INTERNATIONAL STANDARD

ISO/IEC 14496-3

First edition
1999-12-15

Information technology — Coding of audio-visual objects —

Part 3: **Audio**

Technologies de l'information — Codage des objets audiovisuels —

Partie 3: Codage audio

Reference number
ISO/IEC 14496-3:1999(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 14496 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14496-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

— *Part 1: Systems*

— *Part 2: Visual*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Reference testing*

— *Part 6: Delivery Multimedia Integration Framework (DMIF)*

Annexes 2.A to 2.C, 3.C, 4.A and 5.A form a normative part of this part of ISO/IEC 14496. Annexes 1.A, 1.B, 2.D, 3.A, 3.B, 3.D to 3.F, 4.B and 5.B to 5.G are for information only.

Due to its technical nature, this part of ISO/IEC 14496 requires a special format as several standalone electronic files and, consequently, does not conform to some of the requirements of the ISO/IEC Directives, Part 3.

Information technology — Coding of audio-visual objects —

Part 3: Audio

Subpart 1: Main

Structure of this part of ISO/IEC 14496:

This part of ISO/IEC 14496 comprises six subparts:

Subpart 1:	Main
Subpart 2:	Speech coding - HVXC
Subpart 3:	Speech coding - CELP
Subpart 4:	General Audio coding (GA)
Subpart 5:	Structured audio
Subpart 6:	Text to speech interface

For reasons of manageability of large documents, this part of ISO/IEC 14496 is divided into six files, corresponding to the six subparts of the standard:

1. a025035e.pdf contains Subpart 1.
2. b025035e.pdf contains Subpart 2.
3. c025035e.pdf contains Subpart 3.
4. d025035e.pdf contains Subpart 4.
5. e025035e.pdf contains Subpart 5.
6. f025035e.pdf contains Subpart 6.

Contents for Subpart 1

Subpart 1: Main

1.1 Scope

1.1.1 Overview of MPEG-4 Audio

This part of ISO/IEC 14496 (MPEG-4 Audio) is a new kind of audio standard that integrates many different types of audio coding: natural sound with synthetic sound, low bitrate delivery with high-quality delivery, speech with music, complex soundtracks with simple ones, and traditional content with interactive and virtual-reality content. By standardizing individually sophisticated coding tools as well as a novel, flexible framework for audio synchronization, mixing, and downloaded post-production, the developers of the MPEG-4 Audio standard have created new technology for a new, interactive world of digital audio.

MPEG-4, unlike previous audio standards created by ISO/IEC and other groups, does not target a single application such as real-time telephony or high-quality audio compression. Rather, MPEG-4 Audio is a standard that applies to *every* application requiring the use of advanced sound compression, synthesis, manipulation, or playback. The subparts that follow specify the state-of-the-art coding tools in several domains; however, MPEG-4 Audio is more than just the sum of its parts. As the tools described here are integrated with the rest of the MPEG-4 standard, exciting new possibilities for object-based audio coding, interactive presentation, dynamic soundtracks, and other sorts of new media, are enabled.

Since a single set of tools is used to cover the needs of a broad range of applications, *interoperability* is a natural feature of systems that depend on the MPEG-4 Audio standard. A system that uses a particular coder—for example, a real-time voice communication system making use of the MPEG-4 speech coding toolset—can easily share data and development tools with other systems, even in different domains, that use the same tool—for example, a voicemail indexing and retrieval system making use of MPEG-4 speech coding. A multimedia terminal that can decode the Main Profile of MPEG-4 Audio has audio capabilities that cover the entire spectrum of audio functionality available today and in the future.

The remainder of this clause gives a more detailed overview of the capabilities and functioning of MPEG-4 Audio. First, a discussion of concepts that have changed since the MPEG-2 audio standards is presented; then, the MPEG-4 Audio toolset is outlined.

1.1.2 New concepts in MPEG-4 Audio

Many concepts in MPEG-4 Audio are different than those in previous MPEG Audio standards. For the benefit of readers who are familiar with MPEG-1, MPEG-2, and MPEG-AAC, we provide a brief overview here.

- **MPEG-4 has no standard for transport.** In all of the MPEG-4 tools for audio and visual coding, the coding standard ends at the point of constructing a sequence of *access units* that contain the compressed data. The MPEG-4 Systems (ISO/IEC 14496-1) specification describes how to convert the individually coded objects into a bitstream that contains a number of multiplexed sub-streams.

  There is no standard mechanism for transport of this stream over a channel; this is because the broad range of applications that can make use of MPEG-4 technology have delivery requirements that are too wide to easily characterize with a single solution. Rather, what is standardized is an *interface* (the Delivery Multimedia Interface Format, or DMIF, specified in ISO/IEC 14496-6) that describes the capabilities of a transport layer and the communication between transport, multiplex, and demultiplex functions in encoders and decoders. The use of DMIF and the MPEG-4 Systems bitstream specification allows transmission functions that are much more sophisticated than are possible with previous MPEG standards.

  For applications which do not require sophisticated transport functionality, object-based coding, synchronization with other media, or other functions provided by MPEG-4 Systems, a private, not normative transport may be used to deliver a single MPEG-4 Audio stream. An example private transport for this purpose is given in Informative Annex A of subpart 1.

- **MPEG-4 Audio encourages low-bitrate coding.** Previous MPEG Audio standards have focused primarily on transparent (undetectable) or nearly transparent coding of high-quality audio at whatever bitrate was required to provide it. MPEG-4 provides new and improved tools for this purpose, but also standardizes (and has tested) tools that can be used for transmitting audio at the low bitrates suitable for Internet, digital radio, or other bandwidth-limited delivery. The tools specified in MPEG-4 are the state-of-the-art tools available for low-bitrate coding of speech and other audio.

- **MPEG-4 is an object-based coding standard with multiple tools.** Previous MPEG Audio standards provided a single toolset, with different configurations of that toolset specified for use in various applications. MPEG-4 provides several toolsets that have no particular relationship to each other, each with a different target function. The Profiles of MPEG-4 Audio (subclause 5.1) specify which of these tools are used together for various applications.

  Further, in previous MPEG standards, a single (perhaps multi-channel or multi-language) piece of content was all that was transmitted. In MPEG-4, by contrast, the concept of a *soundtrack* is much more flexible. Multiple tools may be used to transmit several *audio objects*; when using multiple tools together, an *audio composition* system is used to create a single soundtrack from the audio substreams. User interaction, terminal capability, and speaker configuration may be used when determining how to produce a single soundtrack from the component objects. This capability allows significant advantages in quality and flexibility in MPEG-4 over previous audio standards.

- **MPEG-4 provides capabilities for synthetic sound.** In natural sound coding, an existing sound is compressed by a server, transmitted and decompressed at the receiver. This type of coding is the subject of many existing standards for sound compression. MPEG-4 also standardizes a novel paradigm in which synthetic sound descriptions, including synthetic speech and synthetic music, are transmitted and then *synthesized* into sound at the receiver. Such capabilities open up new areas of very-low-bitrate but still very-high-quality coding.

