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

ISO/IEC JTC1/SC29/WG11 N2564
December 1998/Roma

Source: **MPEG Subgroups: Requirements, Audio, Delivery, SNHC, Systems, Video, Test**
Status: **Final**
Title: **MPEG-4 Overview - (Roma Version)**
Editor: **Rob Koenen**

Overview of the MPEG-4 Standard

Executive Overview

MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group), the committee which also developed the Emmy Award winning standards known as MPEG-1 and MPEG-2. These standards made interactive video on CD-ROM and Digital Television possible. MPEG-4 is be the result of another international effort involving hundreds of researchers and engineers from all over the world. MPEG-4, whose formal ISO/IEC designation will be ISO/IEC 14496, was finalized in October 1998 and will be an International Standard in the first months of 1999. Currently, MPEG is working on MPEG-4 Version 2, a fully backward compatible extension of Version 1.

MPEG-4 is building 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 will provide 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://cselt.it/mpeg` . 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.

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

Executive Overview 1
Table of Contents 2
1. Scope and features of the MPEG-4 standard 3
1.1 Coded representation of media objects 3
1.2 Composition of media objects 4
1.3 Description and synchronization of streaming data for media objects 5
1.4 Delivery of streaming data 6
1.5 Interaction with media objects 7
1.6 Management and Identification of Intellectual Property 7
2. Detailed technical description of the MPEG-4 standard 7
2.1 DMIF 8
2.2 Demultiplexing, synchronization and description of streaming data 12
2.3 Syntax Description 15
2.4 Coding of Audio Objects 16
2.5 Coding of Visual Objects 19
2.6 Scene description 29
2.7 User interaction 30
2.8 Content-related IPR identification and protection 31
2.9 Object Content Information 32
3. List of major functionalities provided by MPEG-4 in Version 1 32
3.1 DMIF 32
3.2 Systems 32
3.3 Audio 33
3.4 Visual 34
4. Verification Test: checking MPEG's performance 36
5. Profiles in MPEG-4 Version 1 36
5.1 Visual Profiles 37
5.2 Audio Profiles 37
5.3 Graphics Profiles 38
5.4 Scene Description Profiles 38
5.5 Object Descriptor Profile 39
6. Version 2 of MPEG-4: Ready one year after Version 1 39
6.1 Systems 39
6.2 Visual 41
6.3 Audio 43
6.4 DMIF 43
7. Annexes 45
Annex A - The MPEG-4 development process 45
Annex B - Organization of work in MPEG 46
Annex C - Glossary and Acronyms 47

1. Scope and features of the MPEG-4 standard

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

- For *authors*, MPEG-4 will enable 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 will be possible to better manage and protect content owner rights.
- For *network service providers* MPEG-4 will offer transparent information which will be interpreted and translated into the appropriate native signaling messages of each network with the help of relevant standards bodies. However the foregoing excludes Quality of Service considerations, for which MPEG-4 will provide 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 be defined by network providers. Signaling of the MPEG-4 media QoS descriptors end-to-end, will enable transport optimization in heterogeneous networks.
- For *end users*, MPEG-4 will enable many functionalities which could potentially be accessed on a single compact terminal and higher levels of interaction with content, within the limits set by the author. An MPEG-4 applications document exists which describes many end user applications including, among others, real time communications, surveillance and mobile multimedia.

For all parties involved, MPEG wants to avoid the emergence of 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

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);
- etc.

MPEG 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;
- synthetic sound

A media object in its coded form consists of descriptive elements that allow to handle 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 gives an example that highlights 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 a 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. moving texture belonging to an object; animation parameters animating a moving head);
- change, interactively, the user's viewing and listening points anywhere in the scene.

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

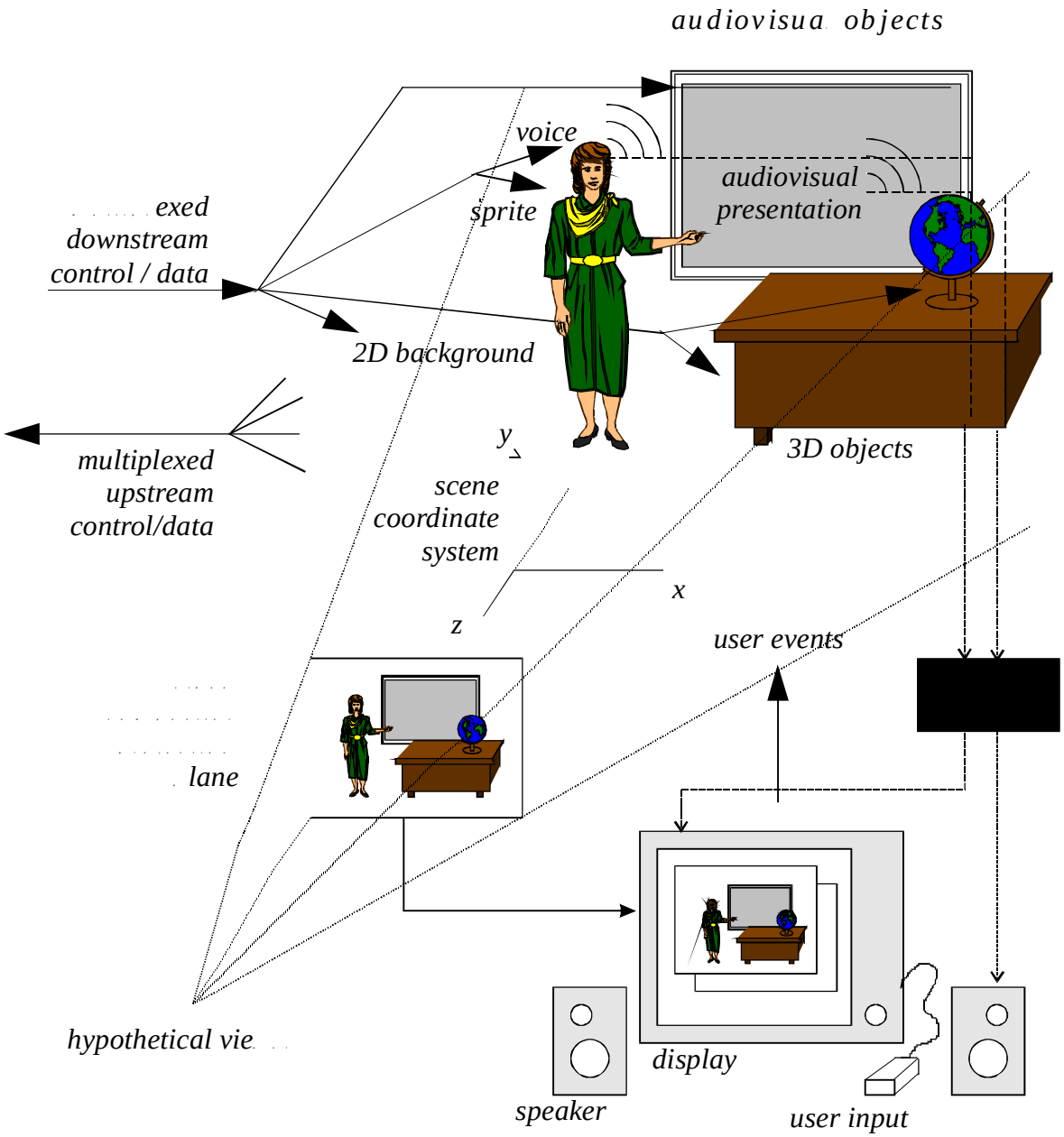


Figure 1 - an example of an MPEG-4 Scene

1.3 Description and synchronization of streaming data for media objects

Media objects may rely on streaming data that is conveyed in one or more elementary streams. All streams associated to one media object are identified by an object descriptor. This allows handling hierarchically encoded data as well as the association of meta information about the content (object content information) and the intellectual property rights associated with it.

Each stream is characterized itself by a set of descriptors conveying configuration information, e.g., to determine the required decoder resources and the precision of encoded timing information. Furthermore the descriptors may convey 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 identification of such access units and the time stamping is accomplished by the synchronization layer. Independent of the media type, this layer allows identification of access units (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 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 aforementioned 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. 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.

