ELSEVIER

Signal Processing: *Image Communication* 15 (2000) 463–478

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

Profiles and levels in MPEG-4: Approach and overview

Rob Koenen

KPN Research, St. Paulusstraat, 4, 2264 XZ Leidschendam, Netherlands

Abstract

Profiles and levels in MPEG-4 are standardised in order to give users a number of well-defined and well-chosen conformance points. They serve two main purposes: (1) ensuring interoperability between MPEG-4 implementations, and (2) allowing conformance to the standard to be tested. Profiles exist not only for the Audio and Visual parts of the standard (*audio profiles* and *visual profiles*), but also for the Systems part of the standard, in the form of *graphics profiles*, *scene graph profiles*, and an *object descriptor profile*. Different profiles are created for different application environments. The policy for defining profiles is that they should enable as many applications as possible while keeping the number of different profiles low. MPEG has defined a first set of profiles for MPEG-4, but more are expected. MPEG will be restrictive in defining any new profiles, listening carefully to what its users have to say.

Keywords: MPEG-4; Profile; Level; Conformance; MPEG-4 video; MPEG-4 audio; MPEG-4 Systems; Interoperability; Interworking

1. Introduction

An often encountered criticism of the MPEG-4 Standard concerns its size and complexity. The complaint is that the standard is too big and unwieldy to implement. This is true – at least it would have been without the existence of profiles and levels. (See [8,9] and the other articles in this issue for an overview of the object-based multimedia content representation standard MPEG-4.) This paper will describe MPEG-4's Profiles and Levels and the philosophy behind their definition.

Profiles are known from MPEG-2 Video [3] where the most used profiles are 'Main' in end user systems, and more recently '4 : 2 : 2' for professional purposes. Both in MPEG-2 and MPEG-4, profiles limit the tool set that needs to be implemented; they are created for users that wish to use only a part of the standard. (Such 'users' are usually industrial consortia rather than end-users.) In fact, profiles can be regarded as a compromise between maximal interoperability and minimal implementation overhead. The essence of Profiling the MPEG-4 standard is, as this article will argue, about finding optimal balances and making the right trade-offs. The larger and more diverse a standard like MPEG-4 becomes, the more difficult it is to make the right choices and to arrive at a transparent and usable division into subsets while maintaining interoperability between systems.

MPEG-4 not only defines visual (as in MPEG-2) and audio profiles, but also graphics profiles, scene graph profiles, and one object descriptor profile. Visual profiles are, logically, defined in the visual part of the standard (part 2 [4]); audio profiles in

E-mail address: r.h.koenen@research.kpn.com (R. Koenen)

0923-5965/00/$ - see front matter
PII: S0923-5965(99)00058-2

the audio part (part 3 [5]) and the other three types can be found in the Systems part of MPEG-4 (part 1 [6]). The DMIF part of the MPEG-4 Standard [7] does not have profiles; they simply are not needed, since implementing the whole part does not incur significant complexity over only implementing a subset.

First, we will explain the most important concepts in Section 2. Section 3 will then describe the procedure and policy that was developed for choosing the needed profiles, and, next, in Section 4, we will describe the profiles that are currently defined and their envisioned application areas. Section 5 will then give an outlook to what may follow in MPEG-4 Version 2 and beyond.

2. Profiling principles

The goal of defining profiles and levels is twofold: The first goal is to *ensure interoperability*. Implementations of a profile at a certain level result in a decoder that behaves in a predictable way. Content encoded (e.g. by a real-time encoder) or authored (e.g. for streaming from a server) for such a combination will work on any decoder implementation thereof. The second goal is to allow *conformance testing* to take place. A profile/level combination gives a well-defined conformance point. For such a conformance point, tests can be devised to determine whether implementations of the standard really operate as the standard specifies. Typically, such tests define input (bitstreams) and expected decoder output (e.g. waveforms or pixel values for decoded audio and video objects, respectively).

Profiles by themselves do not constitute a conformance point for the standard. To define a conformance point, a level also is needed. Whereas the profile restricts the tool set, the level defines the bounds of complexity that can be expected in the bitstream for a particular profile. Without a level definition, the complexity that needs to be handled by, say, a video decoder, could still be arbitrarily complex as a profile does not specify, e.g. maximum bit-rates, frequency, etc. The bound set by the level needs to be observed by both the encoder and the decoder. The decoder should at all times be capable of handling a bitstream with maximum complexity, and thus the level gives minimum implementation bounds. For decoding *hardware*, the profile/level combination gives minimum performance constraints to be observed at design and manufacture time. For decoding *software*, the combination may also imply resource availability to be monitored at run time. (Note that a Profile/Level combination is usually referred to as 'profile@level', to be pronounced as 'a profile at a level'.)

The MPEG-4 Standard, like MPEG-1 and MPEG-2, only defines the decoding process and the syntax and semantics of the bitstream. The encoding process is not specified, with the restriction that a valid bitstream must result. In spite of this fact, profile/level combinations do impact encoding systems. To encoders, regardless of whether implemented in hardware or software, a profile/level combination gives implicit bounds to observe while encoding a bitstream. The encoder should constantly check whether the output bitstream is still within the limits defined by the level. In practice, for 'encoder' one should actually read the more general term 'authoring system' because MPEG-4 content could well be created from different pre-encoded objects. Summarising, a profile@level is an *upper* bound on the complexity of the bitstream (to be observed by the encoder), and a *lower* bound on the capabilities of the decoder.

An important note: MPEG-4 is an object-based standard, and audiovisual scenes are composed of different objects. In this context, it is that Audio and Visual Profiles@Levels do *not* define the maximum complexity per individual MPEG-4 object but rather give bounds on the *total of all objects* in the scene. We will explain below why this approach was chosen.