As with previous MPEG standards, MPEG-4 does not standardize methods for encoding sound. Thus, content authors are left to their own decisions for the best method of creating bitstreams. At the present time, it is an open problem how to automatically convert natural sound into synthetic or multi-object descriptions; therefore, most immediate solutions will involve hand-authoring the content stream in some way. This process is similar to current schemes for MIDI-based and multi-channel mixdown authoring of soundtracks.

1.1.3 MPEG-4 Audio capabilities

1.1.3.1 Overview of capabilities

The MPEG-4 Audio tools can be broadly organized into several categories:

1. *Speech* tools for the transmission and decoding of synthetic and natural speech
2. *Audio* tools for the transmission and decoding of recorded music and other audio soundtracks
3. *Synthesis* tools for very low bitrate description and transmission, and terminal-side synthesis, of synthetic music and other sounds
4. *Composition* tools for object-based coding, interactive functionality, and audiovisual synchronization
5. *Scalability* tools for the creation of bitstreams that can be transmitted, without recoding, at several different bitrates

Each of these types of tools will be described in more detail in the following subclauses.

1.1.3.2 MPEG-4 speech coding tools

1.1.3.2.1 Introduction

Two types of speech coding tools are provided in MPEG-4. The *natural* speech tools allow the compression, transmission, and decoding of human speech, for use in telephony, personal communication, and surveillance applications. The *synthetic* speech tool provides an interface to text-to-speech synthesis systems; using synthetic speech provides very-low-bitrate operation and built-in connection with facial animation for use in low-bitrate videoteleconferencing applications. Each of these tools will be discussed.

1.1.3.2.2 Natural speech coding

The MPEG-4 speech coding toolset covers the compression and decoding of natural speech sound at bitrates ranging between 2 and 24 kbit/s. When the variable bitrate coding is allowed, coding at even less than 2 kbit/s, such as average bitrate of 1.2 kbit/s, is also supported. Two basic speech coding techniques are used: One is a parametric speech coding algorithm, HVXC (Harmonic Vector eXcitation Coding), for very low bit rates; and the other is a CELP (Code Excited Linear Prediction) coding technique. The MPEG-4 speech coder targets applications from mobile and satellite communications, to Internet telephony, to packaged media and speech databases. It meets a wide range of requirements covering bitrates, functionality and sound quality and is specified in subparts 2 and 3.

MPEG-4 HVXC operates at fixed bitrates between 2.0 kbit/s and 4.0 kbit/s, using a bitrate scalability technique. It also operates at lower bitrates, typically 1.2-1.7 kbit/s, in variable bitrate mode. HVXC provides communications-quality to near-toll-quality speech in the 100-3800 Hz band at 8kHz sampling rate. HVXC also allows independent change of speed and pitch during decoding, which is a powerful functionality for fast access to speech databases.

MPEG-4 CELP is a well-known coding algorithm with new functionality. Conventional CELP coders offer compression at a single bit rate and are optimized for specific applications. Compression is one of the functionalities provided by MPEG-4 CELP, but MPEG-4 also enables the use of one basic coder in multiple applications. It provides scalability in bitrate and bandwidth, as well as the ability to generate bitstreams at arbitrary bitrates. The MPEG-4 CELP coder supports two sampling rates, namely, 8 and 16 kHz. The associated bandwidths are 100 – 3800 Hz for 8 kHz sampling and 50 – 7000 Hz for 16 kHz sampling.

MPEG has conducted extensive verification testing in realistic listening conditions in order to prove the efficacy of the speech coding toolset.

1.1.3.2.3 Text-to-speech interface

Text-to-speech (TTS) capability is becoming a rather common media type and plays an important role in various multi-media application areas. For instance, by using TTS functionality, multimedia content with narration can be easily created without recording natural speech sound. Before MPEG-4, however, there was no way for a multimedia content provider to easily give instructions to an unknown TTS system. In MPEG-4, a single common interface for TTS systems is standardized. This interface allows speech information to be transmitted in the International Phonetic Alphabet (IPA), or in a textual (written) form of any language. It is specified in subpart 6.

The MPEG-4 TTS package, Hybrid/Multi-Level Scalable TTS Interface, can be considered as a superset of the conventional TTS framework. This extended TTS Interface can utilize prosodic information taken from natural speech in addition to input text and can thus generate much higher-quality synthetic speech. The interface and its bitstream format is strongly scalable in terms of this added information; for example, if some parameters of prosodic information are not available, a decoder can generate the missing parameters by rule. Normative algorithms for speech synthesis and text-to-phoneme translation are not specified in MPEG-4, but to meet the goal that underlies the MPEG-4 TTS Interface, a decoder should fully utilize all the provided information according to the user's requirements level.

As well as an interface to Text-to-speech synthesis systems, MPEG-4 specifies a joint coding method for phonemic information and facial animation (FA) parameters and other animation parameters (AP). Using this technique, a single bitstream may be used to control both the Text-to-Speech Interface and the Facial Animation visual object decoder (see ISO/IEC 14496-2 Annex C). The functionality of this extended TTS thus ranges from conventional TTS to natural speech coding and its application areas, from simple TTS to audio presentation with TTS and motion picture dubbing with TTS.

1.1.3.3 MPEG-4 general audio coding tools

MPEG-4 standardizes the coding of natural audio at bitrates ranging from 6 kbit/s up to several hundred kbit/s per audio channel for mono, two-channel-, and multi-channel-stereo signals. General high-quality compression is provided by the use of the MPEG-2 AAC standard (ISO/IEC 13818-7), with certain improvements, within the MPEG-4 tool set. At 64 kbit/s/channel and higher ranges, this coder has been found in verification testing under rigorous conditions to meet the criterion of "indistinguishable quality" as defined by the European Broadcasting Union.

Subpart 4 of MPEG-4 specifies the AAC tool set, in the General Audio coder. This coding technique uses a perceptual filterbank, a sophisticated masking model, noise-shaping techniques, channel coupling, and noiseless coding and bit-allocation to provide the maximum compression within the constraints of providing the highest possible quality. Psychoacoustic coding standards developed by MPEG have represented the state-of-the-art in this technology for nearly 10 years; MPEG-4 General Audio coding continues this tradition.

For bitrates from 6 kbit/s up to 64 kbit/s per channel, the MPEG-4 standard provides extensions to AAC and the TwinVQ tools that allow the content author to achieve highest quality by altering the tool used depending on the bit rate. Furthermore, various bit rate scalability options are available within the GA coder (see subclause 1.1.3.6.). The low-bitrate techniques and scalability modes provided with this tool set have also been verified in formal tests by MPEG.

1.1.3.4 MPEG-4 Audio synthesis tools

The MPEG-4 toolset providing general audio synthesis capability is called MPEG-4 Structured Audio, and it is described in subpart 5 of ISO/IEC 14496-3. (There is also a tool for the transmission of synthetic speech; it is

described above in subclause 1.2.2 and in subpart 6). MPEG-4 Structured Audio (the SA coder) provides very general capabilities for the description of synthetic sound, and the normative creation of synthetic sound in the decoding terminal. High-quality stereo sound can be transmitted at bitrates from 0 kbit/s (no continuous cost) to 2-3 kbit/s for extremely expressive sound using these tools.