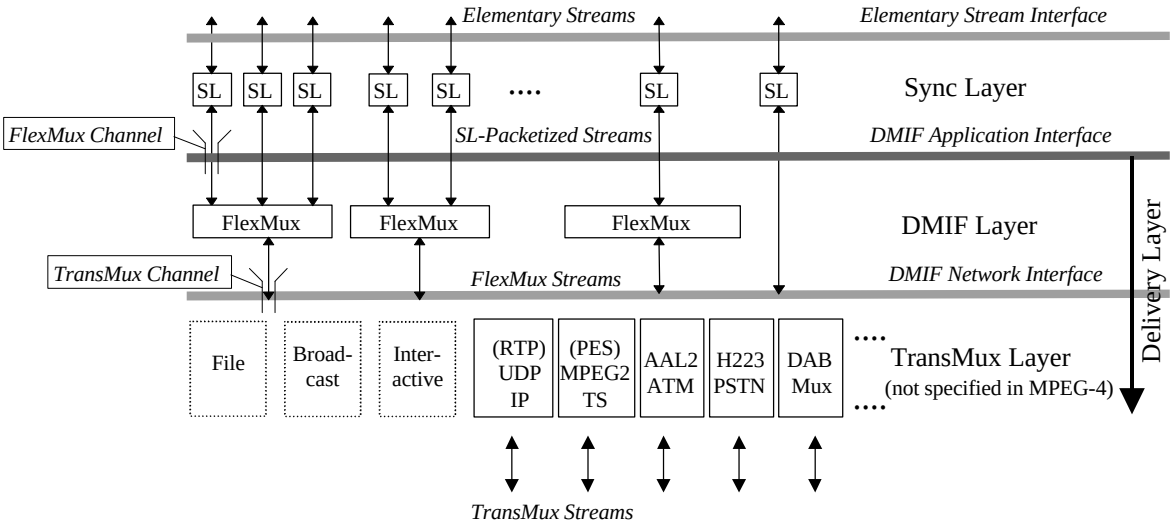


Figure 2 - The MPEG-4 System Layer Model

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 will be possible to:
1. identify access units, transport timestamps and clock reference information and identify data loss.
2. optionally interleave data from different elementary streams into FlexMux streams
3. 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

Part of the control functionalities will be 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 being coded into MPEG-4 media objects. Therefore, MPEG works 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, that 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 people who want to use systems that control access to intellectual property a standardized interface that is integrated tightly into the Systems layer. With this interface, proprietary control systems can be easily amalgamated with the standardized part of the decoder.

2. Detailed technical description of the MPEG-4 standard

As shown in Figure 3, 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. This is described in Section 2.2. The Elementary Streams (ESs) are

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

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 2.4 and 2.5. 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 is explained in Section 2.6, and Composition in Section 2.7. The user can, to the extent allowed by the author, interact with the scene which is eventually rendered and presented. Section 2.8 describes this interaction.

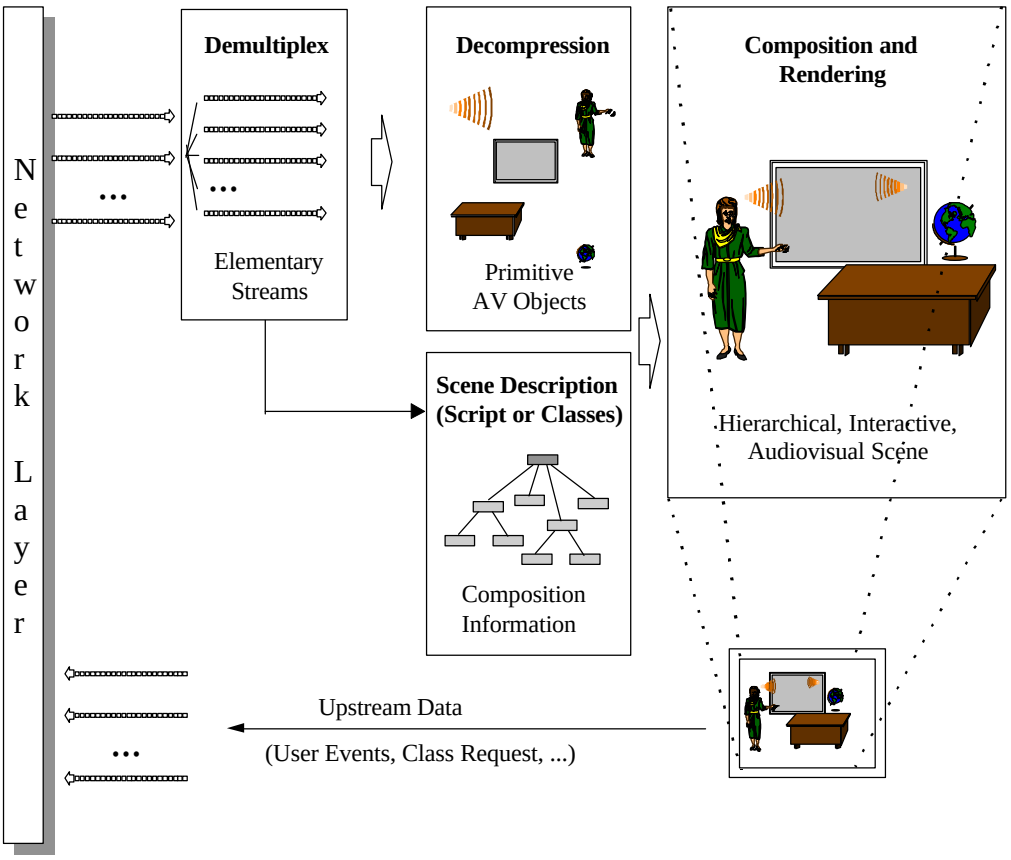


Figure 3- Major components of an MPEG-4 terminal (receiver side)

2.1 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 (but 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 establishes the connection themselves).

Compared to FTP, DMIF is both a framework and a protocol. The functionality provided by DMIF are expressed by an interface called DAI (DMIF-Application Interface), 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 (including 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 4 below.

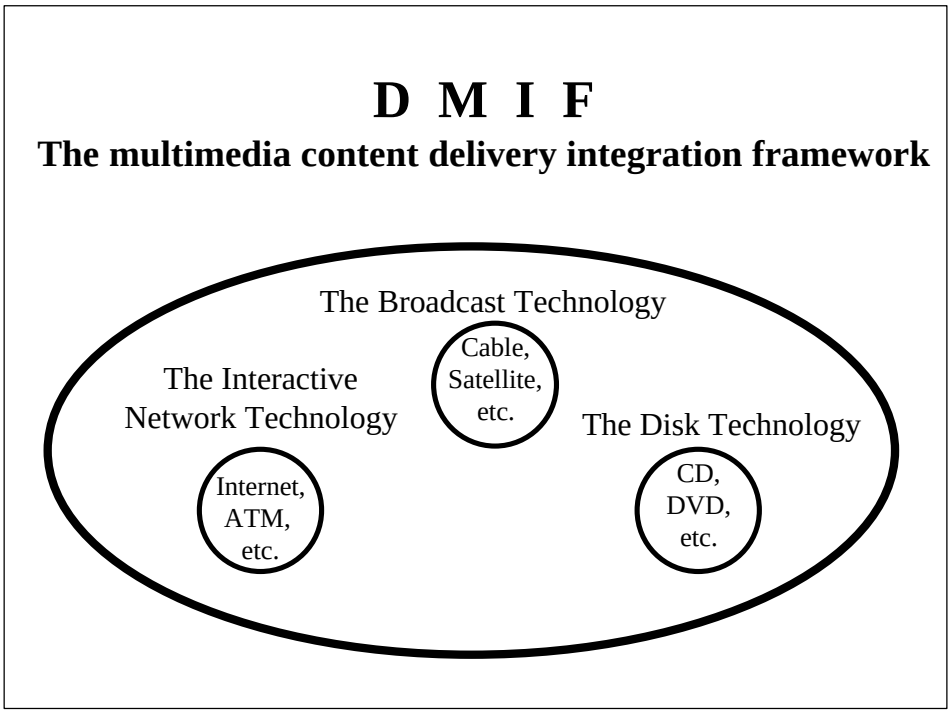


Figure 4 - DMIF addresses the delivery integration of three major technologies

The DMIF architecture is such that applications which 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 4 represents the above concept. An application accesses data through the DMIF-Application Interface, irrespectively 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 than 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. However in the former case the messages exchanged between the two entities have to be normatively defined to ensure interoperability (these are the DMIF Signalling messages), while in the latter case 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 scenario 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).

