description information. This is a departure from MPEG’s past approach of utilizing pseudo C. This language is an extension of C++, and is used to describe the syntactic representation of objects and the overall media object class definitions and scene description information in an integrated way. This provides a consistent and uniform way of describing the syntax in a very precise form, while at the same time simplifying bitstream compliance testing. Software tools can be used to process the syntactic description and generate the necessary code for programs that perform validation.

10.6 Binary Format for Scene description: BIFS

In addition to providing support for coding individual objects, MPEG-4 also provides facilities to compose a set of such objects into a scene. The necessary composition information forms the scene description, which is coded and transmitted together with the media objects. Starting from VRML (the Virtual reality Modeling Language), MPEG has developed a binary language for scene description called BIFS. BIFS stands for **BI**nary **F**ormat for **S**cenes.

In order to facilitate the development of authoring, manipulation and interaction tools, scene descriptions are coded independently from streams related to primitive media objects. Special care is devoted to the identification of the parameters belonging to the scene description. This is done by differentiating parameters that are used to improve the coding efficiency of an object (e.g., motion vectors in video coding algorithms), and the ones that are used as modifiers of an object (e.g., the position of the object in the scene). Since MPEG-4 should allow the modification of this latter set of parameters without having to decode the primitive media objects themselves, these parameters are placed in the scene description and not in primitive media objects.
The following list gives some examples of the information described in a scene description.

How objects are grouped together: An MPEG-4 scene follows a hierarchical structure, which can be represented as a directed acyclic graph. Each node of the graph is a media object, as illustrated in Figure 10 (note that this tree refers back to Figure 1). The tree structure is not necessarily static; node attributes (e.g., positioning parameters) can be changed while nodes can be added, replaced, or removed.

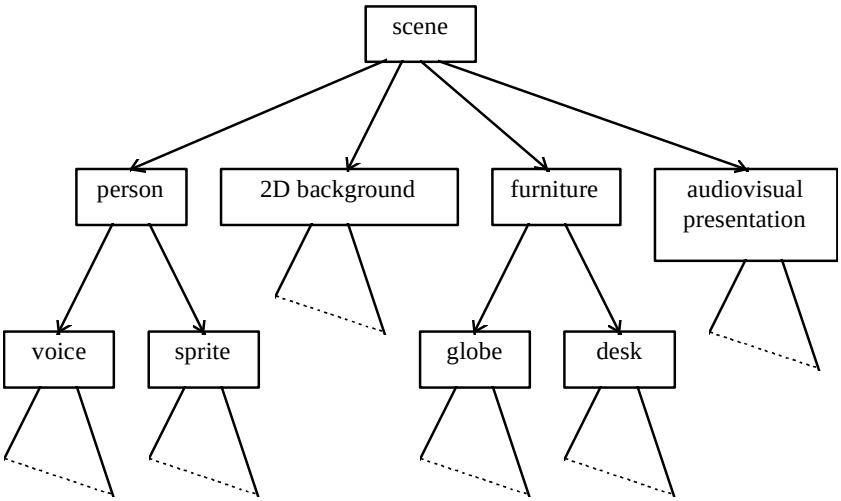


Figure 10- Logical structure of a scene

How objects are positioned in space and time: In the MPEG-4 model, audiovisual objects have both a spatial and a temporal extent. Each media object has a local coordinate system. A local coordinate system for an object is one in which the object has a fixed spatio-temporal location and scale. The local coordinate system serves as a handle for manipulating the media object in space and time. Media objects are positioned in a scene by specifying a coordinate transformation from the object's local coordinate system into a global coordinate system defined by one more parent scene description nodes in the tree.

Attribute Value Selection: Individual media objects and scene description nodes expose a set of parameters to the composition layer through which part of their behavior can be controlled. Examples include the pitch of a sound, the color for a synthetic object, activation or deactivation of enhancement information for scaleable coding, etc.

Other transforms on media objects: As mentioned above, the scene description structure and node semantics are heavily influenced by VRML, including its event model. This provides MPEG-4 with a very rich set of scene construction operators, including graphics primitives that can be used to construct sophisticated scenes.

10.6.1 Advanced BIFS

The version 2 BIFS (Advanced BIFS) includes the following new functionalities:

- *Advanced sound environment modeling* in interactive virtual scenes, where properties such as room reflections, reverberation, Doppler effect, and sound obstruction caused by objects appearing between the source and the listener are computed for sound sources in a dynamic environment in real time. Also enhanced source directivity modeling is made possible enabling inclusion of realistic sound sources into 3-D scenes.
- Body animation of either a default body model present at the decoder or of a downloadable body model. The animation of the body is performed by sending animation parameters to it in a bitstream. (Also See Section 11.5.2)
- Chroma keying which is used to generate a shape mask and a transparency value for an image or a video sequence.
- Inclusion of hierarchical 3-D meshes to BIFS scenes.
- Associating interactive commands to media nodes. The commands are passed to server over a back channel for specified processing.
- PROTOs and EXTERNPROTOs

10.6.2 Textual Format

The Extensible MPEG-4 Textual format (XMT) is a framework for representing MPEG-4 scene description using a textual syntax. The XMT allows the content authors to exchange their content with other authors, tools or service providers, and facilitates interoperability with both the Extensible 3D (X3D) being developed by the Web3D Consortium, and the Synchronized Multimedia Integration Language (SMIL) from the W3C consortium.

The XMT format can be interchanged between SMIL players, VRML players, and MPEG-4 players. The format can be parsed and played directly by a W3C SMIL player, preprocessed to Web3D X3D and played back by a VRML player, or compiled to an MPEG-4 representation such as mp4, which can then be played by an MPEG-4 player. See below for a graphical description of interoperability of the XMT. It encompasses MPEG-4, a large part of SMIL, Scalable Vector Graphics, X3D and also gives a textual representation for MPEG-7 Descriptions (see http://mpeg.telecomitalialab.com for documentation on the MPEG-7 Content Description Standard)

The XMT framework consists of two levels of textual syntax and semantics: the XMT-A format and the XMT-Ω format.
The XMT-A is an XML-based version of MPEG-4 content, which contains a subset of the X3D. Also contained in XMT-A is an MPEG-4 extension to the X3D to represent MPEG-4

specific features. The XMT-A provides a straightforward, one-to-one mapping between the textual and binary formats.

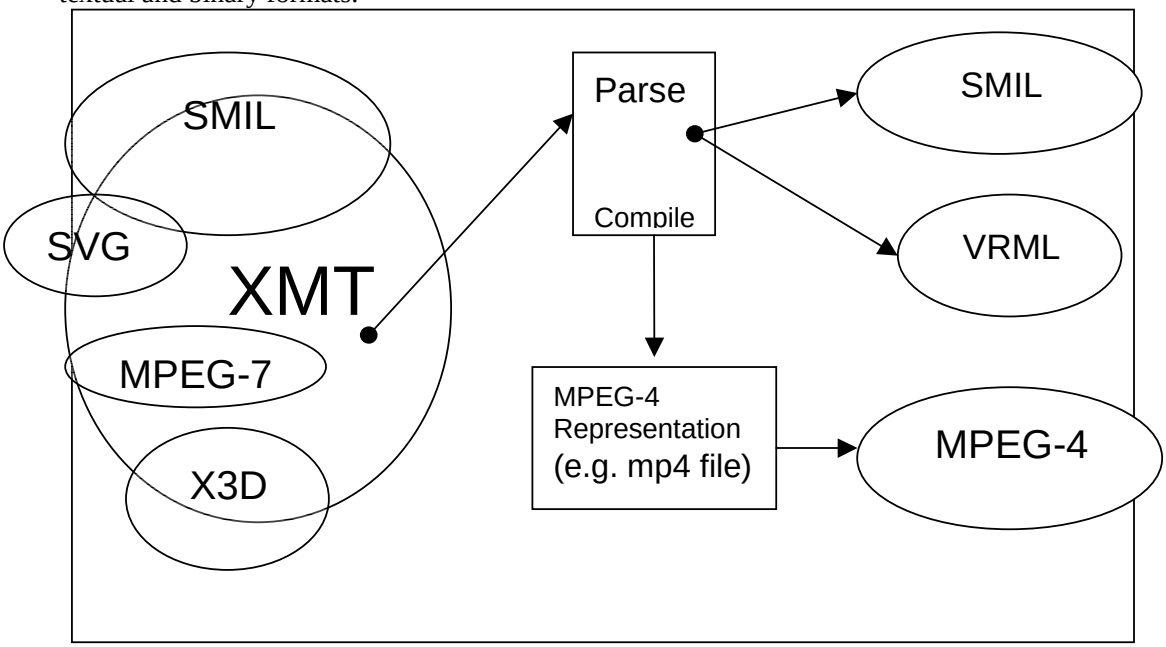


Figure 11 - the eXtensible MPEG-4 Textual format

The XMT-Ω is a high-level abstraction of MPEG-4 features based on the W3C SMIL. The XMT provides a default mapping from Ω to A, for there is no deterministic mapping between the two, and it also provides content authors with an escape mechanism from Ω to A.

10.7 User interaction

MPEG-4 allows for user interaction with the presented content. This interaction can be separated into two major categories: client-side interaction and server-side interaction. Client-side interaction involves content manipulation, which is handled locally at the end-user's terminal, and can take several forms. In particular, the modification of an attribute of a scene description node, e.g., changing the position of an object, making it visible or invisible, changing the font size of a synthetic text node, etc., can be implemented by translating user events. A user event can be a mouse clicks or keyboard command) to scene description updates. The MPEG-4 terminal can process the commands in exactly the same way as if they originated from the original content source. As a result, this type of interaction does not require standardization.

Other forms of client-side interaction require support from the scene description syntax, and are specified by the standard. The use of the VRML event structure provides a rich model on which content developers can create compelling interactive content.

Server-side interaction involves content manipulation that occurs at the transmitting end, initiated by a user action. This, of course, requires that a back-channel is available.

10.8 Content-related IPR identification and protection

MPEG-4 provides mechanisms for protection of intellectual property rights (IPR), as outlined in section 1.5. This is achieved by supplementing the coded media objects with an optional Intellectual Property Identification (IPI) data set, carrying information about the contents, type of content and (pointers to) rights holders. The data set, if present, is part of an elementary stream descriptor that describes the streaming data associated to a media object. The number of data sets to be associated with each media object is flexible; different media objects can share the same data sets or have separate data sets. The provision of the data sets allows the implementation of mechanisms for audit trail, monitoring, billing, and copy protection.

Next to identifying rights, each of the wide range of MPEG-4 applications has a set of requirements regarding protection of the information it manages. These applications can have different security requirements. For some applications, users exchange information that has no intrinsic value but that must still be protected to preserve various rights of privacy. For other applications, the managed information has great value to its creator and/or distributors requiring high-grade management and protection mechanisms. The implication is that the design of the IPMP framework must consider the complexity of the MPEG-4 standard and the diversity of its applications. This IPMP framework leaves the details of IPMP systems designs in the hands of applications developers. The level and type of management and protection required depends on the content's value, complexity, and the sophistication of the associated business models.

The approach taken allows the design and use of domain-specific IPMP systems (IPMP-S). While MPEG-4 does not standardize IPMP systems themselves, it does standardize the MPEG-4 IPMP interface. This interface consists of IPMP-Descriptors (IPMP-Ds) and IPMP-Elementary Streams (IPMP-ES).

IPMP-Ds and IPMP-ESs provide a communication mechanism between IPMP systems and the MPEG-4 terminal. Certain applications may require multiple IPMP systems. When MPEG-4 objects require management and protection, they have IPMP-Ds associated with them. These IPMP-Ds indicate which IPMP systems are to be used and provide information to these systems about how to manage and protect the content. (See Figure 12)

Besides enabling owners of intellectual property to manage and protect their assets, MPEG-4 provides a mechanism to identify those assets via the Intellectual Property Identification Data Set (IPI Data Set). This information can be used by IPMP systems as input to the management and protection process.