We will now first give the relevant definitions and terminology as used in MPEG before going into more detail. These are extracted from the MPEG-4 Requirements Document [12].

> **Object Type** – *An Object Type defines the syntax of the bitstream for one single object that can represent a meaningful entity in the* (*Audio or Visual*) *scene.* (*Note that this corresponds to a list of tools. There are Audio Object Types and Visual Object Types.*)

Profile – *A Profile defines the set of a certain type of tools that can be used in a certain MPEG-4 terminal. There are Audio, Visual, Graphics, Scene Description,*[1] *and Object Descriptor Profiles.*
Level – *A level is a specification of the constraints and performance criteria on an Audio, Visual, Graphics, Scene Description, or Object Descriptor Profile and thus on the corresponding tools.*
Conformance Point – *A Conformance point is a specification of a particular Audio, Visual, Graphics, Scene Description, or Object Descriptor Profile at a certain Level at which conformance may be tested. Conformance Points establish normative parts of the MPEG-4 Standard.*

Audio and visual profiles are more than just a list of tools. They define the kinds of audio and visual objects that the MPEG-4 terminal needs to be able to decode and, hence, give a list of admissible Elementary Stream types – perhaps with a restriction on how they can be combined. (See [2] for a discussion on Elementary Streams in MPEG-4.)

In MPEG-2, (video) profiles can be thought of as containing only one single, rectangular object. In MPEG-4, scenes can contain more than one object, and the objects can be of different nature. Therefore, the concept of an object type is introduced, as an intermediate level of definition between tools and profiles. Object types not only define which tools are needed to create an object in the scene, but also how they can be combined. When a profile consisted only of a list of tools, many more combinations of tools would be possible than now allowed in the pre-defined object types. Not all of these would make sense and some of them would be very hard to implement. Hence, object types are a required step in the definition of a profile.

Graphics profiles define, in terms of BIFS nodes (Binary Format for Scenes, see [14]), which graphical elements can be used in the scene. Scene Description profiles define the scene description capabilities required in the terminal, also in the form of allowed BIFS nodes in the bitstream. Note that there are two different BIFS nodes. The first type of nodes is used to create objects in the scene, or to refer to elementary streams associated with media objects. This is the type of node found in the graphics profile. The second type of nodes is used to build the scene structure and to define object and user interactions. These are called "scene graph elements", and are found in the scene graph profiles. Lastly, the object descriptor profiles define required terminal capabilities in terms of object descriptor and synchronisation layer tools. (See [2] for a detailed explanation about these tools.) The audio, visual and graphics profiles can be called *media profiles* as they govern the media elements in the scene. Note that MPEG has chosen not to prescribe which combinations of audio, visual and graphics media profiles are allowed. MPEG wants to let the market decide this.

From the above, it can be concluded that profiles can only be made in combination with a certain level. Some profiles, however, currently have only one level defined in which case mentioning the (default) level could be omitted.

3. Profile policy and version management

The policy in defining MPEG-4 profiles is aimed at obtaining a minimum amount of profiles/level combinations that are as widely usable as possible. This means both a low total number (giving a 'global' optimum) as well as a low number of different conformance points that address the same application type (giving a 'local' optimum). The profile policy should be understood together with the approach to version management. Let us first discuss the version management before returning to the profile policy. The fact that the MPEG-4 Standard is delivered in versions necessitates the existence of version management procedures [10]. Currently there is MPEG-4 version 1, with version 2 scheduled for the end of 1999. New tools are under consideration that are not part of version 2, so it is likely that a version 3 will also see the light of day. Technically, such a new version is issued as an amendment of the standard. Versions of MPEG-4 are meant for major improvements and

[1] Note that scene description profiles are in the MPEG-4 standard referred to as scene graph profiles. This is also the term used in this article.

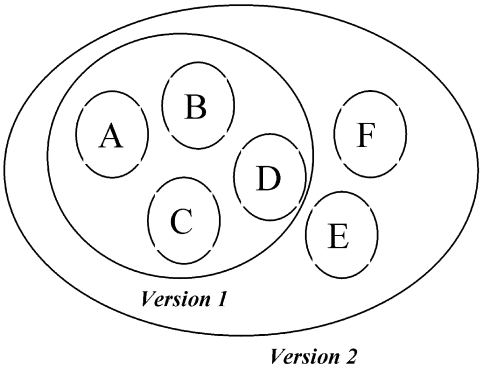

Fig. 1. Version 2 (e.g. of MPEG-4 Visual) includes all Version 1 profiles and adds new ones. Profile E could be a superset of D, but it could also be a subset of an existing profile with new tools added. (E could even be a subset of D, which means that only version 2 tools are used in this profile.)

enhancements of the tool set. New tools are only added if they bring new functionalities or significant gains in performance for the functionalities already provided. This implies that a marginal increase in, e.g., coding efficiency is not enough reason for adding a tool, which would make the standard more complex, more expensive to implement, and less stable. A new version, by definition, extends the standard in a backward compatible way. This compatibility, notably in the area of (audio and visual) coding tools, is preserved by adding new profiles to the existing set. These *may* be supersets of existing profiles, but they do not have to be. It is important to note that an existing profile will not be modified in a new version. The version management procedures as documented within MPEG define this as follows: '*New Versions, notably in the area of coding tools, are managed by adding new profiles, significantly different from existing ones. New Versions will not make changes to existing profiles*' [10]. See Fig. 1. (Another aspect of version management is software version management; MPEG requires software implementations to be made available for all tools that are included in the standard. A discussion of the MPEG software version management process is outside the scope of this article.)