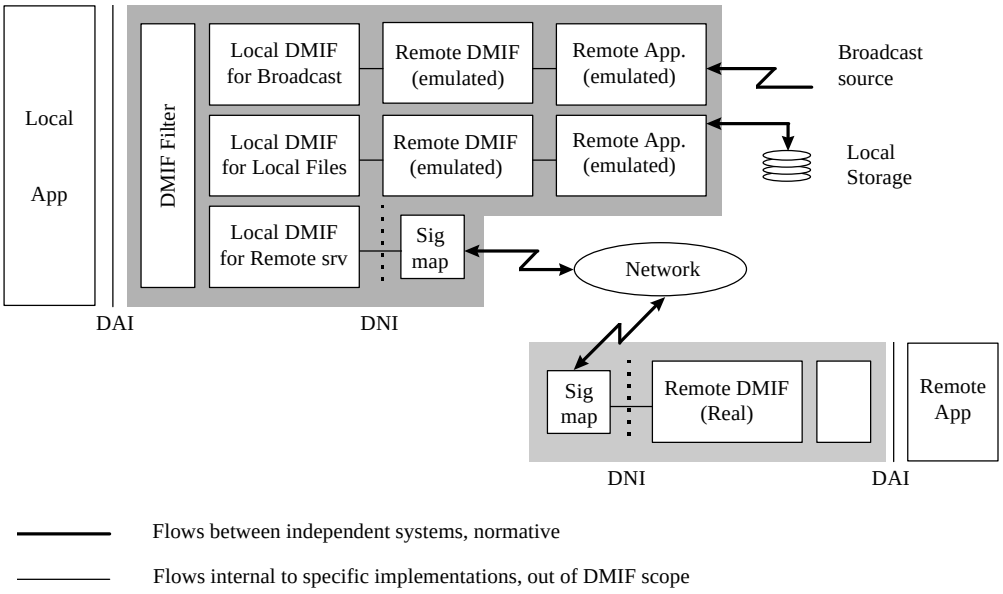


Figure 5 - DMIF communication architecture

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 (”Signalling mapping” in the figure) takes care of mapping the DNI primitives into signalling 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 5 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

2.1.1 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 6 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.

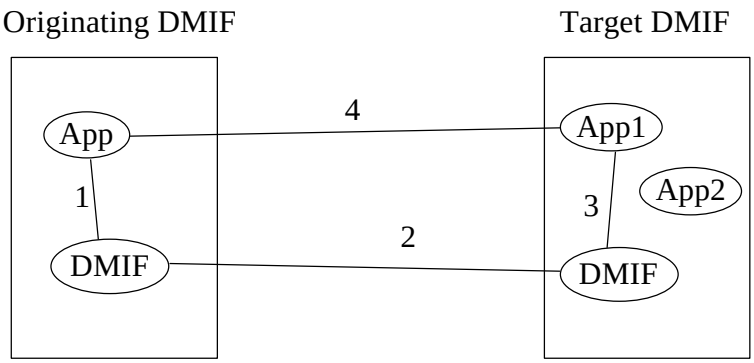


Figure 6 — DMIF Computational Model

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.

2.2 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-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 described 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 which 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.

2.2.1 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 links 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 process has been started.

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.

2.2.2 Synchronization and description of elementary streams

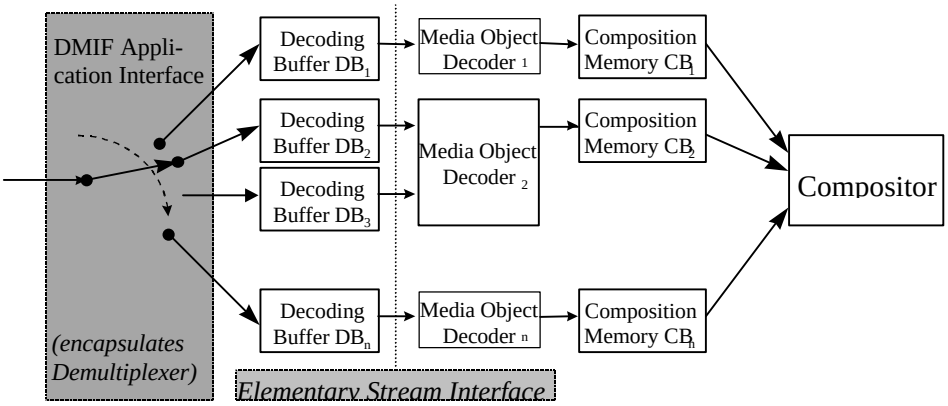


Figure 7 - Buffer architecture of the System Decoder Model

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 presentation. 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 spatiotemporal layout of the presentation over time.

2.2.3 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.

2.2.4 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 be 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 (relative reduction or extension of their time scale).

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

2.3 Syntax Description

MPEG-4 defines a *syntactic description language* to describe the exact binary syntax of a bit stream conveying data for a media object representation as well as that of the scene description information. This is a departure from MPEG's traditional 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.

2.4 Coding of Audio Objects

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. 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.

2.4.1 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 will provide 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 8.

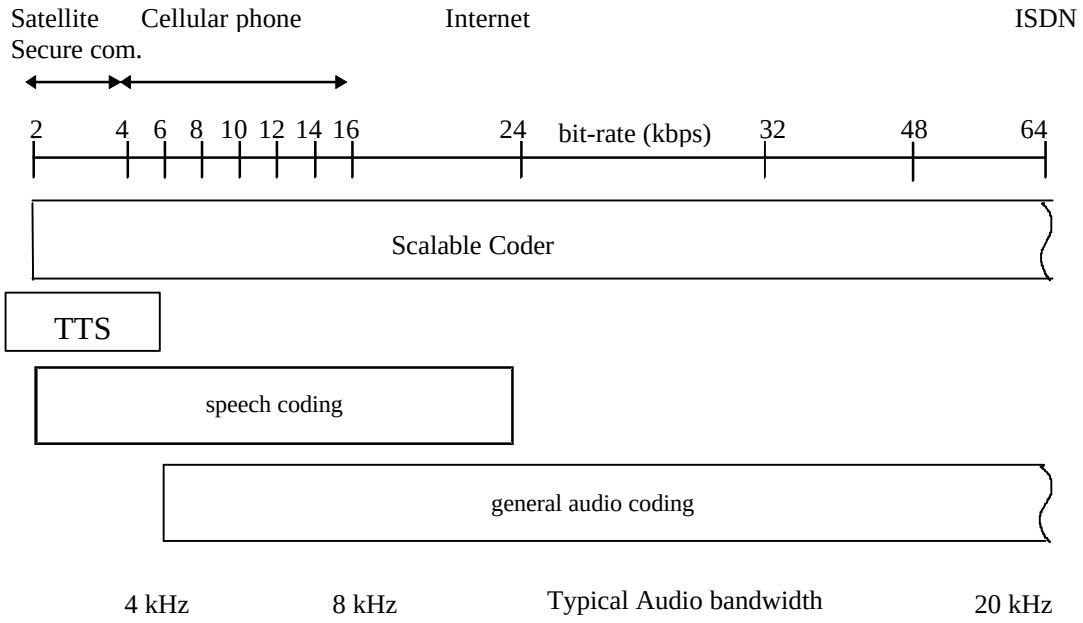


Figure 8 - General block diagram of MPEG-4 Audio

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 bit stream 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 facilitates the use, and signaling, of different tools , and thus also codecs according to existing standards, e.g. MPEG-2 AAC, can be accommodated. 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 a functionality within an individual coder is speed or pitch change within HVXC.

2.4.2 Synthesized Sound

Decoders are also available for generating sound based on 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 which 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) rather than a normative TTS synthesizer itself.

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 method” of synthesis, but rather a method of describing 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

which 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.

2.5 Coding of Visual Objects

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

2.5.1 Natural Textures, Images and Video

The tools for representing natural video in the MPEG-4 visual standard aim at providing standardized core technologies allowing efficient storage, transmission and manipulation of textures, images and video data for multimedia environments. These tools will 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 2D and 3D meshes
- efficient compression of implicit 2D 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

The visual part of the MPEG-4 standard will provide a toolbox containing tools and algorithms bringing solutions to the above mentioned functionalities and more.