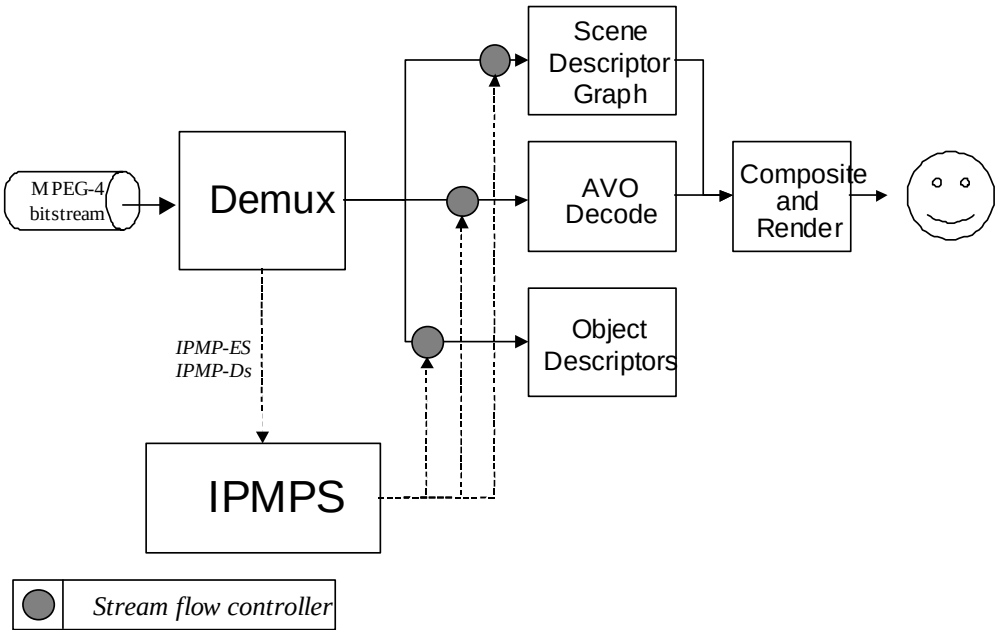


Figure 12 - The IPMP Interfaces within an MPEG-4 System

10.9 MPEG-4 File Format

The MP4 file format is designed to contain the media information of an MPEG-4 presentation in a flexible, extensible format which facilitates interchange, management, editing, and presentation of the media. This presentation may be 'local' to the system containing the presentation, or may be via a network or other stream delivery mechanism (a TransMux). The file format is designed to be independent of any particular delivery protocol while enabling efficient support for delivery in general. The design is based on the QuickTime® format from Apple Computer Inc.

The following diagram gives an example of a simple interchange file, containing three streams.

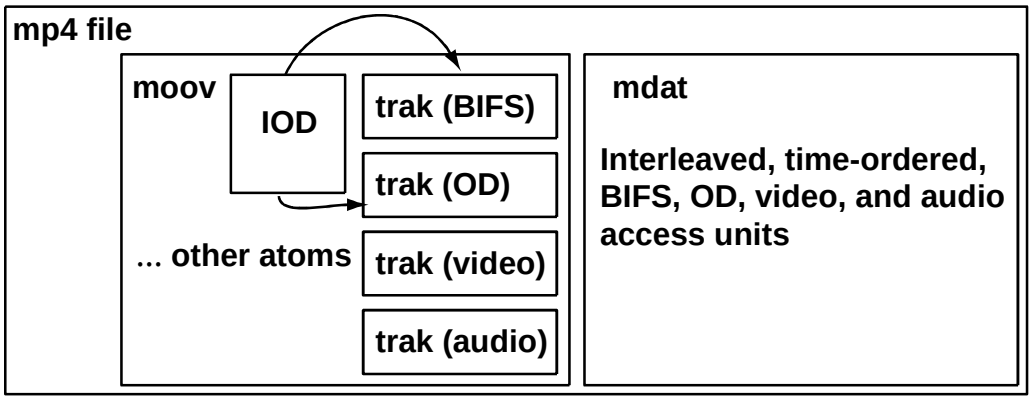


Figure 13 - Example of a simple interchange file

The MP4 file format is composed of object-oriented structures called ‘atoms’. A unique tag and a length identify each atom. Most atoms describe a hierarchy of metadata giving information such as index points, durations, and pointers to the media data. This collection of atoms is contained in an atom called the ‘movie atom’. The media data itself is located elsewhere; it can be in the MP4 file, contained in one or more ‘mdat’ or media data atoms, or located outside the MP4 file and referenced via URL’s.

The following diagram shows a more complex file with external media data:

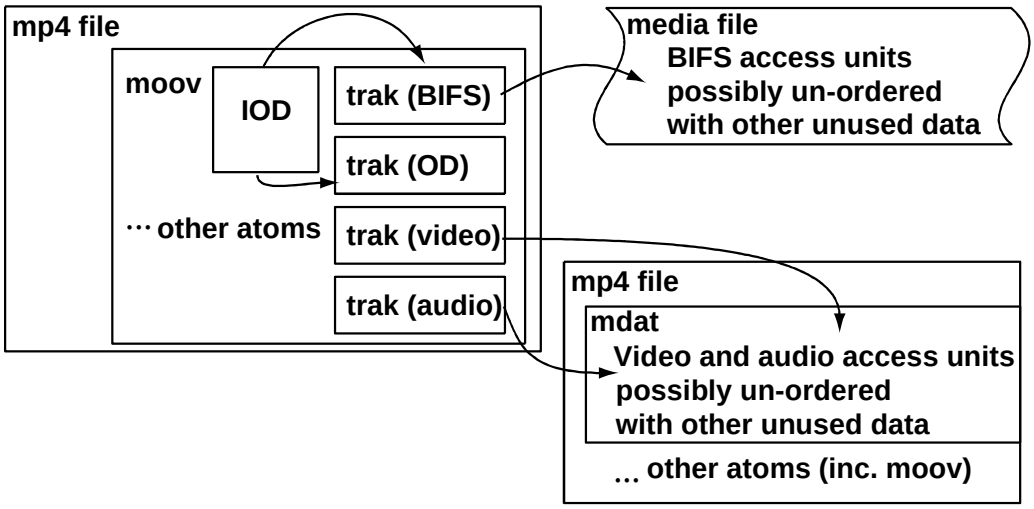


Figure 14 -- Complex file with external media data

The following diagram shows the container relationship between the different objects or atoms:

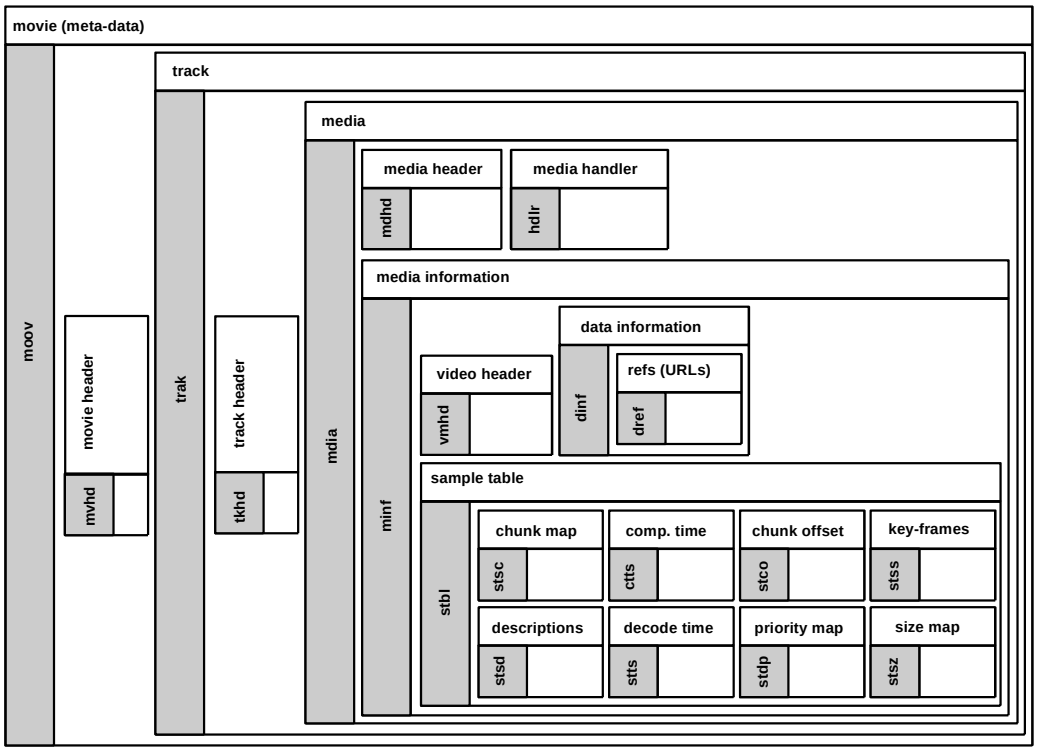


Figure 15 – Relation between different objects (or atoms)

The file format is a streamable format, as opposed to a streaming format. That is, the file format does not define an on-the-wire protocol, and is never actually streamed over a transmission medium. Instead, metadata in the file known as ‘hint tracks’ provide instructions, telling a server application how to deliver the media data over a particular delivery protocol. There can be multiple hint tracks for one presentation, describing how to deliver over various delivery protocols. In this way, the file format facilitates streaming without ever being streamed directly

The following diagram shows the container relationship with RTP protocol hint tracks to stream a simple video movie:

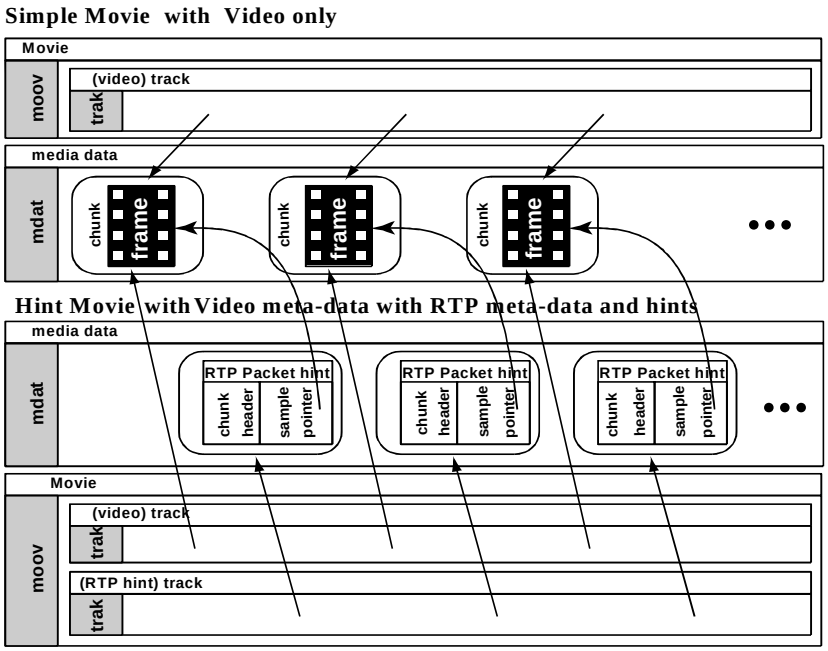


Figure 16 - Relationship with RTP protocol hint tracks to stream a simple video movie

The metadata in the file, combined with the flexible storage of media data, allows the MP4 format to support streaming, editing, local playback, and interchange of content, thereby satisfying the requirements for the MPEG4 Intermedia format.

10.10 MPEG-J

The MPEG-J is a programmatic system (as opposed to the parametric system offered by MPEG-4 Version 1) which specifies API for interoperation of MPEG-4 media players with Java code. By combining MPEG-4 media and safe executable code, content creators may embed complex control and data processing mechanisms with their media data to intelligently manage the operation of the audio-visual session. A block diagram of the MPEG-J player in an MPEG-4 system player environment is shown in Figure 17. The lower half of this drawing depicts the parametric MPEG-4 System player also referred to as the Presentation Engine. The MPEG-J subsystem controlling the Presentation Engine, also referred to as the Application Engine, is depicted in the upper half of Figure 17.

The Java application is delivered as a separate elementary stream to the MPEG-4 terminal. There it will be directed to the MPEG-J run time environment, from where the MPEG-J program will have access to the various components and data of the MPEG-4 player, in addition to .the basic packages of the language (java.lang, java.io, java.util). MPEG-J specifically does *not* support downloadable decoders.

For the above-mentioned reason, the group has defined a set of APIs with different scopes. For Scene graph API the objective is to provide access to the scene graph: to inspect the graph, to alter nodes and their fields, and to add and remove nodes within the graph. The Resource Manager API is used for regulation of performance: it provides a centralized facility for managing resources. The Terminal Capability API is used when program execution is contingent upon the terminal configuration and its capabilities, both static (that do not change during execution) and dynamic. Media Decoders API allow the control of the decoders that are present in the terminal. The Network API provides a way to interact with the network, being compliant to the MPEG-4 DMIF Application Interface. Complex applications and enhanced interactivity are possible with these basic packages.