Now that the versioning is clear, let us again consider the profile policy. When the potential profiles for MPEG-4 were first discussed in MPEG, there were many requests for different profiles. Sometimes the proposals were very close in terms of tools to be included. With these many requests, it was very clear that a strict policy was needed in order to limit the amount of different profiles, especially in the visual part of the standard. Having too many profiles confuses the market place and is very unhelpful in achieving interoperability. While profiling information is an integral part of the standard, defining a profile or a level is different than defining a tool because a profile does not affect the syntax of the bitstream. (There is one exception: a new value needs to be defined for the field that signals the profile in the object descriptor.) This implies that there is no 'hurry' in defining profiles and that they can be added at a relatively late stage. This is in contrast to the tools themselves, which need to be rigorously tested and cross-checked with other tools. In other words, the tools in the standard need to anticipate future needs while the profiles can be defined when the requirements become apparent. This means that MPEG can take a conservative attitude in defining profiles.

As the definitions in Section 1 highlighted, audio and visual profiles are defined as a collection of object types, admissible in the scene. For these object types, basically the same rules apply as for profiles: any newly defined object type should differ from all existing ones, adding functionality or significantly improving existing functionality. Such functionality could be, e.g., error resilience, support for interactivity, scalability, or compression efficiency. Moreover, there needs to be enough evidence that the proposed object type is actually useful and will indeed be used. Potential new object types in version 1 were not considered just on their own merit, but also in relation with the other object types on the table. The same rule applies for version 2: not only need the new object types be different from existing ones, but also from the other object types under consideration; MPEG tries to merge similar object types. The rule is that new objects are only created if such merging is not possible, that is *"the tool in question cannot be added to an object type already under consideration for definition in the same version, without overly burdening it"* as MPEG describes it in [11]. In practice, this means MPEG

seeks to ensure that applications do not have to carry too many unnecessary tools.

A very similar procedure is applied in adding new profiles. They should be significantly different from existing ones and (within a new version) from each other. Also here MPEG looks at whether proposed applications really need a new profile, as they might make good use of an existing, possibly more powerful profile without too much extra burden. For profiles defined after version 1, an extra criterion is applied: there needs to be support from parties that wish to implement the profile in an MPEG-4 product. (While MPEG's requirements group asks companies for their commitment, it will not ask them for precise deployment plans, as these are often confidential.)

For the last step in the process, the definition of levels for a profile, the same strategy is applied: only define levels that are going to be used. As far as producing the standard is concerned, adding levels is a matter of merely 'adding a line to a table'. This can always be done when the need becomes apparent; although the formal process is the same as for any amendment, it requires far less technical work than adding tools would. Thus MPEG is not concerned with completely defining all possible levels within a given profile, but rather only those that are envisaged to be used.

Although, in principle, all the work in MPEG is driven by the requirements, it may turn out that tools in the standard have not found a 'home' in any of the profiles, at least not yet. As long as the number of these tools is low, this is not a problem. The same happened in MPEG-2, where some tools were not included in (Video) profiles. Work on tools is often started in advance of real market need, based on anticipated requirements. When later in the process the exact market requirements take shape, some of the tools may not be as useful as originally thought. (Note that the converse also happens: needs become apparent that are unsupported by tools. In such a case, quick action needs to be taken, especially if the technology to fulfil the requirements exists. In MPEG-4, this was the case for the MPEG-4 file format and the management and protection of Intellectual Property.)

The profiles in MPEG-2 Video are organised as an almost complete hierarchical structure. This implies that decoders capable of dealing with the higher profiles can by definition also understand the ones 'below'. In MPEG-4, hierarchy is implemented where possible, but not pursued at all cost. This policy was adopted because keeping a strict hierarchy in MPEG-4 is much harder than in MPEG-2, since there are more tools that suit more diverse purposes. As an example to illustrate this, consider MPEG-4 visual. A strict hierarchy could be maintained for all the tools addressing compression efficiency for rectangular objects. Different application areas, however, have different needs for choosing combinations of natural and synthetic object types, and using those with or without scalability, error resilience and shape representation. Thus, a strict hierarchy across all of the Profiles cannot be maintained.

4. Overview of profiles in MPEG-4

This section gives an overview of the profiles that are defined in version 1 of MPEG-4. It first lists the media profiles and then the non-media profiles. These non-media profiles are defined in the Systems part of the standard and so are the graphics media profiles. The audio and visual media profiles can be found in the audio [5] and visual [4] parts of the standard, respectively. Fig. 2 provides a graphical representation of the profiles in MPEG-4.

4.1. Media profiles

Media profiles describe the object types that can be used to create the scene, and tools that can be used to create those object types. In this subsection, we will describe the media profiles and give examples of the possible environments they can be used in.

4.1.1. Visual

In this section, we will discuss the visual object types and profiles, as defined in MPEG-4 Visual [4]. Level information will be supplied without too much detail, as this would go beyond the scope of this paper.

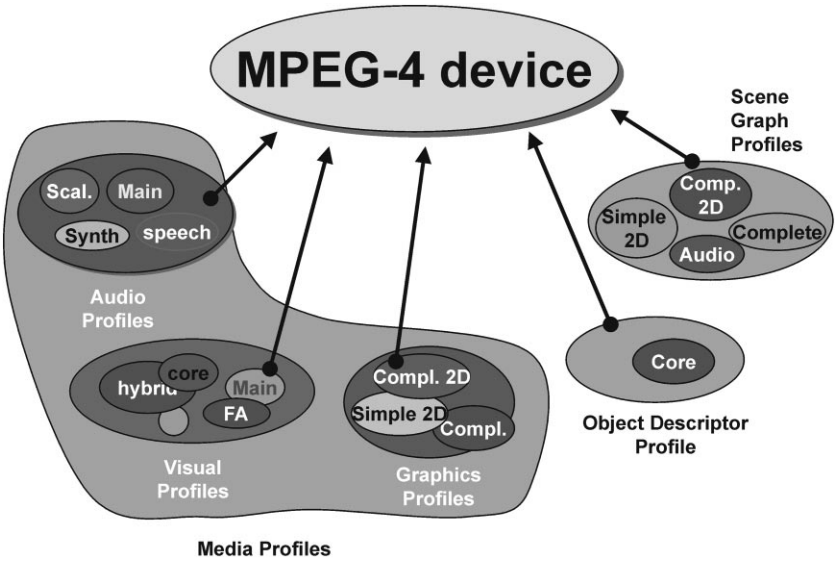

Fig. 2. An overview of the profile structure in MPEG-4. Note that the figure does not list all profiles, notably in the visual area. It is not meant to reflect hierarchical relationships.