2.5.2 Synthetic Objects

Synthetic objects form a subset of the larger class of computer graphics, as an initial focus the following visual synthetic objects will be described:

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

2.5.2.1 facial animation

The face is an object capable of facial geometry ready for rendering and animation. The shape, texture and expressions of the face are generally controlled by the bitstream containing instances of Facial Definition Parameter (FDP) sets and/or Facial Animation Parameter (FAP) sets. Upon construction, the Face object contains a generic face with a neutral expression. This face can already be rendered. It is also immediately capable of receiving the animation parameters from the bitstream, which will produce animation of the face: expressions, speech etc. If definition parameters are received, they are used to transform the generic face into a particular face determined by its shape and (optionally) texture. If so desired, a complete face model can be downloaded via the FDP set as a scene graph for insertion in the face node.

Face Animation in MPEG-4 Version 1 provides for highly efficient coding of animation parameters whose decoding 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. Depending on the application, Facial Animation can be used with or without the MPEG-4 Systems layer.

The Systems Binary Format for Scenes (BIFS) 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.

2.5.2.2 body animation

Body Animation, to be standardised in MPEG-4 Version 2, is being designed into the MPEG-4 fabric to work in a thoroughly integrated way with face/head animation. Decoding and scene description for Body Animation directly mirror technology already proven in Face Animation.

2.5.2.3 2D animated meshes

A 2D *mesh* is a tessellation (or partition) of a 2D 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 2D dynamic mesh refers to 2D mesh geometry and motion information of all mesh node points within a temporal segment of interest. Triangular meshes have long been used for efficient 3D object shape (geometry) modeling and rendering in computer graphics. 2D mesh modeling may be considered as projection of such 3D triangular meshes onto the image plane. An example of a 2D mesh is depicted in Figure 9.

Figure 9- 2D mesh modeling of the "Akiyo" video object

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* . 2D 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 2D 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, and 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 2D 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.

The attractiveness of 2D mesh modeling originates from the fact that 2D meshes can be designed from a single view of an object without requiring range data, while maintaining several of the functionalities offered by 3D mesh modeling. In summary, the 2D 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 3D object modeling when data to construct such models is available. In particular, the 2D mesh representation of video objects enables the following functionalities:

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).

Video Object Compression

- 2D 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 2D mesh information.

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.

2.5.2.4 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 3D 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 Wavelet and DCT based encoders.

2.5.3 Structure of the tools for representing natural video

The MPEG-4 image and video coding algorithms will give an efficient representation of visual objects of arbitrary shape, with the goal to support so-called content-based functionalities. Next to this, it will support most functionalities already provided by MPEG-1 and MPEG-2, including the provision to efficiently compress 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 10 below, with the attempt to cluster bit-rate levels versus sets of functionalities.

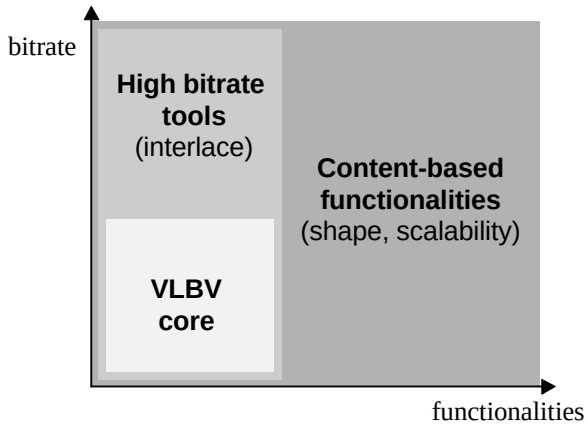


Figure 10 - Classification of the MPEG-4 Image and Video Coding Algorithms and Tools

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) VLBV 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) provisions for “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 - employing identical or similar algorithms and tools as the VLBV Core. The bit-rates envisioned range typically from 64 kbits/s up to 4 Mb/s and applications envisioned include broadcast or interactive retrieval of signals with a quality comparable to digital TV. For these applications at higher bit-rates, tools for coding interlaced signals are specified in MPEG-4.

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 background.

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 HBV Cores are complemented by additional elements.

2.5.4 Support for Conventional and Content-Based Functionalities

The MPEG-4 Video standard will support the decoding of conventional rectangular images and video as well as the decoding of images and video of arbitrary shape. This concept is illustrated in Figure 11 below.

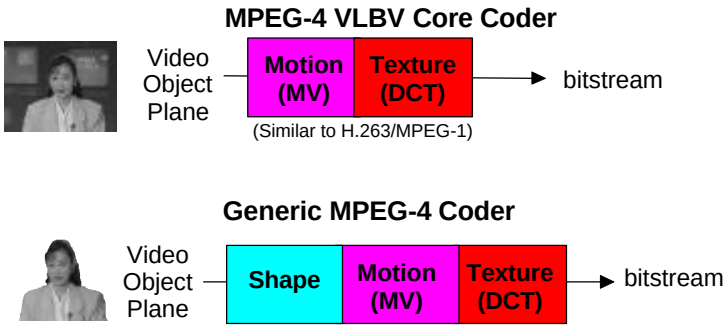


Figure 11 - the VLBV Core and the Generic MPEG-4 Coder

The coding of conventional images and video is achieved similar to conventional MPEG-1/2 coding and 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.

2.5.5 The MPEG-4 Video Image and Coding Scheme

Figure 12 below outlines the basic approach of the MPEG-4 video algorithms to encode rectangular as well as arbitrarily shaped input image sequences.

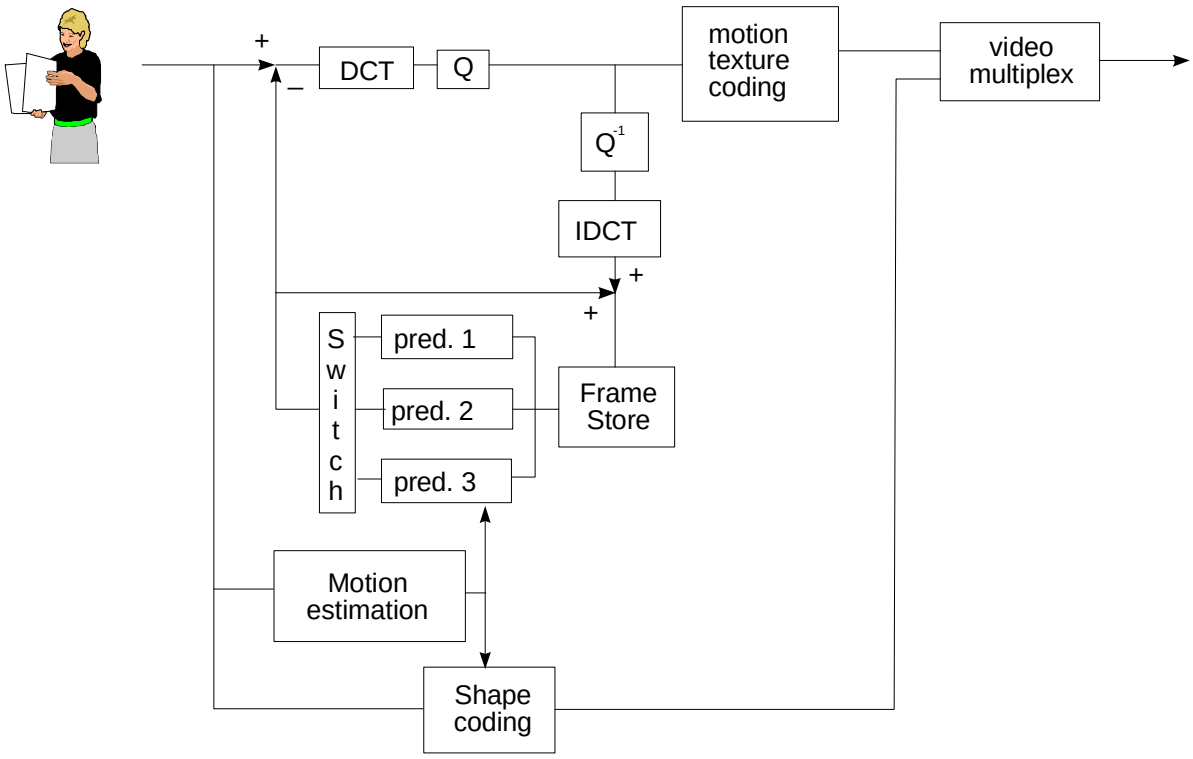


Figure 12 - Basic block diagram of MPEG-4 Video Coder

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 taken by 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.
- Global motion compensation based on the transmission of a static “sprite”. 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.