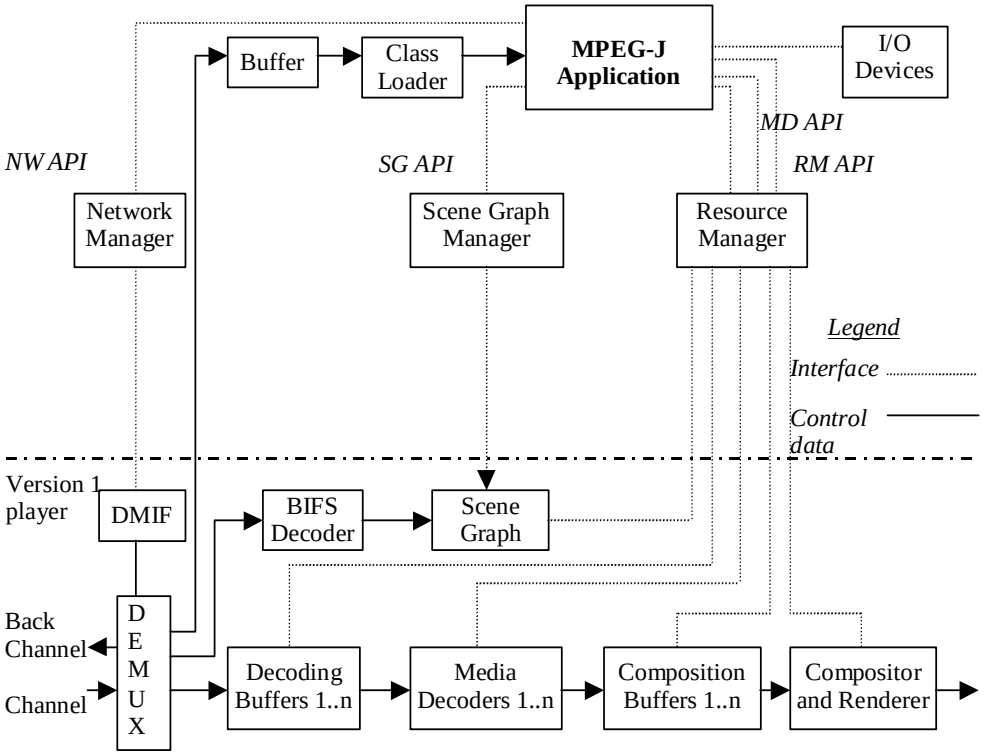


Figure 17 - Location of interfaces in the architecture of an MPEG-J enabled MPEG-4 System

10.11 Object Content Information

MPEG-4 allows attaching information to objects about their content. Users of the standard can use this 'OCI' (Object Content Information) data stream to send textual information along with MPEG-4 content. It is also possible to classify content according to pre-defined tables, which will be defined outside of MPEG. Further possibilities are giving unique labels to content, and storing camera parameters.

11 Detailed technical description of MPEG-4 Visual

Visual objects can be either of natural or of synthetic origin. First, the objects of natural origin are described. Then follow the synthetic ones.

11.1 Natural Textures, Images and Video

The tools for representing natural video in the MPEG-4 visual standard provide standardized core technologies allowing efficient storage, transmission and manipulation of textures, images and video data for multimedia environments. These tools allow the decoding and representation of atomic units of image and video content, called “video objects” (VOs). An example of a VO could be a talking person (without background), which can then be composed with other AVOs (audio-visual objects) to create a scene. Conventional rectangular imagery is handled as a special case of such objects.

In order to achieve this broad goal rather than a solution for a narrow set of applications, functionalities common to several applications are clustered. Therefore, the visual part of the MPEG-4 standard provides solutions in the form of tools and algorithms for:

- Efficient compression of images and video
- Efficient compression of textures for texture mapping on 2-D and 3-D meshes
- Efficient compression of implicit 2-D meshes
- Efficient compression of time-varying geometry streams that animate meshes
- Efficient random access to all types of visual objects
- Extended manipulation functionality for images and video sequences
- Content-based coding of images and video
- Content-based scalability of textures, images and video
- Spatial, temporal and quality scalability
- Error robustness and resilience in error prone environments

As mentioned before, MPEG-4 Video supports conventional rectangular images and video as well as images and video of arbitrary shape. This concept is illustrated in Figure 18 below.

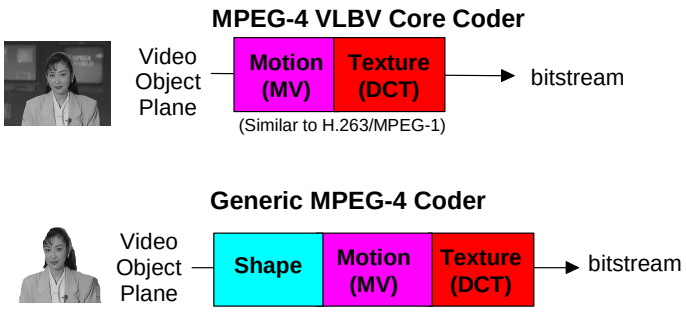


Figure 18 - the VLBV Core and the Generic MPEG-4 Coder

The coding of conventional images and video is similar to conventional MPEG-1/2 coding. It involves motion prediction/compensation followed by texture coding. For the content-based functionalities, where the image sequence input may be of arbitrary shape and location, this approach is extended by also coding shape and transparency information. Shape may be either

represented by an 8 bit transparency component - which allows the description of transparency if one VO is composed with other objects - or by a binary mask.

The extended MPEG-4 content-based approach can be seen as a logical extension of the conventional MPEG-4 VLBV Core or high bit-rate tools towards input of arbitrary shape.

a) Scalable Coding of Video Objects

MPEG-4 supports the coding of images and video objects with spatial and temporal scalability, both with conventional rectangular as well as with arbitrary shape. Scalability refers to the ability to only decode a part of a bitstream and reconstruct images or image sequences with:

- reduced decoder complexity and thus reduced quality
- reduced spatial resolution
- reduced temporal resolution
- with equal temporal and spatial resolution but with reduced quality.

This functionality is desired for progressive coding of images and video sent over heterogeneous networks, as well as for applications where the receiver is not capable of displaying the full resolution or full quality images or video sequences. This could for instance happen when processing power or display resolution is limited.

There are several scalable coding schemes in MPEG-4 Visual: spatial scalability, temporal scalability, fine granularity scalability and object-based spatial scalability. Spatial scalability supports changing the spatial resolution. Object-based spatial scalability extends the 'conventional' types of scalability towards arbitrary shape objects, so that it can be used in conjunction with other object-based capabilities. Thus, a very flexible content-based scaling of video information can be achieved. This makes it possible to enhance SNR, spatial resolution, shape accuracy, etc, only for objects of interest or for a particular region, which can be done dynamically at play-time. Fine granularity scalability (FGS) was developed in response to the growing need on a video coding standard for streaming video over the Internet. FGS and its combination with temporal scalability addresses a variety of challenging problems in delivering video over the Internet. FGS allows the content creator to code a video sequence once and to be delivered through channels with a wide range of bitrates. It provides the best user experience under varying channel conditions. It overcomes the "digital cutoff" problem associated with digital video. In other words, it makes compressed digital video behave similarly to analog video in terms of robustness while maintaining all the advantages of digital video.

b) Robustness in Error Prone Environments

MPEG-4 provides error robustness and resilience to allow accessing image or video information over a wide range of storage and transmission media. In particular, due to the rapid growth of mobile communications, it is extremely important that access is available to audio and video information via wireless networks. This implies a need for useful operation of audio and video compression algorithms in error-prone environments at low bit-rates (i.e., less than 64 kbit/s).

The error resilience tools developed for MPEG-4 can be divided into three major areas: resynchronization, data recovery, and error concealment. It should be noted that these categories are not unique to MPEG-4, but instead have been used by many researchers working in the area error resilience for video. It is, however, the tools contained in these categories that are of interest, and where MPEG-4 makes its contribution to the problem of error resilience.

Resynchronization

Resynchronization tools attempt to enable resynchronization between the decoder and the bitstream after a residual error or errors have been detected. Generally, the data between the synchronization point prior to the error and the first point where synchronization is

reestablished, is discarded. If the resynchronization approach is effective at localizing the amount of data discarded by the decoder, then the ability of other types of tools that recover data and/or conceal the effects of errors is greatly enhanced.

The resynchronization approach adopted by MPEG-4, referred to as a packet approach, is similar to the Group of Blocks (GOBs) structure utilized by the ITU-T standards of H.261 and H.263. In these standards a GOB is defined as one or more rows of macroblocks (MBs). At the start of a new GOB, information called a GOB header is placed within the bitstream. This header information contains a GOB start code, which is different from a picture start code, and allows the decoder to locate this GOB. Furthermore, the GOB header contains information which allows the decoding process to be restarted (i.e., resynchronize the decoder to the bitstream and reset all predictively coded data).
The GOB approach to resynchronization is based on spatial resynchronization. That is, once a particular macroblock location is reached in the encoding process, a resynchronization marker is inserted into the bitstream. A potential problem with this approach is that since the encoding process is variable rate, these resynchronization markers will most likely be unevenly spaced throughout the bitstream. Therefore, certain portions of the scene, such as high motion areas, will be more susceptible to errors, which will also be more difficult to conceal.

The video packet approach adopted by MPEG-4 is based on providing periodic resynchronization markers throughout the bitstream. In other words, the length of the video packets are not based on the number of macroblocks, but instead on the number of bits contained in that packet. If the number of bits contained in the current video packet exceeds a predetermined threshold, then a new video packet is created at the start of the next macroblock.

A resynchronization marker is used to distinguish the start of a new video packet. This marker is distinguishable from all possible VLC codewords as well as the VOP start code. Header information is also provided at the start of a video packet. Contained in this header is the information necessary to restart the decoding process and includes: the macroblock number of the first macroblock contained in this packet and the quantization parameter necessary to decode that first macroblock. The macroblock number provides the necessary spatial resynchronization while the quantization parameter allows the differential decoding process to be resynchronized.

Also included in the video packet header is the header extension code. The HEC is a single bit that, when enabled, indicates the presence of additional resynchronization information; including modular time base, VOP temporal increment, VOP prediction type, and VOP F code. This additional information is made available in case the VOP header has been corrupted.

It should be noted that when utilizing the error resilience tools within MPEG-4, some of the compression efficiency tools are modified. For example, all predictively encoded information must be confined within a video packet so as to prevent the propagation of errors.

In conjunction with the video packet approach to resynchronization, a second method called fixed interval synchronization has also been adopted by MPEG-4. This method requires that VOP start codes and resynchronization markers (i.e., the start of a video packet) appear only at legal fixed interval locations in the bitstream. This helps avoiding the problems associated with start codes emulations. That is, when errors are present in a bitstream it is possible for these errors to emulate a VOP start code. In this case, when fixed interval synchronization is utilized the decoder is only required to search for a VOP start code at the beginning of each fixed

interval. The fixed interval synchronization method extends this approach to be any predetermined interval.

Data Recovery

After synchronization has been reestablished, data recovery tools attempt to recover data that in general would be lost. These tools are not simply error correcting codes, but instead techniques that encode the data in an error resilient manner. For instance, one particular tool that has been endorsed by the Video Group is Reversible Variable Length Codes (RVLC). In this approach, the variable length codewords are designed such that they can be read both in the forward as well as the reverse direction.

An example illustrating the use of a RVLC is given in Figure 19. Generally, in a situation such as this, where a burst of errors has corrupted a portion of the data, all data between the two synchronization points would be lost. However, as shown in the Figure, an RVLC enables some of that data to be recovered. It should be noted that the parameters, QP and HEC shown in the Figure, represent the fields reserved in the video packet header for the quantization parameter and the header extension code, respectively.

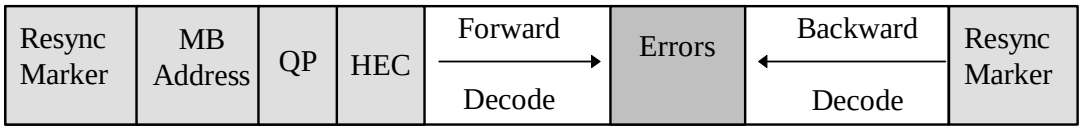


Figure 19 - example of Reversible Variable Length Code

Error Concealment

Error concealment is an extremely important component of any error robust video codec. Similar to the error resilience tools discussed above, the effectiveness of an error concealment strategy is highly dependent on the performance of the resynchronization scheme. Basically, if the resynchronization method can effectively localize the error then the error concealment problem becomes much more tractable. For low bitrate applications, low delay applications the current resynchronization scheme provides very acceptable results with a simple concealment strategy, such as copying blocks from the previous frame.