Rather than specify a particular method of synthesis, SA specifies a flexible language for describing methods of synthesis. This technique allows content authors two advantages. First, the set of synthesis techniques available is not limited to those that were envisioned as useful by the creators of the standard; any current or future method of synthesis may be used in MPEG-4 Structured Audio. Second, the creation of synthetic sound from structured descriptions is normative in MPEG-4, so sound created with the SA coder will sound the same on any terminal.

Synthetic audio is transmitted via a set of *instrument* modules that can create audio signals under the control of a *score*. An instrument is a small network of signal-processing primitives that control the parametric generation of sound according to some algorithm. Several different instruments may be transmitted and used in a single Structured Audio 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. The format for the description of instruments—SAOL, the Structured Audio Orchestra Language—and that for the description of scores—SASL, the Structured Audio Score Language—are specified in subpart 6.

Efficient transmission of sound samples, also called *wavetables*, for use in sampling synthesis is accomplished by providing interoperability with the MIDI Manufacturers Association Downloaded Sounds Level 2 (DLS-2) standard, which is normatively referenced by the Structured Audio standard. By using the DLS-2 format, the simple and popular technique of wavetable synthesis can be used in MPEG-4 Structured Audio soundtracks, either by itself or in conjunction with other kinds of synthesis using the more general-purpose tools. To further enable interoperability with existing content and authoring tools, the popular MIDI (Musical Instrument Digital Interface) control format can be used instead of, or in addition to, scores in SASL for controlling synthesis.

Through the inclusion of compatibility with MIDI standards, MPEG-4 Structured Audio thus represents a unification of the current technique for synthetic sound description (MIDI-based wavetable synthesis) with that of the future (general-purpose algorithmic synthesis). The resulting standard solves problems not only in very-low-bitrate coding, but also in virtual environments, video games, interactive music, karaoke systems, and many other applications.

1.1.3.5 MPEG-4 Audio composition tools

The tools for audio composition, like those for visual composition, are specified in the MPEG-4 Systems standard (ISO/IEC 14496-1). However, since readers interested in audio functionality are likely to look here first, a brief overview is provided.

Audio composition is the use of multiple individual “audio objects” and mixing techniques to create a single soundtrack. It is analogous to the process of recording a soundtrack in a multichannel mix, with each musical instrument, voice actor, and sound effect on its own channel, and then “mixing down” the multiple channels to a single channel or single stereo pair. In MPEG-4, the multichannel mix itself may be transmitted, with each audio source using a different coding tool, and a set of instructions for mixdown also transmitted in the bitstream. As the multiple audio objects are received, they are decoded separately, but not played back to the listener; rather, the instructions for mixdown are used to prepare a single soundtrack from the “raw material” given in the objects. This final soundtrack is then played for the listener.

An example serves to illustrate the efficacy of this approach. Suppose, for a certain application, we wish to transmit the sound of a person speaking in a reverberant environment over stereo background music, at very high quality. A traditional approach to coding would demand the use of a general audio coding at 32 kbit/s/channel or above; the sound source is too complex to be well-modeled by a simple model-based coder. However, in MPEG-4 we can represent the soundtrack as the conjunction of several objects: a **speaking person** passed through a **reverberator** added to a **synthetic music track**. We transmit the speaker’s voice using the CELP tool at 16 kbit/s, the synthetic music using the SA tool at 2 kbit/s, and allow a small amount of overhead (only a few hundreds of bytes as a fixed cost) to describe the stereo mixdown and the reverberation. Using MPEG-4 and an object-based approach thus allows us to describe in less than 20 kbit/s total a bitstream that might require 64 kbit/s to transmit with traditional coding, at equivalent quality.

Additionally, having such structured soundtrack information present in the decoding terminal allows more sophisticated client-side interaction to be included. For example, the listener can be allowed (if the content author desires) to request that the background music be muted. This functionality would not be possible if the music and speech were coded into the same audio track.

With the MPEG-4 Binary Format for Scenes (BIFS), specified in MPEG-4 Systems, a subset tool called AudioBIFS allows content authors to describe sound scenes using this object-based framework. Multiple sources may be mixed and combined, and interactive control provided for their combination. Sample-resolution control over mixing is provided in this method. Dynamic download of custom signal-processing routines allows the content author to exactly request a particular, normative, digital filter, reverberator, or other effects-processing routine. Finally, an interface to terminal-dependent methods of 3-D audio spatialisation is provided for the description of virtual-reality and other 3-D sound material.

As AudioBIFS is part of the general BIFS specification, the same framework is used to synchronize audio and video, audio and computer graphics, or audio with other material. Please refer to ISO/IEC 14496-1 (MPEG-4 Systems) for more information on AudioBIFS and other topics in audiovisual synchronization.

1.1.3.6 MPEG-4 Audio scalability tools

Many of the bitstream types in MPEG-4 are *scalable* in one manner or another. Several types of scalability in the standard are discussed below.

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 bitstream parsing can occur either during transmission or in the decoder. Scalability is available within each of the natural audio coding schemes, or by a combination of different natural audio coding schemes.

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. This is available for the CELP speech coder, where an extension layer converts the narrow band base layer encoder into a wide band speech coder. Also the general audio coding tools which all operate in the frequency domain offer a very flexible bandwidth control for the different coding layers.

Encoder complexity scalability allows encoders of different complexity to generate valid and meaningful bitstreams. An example for this is the availability of a high quality and a low complexity excitation module for the wideband CELP coder allowing to choose between significant lower encoder complexity or optimized coding quality.

Decoder complexity scalability allows a given bitstream to be decoded by decoders of different levels of complexity. A subtype of decoder complexity scalability is *graceful degradation*, in which a decoder dynamically monitors the resources available, and scales down the decoding complexity (and thus the audio quality) when resources are limited. The Structured Audio decoder allows this type of scalability; a content author may provide (for example) several different algorithms for the synthesis of piano sounds, and the content itself decides, depending on available resources, which one to use.

1.2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 14496. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 14496 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 11172-3:1993, *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 3: Audio*.
ITU-T Rec. H.222.0(1995) | ISO/IEC 13818-1:1996, *Information technology - Generic coding of moving pictures and associated audio information: Systems.*
ISO/IEC 13818-3:1998, *Information technology - Generic coding of moving pictures and associated audio information - Part 3: Audio*.
ISO/IEC 13818-7:1997, *Information technology - Generic coding of moving pictures and associated audio information - Part 7: Advanced Audio Coding (AAC).*
(c) 1996 MIDI Manufacturers Association, *The Complete MIDI 1.0 Detailed Specification* v. 96.2.
(c) 1998 MIDI Manufacturers Association, *The MIDI Downloadable Sounds Specification, v. 98.2*.