Figure 13 depicts the basic concept for coding a MPEG-4 video sequence using a sprite panorama image. It is assumed that the foregound 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.

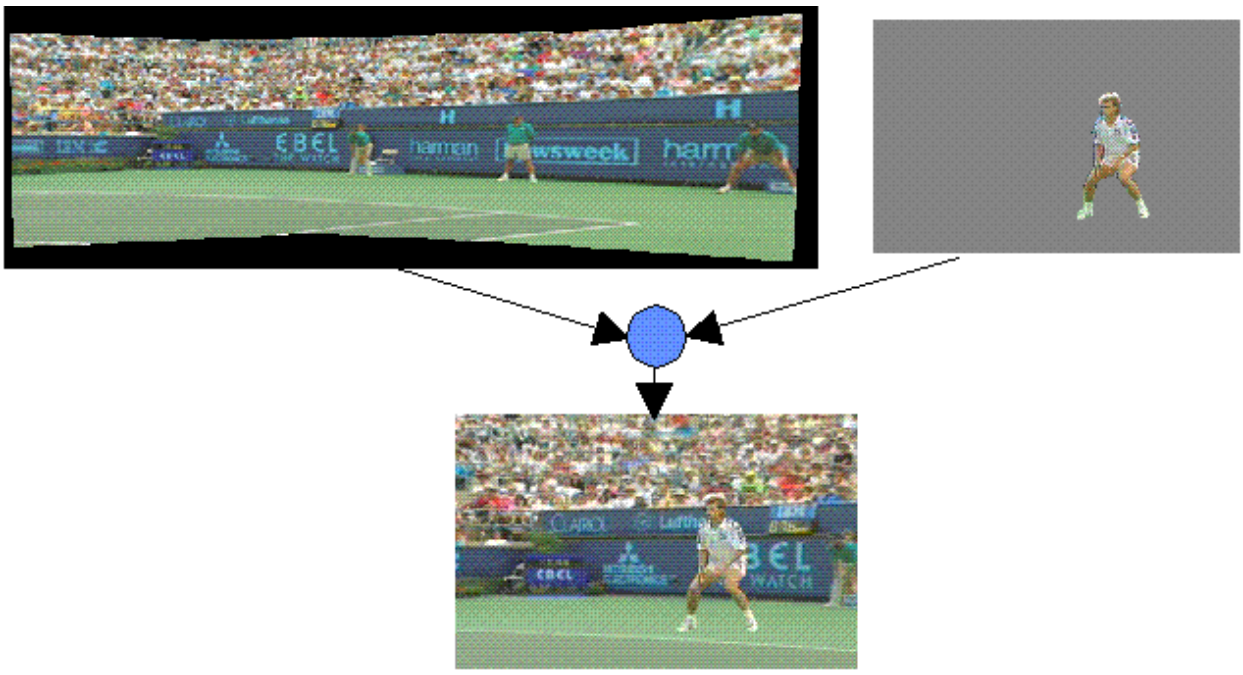


Figure 13 - Example of Sprite Coding of Video Sequence

2.5.6 Coding of Textures and Still Images

Efficient Coding of visual textures and still images 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 3D 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 2D and 3D virtual worlds.

2.5.7 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 bit stream 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 over heterogeneous networks, as well as for applications where the receiver is not willing or 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.

For decoding of still images, the MPEG-4 standard will provide spatial scalability with up to 11 levels of granularity and also quality scalability up to the bit level. For video sequences a maximum of 3 levels of granularity will be supported.

2.5.8 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 Kbps).

The error resilience tools developed for MPEG-4 can be divided into three major areas. These areas or categories include 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.

2.5.8.1 Resynchronization

Resynchronization tools, as the name implies, 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 which 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 distinguished 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 to avoid 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.

2.5.8.2 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 which 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 14. 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.

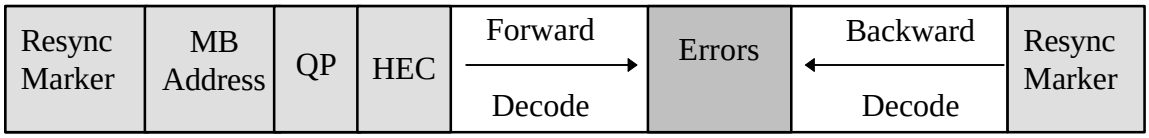


Figure 14 - example of Reversible Variable Length Code

2.5.8.3 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, 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.

2.6 Scene description

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.

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 15 (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.

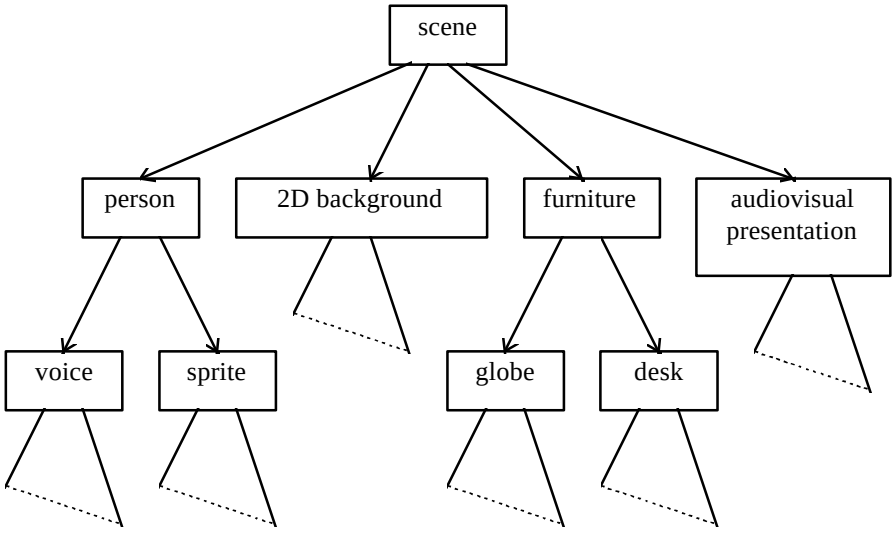


Figure 15- Logical structure of a scene

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: 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.

2.7 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 mous 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.

2.8 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 16)

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.

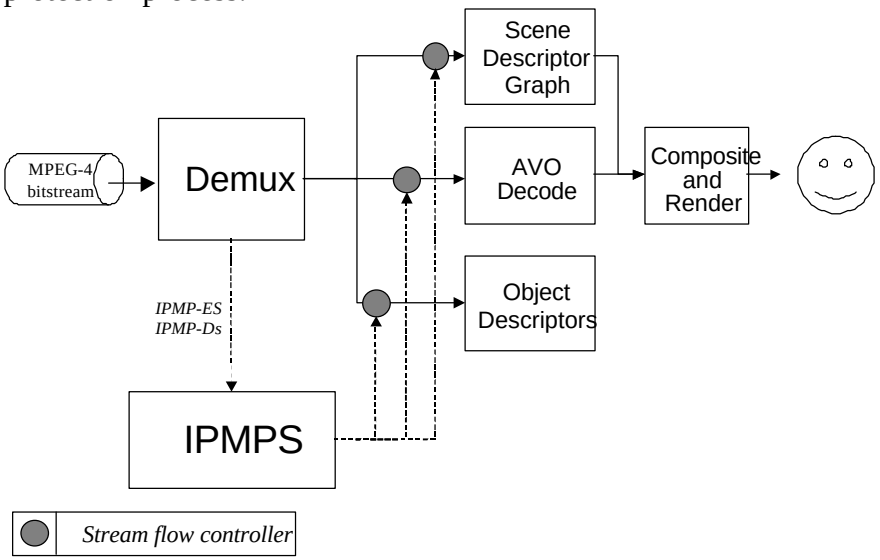


Figure 16 - The IPMP Interfaces within an MPEG-4 System

2.9 Object Content Information

MPEG-4 will allow attaching information to objects about their content. Users of the standard can use this 'OCI' datastream 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.

3. List of major functionalities provided by MPEG-4 in Version 1

This section contains, in an itemized fashion, the major functionalities that the different parts of the MPEG-4 Standard will offer in MPEG-4 Version 1. Description of the functionalities can be found above.

3.1 DMIF

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.