In recognizing the need to provide enhanced concealment capabilities, the Video Group has developed an additional error resilient mode that further improves the ability of the decoder to localize an error.

Specifically, this approach utilizes data partitioning by separating the motion and the texture. This approach requires that a second resynchronization marker be inserted between motion and texture information. If the texture information is lost, this approach utilizes the motion information to conceal these errors. That is, due to the errors the texture information is discarded, while the motion is used to motion compensate the previous decoded VOP.

Fast recovery in real-time coding

A newly developed technique in MPEG, called NEWPRED (for "new prediction"), provides a fast error recovery in real-time coding applications. It uses an upstream channel from the decoder to the encoder. The encoder switches the reference frames adaptively according to the error conditions of the network. NEWPRED does not use intra refresh and it provides the high coding efficiency. This technique has been proven to work under stressful error conditions:

- Burst Error on the wireless networks (averaged bit error rate is 10E-3, 1ms burst length)
- Packet Loss on the internet (packet loss rate is 5%)

c) Improved temporal resolution stability with low buffering delay

A special technique of use in real-time encoding situations is Dynamic Resolution Conversion (DRC), a way to stabilize the transmission buffering delay by minimizing the jitter of the amount of the coded output bits per VOP. Large frame skips are also prevented and the encoder can control the temporal resolution even in highly active scenes. This techniques requires backchannel information to be sent to the encoder, which explains why it is only useful in real-time situations.

11.2 Structure of the tools for representing natural video

The MPEG-4 image and video coding algorithms give an efficient representation of visual objects of arbitrary shape, also supporting so-called content-based functionalities. They support most functionalities already provided by MPEG-1 and MPEG-2, including efficient compression of standard rectangular sized image sequences at varying levels of input formats, frame rates, pixel depth, bit-rates, and various levels of spatial, temporal and quality scalability.

A basic classification of the bit rates and functionalities currently provided by the MPEG-4 Visual standard for natural images and video is depicted in Figure 20 below, which clusters bit-rate levels versus sets of functionalities.

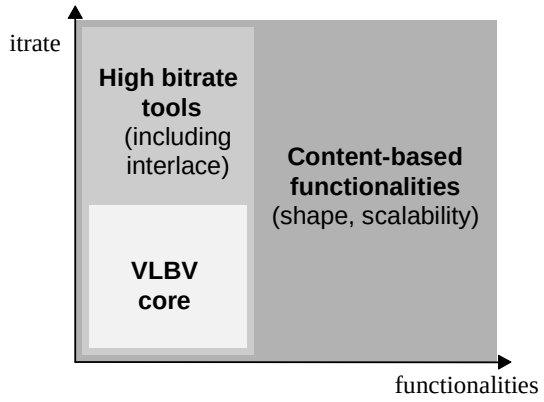


Figure 20 - Classification of the MPEG-4 Image and Video Coding Algorithms and Tools

At the bottom end a “VLBV Core” (VLBV: Very Low Bit-rate Video) provides algorithms and tools for applications operating at bit-rates typically between 5...64 kbits/s, supporting image sequences with low spatial resolution (typically up to CIF resolution) and low frame rates (typically up to 15 Hz). The basic applications specific functionalities supported by the VLBV Core include:

a) coding of conventional rectangular size image sequences with high coding efficiency and high error robustness/resilience, low latency and low complexity for real-time multimedia communications applications, and

b) “random access” and “fast forward” and “fast reverse” operations for VLB multimedia data-base storage and access applications.

The same basic functionalities outlined above are also supported at higher bit-rates with a higher range of spatial and temporal input parameters up to ITU-R Rec. 601 resolutions and

larger - employing identical or similar algorithms and tools as the VLBV Core. The bit-rates envisioned range typically from 64 kbits/s up to 10 Mb/s and applications envisioned include multimedia broadcast or interactive retrieval of signals with a quality comparable to digital TV. For these applications at higher bit-rates, also interlaced can be represented by MPEG-4 coding tools.

Content-based functionalities support the separate encoding and decoding of content (i.e. physical objects in a scene, VOs). This MPEG-4 feature provides the most elementary mechanism for interactivity; flexible representation and manipulation with/of VO content of images or video in the compressed domain, without the need for further segmentation or transcoding at the receiver.

For the hybrid coding of natural as well as synthetic visual data (e.g. for virtual presence or virtual environments) the content-based coding functionality allows mixing a number of VO's from different sources with synthetic objects, such as a virtual backgrounds.

The extended MPEG-4 algorithms and tools for content-based functionalities can be seen as a superset of the VLBV core and high bit-rate tools - meaning that the tools provided by the VLBV and higher bitrate cores are complemented by additional elements.

11.3 The MPEG-4 Video Image Coding Scheme

Figure 21 below outlines the basic approach of the MPEG-4 video algorithms to encode rectangular as well as arbitrarily shaped input image sequences.

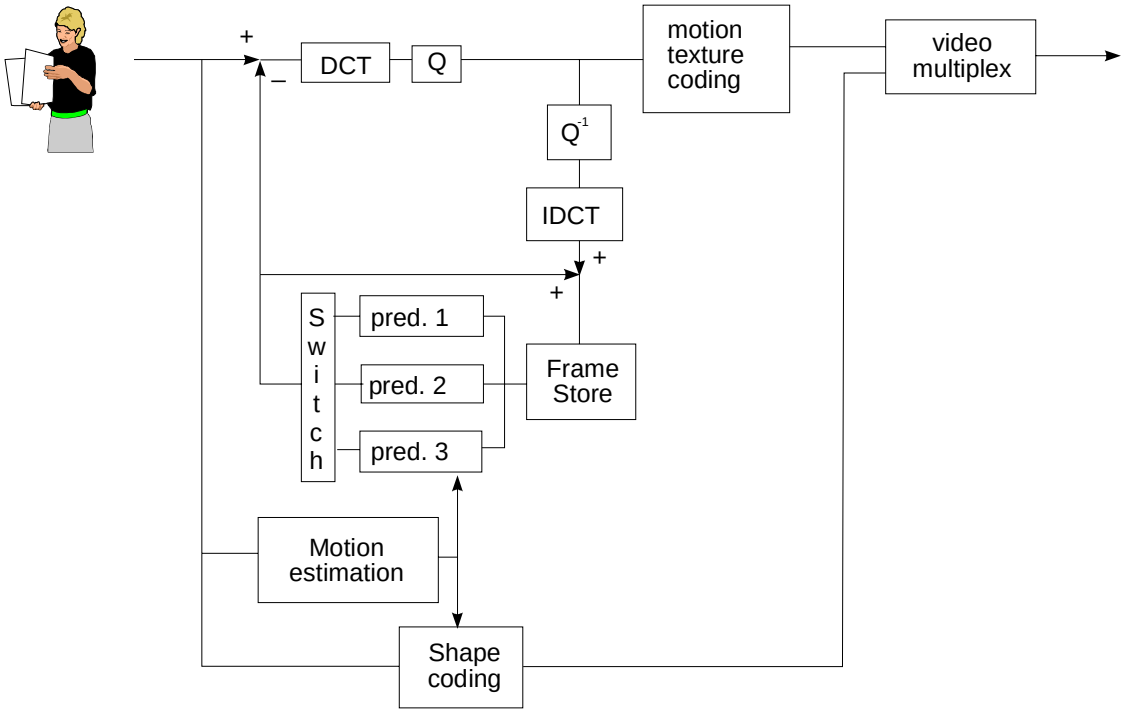


Figure 21 - Basic block diagram of MPEG-4 Video Coder

The basic coding structure involves shape coding (for arbitrarily shaped VOs) and motion compensation as well as DCT-based texture coding (using standard 8x8 DCT or shape adaptive DCT).

The basic coding structure involves shape coding (for arbitrarily shaped VOs) and motion compensation as well as DCT-based texture coding (using standard 8x8 DCT or shape adaptive DCT).

An important advantage of the content-based coding approach MPEG-4 is that the compression efficiency can be significantly improved for some video sequences by using appropriate and dedicated object-based motion prediction "tools" for each object in a scene. A number of motion prediction techniques can be used to allow efficient coding and flexible presentation of the objects:

- Standard 8x8 or 16x16 pixel block-based motion estimation and compensation, with up to ¼ pel accuracy
- Global Motion Compensation (GMC) for video objects: Encoding of the global motion for a object using a small number of parameters. GMC is based on global motion estimation, image warping, motion trajectory coding, and texture coding for prediction errors.
- Global motion compensation based for static "sprites". A static sprite is a possibly large still image, describing panoramic background. For each consecutive image in a sequence, only 8 global motion parameters describing camera motion are coded to reconstruct the object. These parameters represent the appropriate affine transform of the sprite transmitted in the first frame.
- Quarter Pel Motion Compensation enhances the precision of the motion compensation scheme, at the cost of only small syntactical and computational overhead. A accurate motion description leads to a smaller prediction error and, hence, to better visual quality.
- Shape-adaptive DCT: In the area of texture coding, the shape-adaptive DCT (SA-DCT) improves the coding efficiency of arbitrary shaped objects. The SA-DCT algorithm is based on predefined orthonormal sets of one-dimensional DCT basis functions.

Figure 22 depicts the basic concept for coding an MPEG-4 video sequence using a sprite panorama image. It is assumed that the foreground object (tennis player, image top right) can be segmented from the background and that the sprite panorama image can be extracted from the sequence prior to coding. (A sprite panorama is a still image that describes as a static image the content of the background over all frames in the sequence). The large panorama sprite image is transmitted to the receiver only once as first frame of the sequence to describe the background – the sprite remains is stored in a sprite buffer. In each consecutive frame only the camera parameters relevant for the background are transmitted to the receiver. This allows the receiver to reconstruct the background image for each frame in the sequence based on the sprite. The moving foreground object is transmitted separately as an arbitrary-shape video object. The receiver composes both the foreground and background images to reconstruct each frame (bottom picture in figure below). For low delay applications it is possible to transmit the sprite in multiple smaller pieces over consecutive frames or to build up the sprite at the decoder progressively.

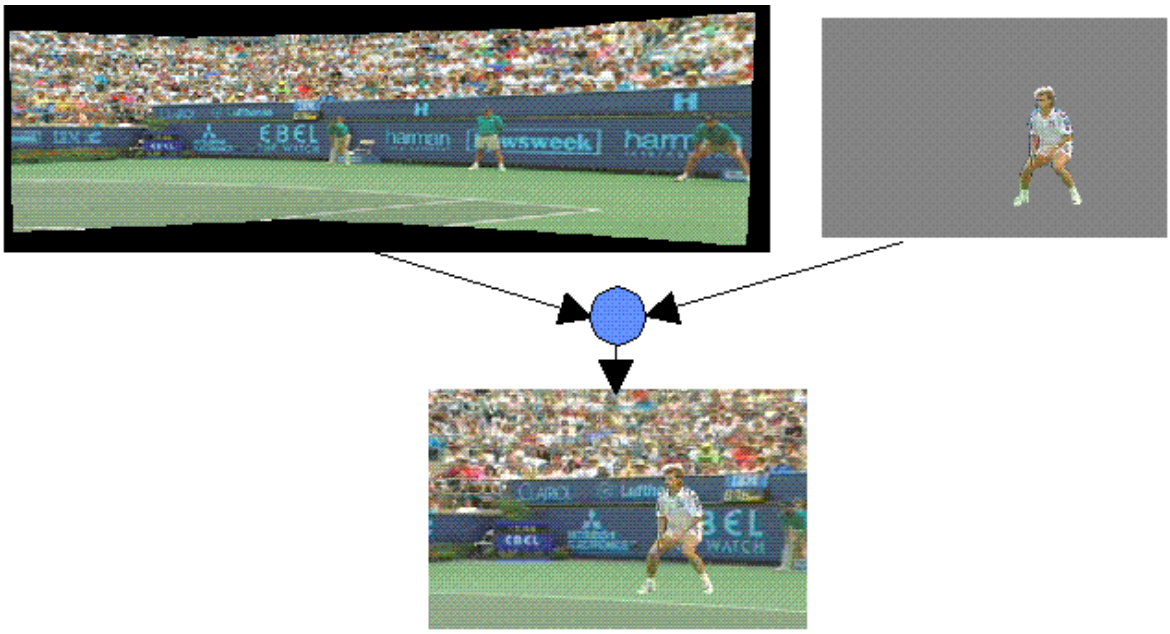


Figure 22 - Example of Sprite Coding of Video Sequence

Subjective evaluation tests within MPEG have shown that the combination of these techniques can result in a bitstream saving of up to 50% compared with the version 1, depending on content type and datarate.