1.3 Terms and definitions

For the purposes of this part of ISO 14496, the following terms and definitions apply.

1.3.1 AAC Advanced Audio Coding.
1.3.2 API: Application Programming Interface.
1.3.3 AudioBIFS: The set of tools specified in ISO/IEC 14496-1 (MPEG-4 Systems) for the composition of audio data in interactive scenes.
1.3.4 audio buffer: A buffer in the system target decoder (see ISO/IEC 13818-1) for storage of compressed audio data.
1.3.5 backward compatibility: A newer coding standard is backward compatible with an older coding standard if decoders designed to operate with the older coding standard are able to continue to operate by decoding all or part of a bitstream produced according to the newer coding standard.
1.3.6 bitrate: The rate at which the compressed bitstream is delivered to the input of a decoder.
1.3.7 bitstream; stream: An ordered series of bits that forms the coded representation of the data.
1.3.8 bitstream verifier: A process by which it is possible to test and verify that all the requirements specified in ISO/IEC 14496-3 are met by the bitstream.
1.3.9 byte aligned: A bit in a coded bitstream is byte-aligned if its position is a multiple of 8-bits from either the first bit in the stream for the Audio_Data_Interchange_Format (see subclause 1.A.2.1) or the first bit in the syncword for the Audio_Data_Transport_Stream Format (see subclause 1.A.2.2).
1.3.10 byte: Sequence of 8-bits.
1.3.11 CELP: Code Excited Linear Prediction.
1.3.12 channel: A sequence of data representing an audio signal intended to be reproduced at one listening position.
1.3.14 coded audio bitstream: A coded representation of an audio signal.
1.3.15 coded representation: A data element as represented in its encoded form.
1.3.16 composition (compositing): Using a scene description to mix and combine several separate audio tracks into a single presentation.
1.3.17 compression: Reduction in the number of bits used to represent an item of data.
1.3.18 constant bitrate: Operation where the bitrate is constant from start to finish of the coded bitstream.
1.3.19 CRC: The Cyclic Redundancy Check to verify the correctness of data.
1.3.20 data element: An item of data as represented before encoding and after decoding.
1.3.21 de-emphasis: Filtering applied to an audio signal after storage or transmission to undo a linear distortion due to emphasis.
1.3.22 decoded stream: The decoded reconstruction of a compressed bitstream.
1.3.23 decoder: An embodiment of a decoding process.
1.3.24 decoding (process): The process that reads an input coded bitstream and outputs decoded audio samples.
1.3.25 digital storage media; DSM: A digital storage or transmission device or system.
1.3.26 downmix: A matrixing of n channels to obtain less than n channels.
1.3.27 editing: The process by which one or more coded bitstreams are manipulated to produce a new coded bitstream. Conforming edited bitstreams must meet the requirements defined in this part of ISO/IEC 14496.
1.3.28 emphasis: Filtering applied to an audio signal before storage or transmission to improve the signal-to-noise ratio at high frequencies.
1.3.29 encoder: An embodiment of an encoding process.
1.3.30 encoding (process): A process, not specified in ISO/IEC 14496, that reads a stream of input audio samples and produces a valid coded bitstream as defined in this part of ISO/IEC 14496.
1.3.31 entropy coding: Variable length lossless coding of the digital representation of a signal to reduce statistical redundancy.
1.3.32 elementary stream (ES): 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 AVObject or a Control Entity in the receiving MPEG-4 Terminal. It flows through one FlexMux Channel.
1.3.33 FFT: Fast Fourier Transformation. A fast algorithm for performing a discrete Fourier transform (an orthogonal transform).
1.3.34 flag: A variable which can take one of only the two values defined in this specification.
1.3.35 forward compatibility: A newer coding standard is forward compatible with an older coding standard if decoders designed to operate with the newer coding standard are able to decode bitstreams of the older coding standard.
1.3.36 Fs: Sampling frequency.
1.3.37 Hann window: A time function applied sample-by-sample to a block of audio samples before Fourier transformation.
1.3.38 HVXC: Harmonic Vector Excitation Coding (very low bit rate speech)
1.3.39 Huffman coding: A specific method for entropy coding.
1.3.40 LPC Linear Predictive Coding.
1.3.41 LSP Line Spectrum Pair.
1.3.41 LTP: Long Term Prediction.

1.3.42 low frequency enhancement (LFE) channel: A limited bandwidth channel for low frequency audio effects in a multichannel system.
1.3.43 MATS: MPEG-4 Audio Transport Stream.
1.3.44 MIDI: The Musical Instrument Digital Interface standards. Certain aspects of the MPEG-4 Structured Audio tools provide interoperability with MIDI standards.
1.3.45 multichannel: A combination of audio channels used to create a spatial sound field.
1.3.46 multilingual: A presentation of dialogue in more than one language.
1.3.47 Nyquist sampling: Sampling at or above twice the maximum bandwidth of a signal.
1.3.48 OD: Object Descriptor.
1.3.49 padding: A method to adjust the average length of an audio frame in time to the duration of the corresponding PCM samples, by conditionally adding a slot to the audio frame.
1.3.50 parameter: A variable within the syntax of this specification which may take one of a range of values. A variable which can take one of only two values is a flag or indicator and not a parameter.
1.3.51 parser: Functional stage of a decoder which extracts from a coded bitstream a series of bits representing coded elements.
1.3.52 PNS: Perceptual Noise Substitution.
1.3.53 prediction error: The difference between the actual value of a sample or data element and its predictor.
1.3.54 prediction: The use of a predictor to provide an estimate of the sample value or data element currently being decoded.
1.3.55 predictor: A linear combination of previously decoded sample values or data elements.
1.3.56 presentation channel: An audio channel at the output of the decoder.
1.3.57 PSNR: Peak Signal to Noise Ratio.
1.3.58 random access: The process of beginning to read and decode the coded bitstream at an arbitrary point.
1.3.59 reserved: The term "reserved" when used in the clauses defining the coded bitstream indicates that the value may be used in the future for ISO/IEC defined extensions.
1.3.60 Sampling Frequency (Fs): Defines the rate in Hertz which is used to digitize an audio signal during the sampling process.
1.3.61 SAOL: The MPEG-4 Structured Audio Orchestra language, used to transmit the description of synthesis algorithms in MPEG-4.
1.3.62 SASBF: The MPEG-4 Structured Audio Sample Bank Format, an efficient format for the transmission of blocks of wavetable (sample data) compatible with the MIDI method for the same.
1.3.63 SASL: The MPEG-4 Structured Audio Score Language, used to transmit synthesis control parameters in MPEG-4.
1.3.64 side information: Information in the bitstream necessary for controlling the decoder.
1.3.65 Structured audio: Audio coding by means of transmitting descriptions that are synthesized into sound as they are received. See subpart 5.
1.3.65 stuffing (bits); stuffing (bytes): Code-words that may be inserted at particular locations in the coded bitstream that are discarded in the decoding process. Their purpose is to increase the bitrate of the stream which would otherwise be lower than the desired bitrate.
1.3.66 surround channel: An audio presentation channel added to the front channels (L and R or L, R, and C) to enhance the spatial perception.
1.3.67 syncword: A code embedded in the audio transport bitstreams defined in the Annex 1.A that identifies the start of a transport frame.
1.3.68 TLSS: Tools for Large Step Scalability. The TLSS tools comprise the frequency selective switch (FSS) tool and the upsampling filter tool, described in subpart 4. In addition, all other methods, which are required to implement all the scalability modes of the generic audio coder, as defined in subpart 4, are also included in the TLSS tools.
1.3.68 TTSI: Text to Speech Interface.
1.3.69 TwinVQ Transform domain Weighted Interleave Vector Quantization.
1.3.70 variable bitrate: Operation where the bitrate varies with time during the decoding of a coded bitstream.
1.3.71 variable length coding: A reversible procedure for coding that assigns shorter code-words to frequent symbols and longer code-words to less frequent symbols.
1.3.72 variable length code (VLC): A code word assigned by variable length encoder (See variable length coding).
1.3.73 variable length decoder: A procedure to obtain the symbols encoded with a variable length coding technique.
1.3.74 variable length encoder: A procedure to assign variable length codewords to symbols.