3.2 Systems

- Scene description for composition (spatio-temporal synchronization with time response behavior) of multiple media objects. The scene description provides a rich set of nodes for 2D and 3D composition operators and graphics primitives.
- Text with international language support, font and font style selection, timing and synchronization.

- 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.
- A tool for interleaving of multiple streams into a single stream, including timing information (FlexMux tool).
- 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 standardisation bodies.
- 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.

3.3 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:

- *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, arealso 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.
- *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. It includes the following functionalities.
    - 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.
- *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 but also includes broadcast quality audio from mono up to multichannel.
- *Synthesized Audio:* Synthetic Audio support is provided by a Structured Audio Decoder implementation that allows the application of score-based control information to musical instruments described in a special language.
- *Bounded-complexity Synthetic Audio:* This is provided by a Structured Audio Decoder implementation that allows the processing of a standardized wavetable format.

Examples of additional functionality are speed control and pitch change for speech signals and scalability in terms of bitrate, bandwidth, error robustness, complexity, etc. as defined below.

- The *speed change functionality* allows the change of the time scale without altering the pitch during the decoding process. This can, for example, be used to implement a “fast forward” function (data base search) or to adapt the length of an audio sequence to a given video sequence, or for practicing dance steps at slower play back speed.
- The *pitch change functionality* allows the change of the pitch without altering the time scale during the encoding or decoding process. This can be used, for example, for voice alteration or Karaoke type applications. This technique only applies to parametric and structured audio coding methods.
- *Bitrate scalability* allows a bitstream to be parsed into a bitstream of lower bitrate such that the combination can still be decoded into a meaningful signal. The bit stream 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.
- *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.
- *Audio Effects* provide the ability to process decoded audio signals with complete timing accuracy to achieve functions for mixing , reverberation, spatialization, etc.

3.4 Visual

The MPEG-4 Visual standard will allow the hybrid coding of natural (pixel based) images and video together with synthetic (computer generated) scenes. This will, for example, allow the virtual presence of videoconferencing participants. To this end, the Visual standard will comprise 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 the MPEG-4 visual standard will support.

3.4.1 Formats Supported

The following formats and bitrates will be supported:
- bitrates: typically between 5 kbit/s and 4 Mbit/s
- Formats: progressive as well as interlaced video
- Resolutions: typically from sub-QCIF to TV

3.4.2 Compression Efficiency

- Efficient compression of video will be supported for all bit rates addressed. 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.

3.4.3 Content-Based Functionalities

- *Content-based coding of images and video* to allow separate decoding and reconstruction of arbitrarily shaped video objects.
- *Random access of content in video sequences* to allow functionalities such as pause, fast forward and fast reverse of stored video objects.
- *Extended manipulation of content in video sequences* to allow 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.

3.4.4 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 bit stream generated by the encoder to reconstruct and display textures, images and video objects at reduced spatial resolution. For textures and still images, a maximum of 11 levels of spatial scalability will be supported. For video sequences, a maximum of three levels will be supported.
- *Temporal scalability* allows decoders to decode a subset of the total bit stream generated by the encoder to reconstruct and display video at reduced temporal resolution. A maximum of three levels will be supported.
- *Quality scalability* allows a bitstream to be parsed into a number of bit stream layers of different bitrate such that the combination of a subset of the layers can still be decoded into a meaningful signal. The bit stream 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.

3.4.5 Shape and Alpha Channel Coding

- *Shape coding* will be supported to assist 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 data bases, interactive games, surveillance, and animation. Efficient techniques are provided that allow efficient coding of binary shape. 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. 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. Efficient techniques are provided, that allow efficient coding of binary as well as gray scale alpha planes. A binary alpha map defines whether or not a pixel belongs to an object. It can be ‘on’ or ‘off’. A gray scale map offers the possibility to define the exact transparency of each pixel.

3.4.6 Robustness in Error Prone Environments

Error resilience will be supported to assist the access of 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). Tools are provided which address both the band limited nature and error resiliency aspects for access over wireless networks.

3.4.7 Face Animation

The 'Face Animation' part of the standard allow sending parameters that calibrate and animate synthetic faces. These models themselves are not standardized by MPEG-4, only the parameters are.

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

3.4.8 Coding of 2D Meshes with Implicit Structure

- Mesh-based prediction and animated texture transfiguration
  - 2D 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:
  - 2D mesh compression with implicit structure & decoder reconstruction

4. Verification Test: checking MPEG's performance

MPEG carries out verification tests to check whether the standard delivers what it promises.
The first formal tests on MPEG-4 Audio codecs were completed, based on collaboration between MPEG and the NADIB (Narrowband Digital Audio Broadcasting) Group. These tests explored the performance of speech and music coders working in the bitrate range 6 kb/s to 24 kb/s, including some scaleable codec options. The results show that a significant improvement in quality can be offered in relation to conventional analogue AM broadcasting and that scaleable coders offer superior performance to simulcast operations.

More verification tests are being prepared. For Video, error robustness, content-based coding and scalability will be evaluated. For Audio: tests are prepared for speech coding and audio on the Internet. The audio tests and the error robustness tests are to be ready in October, the other tests are scheduled for completion in December.

5. Profiles in MPEG-4 Version 1

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:

- The **Simple Visual Profile** provides efficient, error resilient coding of rectangular video objects, suitable for applications on mobile networks, such as PCS and IMT2000.
- 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.
- 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).
- 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.
- 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:

- 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.
- 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.
- The **Basic Animated 2D Texture Visual Profile** provides spatial scalability, SNR scalability, and mesh-based animation for still image (textures) objects and also simple face object animation.
- 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.

5.2 Audio Profiles

Four Audio Profiles have been defined:

- 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.
- 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.
- The **Scalable Profile,** a superset of the Speech Profile, is suitable for scalable coding of speech and music for networks, such as Internet and **N**arrow band **A**udio **DI**gital **B**roadcasting (NADIB). The bitrates range from 6 kbit/s and 24 kbit/s, with bandwidths between 3.5 and 9 kHz.
- The **Main Profile** is a rich superset of all the other Profiles, containing tools for natural and synthetic Audio.

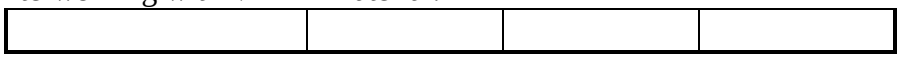
5.3 Graphics Profiles

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:

- **Simple 2D Graphics** Profile The Simple 2D 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.
- **Complete 2D Graphics** Profile The Complete 2D Graphics profile provides two-dimensional graphics functionalities and supports features such as arbitrary two-dimensional graphics and text, possibly in conjunction with visual objects.
- **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.

5.4 Scene Description Profiles

Scene Description Profiles, defined in the Systems part of the standard, allow audiovisual scenes with audio-only, 2-dimensional, 3-dimensional or mixed 2D/3D content. The 3D Profile is called ‘VRML’, as it optimizes interworking with VRML material:



- 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.
- The **Simple 2D** 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 2D Scene Graph profile allows presentation of audio-visual content with potential update of the complete scene but no interaction capabilities. The Simple 2D Scene Graph profile supports applications like broadcast television.
- The **Complete 2D** Scene Graph Profile provides for all the 2D scene description elements of the BIFS tool. It supports features such as 2D transformations and alpha blending. The Complete 2D Scene Graph profile enables 2D applications that require extensive and customized interactivity.
- The **Complete** Scene Graph profile provides the complete set of scene graph elements of the BIFS tool. The Complete Scene Graph profile will enable applications like dynamic virtual 3D world and games.

5.5 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 profiles 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. Version 2 of MPEG-4: Ready one year after Version 1

In October 1998, the first set of MPEG-4 Standards was frozen. Work on MPEG-4 continues for Version 2, which will add tools to the MPEG-4 Standard. Existing tools and profiles from Version 1 will not be replaced in Version 2; technology will be added to MPEG-4 in the form of new profiles. Figure 17 below depicts the relationship between the two versions. Version 2 is a backward compatible extension of Version 1.

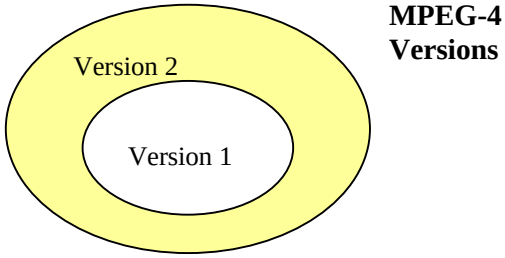


Figure 17 - relation between MPEG-4 Versions

Version 2 builds on Version 1 of MPEG-4. The Systems layer of Version 2 is backward compatible with Version 1. In the area of Audio and Visual, Version 2 will add Profiles to Version 1.