11.4 Coding of Textures and Still Images

Efficient Coding of visual textures and still images (e.g. to be mapped on animated meshes) is supported by the visual texture mode of the MPEG-4. This mode is based on a zerotree wavelet algorithm that provides very high coding efficiency over a very wide range of bitrates. Together with high compression efficiency, it also provides spatial and quality scalabilities (up to 11 levels of spatial scalability and continuous quality scalability) and also arbitrary-shaped object coding. The wavelet formulation provides for scalable bitstream coding in the form of an image resolution pyramid for progressive transmission and temporal enhancement of still images. The coded bitstream is also intended for downloading of the image resolution hierarchy into the terminal to be formatted as 'MIPmap texture' as used in 3-D rendering systems. This technology provides the resolution scalability to deal with a wide range of viewing conditions more typical of interactive applications and the mapping of imagery into 2-D and 3-D virtual worlds.

Wavelet tiling allows an image to be divided into several tiles and each tile to be encoded independently. This means that large images to be encoded/decoded with very low memory requirements, and that random access at the decoder is significantly enhanced.

Scalable shape coding allows encoding of arbitrary shaped textures and still images in a scalable fashion. Using this tool, a decoder can decode an arbitrary shaped image at any desired resolution. This tool enables applications to employ object-based, spatial and quality scalabilities at the same time.

For decoding of still images, the MPEG-4 standard provides spatial scalability with up to 11 levels of granularity and also quality scalability up to the bit level.

11.5 Synthetic Objects

Synthetic objects form a subset of the larger class of computer graphics. MPEG-4 supports the following visual synthetic objects:

- Parametric descriptions of
  a) a synthetic the face and body (body animation in Version 2)
  b) Static and Dynamic Mesh Coding with texture mapping
- Texture Coding for View Dependent applications

These are described in the subsections below.

11.5.1 Face Animation

The 'facial animation object' can be used to render an animated face. The shape, texture and expressions of the face are controlled by Facial Definition Parameters (FDPs) and/or Facial Animation Parameters (FAPs). Upon construction, the face object contains a generic face with a neutral expression. This face can already be rendered. It can also immediately receive the animation parameters from the bitstream, which will produce animation of the face: expressions, speech etc. Meanwhile, definition parameters can be sent to change the appearance of the face from something generic to a particular face with its own shape and (optionally) texture. If so desired, a complete face model can be downloaded via the FDP set.

Face Animation in MPEG-4 Version 1 provides for highly efficient coding of animation parameters that can drive an unlimited range of face models. The models themselves are not normative, although (see above) there are normative tools to describe the appearance of the model. Frame-based and temporal-DCT coding of a large collection of FAPs can be used for accurate speech articulation. Viseme and expression parameters are used to code specific speech configurations of the lips and the mood of the speaker.

The Systems Binary Format for Scenes (BIFS, see section 2.6) provides features to support Face Animation when custom models and specialized interpretation of FAPs are needed:

1 the Face Definition Parameters (FDP) in BIFS (model data downloadable to configure a baseline face model pre-stored in the terminal into a particular face before FAP decoding, or to install a specific face model at the beginning of a session along with the information about how to animate it);

2 the Face Animation Table (FAT) within FDPs (downloadable functional mapping from incoming FAPs to feature control points in the face mesh. This provides piecewise linear mappings of incoming FAPs for controlling facial movements. Example: the FAP could say 'open_jaw (500) and the table then defines what this means in terms of moving the feature points;

3 the Face Interpolation Technique (FIT) in BIFS (downloadable definition of mapping of incoming FAPs into a total set of FAPs before their application to feature points, through weighted rational polynomial functions invoked by conditional evaluation of a Face Interpolation Graph). This can be used for complex cross-coupling of FAPs to link their effects, or to interpolate FAPs missing in the stream using the FAPs that are available in the terminal).

These specialized node types in BIFS effectively provide for tailored face models including calibration of an established face model in a terminal or downloading of a fully custom model including its shape, texture, and color.

11.5.2 Body animation

The Body is an object capable of producing virtual body models and animations in form of a set of 3-D polygonal meshes ready for rendering. Two sets of parameters are defined for the body: Body Definition Parameter (BDP) set, and Body Animation Parameter (BAP) set. The BDP set defines the set of parameters to transform the default body to a customized body with its body surface, body dimensions, and (optionally) texture. The Body Animation Parameters (BAPs), if correctly interpreted, will produce reasonably similar high level results in terms of body posture and animation on different body models, without the need to initialize or calibrate the model.

Upon construction, the Body object contains a generic virtual human body with the default posture. This body can already be rendered. It is also immediately capable of receiving the BAPs from the bitstream, which will produce animation of the body. If BDPs are received, they are used to transform the generic body into a particular body determined by the parameters contents. Any component can be null. A null component is replaced by the corresponding default component when the body is rendered. The default posture is defined by standing posture. This posture is defined as follows: the feet should point to the front direction, the two arms should be placed on the side of the body with the palm of the hands facing inward. This posture also implies that all BAPs have default values.

No assumption is made and no limitation is imposed on the range of motion of joints. In other words the human body model should be capable of supporting various applications, from realistic simulation of human motions to network games using simple human-like models. The work on Body Animation includes the assessment of the emerging standard as applied to hand signing for the listening-impaired.

The Body Animation standard has been developed by MPEG in concert with the Humanoid Animation Working Group within the VRML Consortium, with the objective of achieving consistent conventions and control of body models which are being established by H-Anim.

11.5.3 2-D animated meshes

A 2-D *mesh* is a tessellation (or partition) of a 2-D planar region into polygonal patches. The vertices of the polygonal patches are referred to as the *node points* of the mesh. MPEG4 considers only triangular meshes where the patches are triangles. A 2-D dynamic mesh refers to 2-D mesh geometry and motion information of all mesh node points within a temporal segment of interest. Triangular meshes have long been used for efficient 3-D object shape (geometry) modeling and rendering in computer graphics. 2-D mesh modeling may be considered as projection of such 3-D triangular meshes onto the image plane. An example of a 2-D mesh is depicted in Figure 23.

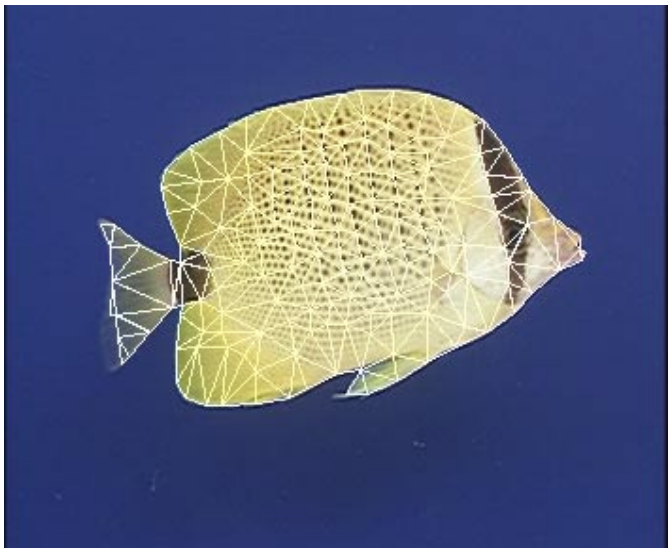


Figure 23- 2-D mesh modeling of the "Breen" video object.
By deforming the mesh, the fish can be animated very efficiently, and be made to 'swim'. Also, a logo could be projected onto the fish, and made to move in accordance with the fish

A dynamic mesh is a forward tracking mesh, where the node points of the initial mesh track image features forward in time by their respective motion vectors. The initial mesh may be regular, or can be adapted to the image content, which is called a *content-based mesh*. 2-D content-based mesh modeling then corresponds to non-uniform sampling of the motion field at a number of salient feature points (node points) along the contour and interior of a video object. Methods for selection and tracking of these node points are not subject to standardization.

In 2-D mesh based texture mapping, triangular patches in the current frame are deformed by the movements of the node points into triangular patches in the reference frame. The texture inside each patch in the reference frame is *warped* onto the current frame using a parametric mapping, defined as a function of the node point motion vectors. For triangular meshes, the affine mapping is a common choice. Its linear form implies texture mapping with low computational complexity. Affine mappings can model translation, rotation, scaling, reflection and shear, and preserve straight lines. The degrees of freedom given by the three motion vectors of the vertices of a triangle match with the six parameters of the affine mapping. This implies that the original 2-D motion field can be compactly represented by the motion of the node points, from which a continuous, piece-wise affine motion field can be reconstructed. At the same time, the mesh structure constrains movements of adjacent image patches. Therefore, meshes are well-suited to represent mildly deformable but spatially continuous motion fields.

2-D mesh modeling is attractive because 2-D meshes can be designed from a single view of an object without requiring range data, while maintaining several of the functionalities offered by 3-D mesh modeling. In summary, the 2-D object-based mesh representation is able to model the shape (polygonal approximation of the object contour) and motion of a VOP in a unified framework, which is also extensible to the 3-D object modeling when data to construct such models is available. In particular, the 2-D mesh representation of video objects enables the following functionalities:

A. Video Object Manipulation

- Augmented reality: Merging virtual (computer generated) images with real moving images (video) to create enhanced display information. The computer-generated images must remain in perfect registration with the moving real images (hence the need for tracking).
- Synthetic-object-transfiguration/animation: Replacing a natural video object in a video clip by another video object. The replacement video object may be extracted from another natural video clip or may be transfigured from a still image object using the motion information of the object to be replaced (hence the need for a temporally continuous motion representation).
- Spatio-temporal interpolation: Mesh motion modeling provides more robust motion-compensated temporal interpolation (frame rate up-conversion).

B. Video Object Compression

- 2-D mesh modeling may be used for compression if one chooses to transmit texture maps only at selected key frames and animate these texture maps (without sending any prediction error image) for the intermediate frames. This is also known as self-transfiguration of selected key frames using 2-D mesh information.

C. Content-Based Video Indexing

- Mesh representation enables animated key snapshots for a moving visual synopsis of objects.
- Mesh representation provides accurate object trajectory information that can be used to retrieve visual objects with specific motion.
- Mesh representation provides vertex-based object shape representation which is more efficient than the bitmap representation for shape-based object retrieval.

11.5.4 3D Meshes

Capabilities for 3-D mesh coding include:

- Coding of generic 3-D polygonal meshes enables the efficient encoding of 3-D polygonal meshes. The coded representation is generic enough to support both manifold and non-manifold meshes.
- Incremental representation enables a decoder to reconstruct a number faces in a mesh proportional to the number of bits in the bit stream that have been processed. This furthermore enables incremental rendering.
- Error resilience enables a decoder to partially recover a mesh when subsets of the bit stream are missing and/or corrupted.
- LOD (Level Of Detail) scalability enables a decoder to reconstruct a simplified version of the original mesh containing a reduced number of vertices from a subset of the bit stream. Such simplified representations are useful to reduce the rendering time of objects which are distant from the viewer (LOD management), but also enable less powerful rendering engines to render the object at a reduced quality.

a) View-dependent scalability

The view-dependent scalability enables to stream texture maps, which are used in realistic virtual environments. It consists in taking into account the viewing position in the 3-D virtual world in order to transmit only the most visible information. Only a fraction of the information is then sent, depending on object geometry and viewpoint displacement. This fraction is computed both at the encoder and at the decoder side. This approach allows to reduce greatly the amount of transmitted information between a remote database and a user, given that a back-

channel is available. This scalability can be applied both with DCT and Wavelet based encoders. It is obviously easily achieved in DCT encoders, in which each 8x8 texture block is DCT transformed and encoded separately. Wavelet based encoders, on the contrary, apply a more global transform on the image, therefore requiring some precautions in texture block selection. Nevertheless, MPEG-4 wavelet-based *Visual Texture Coding* tool (VTC) supports through its error-resilience packetization feature the separation of the coded texture bitstream into its constituent regions of interest. Each packet of the bitstream then corresponds to a specific region of the texture at a certain quality and resolution level. The start of each packet is unambiguously identified by a unique bit-sequence, called texture marker. Detection of such markers therefore allows the selection of texture regions and thus also view-dependent scalability.