1.4 Symbols and abbreviations

The mathematical operators used in this part of ISO/IEC 14496 are similar to those used in the C programming language. However, integer division with truncation and rounding are specifically defined. The bitwise operators are

defined assuming two's-complement representation of integers. Numbering and counting loops generally begin from zero.

1.4.1 Arithmetic operators

+	Addition.
–	Subtraction (as a binary operator) or negation (as a unary operator).
++	Increment.
– –	Decrement.
*	Multiplication.
^	Power.
/	Integer division with truncation of the result toward zero. For example, 7/4 and –7/–4 are truncated to 1 and –7/4 and 7/–4 are truncated to –1.
//	Integer division with rounding to the nearest integer. Half-integer values are rounded away from zero unless otherwise specified. For example 3//2 is rounded to 2, and –3//2 is rounded to –2.
DIV	Integer division with truncation of the result towards $-\infty$.
| |	Absolute value. | x | = x when x > 0 | x | = 0 when x == 0 | x | = –x when x < 0
%	Modulus operator. Defined only for positive numbers.
Sign( )	Sign. Sign(x) = 1 when x > 0 Sign(x) = 0 when x == 0 Sign(x) = –1 when x < 0
INT ( )	Truncation to integer operator. Returns the integer part of the real-valued argument.
NINT ( )	Nearest integer operator. Returns the nearest integer value to the real-valued argument. Half-integer values are rounded away from zero.
sin	Sine.
cos	Cosine.
exp	Exponential.
√	Square root.
$\log_{10}$	Logarithm to base ten.
$\log_e$	Logarithm to base e.
$\log_2$	Logarithm to base 2.

1.4.2 Logical operators

||	Logical OR.
&&	Logical AND.
!	Logical NOT

1.4.3 Relational operators

>	Greater than.
>=	Greater than or equal to.
<	Less than.
<=	Less than or equal to.
==	Equal to.
!=	Not equal to.
max [,...,]	the maximum value in the argument list.
min [,...,]	the minimum value in the argument list.

1.4.4 Bitwise operators

A twos complement number representation is assumed where the bitwise operators are used.

&	AND
|	OR
>>	Shift right with sign extension.
<<	Shift left with zero fill.

1.4.5 Assignment

=	Assignment operator.

1.4.6 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bitstream.

bslbf	Bit string, left bit first, where "left" is the order in which bit strings are written in ISO/IEC 11172. Bit strings are written as a string of 1s and 0s within single quote marks, e.g. '1000 0001'. Blanks within a bit string are for ease of reading and have no significance.
L, C, R, LS, RS	Left, center, right, left surround and right surround audio signals
rpchof	Remainder polynomial coefficients, highest order first. (Audio)
uimsbf	Unsigned integer, most significant bit first.
vlclbf	Variable length code, left bit first, where "left" refers to the order in which the VLC codes are written.
window	Number of the actual time slot in case of block_type==2, 0 <= window <= 2. (Audio)

The byte order of multi-byte words is most significant byte first.

1.4.7 Constants

π	3,14159265358...
e	2,71828182845...

1.4.8 Method of describing bitstream syntax

The bitstream retrieved by the decoder is described in the syntax section of each subpart. Each data item in the bitstream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bitstream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and the definition of the state variables used in their decoding are described in in the sections following the syntax section of each subpart. The following constructs are used to express the conditions when data elements are present, and are in normal type:

Note this syntax uses the 'C'-code convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true.

while (condition) {
  data_element
  . . .
}
	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.

do {
  data_element
  . . .
} while (condition)
	The data element always occurs at least once. The data element is repeated until the condition is not true.

if (condition) {
  data_element
  . . .
}
else {
  data_element
  . . .
}
	If the condition is true, then the first group of data elements occurs next in the data stream.
	If the condition is not true, then the second group of data elements occurs next in the data stream.

for (expr1; expr2; expr3)
{
  data_element
  . . .
}
	Expr1 is an expression specifying the initialisation of the loop. Normally it specifies the initial state of the counter. Expr2 is a condition specifying a test made before each iteration of the loop. The loop terminates when the condition is not true. Expr3 is an expression that is performed at the end of each iteration of the loop, normally it increments a counter.

Note that the most common usage of this construct is as follows:

for (i = 0; i < n; i++) {
  data_element
  . . .
}
	The group of data elements occurs n times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to zero for the first occurrence, incremented to one for the second occurrence, and so forth.

As noted, the group of data elements may contain nested conditional constructs. For compactness, the {} may be omitted when only one data element follows.

Data_element [ ] data_element [ ] is an array of data. The number of data elements is indicated by the context.
Data_element [n] data_element [n] is the n+1th element of an array of data.
Data_element [m][n] data_element [m][n] is the m+1,n+1 th element of a two-dimensional array of data.
Data_element [l][m][n] data_element [l][m][n] is the l+1,m+1,n+1 th element of a three-dimensional array of data.
data_element [m..n] data_element [m..n]is the inclusive range of bits between bit m and bit n in the data_element.

While the syntax is expressed in procedural terms, it should not be assumed that this implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bitstream. Actual decoders must include a means to deal with incorrect bitstreams and to find the start of the described elements.

Definition of bytealigned function
The function bytealigned ( ) returns 1 if the current position is on a byte boundary, that is the next bit in the bitstream is the first bit in a byte. Otherwise it returns 0.

Definition of nextbits function
The function nextbits ( ) permits comparison of a bit string with the next bits to be decoded in the bitstream.

1.5 Technical overview

1.5.1 MPEG-4 Audio Object Types

1.5.1.1 Audio Object Type Definition

Table 1.5.1 shows the audio coding tools related to a specific object type.