4.1.1.1. Visual object types. As the visual profiles are defined using the visual object types, it is necessary to discuss these before discussing the profiles themselves.

There are five different object types for representing natural video information.

1. The **Simple** object type is an error resilient, rectangular natural video object of arbitrary height/width ratio, developed for low bit-rates. It uses relatively simple and inexpensive coding tools, based on I (Intra) and P (Predicted) VOPs (Video Object Planes, the MPEG-4 term for frames).
2. The **Simple Scalable** object type is a scalable extension of Simple, which gives temporal and spatial scalability using Simple as the base layer. The enhancement layer is still rectangular.
3. The **Core** object type uses a tool superset of Simple, giving better quality through the use of bi-directional interpolation (B-VOPs), and it has binary shape. It supports scalability based on sending extra P (predicted) VOPs. Note that binary shape can include a constant transparency but excludes the variable transparency offered by grey-scale shape coding.
4. The **Main** object type is the video object that gives the highest quality. Compared to Core, it also supports grey-scale shape, sprites, and interlaced content in addition to progressive material. (See [1] for an overview of MPEG-4 Video and an explanation of what exactly a sprite is in MPEG-4.)
5. The **N-bit** object type is equal to the Core object type but it can vary the pixel depth from 4 to 12 bits for the luminance as well as the chrominance planes.

As can be seen in Table 1, the Simple object type uses a subset of the tools in Core, and Core in return uses a subset of the tools in Main. The tools in the Simple Scalable object type are a superset of the tools in Simple, while the N-bit object type is a superset of Core (and hence also of Simple).

There is one special object type for representing still natural visual information:

6. The **Still Scalable Texture** object type gives an arbitrary shape still image that uses wavelet

Table 1
Coding tools in the visual object types

Object type → ↓ Tools	Simple	Core	Main	Simple scalable	N-bit	Animated 2D mesh	Basic animated texture	Still scalable texture	Simple face
Basic tools • I-VOP, P-VOP • AC/DC prediction • 4-MV, unrestricted MV	X	X	X	X	X	X			
Error resilience • Slice resynchronization • Data partitioning • Reversible VLC	X	X	X	X	X	X			
Short header	X	X	X		X	X			
B-VOP		X	X	X	X	X			
Method 1/method 2 quantization		X	X		X	X			
P-VOP-based temporal scalability • Rectangular • Arbitrary shape		X	X		X	X			
Binary shape		X	X		X	X	X		
Grey shape			X						
Interlace			X						
Sprite			X						
Temporal scalability (rectangular)				X					
Spatial scalability (rectangular)				X					
N-bit (4–12 bit)					X				
Scalable still texture						X	X	X	
2D dynamic mesh with uniform topology						X	X		
2D dynamic mesh with Delaunay topology						X			
Facial animation parameters									X

coding for scalability and incremental download and build-up.

The following object types use synthetic tools, some of them in combination with natural video texture:

7. The **Animated 2D Mesh** object type combines the synthetic mesh (either rectangular or Delaunay topology) with natural video. The natural video coding uses the same tools as the Core object type. This video can be mapped onto the mesh and deformed by moving the points in the mesh. It gives interesting animation possibilities. Note that the object can be of arbitrary (binary) shape.
8. The **Basic Animated Texture** object type allows mesh animation with arbitrary shape still images (the same images as used for the still scalable texture object type, see above).
9. The last object type is the **Simple Face** object type, which has the tools for facial animation. This object type does not define what the face looks like, and the animation can be applied to any local model of choice. Note that MPEG-4 does include tools to download a pre-defined face to the decoder, but these tools are not mandatory in the simple face object type.

Table 1 (adapted from [4]) gives a list of the tools used by the object types. A detailed explanation of the tools goes beyond this article, please refer to the standard itself [4] or to the article on MPEG-4 Visual in this issue [1].

4.1.1.2. Visual profiles. This visual profiles determine which visual object types can be present in the scene. This is also the way they are defined: as a list of admissible object types. Quite a few of them correspond to the most complicated object that they support, and they also have similar names. Table 2 gives an overview of the visual profiles. Below we will list the profiles and mention some application areas. Note again that these are only suggestions and that profiles were not designed for specific applications. This is also why their names are generic and refer to tools rather than applications or services.