12 Detailed technical description of MPEG-4 Audio

MPEG-4 coding of audio objects provides tools for both representing natural sounds (such as speech and music) and for synthesizing sounds based on structured descriptions. The representation for synthesized sound can be derived from text data or so-called instrument descriptions and by coding parameters to provide effects, such as reverberation and spatialization. The representations provide compression and other functionalities, such as scalability and effects processing.

The MPEG-4 Audio coding tools covering 6kbit/s to 24kbit/s have undergone verification testing for an AM digital audio broadcasting application in collaboration with the NADIB (Narrow Band Digital Broadcasting) consortium. With the intent of identifying a suitable digital audio broadcast format to provide improvements over the existing AM modulation services, several codec configurations involving the MPEG-4 CELP, TwinVQ, and AAC tools have been compared to a reference AM system. (see below for an explanation about these algorithms.) It was found that higher quality can be achieved in the same bandwidth with digital techniques and that scalable coder configurations offered performance superior to a simulcast alternative. Additional verification tests were carried out by MPEG, in which the tools for speech and general audio coding were compared to existing standards.

12.1 Natural Sound

MPEG-4 standardizes natural audio coding at bitrates ranging from 2 kbit/s up to and above 64 kbit/s. When variable rate coding is allowed, coding at less than 2 kbit/s, such as an average bitrate of 1.2 kbit/s, is also supported. The presence of the MPEG-2 AAC standard within the MPEG-4 tool set provides for general compression of audio in the upper bitrate range. For these, the MPEG-4 standard defines the bitstream syntax and the decoding processes in terms of a set of tools. In order to achieve the highest audio quality within the full range of bitrates and at the same time provide the extra functionalities, speech coding techniques and general audio coding techniques are integrated in a common framework:

- Speech coding at bitrates between 2 and 24 kbit/s is supported by using Harmonic Vector eXcitation Coding (HVXC) for a recommended operating bitrate of 2 - 4 kbit/s, and Code Excited Linear Predictive (CELP) coding for an operating bitrate of 4 - 24 kbit/s. In addition, HVXC can operate down to an average of around 1.2 kbit/s in its variable bitrate mode. In CELP coding, two sampling rates, 8 and 16 kHz, are used to support narrowband and wideband speech, respectively. The following operating modes have been subject to verification testing: HVXC at 2 and 4 kbit/s, narrowband CELP at 6, 8.3, and 12 kbit/s, and

wideband CELP at 18 kbit/s. In addition various of the scalable configurations have been verified.

- For general audio coding at bitrates at and above 6 kbit/s, transform coding techniques, namely TwinVQ and AAC, are applied. The audio signals in this region typically have sampling frequencies starting at 8 kHz.

To allow optimum coverage of the bitrates and to allow for bitrate and bandwidth scalability, a general framework has been defined. This is illustrated in Figure 24.

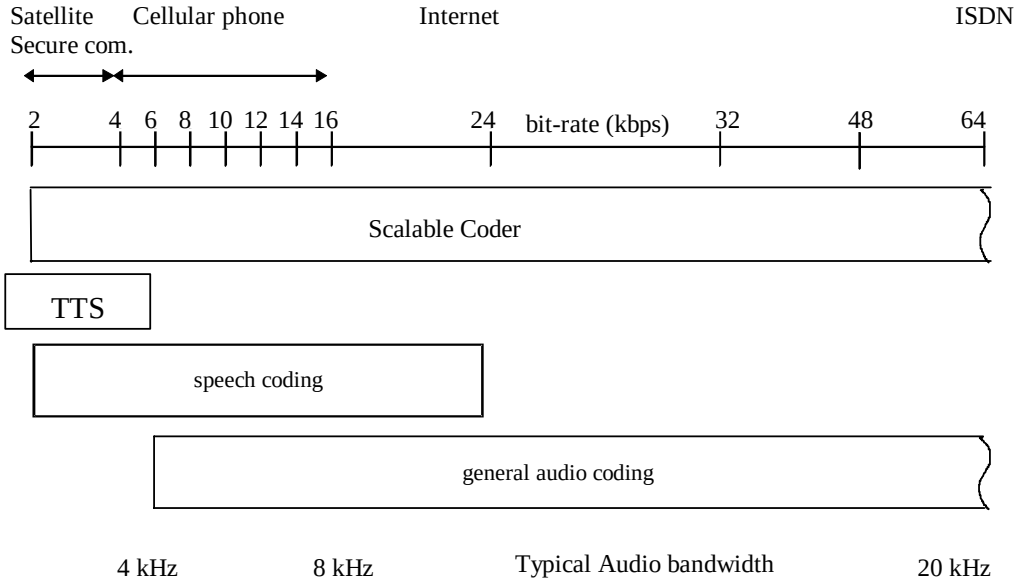


Figure 24 - General block diagram of MPEG-4 Audio

Starting with a coder operating at a low bitrate, by adding enhancements to a general audio coder, both the coding quality as well as the audio bandwidth can be improved.

Bitrate scalability, often also referred to as embedded coding, allows a bitstream to be parsed into a bitstream of lower bitrate that can still be decoded into a meaningful signal. The bitstream parsing can occur either during transmission or in the decoder. Bandwidth scalability is a particular case of bitrate scalability whereby part of a bitstream representing a part of the frequency spectrum can be discarded during transmission or decoding.
Encoder complexity scalability allows encoders of different complexity to generate valid and meaningful bitstreams. The decoder complexity scalability allows a given bitstream to be decoded by decoders of different levels of complexity. The audio quality, in general, is related to the complexity of the encoder and decoder used. Scalability works within some MPEG-4 tools, but can also be applied to a combination of techniques, e.g. with CELP as a base layer and AAC for the enhancement layer(s).

The MPEG-4 systems layer allows codecs according to existing (MPEG) standards, e.g. MPEG-2 AAC, to be used. Each of the MPEG-4 coders is designed to operate in a stand-alone mode with its own bitstream syntax. Additional functionalities are realized both within individual coders, and by means of additional tools around the coders. An example of such a functionality within an individual coder is speed or pitch change within HVXC.

12.1.1 Error Robustness

The error robustness tools provide improved performance on error-prone transmission channels. They can be distinguished into codec specific error resilience tools and a common error protection tool.

Improved error robustness for AAC is provided by a set of error resilience tools. These tools reduce the perceived deterioration of the decoded audio signal that is caused by corrupted bits in the bit stream. The following tools are provided to improve the error robustness for several parts of an AAC frame:

- Virtual CodeBook tool (VCB11)
- Reversible Variable Length Coding tool (RVLC)
- Huffman Codeword Reordering tool (HCR)

Improved error robustness capabilities for all coding tools are provided through the error resilient bit stream payload syntax. It allows advanced channel coding techniques, which can be adapted to the special needs of the different coding tools. This error resilient bit stream payload syntax is mandatory for all Version 2 object types.

The error protection tool (EP tool) provides error protection for all MPEG-4 Audio version 2 audio objects with flexible configuration applicable for wide range of channel error conditions. The main features of the EP tool are as follows:

- providing a set of error correcting/detecting codes with wide and small-step scalability, in performance and in redundancy
- providing a generic and bandwidth-efficient error protection framework which covers both fixed-length frame bit streams and variable-length frame bit streams
- providing an Unequal Error Protection (UEP) configuration control with low overhead

MPEG-4 Audio version 2 coding algorithms provide a classification of each bit stream field according to its error sensitivity. Based on this, the bit stream is divided into several classes, which can be separately protected by the EP tool, such that more error sensitive parts are protected more strongly.

12.1.2 Low-Delay Audio Coding

While the MPEG-4 General Audio Coder provides very efficient coding of general audio signals at low bit rates, it has an algorithmic encoding/decoding delay of up to several 100ms and is thus not well suited for applications requiring low coding delay, such as real-time bi-directional communication. As an example, for the General Audio Coder operating at 24 kHz sampling rate and 24 kbit/s, this results in an algorithmic coding delay of about 110 ms plus up to additional 210 ms for the use of the bit reservoir. To enable coding of general audio signals with an algorithmic delay not exceeding 20 ms, MPEG-4 Version 2 specifies a Low-Delay Audio Coder which is derived from MPEG-2/4 Advanced Audio Coding (AAC). . Compared to speech coding schemes, this coder allows compression of general audio signal types, including music, at a low delay. It operates at up to 48 kHz sampling rate and uses a frame length of 512 or 480 samples, compared to 1024 or 960 samples used in standard MPEG-2/4 AAC. Also the size of the window used in the analysis and synthesis filter bank is reduced by a factor of 2. No block switching is used to avoid the ”look-ahead" delay due to the block switching decision. To reduce pre-echo artifacts in case of transient signals, window shape switching is provided instead. For non-transient parts of the signal a sine window is used, while a so-called low overlap window is used in case of transient signals. Use of the bit reservoir is minimized in the encoder in order to reach the desired target delay. As one extreme case, no bit reservoir is used at all. Verification tests have shown that the reduction in coding delay comes at a very moderate cost in compression performance.

12.1.3 Fine granularity scalability

Bit rate scalability, also known as embedded coding, is a very desirable functionality. The General Audio Coder of Version 1 supports large step scalability where a base layer bit stream can be combined with one or more enhancement layer bit streams to utilize a higher bit rate and thus obtain a better audio quality. In a typical configuration, a 24 kbit/s base layer and two 16 kbit/s enhancement layers could be used, permitting decoding at a total bit rate of 24 kbit/s (mono), 40 kbit/s (stereo), and 56 kbit/s (stereo). Due to the side information carried in each layer, small bit rate enhancement layers are not efficiently supported in Version 1. To address this problem and to provide efficient small step scalability for the General Audio Coder, the Bit-Sliced Arithmetic Coding (BSAC) tool is available in Version 2. This tool is used in combination with the AAC coding tools and replaces the noiseless coding of the quantized spectral data and the scale factors. BSAC provides scalability in steps of 1 kbit/s per audio channel, i.e. 2 kbit/s steps for a stereo signal. One base layer bit stream and many small enhancement layer bit streams are used. The base layer contains the general side information, specific side information for the first layer and the audio data of the first layer. The enhancement streams contain only the specific side information and audio data for the corresponding layer. To obtain fine step scalability, a bit-slicing scheme is applied to the quantized spectral data. First the quantized spectral values are grouped into frequency bands. Each of these groups contains the quantized spectral values in their binary representation. Then the bits of a group are processed in slices according to their significance. Thus first all most significant bits (MSB) of the quantized values in a group are processed, etc. These bit-slices are then encoded using an arithmetic coding scheme to obtain entropy coding with minimal redundancy. Various arithmetic coding models are provided to cover the different statistics of the bit-slices. The scheme used to assign the bit-slices of the different frequency bands to the enhancement layer is constructed in a special way. This ensures that, with an increasing number of enhancement layers utilized by the decoder, providing more of the less significant bits refines quantized spectral data. But also providing bit-slices of the spectral data in higher frequency bands increases the bandwidth.

Verification tests have shown that the scalability aspect of this tool performs well over a wide range of rates. At the highest rate it is as good as AAC main profile operating at the same rate, while at the lowest rate the scalability function requires a moderate overhead relative to AAC main profile operating at the same rate.

12.1.4 Parametric Audio Coding

The Parametric Audio Coding tools combine very low bit rate coding of general audio signals with the possibility of modifying the playback speed or pitch during decoding without the need for an effects processing unit. In combination with the speech and audio coding tools of Version 1, improved overall coding efficiency is expected for applications of object based coding allowing selection and/or switching between different coding techniques.

Parametric Audio Coding uses the Harmonic and Individual Lines plus Noise (HILN) technique to code general audio signals at bit rates of 4 kbit/s and above using a parametric representation of the audio signal. The basic idea of this technique is to decompose the input signal into audio objects, which are described by appropriate source models and represented by model parameters. Object models for sinusoids, harmonic tones, and noise are utilized in the HILN coder.

This approach allows to introduce a more advanced source model than just assuming a stationary signal for the duration of a frame, which motivates the spectral decomposition used e.g. in the MPEG-4 General Audio Coder. As known from speech coding, where specialized source models based on the speech generation process in the human vocal tract are applied, advanced source models can be advantageous in particular for very low bit rate coding schemes.