6.1 Systems

Version 2 of the MPEG-4 systems extends Version 1 to cover issues like multi user interaction, content protection, and Java (MPEG-J) support. Version 2 also specifies a file format to store MPEG-4 content. Below the new elements will be briefly introduced.

6.1.2 Advanced BIFS

In MPEG-4 Version 2 Binary Scene Description the following topics have been discussed and included in the Verification Model and Working draft of the standard:

1. Multi user functionality to enable several users to access the same scene and interact with its content. Multi User Interactions are considered as extension of single user systems rather than targeting to a specific application domain.
2. Advanced Audio BIFS for enabling more natural sound source and sound environment modeling than is possible in Version 1 BIFS. The functionalities of the new sound source include modeling of air absorption and more natural distance dependent attenuation, as well as sound source

directivity modeling. In Version 2 BIFS also takes into account the response of the environment so that the sound can be rendered so that it corresponds to the visual parts of the scene.
3. Face and body animation by a Body node that contains information about the physical description (visual appearance) of a body as well as description of its animation. This enables either use of a default body with default initial position or description of user-defined body and positions.
4. Proto and Externprotoand Script VRML constructs, that enable utilizing the functionality of VRML prototypes and scripts, as well as enabling a maximal compatibility with VRML.
5. Other VRML nodes that are not specified in the current Version of BIFS.

6.1.3 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 TransMux while enabling efficient support for TransMuxes in general. The design is based on the QuickTime® format from Apple Computer Inc.

The MP4 file format is composed of object-oriented structures called 'atoms'. Each atom is identified by a unique tag and a length. 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

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 TransMux. There can be multiple hint tracks for one presentation, describing how to deliver over various TransMuxes. In this way, the file format facilitates streaming without ever being streamed directly.

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.

6.1.4 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 18. 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 2.

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.

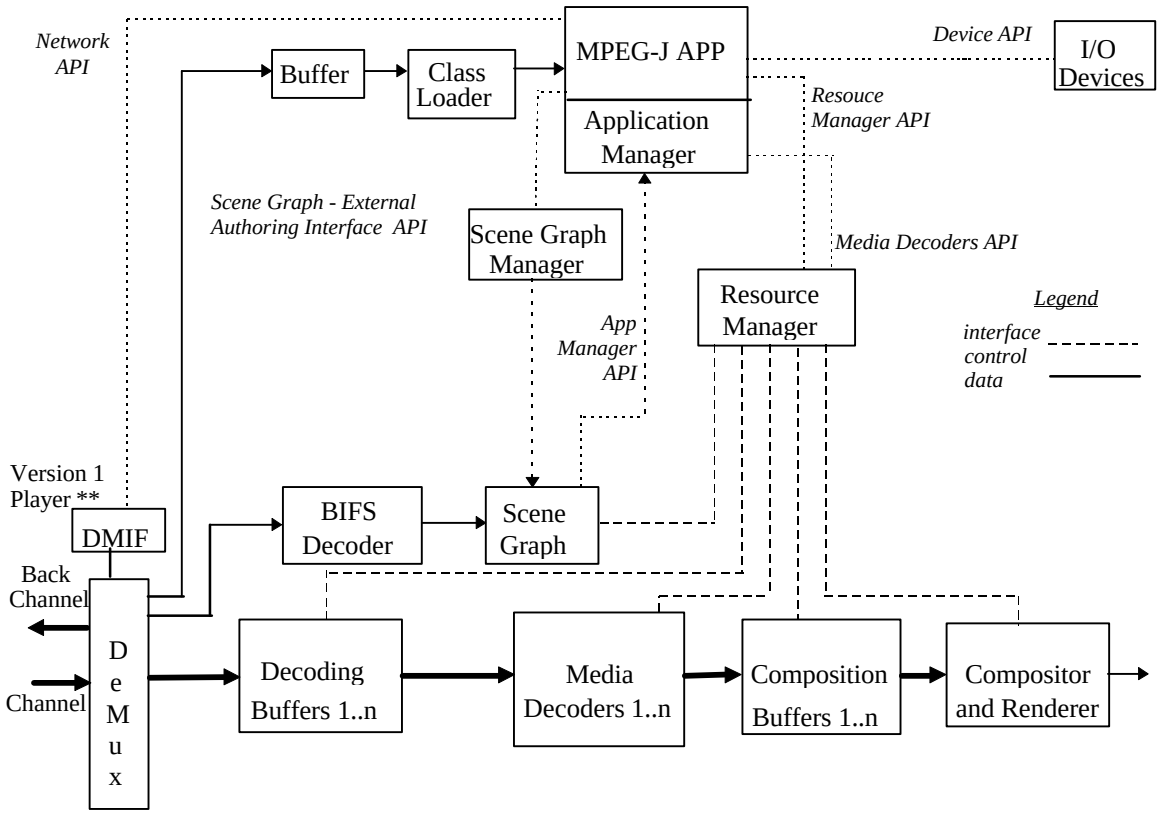


Figure 18 - Location of interfaces in the architecture of an MPEG-J enabled MPEG-4 System

6.2 Visual

The Visual part of the MPEG-4 standard will be extended with tools in the following areas:

6.2.1 Natural Video

- increased flexibility in object-based scalable coding,
- improved coding efficiency, especially for very low bitrate applications requiring simple decoders;

- improved error robustness;
- Coding of Multiple Views: *Intermediate views or stereoscopic views* will be supported based on the efficient coding of multiple images or video sequences. A particular example is the coding of stereoscopic images or video by redundancy reduction of information contained between the images of different views.

6.2.2 Body animation

The Body is an object capable of producing virtual body models and animations in form of a set of 3D polygon 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)h, 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 is being 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.

6.2.3 Coding of 3D Meshes

Version 2 MPEG-4 is developing a suite of tools for the coded transmission of the elements of 3D models. The underlying technologies compress the structure or connectivity, geometry or shape, and properties such as shading normals, colors and texture coordinates of 3D models. Targeted capabilities for 3D model coding include:

- *Coding of generic 3D meshes* allows to efficiently code synthetic 3D objects.
- *LOD (Level Of Detail) scalability* allows the decoder to decode a subset of the total bit stream generated by the encoder to reconstruct a simplified version of the mesh containing less vertices than the original. Such simplified representations are useful to reduce the rendering time of objects which are distant from the viewer (LOD management), but also allow less powerful rendering engines to render the object at a reduced quality.

- *Spatial scalability* allows the decoder to decode a subset of the total bit stream generated by the encoder to reconstruct the mesh at a reduced spatial resolution. This feature is most useful when combined with LOD scalability.
- 3D progressive geometric meshes (temporal enhancement of 3D mesh detail through progressive transmission of differential geometry and structure - similar in functionality to progressive texture).

6.3 Audio

The following additional functionalities will be supported by MPEG-4 Audio Version 2:

- Error resilience methods

  Version 2 will add new tools to the audio algorithms to improve their error resilience. There are two classes of tools: The first class contains algorithms to improve the robustness of the source coding itself, e.g. Huffman codeword reordering for AAC. The second class consists of general tools for error protection, allowing equal and unequal error protection of the MPEG-4 audio coding schemes. Since these tools are based on convolutional codes, they allow very flexible use of different error correction overheads and capabilities, thus accommodating very different channel conditions.
- Environmental spatialization.

  These new tools allow parametrization of the acoustical properties of an MPEG-4 scene (e.g. a 3-D model of a furnished room or a concert hall) created with the BIFS scene description tools. Such properties are, for example, room reverberation time, speed of sound, boundary material properties (reflection, transmission), and sound source directivity. New functionality made possible with these scene description parameters includes advanced and immersive audiovisual rendering, detailed room acoustical modeling, and enhanced 3-D sound presentation
- Low delay general audio coding

  This functionality supports transmission of general audio signals in applications with bi-directional communication. Compared to version 1, a significantly reduced coding/decoding delay will be provided with only a slight reduction of coding efficiency.
- Syntax for a Backchannel, for adaptive coding and play-out of Audio objects
- Small step scalability

  This tool allows scalable coding with very fine granularity, i.e. embedded coding with very small bitrate steps. This is achieved by Bit-Sliced Arithmetic Coding (BSAC) in conjunction with the general audio coding tools of version 1.
- Parametric audio coding

  These tools offer the possibility of modifying the playback speed or pitch during decoding due to a parametric signal representation without the need of a special effects processing unit. A combination with the HVXC speech coding tool will be possible, which is also based on a parametric signal representation. Furthermore, for applications of object based coding allowing selection and/or switching between different coding techniques, an improvement of the overall coding efficiency is expected in conjunction with the general audio coding tools of version 1.