1. The **Simple Profile** only accepts objects of type Simple, and was created with low-complexity applications in mind. The first usage is mobile use of (audio)visual services, and the second is putting very low-complexity video on the Internet. Also small camera devices recording moving video to, e.g., disk or memory chips, can make good use of this profile. It supports up to four objects in the scene with, at the lowest level, a maximum total surface of a QCIF picture. There are three levels for the Simple Profile with bit-rates from 64 to 384 kbit/s. The levels also define the maximum total surface for the objects and the amount of macroblocks per second that the decoder needs to be able to decode. Further, they define the size of various (hypothetical) buffers needed for decoding. While the maximum total object size *is* defined, the aspect ratio is not prescribed. This gives maximum creative freedom. It could be used for instance in a personal computer screen, where a very wide or a very tall object could be created, or several smaller objects in various places on the screen, not confined to a typical QCIF area. The same level philosophy is followed for restricting the complexity of the natural video objects in all the visual profiles.
2. The **Simple Scalable Profile** can supply scalable coding in the same operational environments as foreseen for Simple, and has two levels defined.
3. The **Core Profile** accepts Core and Simple object types. It is useful for higher-quality interactive services, combining good quality with limited complexity and supporting arbitrary shape objects. Also mobile broadcast services could be supported by this profile. The maximum bit-rate is 384 kbit in Level 1 and 2 Mbit/s in Level 2. While the levels do not prescribe the visual session size, they are created with a certain session size in mind, called the 'typical visual session size'. For Simple this was QCIF, for Core it is QCIF and CIF for the two levels, respectively. The amount of macroblocks is chosen such that a scene using this typical session size can have overlapping objects and still be 'filled'.
4. The **Main Profile** was created with broadcast services in mind, addressing progressive as well as interlaced material. It combines the highest quality with the versatility of arbitrarily shaped

Table 2
Object types supported in the visual profiles

Profile → ↓ Object types	Simple	Simple scaleable	Core	Main	N-bit	Scaleable texture	Simple FA	Basic animated texture	Hybrid
Simple	✓	✓	✓	✓	✓				✓
Simple scalable		✓							
Core			✓	✓	✓				✓
Main				✓					
N-bit					✓				
Animated 2D mesh									✓
Basic animated texture								✓	✓
Scalable texture				✓		✓		✓	✓
Simple face							✓	✓	✓
Number of levels	3	2	2	3	1	3	2	2	2

object using grey-scale coding. The highest level accepts up to 32 objects (of Simple, Core or Main type) for a maximum total bit-rate of 38 Mbit/s.

5. The **N-bit** profile is useful for areas that use thermal imagers, such as surveillance applications. Also medical applications may want to use the enhanced pixel depth giving a larger dynamic range in colour and luminance. It accepts objects of type Simple, Core and N-bit. Currently only one level is defined.
6. The **Scaleable Texture Profile** is meant for audiographic applications. It was requested by companies that want to build mobile devices, which combine sound with synchronously displayed pictures, and possibly BIFS-based graphics, in very simple terminals.
7. The **Simple Face Profile** accepts only objects of type Simple Face. Depending on the level, either one or a maximum of 4 faces can appear in the scene, e.g., for a virtual meeting. Bit-rates remain very low; even for the second level, 32 kbit/s is more than adequate for driving the four faces.
8. The **Hybrid** Profile allows combining both natural and synthetic objects in the same scene while keeping complexity reasonable. On the natural side, it compares to the Core Profile, while on the synthetic side, it adds animated meshes, scalable textures, and animated faces – a rich set of tools for creating attractive hybrid natural and synthetic content. This profile can be used to place 'real' objects into a synthetic world and also to do the opposite, adding synthetic objects to a natural environment.
9. The **Basic Animated Texture** Profile allows animation of still pictures and facial animation. Attractive content can be created at very low bit-rates.

A partial hierarchy exists in the visual profiles, the same hierarchy that we described above for the corresponding object types. This means that Main is a superset of Core, which in itself is a superset of Simple. N-bit is a superset of Core. Simple Scalable is a superset of Simple, in such a way that the Simple profile can decode the base layer of Simple Scalable bitstream.

4.1.2. Audio

We will first describe the object types and then show how they are grouped into profiles. While there are quite many object types in the audio part of the standard (13 plus the Null object type), the amount of different profiles is low compared to the visual side: four. Again, level information will only be presented in broad lines.

4.1.2.1. Audio object types. For coding natural sound, MPEG-4 includes the Advanced Audio Coding (AAC) and Twin Vector Quantisation (Twin VQ) algorithms. The following object types exist:

1. The **Advanced Audio Coding (AAC) Main** object type is very similar to – and compatible with – the AAC Main profile that is defined in MPEG-2 (ISO/IEC 13818-7). MPEG-4 AAC adds the Perceptual Noise Shaping tool. The object type has multi-channel capability, to give five full channels plus a separate low-frequency channel in one object.
2. The **MPEG-4 AAC Low Complexity** object type is a low complexity version of the AAC Main Object type.
3. The **MPEG-4 AAC Scalable Sampling Rate** object type is the counterpart to the MPEG-2 AAC Scalable Sampling Rate profile.
4. The **MPEG-4 AAC LTP** object type is similar to the AAC Main object type with the long term predictor replacing the MPEG-2 AAC predictor. This gives the same efficiency with significantly lower implementation cost.
5. The **AAC Scalable** object type allows a large number of scalable combinations including combinations with TwinVQ and CELP coder tools as the core coders (see below). It supports only mono or 2-channel stereo sound.
6. The **TwinVQ** object type is based on fixed-rate vector quantisation instead of the Huffman coding used in AAC. It operates at lower bit-rates than AAC, supporting mono and stereo sound. (TwinVQ stands for Transform domain Weighted Interleave Vector Quantization).

MPEG-4 includes two different algorithms for coding speech, each operating at different bit-rates, plus a Text-to-Speech Interface:

7. The **CELP** object type uses Code Excited Linear Prediction. It supports 8 kHz and 16 kHz sampling rates at bit-rates from 4 to 24 kbit/s. CELP bitstreams can be coded in a scalable way using bit-rate scalability and bandwidth scalability.
8. The **HVXC** (Harmonic Vector Excitation Coding) object type gives a parametric representation of 8 kHz, mono speech at fixed bit-rates between 2 and 4 kbit/s and below 2 kbit/s using a variable bit-rate mode, supporting pitch and speed changes.
9. The **TTSI** (Text-to-Speech Interface) object type gives an extremely low-bit-rate phonemic representation of speech. The actual text-to-speech synthesis is not specified; only the interface is defined. Bit-rates range from 0.2 to 1.2 kbit/s. The synthesised speech can be synchronised with a facial animation object (see above).