Due to the very low target bit rates, only the parameters for a small number of objects can be transmitted. Therefore a perception model is employed to select those objects that are most important for the perceptual quality of the signal.
In HILN, the frequency and amplitude parameters are quantized according to the "just noticeable differences" known from psychoacoustics. The spectral envelope of the noise and the harmonic tone is described using LPC modeling as known from speech coding. Correlation between the parameters of one frame and between consecutive frames is exploited by parameter prediction. The quantized parameters are finally entropy coded and multiplexed to form a bit stream.
A very interesting property of this parametric coding scheme arises from the fact that the signal is described in terms of frequency and amplitude parameters. This signal representation permits speed and pitch change functionality by simple parameter modification in the decoder. The HILN parametric audio coder can be combined with MPEG-4 parametric speech coder (HVXC) to form an integrated parametric coder covering a wider range of signals and bit rates. This integrated coder supports speed and pitch change. Using a speech/music classification tool in the encoder, it is possible to automatically select the HVXC for speech signals and the HILN for music signals. Such automatic HVXC/HILN switching was successfully demonstrated and the classification tool is described in the informative Annex of the Version 2 standard.
Verification test have shown that HILN coding has performance comparable to other MPEG-4 coding technology operating at similar bit rates while providing the additional capability of independent audio signal speed or pitch change when decoding. The test has also shown that the scalable HILN coder provides quality comparable to that of a fixed-rate HILN coder at the same bit rate.

12.1.5 CELP Silence Compression

The silence compression tool reduces the average bit rate thanks to a lower bit rate compression for silence. In the encoder, a voice activity detector is used to distinguish between regions with normal speech activity and those with silence or background noise. During normal speech activity, the CELP coding as in Version 1 is used. Otherwise a Silence Insertion Descriptor (SID) is transmitted at a lower bit rate. This SID enables a Comfort Noise Generator (CNG) in the decoder. The amplitude and spectral shape of this comfort noise is specified by energy and LPC parameters similar as in a normal CELP frame. These parameters are an optional part of the SID and thus can be updated as required.

12.1.6 Error Resilient HVXC

The Error Resilient (ER) HVXC object is supported by the Parametric speech coding (ER HVXC) tools, which provides fixed bit-rate modes(2.0-4.0kbps) and variable bit-rate mode(<2.0kbps, <4.0kbps) both in a scalable and non-scalable scheme. In the Version 1 HVXC, variable bit rate mode of 2.0 kbit/s maximum is already supported and the variable bit rate mode of 4.0 kbit/s maximum is additionally supported in Version 2 ER HVXC. In the variable bit rate modes, non-speech parts are detected in unvoiced signals, and a smaller number of bits is used for these non-speech parts to reduce the average bit rate. ER HVXC provides communications-quality to near-toll-quality speech in the 100-3800 Hz band at 8kHz sampling rate. When the variable bit-rate mode is allowed, operation at lower average bit-rate is possible. Coded speech with variable bit-rate mode at typical bit-rate of 1.5kbps average, and at typical bit-rate of 3.0kbps average has essentially the same quality as 2.0 kbps fixed rate and 4.0 kbps fixed rate respectively. The functionality of pitch and speed change during decoding is supported for all modes. ER HVXC has the syntax with the error sensitivity classes to be used with the EP-Tool, and the error concealment functionality are supported for the use for error-prone channel like mobile communication channels. The ER HVXC speech coder targets

applications from mobile and satellite communications, to Internet telephony, to packaged media and speech databases.

12.1.7 Environmental Spatialization

The Environmental Spatialization tools enable composition of an audio scene with more natural sound source and sound environment modeling than is possible in Version 1. Both, a physical and a perceptual approach to spatialization are supported. The *physical approach* is based on a description of the acoustical properties of the environment (e.g. room geometry, material properties, position of sound source) and can be used in applications like 3-D virtual reality. The *perceptual approach* on the other hand permits a high level perceptual description of the audio scene based on parameters similar to those of a reverberation effects unit. Thus, the audio and the visual scene can be composed independently as usually required by applications like movies. Although the Environmental Spatialization tools are related to audio, they are part of the BInary Format for Scene description (BIFS) in MPEG-4 Systems and are referred to as Advanced AudioBIFS.

12.1.8 Back channel

The back channel allows a request of client and/or client terminal to server. With this capability, interactivity can be achieved. In MPEG-4 System, the need for an up-stream channel (back channel) is signaled to the client terminal by supplying an appropriate elementary stream descriptor declaring the parameters for that stream. The client terminal opens this upstream channel in a similar manner as it opens the down-stream channels. The entities (e.g. media encoders & decoders) that are connected through an upstream channel are known from the parameters in its elementary stream descriptor and from the association of the elementary stream descriptor to a specific object descriptor. In MPEG-4 Audio, the back channel allows feedback for bit rate adjustment, the scalability and error protection adaptation.

12.1.9 Audio transport stream

The MPEG-4 Audio transport stream defines a mechanism to transport MPEG-4 Audio streams without using MPEG-4 Systems and is dedicated for audio-only applications. The transport mechanism uses a two-layer approach, namely a multiplex layer and a synchronization layer. The multiplex layer (Low-overhead MPEG-4 Audio Transport Multiplex: LATM) manages multiplexing of several MPEG-4 Audio payloads and audio specific configuration information. The synchronization layer specifies a self-synchronized syntax of the MPEG-4 Audio transport stream which is called Low Overhead Audio Stream (LOAS). The Interface format to a transmission layer depends on the conditions of the underlying transmission layer.

12.2 Synthesized Sound

MPEG-4 defines decoders for generating sound based on several kinds of ‘structured’ inputs. Text input is converted to speech in the Text-To-Speech (TTS) decoder, while more general sounds including music may be normatively synthesized. Synthetic music may be delivered at extremely low bitrates while still describing an exact sound signal.

Text To Speech. TTS coders bitrates range from 200 bit/s to 1.2 Kbit/s, which allows a text or a text with prosodic parameters (pitch contour, phoneme duration, and so on) as its inputs to generate intelligible synthetic speech. It supports the generation of parameters that can be used to allow synchronization to associated face animation, international languages for text and international symbols for phonemes. Additional markups are used to convey control information within texts, which is forwarded to other components in synchronization with the

synthesized text. Note that MPEG-4 provides a standardized interface for the operation of a Text To Speech coder (TTSI = Text To Speech Interface), but *not* a normative TTS synthesizer itself.

An itemized overview:

- Speech synthesis using the prosody of the original speech
- Lip synchronization control with phoneme information.
- Trick mode functionality: pause, resume, jump forward/backward.
- International language and dialect support for text. (i.e. it can be signaled in the bitstream which language and dialect should be used)
- International symbol support for phonemes.
- support for specifying age, gender, speech rate of the speaker
- support for conveying facial animation parameter(FAP) bookmarks.

Score Driven Synthesis.

The Structured Audio tools decode input data and produce output sounds. This decoding is driven by a special synthesis language called SAOL (Structured Audio Orchestra Language) standardized as a part of MPEG-4. This language is used to define an "orchestra" made up of "instruments" (downloaded in the bitstream, not fixed in the terminal) which create and process control data. An instrument is a small network of signal processing primitives that might emulate some specific sounds such as those of a natural acoustic instrument. The signal-processing network may be implemented in hardware or software and include both generation and processing of sounds and manipulation of pre-stored sounds.

MPEG-4 does not standardize "a single method" of synthesis, but rather a way to describe methods of synthesis. Any current or future sound-synthesis method can be described in SAOL, including wavetable, FM, additive, physical-modeling, and granular synthesis, as well as non-parametric hybrids of these methods.

Control of the synthesis is accomplished by downloading "scores" or "scripts" in the bitstream. A score is a time-sequenced set of commands that invokes various instruments at specific times to contribute their output to an overall music performance or generation of sound effects. The score description, downloaded in a language called SASL (Structured Audio Score Language), can be used to create new sounds, and also include additional control information for modifying existing sound. This allows the composer finer control over the final synthesized sound. For synthesis processes that do not require such fine control, the established MIDI protocol may also be used to control the orchestra.

Careful control in conjunction with customized instrument definition, allows the generation of sounds ranging from simple audio effects, such as footsteps or door closures, to the simulation of natural sounds such as rainfall or music played on conventional instruments to fully synthetic sounds for complex audio effects or futuristic music.

For terminals with less functionality, and for applications which do not require such sophisticated synthesis, a "wavetable bank format" is also standardized. Using this format, sound samples for use in wavetable synthesis may be downloaded, as well as simple processing, such as filters, reverbs, and chorus effects. In this case, the computational complexity of the required decoding process may be exactly determined from inspection of the bitstream, which is not possible when using SAOL.

13 Detailed Description of the Animation Framework eXtension (AFX)

The Animation Framework extension (AFX – pronounced 'effects') provides an integrated toolbox for building attractive and powerful synthetic MPEG-4 environments. The framework defines a collection of interoperable tool categories that collaborate to produce a reusable architecture for interactive animated contents. In the context of AFX, a tool represents functionality such as a BIFS node, a synthetic stream, or an audio-visual stream.

AFX utilizes and enhances existing MPEG-4 tools, while keeping backward-compatibility, by offering:

- Higher-level descriptions of animations (e.g. inverse kinematics)
- Enhanced rendering (e.g. multi-texturing, procedural texturing)
- Compact representations (e.g. piecewise curve interpolators, subdivision surfaces)
- Low bitrate animations (e.g. using interpolator compression and dead-reckoning)
- Scalability based on terminal capabilities (e.g. parametric surfaces tessellation)
- Interactivity at user level, scene level, and client-server session level
- Compression of representations for static and dynamic tools

Compression of animated paths and animated models is required for improving the transmission and storage efficiency of representations for dynamic and static tools.

AFX defines a hierarchy made of 6 categories of models that rely on each other. Each model may have many tools. For example, BIFS tools prior to this specification belong to the lowest category of models of this pyramid. The 6 categories are:

1. *Geometric models.*

Geometric models capture the form and appearance of an object. Many characters in animations and games can be quite efficiently controlled at this low-level; familiar tools for generating motion include key framing, and motion capture. Due to the predictable nature of motion, building higher-level models for characters that are controlled at the geometric level is generally much simpler.

2. Modeling models.

These are an extension of geometric models and add linear and non-linear deformations to them. They capture transformation of the models without changing its original shape. Animations can be made on changing the deformation parameters independently of the geometric models.

3. Physical models.

They capture additional aspects of the world such as an object's mass inertia, and how it responds to forces such as gravity. The use of physical models allows many motions to be created automatically and with unparallel realism. The cost of simulating the equations of motion may be important in a real-time engine and in many games, a physically plausible approach is often preferred. Applications such as collision restitution, deformable bodies, and rigid articulated bodies use these models intensively.

4. Biomechanical models.

Real animals have muscles that they use to exert forces and torques on their own bodies. If we already have built physical models of characters, they can use virtual muscles to move themselves around. In his pioneering work on Artificial Intelligence for games and animation,

John Funge refers to the character's ability to exert some control over their motions as actuated (or animate) objects. These models have their roots in control theory.

5. Behavioral models.

After simple locomotion comes a character's behavior. A character may expose a reactive behavior when its behavior is solely based on its perception of the current situation (i.e. no memory of previous situations). Reactive behaviors can be implemented using stimulus-response rules, which are very used in games. Finite-States Machines (FSMs) are often used to encode deterministic behaviors based on multiple states. Goal-directed behaviors can be used to define a cognitive character's goals. They can also be used to model flocking behaviors.

6. Cognitive models.

If the character is able to learn from stimuli from the world, it may be able to adapt its behavior. These models are related to artificial intelligence techniques.

The models are hierarchical; each level relies on the next-lower one. For example, an autonomous agent (category 5) may respond to stimuli from the environment he is in and may decide to adapt his way of walking (category 4) that can modify physics equation (for example skin modeled with mass-spring-damp properties) or have influence on some underlying deformable models (category 2) or may even modify the geometry (category 1). If the agent is clever enough, it may also learn from the stimuli (category 6) and adapt or modify his behavioral models.

14 Annexes

A The MPEG-4 development process

The Moving Picture Coding Experts Group (MPEG) is a working group of ISO/IEC in charge of the development of international standards for compression, decompression, processing, and coded representation of moving pictures, audio and their combination.
The purpose of MPEG is to produce standards. The first two standards produced by MPEG were:

- **MPEG-1**, a standard for storage and retrieval of moving pictures and audio on storage media (officially designated as ISO/IEC 11172, in 5 parts)
- **MPEG-2**, a standard for digital television (officially designated as ISO/IEC 13818, in 9 parts).