Table 1.5.1 - Audio Object Types

Tools / Audio Object Type	13818-7 main	13818-7 LC	13818-7 SSR	PNS	LTP	TLSS	Twin VQ	CELP	HVXC	TTSI	SA tools	SASBF	MIDI	GA Bitstream Syntax Type	Hierarchy	Object Type ID
Null																**0**
AAC main	X			X										ISO/IEC 13818-7 Style	**contains AAC LC**	**1**
AAC LC		X		X										ISO/IEC 13818-7 Style		**2**
AAC SSR			X	X										ISO/IEC 13818-7 Style		**3**
AAC LTP		X		X	X									ISO/IEC 13818-7 Style	**contains AAC LC**	**4**
Reserved																**5**
AAC Scalable		X		X	X	X								Scalable		**6**
TwinVQ					X		X							Scalable		**7**
CELP								X								**8**
HVXC									X							**9**
Reserved																**10**
Reserved																**11**
TTSI										X						**12**
Main synthetic											X	X	X		**contains W/T & Algor. synthesis**	**13**
Wavetable synthesis												X	X		**contains General MIDI**	**14**
General MIDI													X			**15**
Algorithmic Synthesis and Audio FX											X					**16**
Reserved																**17-31**

1.5.1.2 Description

1.5.1.2.1 NULL Object

The NULL object provides the possibility to feed raw PCM data directly to the audio compositor. No decoding is involved, However, an audio object descriptor is used to specify the sampling rate and the audio channel configuration.

1.5.1.2.2 AAC - Main Object

The AAC Main object is very similar to the AAC Main Profile that is defined in ISO/IEC 13818-7. However, additionally the PNS tool is available. The AAC Main object type bitstream syntax is compatible with the syntax defined in ISO/IEC 13818-7. All the MPEG-2 AAC multi-channel capabilities are available. A decoder capable to decode a MPEG-4 main object stream can also parse and decode a MPEG-2 AAC raw data stream. On the other hand, although a MPEG-2 AAC coder can parse an MPEG-4 AAC Main bitstream, decoding may fail, since PNS might have been used.

1.5.1.2.3 AAC - Low Complexity (LC) Object

The MPEG-4 AAC Low Complexity object type is the counterpart to the MPEG-2 AAC Low Complexity Profile, with exactly the same restrictions as mentioned above for the AAC Main object type.

1.5.1.2.4 AAC - Scalable Sampling Rate (SSR) Object

The MPEG-4 AAC Scalable Sampling Rate object type is the counterpart to the MPEG-2 AAC Scalable Sampling Rate Profile, with exactly the same restrictions as mentioned above for the AAC Main object type.

1.5.1.2.5 AAC - Long Term Predictor (LTP) Object

The MPEG-4 AAC LTP object type is similar to the AAC Main object type. However, a Long Term Predictor replaces the MPEG-2 AAC predictor. The LTP achieves a similar coding gain, but requires significantly lower implementation complexity. The bitstream syntax for this object type is very similar to the syntax defined in ISO/IEC 13818-7. An MPEG-2 AAC LC profile bitstream can be decoded without restrictions using an MPEG-4 AAC LTP object decoder.

1.5.1.2.6 AAC Scalable Object

The scalable AAC object uses a different bitstream syntax to support bitrate- and bandwidth- scalability. A large number of scalable combinations are available, including combinations with TwinVQ and CELP coder tools. However, only mono, or 2-channel stereo objects are supported.

1.5.1.2.7 TwinVQ Object

The TwinVQ object belongs to the GA coding scheme that quantizes the MDCT coefficients. This coding scheme is based on fixed rate vector quantization instead of Huffman coding in AAC.

Low bit rate mono and stereo audio coding is available. Scalable audio coding schemes are also available in the GA LC scalable profile combined with AAC scalable object type.

1.5.1.2.8 CELP Object

The CELP object is supported by the CELP speech coding tools, which provide coding at 8 kHz and 16 kHz sampling rate at bit rates in the range 4-24 kbit/s. Additionally, bit rate scalability and bandwidth scalability are available in order to provide scalable decoding of CELP bitstreams. CELP Object always contains exactly one mono audio signal.

1.5.1.2.9 HVXC Object

The HVXC object is supported by the parametric speech coding (HVXC) tools, which provide fixed bitrate modes (2.0-4.0kbit/s) in a scalable and a non-scalable scheme, a variable bitrate mode (< 2.0kbit/s) and the functionality of pitch and speed change. Only 8 kHz sampling rate and mono audio channel are supported.

1.5.1.2.10 TTSI Object

The TTSI object is supported by the TTSI tools described in subpart 6. It allows very-low-bitrate phonemic descriptions of speech to be transmitted in the bitstream and then synthesized into sound. No particular speech synthesis method is specified in MPEG-4; rather, the TTSI tools specify an *interface* to non-normative synthesis methods. This method has a bit rate ranging 200 ~ 1200 bit/s. The TTSI object also supports synchronization of the synthesized speech with the facial animation defined in ISO/IEC 14496-2.

1.5.1.2.11 Main Synthetic Object

The main synthetic object allows the use of all MPEG-4 Structured Audio tools (described in subpart 5 of this standard). It supports flexible, high-quality algorithmic synthesis using the SAOL music-synthesis language; efficient wavetable synthesis with the SASBF sample-bank format; and enables the use of high-quality mixing and postproduction in the Systems AudioBIFS toolset. Sound can be described at 0 kbit/s (no continuous cost) to 3-4 kbit/s for extremely expressive sounds in the MPEG-4 Structured Audio format.

1.5.1.2.12 Wavetable Synthesis Object

The wavetable synthesis object is supported only by the SASBF format and MIDI tools. It allows the use of simple "sampling synthesis" in presentations where the quality and flexibility of the full synthesis toolset is not required.

1.5.1.2.13 General Midi Object

The General MIDI object is included only to provide interoperability with existing content. Normative sound quality and decoder behavior are not provided with the General MIDI object.

1.5.1.2.14 Algorithmic Synthesis and Audio FX Object

The Algorithmic Synthesis object provides SAOL-based synthesis capabilities for very low-bitrate terminals. It is also used to support the AudioBIFS **AudioFX** node in content where sound synthesis capability is not needed.

1.5.2 Audio Profiles and Levels

1.5.2.1 Profiles

Four Audio Profiles have been defined:

1. The **Speech Audio Profile** provides a parametric speech coder, a CELP speech coder and a Text-To-Speech interface.
2. The **Synthesis Audio Profile** provides the capability to generate sound and speech at very low bitrates.
3. The **Scalable Audio Profile,** a superset of the Speech Profile, is suitable for scalable coding of speech and music, for transmission methods such as Internet and Digital Broadcasting.
4. The **Main Audio Profile** is a rich superset of all the other Profiles, containing tools for natural and synthetic audio. The **Main Audio Profile** is a superset of the other three profiles (scalable, speech, synthesis).