Lastly, a number of different object types exist for synthetic sound.

10. The **Main Synthetic** object type collects all MPEG-4 Structured Audio tools. Structured Audio is a way to describe methods of synthesis, see [13]. It supports flexible, high-quality algorithmic synthesis using the Structured Audio Orchestra Language (SAOL) music-synthesis language, efficient wavetable synthesis with the Structured Audio Sample-Bank Format (SASBF), and enables the use of high-quality mixing and postproduction in the Systems Audio BIFS tool set. Sound can be described at '0 kbit/s' (meaning that sound continues without input – until it is stopped by an explicit command) to 3–4 kbit/s for extremely expressive sounds in the MPEG-4 Structured Audio format.
11. The **Wavetable Synthesis** object type is a subset of the Main Synthetic object type, making use of the SASBF format and MIDI tools. It provides relatively simple sampling synthesis. (MIDI means Musical Instrument Digital Interface, a popular wavetable format in wide use, [15]).
12. The **General MIDI** object type gives interoperability with existing content (see above). Unlike the *Main Synthetic* or *Wavetable Synthesis* object types, it does not give completely

Table 3
Coding tools in the audio object types

Object type → / ↓ Tools	Null	AAC main	AAC LC	AAC SSR	AAC LTP	AAC scalable	TwinVQ	CELP	HVXC	TTSI	Main synthetic	Wavetable synthesis	General MIDI	Algorithmic synthesis and audio FX
MPEG-2 AAC main		X												
MPEG-2AAC LC			X		X	X								
MPEG-2AAC SSR				X										
Noise shaping		X	X	X	X	X								
Long term prediction					X	X	X							
Tools for large-step scalability						X								
TwinVQ							X							
CELP								X						
HVXC									X					
TTSI										X				
Structured audio tools											X			X
SA sample bank format											X	X		
MIDI														

predictable (i.e., normative) sound quality and decoder behaviour.

13. The **Algorithmic Synthesis and AudioFX** object type provides SAOL-based synthesis capabilities for very low-bit-rate terminals. (Note that 'FX' stands for 'effects'.)
14. The **NULL** object type provides the possibility to feed raw PCM data directly to the MPEG-4 audio compositor in order to allow mixing in of local sound at the decoder. This means that support for this object type is in the compositor, not in the decoder, which explains why it does not show up in the profile table below.

Table 3 gives a list of all the audio object types and the tools they use. The list is taken from [5], with 'reserved' lines removed. For a complete explanation of the audio tools, please refer to [5].

4.1.2.2. Audio profiles. While there are quite a few different object types in the audio area, there are only four different profiles. They are explained below, and in Table 4, which gives the object types that are supported by each of the audio profiles. Remember that codec builders cannot claim conformance to object types, but only to profiles at a certain level.

Table 4
Object types in the audio profiles

Profiles → ↓ Object types	Speech	Scalable	Main	Synthetic
AAC main			✓	
AAC SSR			✓	
AAC LC		✓	✓	
AAC LTP		✓	✓	
AAC scalable		✓	✓	
TwinVQ		✓	✓	
CELP	✓	✓	✓	
HVXC	✓	✓	✓	
TTSI	✓	✓	✓	
Main synthetic			✓	✓
Wavetable synthesis			✓	✓
General MIDI			✓	✓
Algorithmic synthesis			✓	✓
Number of levels	2	4	4	3

1. The application area for the **Speech** profile can easily be deduced from its name. Two levels are defined, determining whether either one or a maximum of 20 objects can be present in the (audio) scene.
2. A prime reason for defining the **Scalable** profile was to allow good-quality, reasonable complexity, low bit-rate audio on the Internet, an environment in which bit-rate varies from user to user and from one minute to the next. Scalability allows making optimal use of available, and even dynamically changing, bandwidth while only having to encode and store the material once. The scalable profile was not defined exclusively for the Internet, however. Also in, e.g., broadcast situations, scalability can be a desirable feature. The scalable profile has four levels that restrict the amount of objects in the scene, the total amount of channels, and the sampling frequency. The highest level employs the novel concept of complexity units, to be explained below.
3. Also the **Synthetic** profile has a telling name: it groups all the synthetic object types. The main application areas are found where good-quality sound is needed at very low data rates, while the sound source is usually not a microphone. There are three levels which define the amount of memory for data, the sampling rates, the amount of TTSI objects, and some further processing restrictions.
4. The **Main** profile includes all object types. It is useful in environments where processing power is available to create very rich, highest-quality audio scenes that may combine microphone-recorded sources with synthetic ones. Example application areas are the DVD and multimedia broadcast. This profile has four levels, defined in terms of complexity units. There are two different types of complexity units: processor complexity units (PCU), specified in millions operations/s, and RAM complexity units (RCU), specified in terms of number of kWords. The standard also specifies the complexity units required for each object type. In this way, authors have maximum freedom in choosing the right object types and allocating resources among them. An example makes this clear. The profile could contain main AAC and wavetable synthesis object types. A level could specify a maximum of two of each. This would prevent the resources reserved for the AAC objects to be used for a third and fourth wavetable synthesis object, even though it would not break the decoder. With the complexity units, the author is completely free to use decoder resources for any combination of object types, as long as the types are supported by the profile. (Note that in the scalable profile, this type of level definition was only used for the highest level, because the other three levels can be expressed in simpler ways. Here, only for the highest level will the decoder be complex enough to benefit from the type of flexibility offered by the resource-based way of defining levels.) For further details on audio levels, see the Audio Final Draft International Standard [5].