MPEG is has finalized MPEG-4 Version 1 in November 1998, and it reached the status of International Standard a few months later.
MPEG-7, a content representation standard for information search, was completed in its first version in Fall 2001. The Call for Proposals was issued in October 1998.
MPEG-21, a Multimedia Framework standard, is being completed in phases. The first parts are almost done, while new work items are continuously being added.
MPEG-1 has been a very successful standard. It is the de-facto form of storing moving pictures and audio on the World Wide Web and is used in millions of Video CDs.
MPEG-2 has been the timely response for the satellite broadcasting and cable television industries in their transition from analogue to digital. Hundreds of millions of set-top boxes incorporating MPEG-2 decoders have been sold in the last 6 years.

This is the MPEG-4 work plan:

Part	Title	WD	CD	FCD	FDIS	IS
1	Systems		97/11	98/03	98/10	99/04
1/Amd x	Systems v.2	97/10	99/03	99/07	99/12	00/02
1/Amd 1	Extended BIFS			00/03	01/01	01/03
1/Amd 2	Textual Format			01/03	02/03	02/05
1/Amd 3	IPMP specific tools		01/07	02/03	02/07	02/09
1/Amd 4	AFX and MUW		01/12	02/05	02/10	02/12
1/Amd 5	MP4 Common Text		01/12	02/03	02/07	02/09

1/Amd 6	MP4 MPEG-4 Specific Text		01/12	02/03	02/07	02/09
2	Visual		97/11	98/03	98/10	99/04
2/Amd x	Visual v.2	97/10	99/03	99/07	99/12	00/02
2/Amd 1	Studio Profile		00/03	00/07	01/01	01/03
2/Amd 2	Streaming Video Profiles		00/03	00/07	01/01	01/03
2/Amd 3	New levels and tools	01/07	01/12	02/03	02/10	02/12
3	Audio		97/11	98/03	98/10	99/04
3/Amd x	Audio v.2	97/10	99/03	99/07	99/12	00/02
3/Amd 1	Audio Compatible Extensions	01/12	02/05	02/10	03/03	03/05
4	Conformance Testing	97/10	98/12	99/07	99/12	00/02
4/Amd 1	Conformance Testing Extensions (v.2)	98/12	99/12	00/07	01/01	01/03
5	Reference Software		97/11	98/03	99/03	99/05
5/Amd 1	Reference Software Extensions	97/10	99/07	99/12	01/01	01/03
6	Delivery Multimedia Integration Framework (DMIF)	97/07	97/11	98/03	98/10	99/04
6/Amd 1	DMIF v.2	97/10	99/03	99/07	99/12	00/08
7	Optimized Software for MPEG-4 tools (video encoders)				1/12	02/02
8	4 on IP framework		01/03	01/07	01/12	02/02
9	Reference Hardware Description	t.b.d.				
10	Advanced Video Coding	01/12	02/05	02/07	02/12	03/02

AMD stands for Amendment. The amendment numbering changes as new editions are issued, it is always relative to the latest version of the part in question. Past amendments that have been folded into the most recent editions are denoted by 'Amd x'

Table 1 - MPEG-4 work plan (The abbreviations are explained below)

Because of the complexity of the work item, it took 2 years before a satisfactory definition of the scope could be achieved and half a year more before a first call for proposals was issued. This call, like all MPEG calls, was open to all interested parties, no matter whether they were within or outside of MPEG. It requested technology that proponents felt could be considered by MPEG for the purpose of the developing the MPEG-4 standard. After that first call, other calls were issued for other technology areas.
The proposals of technology received were assessed and, if found promising, incorporated in the so-called Verification Models (VMs). A VM describes, in text and some sort of programming language, the operation of encoder and decoder. VMs are used to carry out simulations with the
aim to optimize the performance of the coding schemes.
Because (next to the envisaged hardware environments for MPEG-4) software platforms are gaining in importance for multimedia standards, MPEG decided to maintain software implementations of the different standard parts. These can be used for the purpose of development of the standard and for commercial implementations of the standard. At the Maceió meeting in November '96 MPEG reached sufficient confidence in the stability of the standard under development, and produced the Working Drafts (WDs). Much work has been done since, resulting in the production of Committee Drafts (CD), Final Committee Drafts (FCD) and finally Final Drafts of International Standard (FDIS), in Atlantic City, October 1998.
The WDs already had the structure and form of a standard but they were kept internal to MPEG for revision. Starting from the Seville meeting in February '97, MPEG decided to publish the WDs to seek first comments from industry. The CD underwent a formal ballot by national Bodies and the same happened to the FCD. This applies to all parts of the standard, although 'Conformance' is scheduled for later completion than the other parts.

Ballots by NBs are usually accompanied by technical comments. These ballots were considered at the March '98 meeting in Tokyo for CD and in Atlantic City (October 1998) for the FCD. This process entailed making changes. Following the Atlantic City meeting, the standard was sent out for a final ballot, where NBs could only cast a yes/no ballot, without comments, within two months. After that, the FDIS became International Standard (IS) and was sent to the ISO Central Secretariat for publication. A similar process was followed for the amendments that form MPEG-4 Version 2. In December '99, at the Maui meeting, MPEG-4 Version 2 has acquired the status of FDIS. After that, several extensions were added to the standard, and the concept of 'Version' became less used.

B Organization of work in MPEG

Established in 1988, MPEG has grown to form an unusually large committee. Some 300 to 400 experts take part in MPEG meetings, and the number of people working on MPEG-related matters without attending meetings is even larger.
The wide scope of technologies considered by MPEG and the large body of available expertise, require an appropriate organization. Currently MPEG has the following subgroups:

1. Requirements	Develops requirements for the standards under development

	(currently, MPEG-4 and MPEG-7).
2. Systems	Develops standards for the coding of the combination of individually coded audio, moving images and related information so that the combination can be used by any application.
3. Video	Develops standards for coded representation of moving pictures of natural origin.
4. Audio	Develops standards for coded representation of audio of natural origin.
5. SNHC (Synthetic- Natural Hybrid Coding)	Develops standards for the integrated coded representation of audio and moving pictures of natural and synthetic origin. SNHC concentrates on the coding of synthetic data.
6. Multimedia Description	Develops Structures for multimedia descriptions. This group only works for MPEG-7,
7. Test	Develops methods for and the execution of subjective evaluation tests of the quality of coded audio and moving pictures, both individually and combined, to test the quality of moving pictures and audio produced by MPEG standards
8. Implementation	Evaluates coding techniques so as to provide guidelines to other groups upon realistic boundaries of implementation parameters.
9. Liaison	Handles relations with bodies external to MPEG.
10. HoD (Heads of Delegation)	The group, consisting of the heads of all national delegations, acts in advisory capacity on matters of general nature.

Work for MPEG takes place in two different instances. A large part of the technical work is done at MPEG meetings, usually lasting one full week. Members electronically submit contributions to the MPEG FTP site (several hundreds of them at every meeting). Delegates are then able to come to meetings well prepared without having to spend precious meeting time to study other delegates' contributions.
The meeting is structured in 3 plenary meetings (on Monday morning, on Wednesday morning and on Friday afternoon) and in parallel subgroup meetings.
About 100 output documents are produced at every meeting; these capture the agreements reached. Documents of particular importance are:

- Drafts of the different parts of the standard under development;
- New versions of the different Verification Models, that are used to develop the respective parts of the standard.
- "Resolutions", which document the outline of each agreement and make reference to the documents produced;
- "Ad-hoc groups", groups of delegates agreeing to work on specified issues, usually until the following meeting;

Output documents are also stored on the MPEG FTP site. Access to input and output documents is restricted to MPEG members. At each meeting, however, some output documents are released for public use. These can be accessed from the home page: www.cselt.it/mpeg
Equally important is the work that is done by the ad-hoc groups in between two MPEG meetings. They work by e-mail under the guidance of a Chairman appointed at the Friday (closing) plenary meeting. In some exceptional cases, they may hold physical meetings. Ad-hoc groups produce recommendations that are reported at the first plenary of the MPEG week and function as valuable inputs for further deliberation during the meeting.

C Glossary and Acronyms

AAC	Advanced Audio Coding
AAL	ATM Adaptation Layer
Access Unit	A logical sub-structure of an Elementary Stream to facilitate random access or bitstream manipulation
ACE	Advanced Coding Efficiency (Profile)
AFX	Animation Framework eXtension
Alpha plane	Image component providing transparency information
Amd	Amendment
AOI	Area Of Interest
API	Application Programming Interface
ARTS	Advanced Real-time Simple (Profile)
ATM	Asynchronous Transfer Mode
BAP	Body Animation Parameters
BAP	Body Animation Parameter
BDP	Body Definition Parameters
BDP	Body Definition Parameter
BIFS	Binary Format for Scenes
BSAC	Bit-Sliced Arithmetic Coding
CD	Committee Draft
CE	Core Experiment
CELP	Code Excited Linear Prediction
CIF	Common Intermediate Format
CNG	Comfort Noise Generator
DAI	DMIF-Application Interface
DCT	Discrete Cosine Transform
DMIF	Delivery Multimedia Integration Framework
DNI	DMIF Network Interface
DRC	Dynamic Resolution Conversion
DS	DMIF signaling
EP	Error Protection
ER	Error Resilient
ES	Elementary Stream: A sequence of data that originates from a single producer in the transmitting MPEG-4 Terminal and terminates at a single recipient, e.g. an media object or a Control Entity in the receiving MPEG-4 Terminal. It flows through one FlexMux Channel.
FAP	Facial Animation Parameters
FBA	Facial and Body Animation
FCD	Final Committee Draft
FDIS	Final Draft International Standard
FDP	Facial Definition Parameters
FlexMux stream	A sequence of FlexMux packets associated to one or more FlexMux Channels flowing through one Trans-Mux Channel
FlexMux tool	A Flexible (Content) Multiplex tool
FSM	Finite-States Machine
FTTC	Fiber To The Curb
GMC	Global Motion Compensation

GSTN	General Switched Telephone Network
HCR	Huffman Codeword Reordering
HFC	Hybrid Fiber Coax
HILN	Harmonic Individual Line and Noise
HTTP	HyperText Transfer Protocol
HVXC	Harmonic Vector Excitation Coding
IP	Internet Protocol
IPI	Intellectual Property Identification
IPMP	Intellectual Property Management and Protection
IPR	Intellectual Property Rights
IS	International Standard
ISDN	Integrated Service Digital Network
LAR	Logarithmic Area Ratio
LATM	Low-overhead MPEG-4 Audio Transport Multiplex:
LC	Low Complexity
LOAS	Low Overhead Audio Stream
LOD	Level Of Detail
LPC	Linear Predictive Coding
LSP	Line Spectral Pairs
LTP	Long Term Prediction
M4IF	MPEG-4 Industry Forum
MCU	Multipoint Control Unit
Mdat	media data atoms
Mesh	A graphical construct consisting of connected surface elements to describe the geometry/shape of a visual object.
MIDI	Musical Instrument Digital Interface
MPEG	Moving Pictures Experts Group
MPEG-J	Framework for MPEG Java API's
MSB	Most Significant Bits
NB	National Body
OCI	Object Content Information
OD	Object descriptor
OD	Object Descriptor
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PSNR	Peak Signal to Noise Ratio
QCIF	Quarter Common Intermediate Format
QoS	Quality of Service
Rendering	The process of generating pixels for display
RTP	Real Time Transport Protocol
RTSP	Real Time Streaming Protocol
RVLC	Reversible Variable Length Coding
SA-DCT	shape-adaptive DCT
SID	Silence Insertion Descriptor
SL	Sync(hronization) layer
SMIL	Synchronized Multimedia Integration Language
SNHC	Synthetic- Natural Hybrid Coding
SNR	Signal to Noise Ratio
Sprite	A static sprite is a - possibly large - still image, describing panoramic background.

SRM	Session Resource Manager
SVG	Scalable Vector Graphics
T/F coder	Time/Frequency Coder
TCP	Transmission Control Protocol
TransMux	generic abstraction for any transport multiplex scheme
TTS	Text-to-speech
UDP	User Datagram Protocol
UEP	Unequal Error Protection
UMTS	Universal Mobile Telecommunication System
VCB	Virtual CodeBook
Viseme	Facial expression associated to a specific phoneme
VLBV	Very Low Bitrate Video
VM	Verification Model
VOP	Video Object Plane
VRML	Virtual Reality Modeling Language
W3C	World Wide Web Consortium
WD	Working Draft
WWW	World Wide Web
XMT	Extensible MPEG-4 textual format