Table 1.5.2 - Main profile

Tools / Audio Object Types	13818-7 main	13818-7 LC	13818-7 SSR	PNS	LTP	TLSS	Twin VQ	CELP	HVXC	TTSI	SA tools	SASBF	MIDI
AAC main	X			X									
AAC LC		X		X									
AAC SSR			X	X									
AAC LTP		X		X	X								
AAC Scalable		X		X	X	X							
TwinVQ					X		X						
CELP								X					
HVXC									X				
TTSI										X			
Main synthetic											X	X	X

Table 1.5.3 - Scalable Profile

Tools / Audio Object Types	13818-7 main	13818-7 LC	13818-7 SSR	PNS	LTP	TLSS	Twin VQ	CELP	HVXC	TTSI	SA tools	SASBF	MIDI
AAC LC		X		X									
AAC LTP		X		X	X								
AAC Scalable		X		X	X	X							
TwinVQ					X		X						
CELP								X					
HVXC									X				
TTSI										X			

Table 1.5.4 - Speech profile

Tools / Audio Object Types	13818-7 main	13818-7 LC	13818-7 SSR	PNS	LTP	TLSS	Twin VQ	CELP	HVXC	TTSI	SA tools	SASBF	MIDI
CELP								X					
HVXC									X				
TTSI										X			

Table 1.5.5 - Synthetic Audio Profile

Tools / Audio Object Types	13818-7 main	13818-7 LC	13818-7 SSR	PNS	LTP	TLSS	Twin VQ	CELP	HVXC	TTSI	SA tools	SASBF	MIDI
TTSI										X			
Main synthetic											X	X	X

1.5.2.2 Complexity Units

Complexity units are defined to give an approximation of the decoder complexity in terms of processing power and RAM usage required for processing MPEG-4 Audio bitstreams in dependence of specific parameters.

The approximated processing power is given in „Processor Complexity Units“ (PCU), specified in integer numbers of MOPS. The approximated RAM usage is given in „RAM Complexity Units“ (RCU), specified in (mostly) integer numbers of kWords (1000 words). The RCU numbers do not include working buffers that can be shared between different objects and/or channels.

If a profile level is specified by the maximum number of complexity units, then a flexible configuration of the decoder handling different types of objects is allowed under the constraint that both values for the total complexity for decoding and sampling rate conversion (if needed) do not exceed this limit.

The following table gives complexity estimates for the different object types:

Table 1.5.6 - Complexity of Object Types

Object Type	Parameters	PCU (MOPS)	RCU (kWords)	Remarks
AAC Main	fs = 48 kHz	5	5	1)
AAC LC	fs = 48 kHz	3	3	1)
AAC SSR	fs = 48 kHz	4	3	1)
AAC LTP	fs = 48 kHz	4	4	1)
AAC Scalable	fs = 48 kHz	5	4	1), 2)
TwinVQ	fs = 24 kHz	2	3	1)
CELP	fs = 8 kHz	1	1	
CELP	fs = 16 kHz	2	1	
CELP	fs = 8/16 kHz (bandwidth scalable)	3	1	
HVXC	fs = 8 kHz	2	1	
TTSI		-	-	4)
Wavetable Synthesis	fs = 22.05 kHz,	depends on bitstreams (3)	depends on bitstreams (3)	
Main Synthetic		depends on bitstreams (3)	depends on bitstreams (3)	
General MIDI		4	1	
Sampling Rate Conversion	rf = 2, 3, 4, 6	2	0.5	

Definitions:
- fs = sampling frequency
- rf = ratio of sampling rates

Notes -
1) PCU proportional to sampling frequency.
2) Includes core decoder.
3) See ISO/IEC 14496-4.
4) The complexity for speech synthesis is not taken into account.

1.5.2.3 Levels within the Profiles

- **Levels for Main Profile**

  Complexity units define four levels:
  1. PCU < 40, RCU < 20
  2. PCU < 80, RCU < 64
  3. PCU < 160, RCU < 128
  4. PCU < 320, RCU < 64000

- **Levels for Scalable Profile**

  Four levels are defined by configuration; complexity units define the fourth level:
  1. Maximum 24 kHz of sampling rate, one mono object (all object Types).
  2. Maximum 24 kHz of sampling rate, one stereo object or two mono objects (all object Types).
  3. Maximum 48 kHz of sampling rate, one stereo object or two mono objects (all object Types).
  4. Maximum 48 kHz of sampling rate, one 5.1 channels object or multiple objects with at maximum one integer factor sampling rate conversion for a maximum of two channels.
     Flexible configuration is allowed with PCU < 30 and RCU < 19.

- **Levels for Speech Profile**

  Two levels are defined by number of objects
  1. One speech object.
  2. Up to 20 speech objects.

- **Levels for Synthesis Profile**

  Three levels are defined

  1. Full Synthetic Audio 1 : All bitstream elements may be used with :
     - 64 KB memory for data (not including decoder)
     - “Low” processing (exact numbers in ISO/IEC 14496-4)
     - Only core sample rates may be used
     - no more than one TTSI object

  2. Full Synthetic Audio 2 : All bitstream elements may be used with :
     - 1 MB memory for data (not including decoder).
     - “High” processing (exact numbers in ISO/IEC 14496-4).
     - Only core sample rates may be used.
     - no more than four TTSI objects.

  3. Full Synthetic Audio 3 : All bitstream elements may be used with :
     - 16 MB memory for data (not including decoder).
     - “High” processing (exact numbers in ISO/IEC 14496-4).
     - no more than twelve TTSI objects.

  Note: For the case of scalable coding schemes, only the first instantiation of each object type will be counted to determine the number of objects relevant to the level definition and complexity metric. For example, in a scalable coder consisting of a CELP core coder and two enhancement layers implemented by means of GA LC scalable objects, one CELP object and one GA LC scalable object and their associated complexity metrics is counted since there is almost no overhead associated with the second (and any further) GA enhancement layer.

1.6 Interface to MPEG-4 Systems

1.6.1 Introduction

The header streams are transported via MPEG-4 systems. These streams contain configuration information, which is necessary for the decoding process and parsing of the raw data streams. However, an update is only necessary if there are changes in the configuration.
The payloads contain all information varying on a frame to frame basis and therefore carry the actual audio information.

1.6.2 Syntax

1.6.2.1 Audio DecoderSpecificInfo

In case that DecoderConfigDescriptor() (see ISO/IEC 14496-1) is used for MPEG-4 Audio decoders, the array specificInfoByte[] shall contain AudioSpecificInfo() defined as follows:

Table 1.6.1 - Syntax of AudioSpecificConfig()

Syntax	No. of bits	Mnemonic
AudioSpecificConfig()		
{		
AudioObjectType	**5**	**bslbf**
samplingFrequencyIndex	**4**	**bslbf**
if(samplingFrequencyIndex==0xf)		
samplingFrequency	**24**	**uimsbf**
channelConfiguration	**4**	**bslbf**
if(AudioObjectType == 1 || AudioObjectType == 2 ||		
AudioObjectType == 3 || AudioObjectType == 4 ||		
AudioObjectType == 6 || AudioObjectType == 7)		
GASpecificConfig()		
if(AudioObjectType == 8)		
CelpSpecificConfig()		
if(AudioObjectType == 9)		
HvxcSpecificConfig()		

```
    if( AudioObjectType == 12 )
        TTSSpecificConfig()
    if(   AudioObjectType == 13 || AudioObjectType == 14 ||
        AudioObjectType == 15 )
        StructuredAudioSpecificConfig()
}
```

1.6.2.2 HvxcSpecificConfig

Defined in subpart 2.