4.1.3. Graphics

Graphics profiles regulate which of the graphics and textual elements can be used to build a scene. They are expressed in terms of BIFS nodes. Although these are defined in the Systems part of

Table 5
BIFS nodes in the graphics profiles

Graphics tools (= BIFS Nodes)	Graphics profiles		
	Simple 2D	Complete 2D	Complete
Appearance	X	X	X
Box			X
Bitmap	X	X	X
Background			X
Background2D		X	X
Circle		X	X
Color		X	X
Cone			X
Coordinate			X
Coordinate2D		X	X
Curve2D		X	X
Cylinder			X
DirectionalLight			X
ElevationGrid			X
Expression			X
Extrusion			X
Face			X
FaceDefMesh			X
FaceDefTable			X
FaceDefTransform			X
FAP			X
FDP			X
FIT			X
Fog			X
FontStyle		X	X
IndexedFaceSet			X
IndexedFaceSet2D		X	X
IndexedLineSet			X
IndexedLineSet2D		X	X
LineProperties		X	X
Material			X
Material2D		X	X
Normal			X
PixelTexture		X	X
PointLight			X
PointSet			X
PointSet2D		X	X
Rectangle		X	X
Shape	X	X	X
Sphere			X
SpotLight			X
Text		X	X
TextureCoordinate		X	X
TextureTransform		X	X
Viseme			X

the standard, they are really media profiles like the audio and visual ones, and hence we list them now. Three hierarchical graphics profiles are defined in MPEG-4: **Simple 2D**, **Complete 2D** and **Complete**. They differ in the routines (BIFS Nodes) to be supported at the decoder. **Simple 2D** provides the basic functionalities needed to create a visual scene with visual objects, without giving additional graphical elements. **Complete 2D** allows elements like bitmaps, backgrounds, circles, boxes and lines, all in a flat space. These elements have characteristics like line width and colour. Note that BIFS commands exist to change attributes, e.g., the colour of a circle. **Complete** contains the full set of BIFS graphics nodes with which complete and elaborate three-dimensional graphics can be created. It adds to Complete 2D, for example, the sphere, the cone, 3D boxes, etc., but also directional lighting. Note that 'flat' video material can be projected into this space on an arbitrary plane.

Table 5 lists the BIFS nodes that need to be implemented to comply to each of the graphics profiles. The table was taken from [6]; for an explanation of the table see either [6], or [14] in this issue.

4.2. Systems profiles

4.2.1. Scene graph

The scene graph profiles define what types of transformational capabilities need to be supported by the terminal. As in the case of the graphics profiles, this is defined in terms of the scene graph elements (BIFS nodes) that the decoding terminal needs to be able to understand. Examples are translations, (3-D) rotations, but also elements like input sensors with which interactive behaviour can be created. The scene graph profiles follow a structure similar to the graphics profiles, with one profile added for audio-only scenes. Thus we have **Audio**, **Simple 2D**, **Complete 2D** and **Complete**. Also the target applications are very much the same as those for the graphics profiles with Simple 2D providing placement capabilities, Complete 2D adding, for instance, rotation to that, and Complete giving the capability to do arbitrary transformations in a 3D space.

Table 6
BIFS nodes for the scene graph profiles

Scene graph tools	Scene graph profiles			
	Audio	Simple 2D	Complete 2D	Complete
Anchor			X	X
AudioBuffer	X		X	X
AudioDelay	X		X	X
AudioFX	X		X	X
AudioMix	X		X	X
AudioSwitch	X		X	X
Billboard				X
Collision				X
Composite2DTexture			X	X
Composite3DTexture				X
Form			X	X
Group	X	X	X	X
Inline			X	X
Layer2D			X	X
Layer3D				X
Layout			X	X
ListeningPoint			X	X
LOD				X
NavigationInfo				X
OrderedGroup		X	X	X
QuantizationParameter			X	X
Sound				X
Sound2D	X	X	X	X
Switch			X	X
Transform				X
Transform2D		X	X	X
Viewpoint				X
WorldInfo			X	X
Node Update			X	X
Route Update			X	X
Scene Update	X	X	X	X
AnimationStream			X	X
Script			?	X
ColorInterpolator			X	X
Conditional			X	X
CoordinateInterpolator2D			X	X
CoordinateInterpolator				X
CylinderSensor				X
DiscSensor			X	X
NormalInterpolator				X
OrientationInterpolator				X
PlaneSensor2D			X	X
PlaneSensor				X
PositionInterpolator				X
PositionInterpolator2D			X	X
ProximitySensor				X
ProximitySensor2D			X	X
ROUTE			X	X
ScalarInterpolator			X	X
SphereSensor				X
TermCap			X	X
TimeSensor			X	X
TouchSensor			X	X
VisibilitySensor				X
Valuator			X	X

Table 6 lists the BIFS nodes for the scene graph profiles. Also this table was based on [6]; for an explanation of the table and the BIFS nodes, see either [6], or [14] in this issue.

For a few BIFS nodes, their presence is not just inferred from the scene graph profile but follows logically from the audio and visual profiles that the terminal implements. For instance, any audio profile would require AudioClip and AudioSource to be present, and the Core visual profile will require the Texture, Background 2D, Background and MovieTexture BIFS nodes. The presence of these nodes can always be inferred from the chosen audio and visual profiles, and the Systems part of the MPEG-4 standard provides a few tables that list these required nodes.