6.4 DMIF

The Delivery efforts for MPEG-4 Version 2 will consider the following issues.

- *Addition of Mobile Operation with DMIF*

DMIF V1 will be reviewed to ensure that it can make use of the proper stacks required in mobile operation such as H.223 and H.245 and allow the addition of MPEG-4 to H.324 terminals.

- *Extend DMIF V1 QoS to Access Unit Loss and Delay parameters at the DAI*

This requires mapping of those parameters to the Network QoS parameters such as IETF Int Serv and Diff Serv or ATM Q.2931 QoS or to mobile. Subsequent monitoring of the performance delivered on those parameters and actions taken based on stream priority and stream dependencies.

- *Invoke SRM (Session and Resource Management) on demand after an initial Session has been established (with the tools present in DMIF v.1)*

This allows a seamless transition of a session on a homogeneous network using DMIF v.1 to a one more involved using DMIF SRM, see below. A homogeneous network is a network composed of one transport technology only.

- *Allow heterogeneous connections with end-to-end agreed QoS level*

A heterogeneous Network is composed of different transport technologies which are connected in tandem through InterWorking Units. Typically, an end-to-end connection may be comprised of access distribution networks at both ends to which peers are connected to and a core network between them. No restrictions are enforced on the transport technology that each segment of the end-to-end connection may use. Also, peers can try to achieve a best effort as well as a guaranteed end-to-end QoS on the end-to-end connection. It is possible that a DMIF end-to-end connection uses an Internet core with RSVP as opposed to an ATM core. This work will also incorporate network processing resources in an end-to-end connection such as transcoders/audio and video bridges/multicast servers/switched broadcast servers within a DMIF network session. A standardized signaling between SRM and InterWorking Units will be developed.

- *Integrate with IETF specified Network Servers*

This extends the integration started in DMIF V1 to network servers specified by IETF which provide SRM like functions. The DMIF SRM, like the DMIF signaling will only complement the functions required by DMIF missing or not adopted by existing servers.

- *Fully symmetric consumer and producer operations within a single device.*

DMIF extends the DSM-CC SRM with the concept of peers carrying receiver (consumer) or sender (producer) roles as opposed to mere Clients or Servers. This is in line with symmetric video codecs and allows for the initiation of session and adding a connection by any peer. The role of a peer as a consumer or a producer is defined after a session is established. This allows DMIF v.2 to be used in conversational as well as (DSM-CC) multimedia retrieval applications.

- *End-to-end "session" across multiple network provider implementations*

  In this case many Session and Resource Managers (SRM) each belonging to a different administrative entity and having its own subscribed peers will interoperate, such that a session may be established with peers across different SRM nodes. This will require standardized signaling to be developed between the SRM nodes. DMIF SRM groups the resources used in a service instance using session ID tags. Resources are not limited to network resources and anything that is dynamically allocated/de-allocated can be treated as a resource. The resources used in a DMIF network session can be logged, e.g. for billing. The session ID also is used for retiring the resource for later reuse once the session is terminated. All the connection resources are set up and cleared as one unit.

  DMIF SRM allows the collection of accounting logs so that the revenues collected from the peers in a DMIF session are properly disbursed to those providers that supply the resources within that session.

7. Annexes

Annex 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 recently finalized MPEG-4 Version 1, a standard for multimedia applications, that will officially reach the status of International Standard in February 1999, with the ISO number 14496.

MPEG has also started work on a new standard known as MPEG-7: a content representation standard for information search, scheduled for completion in Fall 2001. The Call for Proposals was issued in October 1998.

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. Digital Audio Broadcasting (DAB) is a new consumer market that makes use of MPEG-1 audio coding.

MPEG-2 has been the timely response for the satellite broadcasting and cable television industries in their transition from analogue to digital. Millions of set-top boxes incorporating MPEG-2 decoders have been sold in the last 3 years.

Since July 1993 MPEG is working on its third standard, called MPEG-4. MPEG considers of vital importance to define and maintain, without slippage, a work plan. This is the MPEG-4 Version 1 work plan:

Part	Title	WD	CD	FCD	FDIS	IS
1	Systems		97/11	98/03	98/10	99/02
2	Visual		97/11	98/03	98/10	99/02
3	Audio		97/11	98/03	98/10	99/02
4	Conformance Testing	97/10	98/12	99/07	99/12	00/02
5	Reference Software		97/11	98/03	99/03	99/05
6	Delivery Multimedia Integration Framework (DMIF)	97/07	97/11	98/03	98/10	99/02

Table 1 - MPEG-4 Version 1 work plan
(NB: 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 could be 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 Sevilla 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 is now at FDIS stage. It will now be sent to National Bodies for a final ballot where NBs are only allowed to cast a yes/no ballot without comments. This must happen within two months. If the number of positive votes is above 75%, the FDIS will become International Standard (IS) and is sent to the ISO Central Secretariat for publication.

Annex B - Organization of work in MPEG

Established in 1988, MPEG has grown to form an unusually large committee. Some 300 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 expertise available 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. DSM (Digital Storage Media)	Develops standards for interfaces between DSM servers and clients for the purpose managing DSM resources, and controlling the delivery of MPEG bitstreams and associated data.
3. Delivery	Develops standards for interfaces between MPEG-4 applications and peers or broadcast media, for the purpose of managing transport resources.
4. 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.
5. Video	Develops standards for coded representation of moving pictures of natural origin.
6. Audio	Develops standards for coded representation of audio of natural origin.

7. 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.
8. 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
9. Implementation	Evaluates coding techniques so as to provide guidelines to other groups upon realistic boundaries of implementation parameters.
10. Liaison	Handles relations with bodies external to MPEG.
11. HoD Heads of Delegation	The group 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, when reasons of urgency so require, they are authorized to hold physical meetings. Ad-hoc groups produce recommendations that are reported at the first plenary of the MPEG week and become valuable inputs for further deliberation during the meeting.

Annex 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
Alpha plane	Image component providing transparency information ??? (Video)

API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BAP	Body Animation Parameters
BDP	Body Definition Parameters
BIFS	Binary Format for Scene description
BSAC	Bit-Sliced Arithmetic Coding
CE	Core Experiment
CELP	Code Excited Linear Prediction
DAI	DMIF-Application Interface
DMIF	Delivery Multimedia Integration Framework
DNI	DMIF Network Interface
DS	DMIF signalling
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 objectbject 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
FDP	Facial Definition Parameters
FlexMux tool	A Flexible (Content) Multiplex tool
FlexMux stream	A sequence of FlexMux packets associated to one or more FlexMux Channels flowing through one TransMux Channel
FTTC	Fiber To The Curb
GSTN	General Switched Telephone Network
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
IPR	Intellectual Property Rights
ISDN	Integrated Service Digital Network
LAR	Logarithmic Area Ratio
LC	Low Complexity
LPC	Linear Predictive Coding
LSP	Line Spectral Pairs
LTP	Long Term Prediction
mesh	A graphical construct consisting of connected surface elements to describe the geometry/shape of a visual object.
MCU	Multipoint Control Unit
MIDI	Musical Instrument Digital Interface
MPEG	Moving Pictures Experts Group
MPEG-J	Framework for MPEG Java API’s
OD	Object descriptor
PSNR	Peak Signal to Noise Ratio
QoS	Quality of Service

RTP	Real Time Transport Protocol
RTSP	Real Time Streaming Protocol
Rendering	The process of generating pixels for display
SL	Sync(hronization) layer
Sprite	A static sprite is a - possibly large - still image, describing panoramic background.
TCP	Transmission Control Protocol
T/F coder	Time/Frequency Coder
TransMux	generic abstraction for any transport multiplex scheme
TTS	Text-to-speech
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunication System
Viseme	Facial expression associated to a specific phoneme
VLBV	Very Low Bitrate Video
VRML	Virtual Reality Modeling Language