4.2.2. Object descriptor profiles

The last type of profile is the object descriptor profile. According to the Systems of the MPEG-4 standard [6], such a profile specifies allowed configurations of the object descriptor and sync layer tools. The object descriptor contains all descriptive information, while the sync layer tool provides the syntax to convey, among others, timing information for elementary streams. The main reason for wanting to subject the object descriptor to profiling lies reducing the amount of asynchronous operations and the necessary permanent storage. Currently, only one, default object descriptor profile exists, termed **Core**. It was created to allow the creation of levels, which in turn is necessary to reduce, e.g., the amount of different time bases that a system needs to support simultaneously. No levels have been defined yet, however. This may happen in Version 2 of the MPEG-4 standard.

5. Version 2 and beyond

Profiling in Version 1 has proved difficult, mostly because the tool set is so wide and diverse. It is possible that not every defined profile will be used, notably in the visual area. MPEG has decided to be very strict in the definition of new profiles. Companies asking for a new profile are required to clearly state that they want to use it for real products. Companies are also required to show that the already existing profiles are not suitable for the target application. New profiles are currently under consideration. In the audio area, probably one or more new, error resilient profiles will be defined. In the visual area, new profiles will likely include one targeted at real-time operation in error-prone environments, one or more profiles with additional scalability and perhaps a new profile, intended for high-end services, including multimedia broadcast.

Work in MPEG is ongoing for amendments (additions) to the standard, beyond Version 2. MPEG has started research on MPEG-4 for Studio applications, notably in the visual area, which requires considerably higher bit-rates than are currently supported. If this work is indeed continued (and there is every reason to believe that it will) then several new (visual) profiles are anticipated. Another research item probably leading to one or more new visual profiles is visual fine-grain scalability, a much desired feature already present in MPEG-4 audio.

6. Conclusion

MPEG-4 profiles and levels are meant for interoperability and conformance checking. MPEG-4 profiles start from MPEG-2 principles, consistently applied to all parts of the standard. The fact that an MPEG-4 scene is potentially composed of multiple objects implies that profiles and levels must give bounds for the total of objects in the scene rather than for individual ones. MPEG-4 profiling is not complete yet; more profiles are expected, although MPEG will not add profiles liberally – there must be plans for deploying new profiles and levels before they can get accepted and standardised.

Acknowledgements

The author would like to thank the MPEG community for the interesting discussions and meetings, that led to this paper. Special thanks go to Fernando Pereira, for his tireless efforts with respect to MPEG Profiles and for his thorough review of and contributions to this paper.

References

[1] E. Ebrahimi, C. Horne, MPEG-4 natural video coding – An overview, Signal Processing: Image Communication 15 (4–5) (2000) 365–385.
[2] C. Herpel, A. Eleftheriadis, MPEG-4 systems: Elementary stream management, Signal Processing: Image Communication 15 (4–5) (2000) 299–320.
[3] ISO/IEC, ISO/IEC International Standard 13813-2, MPEG-2 Video, Spring 1999.
[4] ISO/IEC, ISO/IEC International Standard 14469-2, MPEG-4 Visual, Spring 1999.
[5] ISO/IEC, ISO/IEC International Standard 14469-3, MPEG-4 Audio, Spring 1999.
[6] ISO/IEC, ISO/IEC International Standard 14469-1, MPEG-4 Systems, Spring 1999.
[7] ISO/IEC, ISO/IEC International Standard 14469-1, MPEG-4 DMIF, Spring 1999.
[8] R.H. Koenen, MPEG-4 – Multimedia for our time, IEEE Spectrum 36 (2) (February 1999) 26–33.
[9] R. Koenen, F. Pereira, L. Chiariglione, MPEG-4: Context and objectives, Signal Processing: Image Communication 9 (4) (1997) 295–304.
[10] MPEG Requirements Group, MPEG-4 Version Management Procedures, Doc. ISO/MPEG N2200. MPEG Tokyo Meeting, March 1998.
[11] MPEG Requirements Group, MPEG-4 Profiling Policy, Doc. ISO/MPEG N2565. MPEG Roma Meeting, December 1998.
[12] MPEG Requirements Group, MPEG-4 Requirements Document, Doc. ISO/MPEG N2723. MPEG Seoul Meeting, March 1999.
[13] E.D. Scheirer, Y. Lee, J.-W. Yang, Synthetic and SNHC audio in MPEG-4, Signal Processing: Image Communication 15 (4–5) (2000) 445–461.
[14] J. Signès, Y. Fisher, A. Eleftheriadis, MPEG-4's binary format for scene description, Signal Processing: Image Communication 15 (4–5) (2000) 321–345.
[15] www.midi.org

Rob Koenen received his 'Ingenieur' (MSEE) degree from Delft University of Technology, the Netherlands, in 1989. He studied electrical engineering, specialising in information theory. After having worked for this university for one year, developing knowledge-based advisory systems, he joined the Video Coding group of KPN Research in 1990. There he has researched various aspects of audiovisual communication, working as a project manager and later coordinator of the group. His projects have addressed: image coding research, audiovisual communication for the hard-of-hearing and for elderly people, interactive broadband multimedia services for residential users, mobile multimedia services, the strategic deployment of new multimedia services and more recently subjective and objective methods of audiovisual quality assessment. He has participated in several European collaborative research projects dealing with MPEG technology, and was a member of the Management Committee of COST 211 ter for several years. As an MPEG delegate, he has played a key role in the development of the MPEG-4 standard since 1993, and in defining the upcoming MPEG-7 standard since 1995.

Ir. Koenen now works as a senior advisor/project leader with the Multimedia Technology group of KPN Research. He is the chairman of the MPEG Requirements subgroup. He is also an associate editor of the IEEE Transactions on Circuits and Systems for Video Technology.