1.6.2.3 CelpSpecificConfig

Defined in subpart 3.

1.6.2.4 GASpecificConfig

Defined in subpart 4.

1.6.2.5 StructuredAudioSpecificConfig

Defined in subpart 5.

1.6.2.6 TTSSpecificConfig

Defined in subpart 6.

1.6.2.7 Payloads

For the NULL object the payload shall be 16 bit signed integer in the range from -32768 to +32767. The payloads for all other audio object types are defined in the corresponding subparts. These are the basic entities to be carried by the systems transport layer. Note that for all natural audio coding schemes the output is scaled for a maximum of 32767/-32768. However, the MPEG-4 System compositor expects a scaling.

1.6.3 Semantics

1.6.3.1 AudioObjectType

A five bit field indicating the audio object type. This is the master switch which selects the actual bitstream syntax of the audio data. In general, different object type use a different bitstream syntax. The interpretation of this field is given in the Audio Object Type table in subclause 1.5.1.1.

1.6.3.2 samplingFrequency

The sampling frequency used for this audio object. Either transmitted directly, or coded in the form of **samplingFrequencyIndex.**

1.6.3.3 samplingFrequencyIndex

A four bit field indicating the sampling rate used. If samplingFrequencyIndex equals 15 then the actual sampling rate is signaled directly by the value of **samplingFrequency**. In all other cases **samplingFrequency** is set to the value of the corresponding entry in table 1.6.2.

Table 1.6.2 - Sampling Frequency Index

Value	samplingFrequencyIndex
0x0	96000
0x1	88200
0x2	64000
0x3	48000
0x4	44100
0x5	32000
0x6	24000

0x7	22050
0x8	16000
0x9	12000
0xa	11025
0xb	8000
0xc	7350
0xd	reserved
0xe	reserved
0xf	escape value

1.6.3.4 channelConfiguration

A four bit field indicating the audio output channel configuration:

Table 1.6.3 - Channel Configuration

value	number of channels	channel to speaker mapping
0	-	defined in GASpecificConfig
1	1	center front speaker
2	2	left, right front speakers
3	3	center front speaker, left, right front speakers
4	4	center front speaker, left, right center front speakers, rear surround speakers
5	5	center front speaker, left, right front speakers, left surround, right surround rear speakers
6	5+1	center front speaker, left, right front speakers, left surround, right surround rear speakers, front low frequency effects speaker
7	7+1	center front speaker left, right center front speakers, left, right outside front speakers, left surround, right surround rear speakers, front low frequency effects speaker
8-15	-	reserved

Annex 1.A
(informative)
Audio interchange formats

1.A.1 Introduction

The full capabilities and flexibility of MPEG-4 Audio, like the composition of audio scenes out of multiple audio objects, synthetic audio, and text to speech conversion, is available only if MPEG-4 Audio is used together with MPEG-4 Systems (ISO/IEC-14496-1). The interchange formats, defined in here in Annex A, only support a small subset of the capabilities of MPEG-4 Audio, by defining formats for the storage and transmission of a single mono or stereo or multi-channel audio object, very similar to the formats defined in MPEG-1 and MPEG-2.
As already stated in the introduction to subpart 1, the normative elements in MPEG-4 Audio end with the definition of the payloads (roughly equivalent to a bitstream frame in MPEG-1 and MPEG-2), and the coder configuration structures (resembling the MPEG-1/2 header information). However, there is no normative definition in MPEG-4 Audio how these elements are multiplexed, as this is required only for a limited number of applications. Nevertheless, this informative annex describes such a multiplex. However, MPEG-4 decoders are not obliged to comply to these interface formats.

1.A.2 Interchange format streams

1.A.2.1 MPEG-2 AAC Audio_Data_Interchange_Format, ADIF

Table 1.A.1 – Syntax of adif_sequence

Syntax	No. of bits	Mnemonic

adif_sequence()
{
  adif_header()
  raw_data_stream()
}		

Table 1.A.2 - Syntax of adif_header()

Syntax	No. of bits	Mnemonic
adif_header() {		
adif_id	**32**	**bslbf**
copyright_id_present	**1**	**bslbf**
if(copyright_id_present) **copyright_id**	**72**	**bslbf**
original_copy	**1**	**bslbf**
home	**1**	**bslbf**
bitstream_type	**1**	**bslbf**
bitrate	**23**	**uimsbf**
num_program_config_elements	**4**	**bslbf**

for (i = 0; i < num_program_config_elements + 1; i++) {
  if(bitstream_type == '0')
  adif_buffer_fullness
  program_config_element()
  }
	20	**uimsbf**

}		

Table 1.A.3 - Syntax of raw_data_stream()

Syntax	No. of bits	Mnemonic

raw_data_stream()
{
  while (data_available()) {
  raw_data_block()
  byte_alignment()
  }
}		

1.A.2.2 Audio_Data_Transport_Stream frame, ADTS

Table 1.A.4 - Syntax of adts_sequence()

Syntax	No. of bits	Mnemonic
adts_sequence()		
{		
while (nextbits()==syncword) {		
adts_frame()		
}		
}		

Table 1.A.5 - Syntax of adts_frame()

Syntax	No. of bits	Mnemonic
adts_frame()		
{		
byte_alignment()		
adts_fixed_header()		
adts_variable_header()		
adts_error_check()		
for(i=0; i<no_raw_data_blocks_in_frame+1; i++) {		
raw_data_block()		
}		
}		

1.A.2.2.1 Fixed Header of ADTS

Table 1.A.6 - Syntax of adts_fixed_header()

Syntax	No. of bits	Mnemonic
adts_fixed_header()		
{		
Syncword	**12**	**bslbf**
ID	**1**	**bslbf**
Layer	**2**	**uimsbf**
protection_absent	**1**	**bslbf**
Profile	**2**	**uimsbf**
sampling_frequency_index	**4**	**uimsbf**
private_bit	**1**	**bslbf**
channel_configuration	**3**	**uimsbf**
original/copy	**1**	**bslbf**
Home	**1**	**bslbf**
Emphasis	**2**	**bslbf**
}		

1.A.2.2.2 Variable Header of ADTS

Table 1.A.7 - Syntax of adts_variable_header()

Syntax	No. of bits	Mnemonic
adts_variable_header()		
{		
copyright_identification_bit	**1**	**bslbf**
copyright_identification_start	**1**	**bslbf**
aac_frame_length	**13**	**bslbf**
adts_buffer_fullness	**11**	**bslbf**
no_raw_data_blocks_in_frame	**2**	**uimsbf**
}		

1.A.2.2.3 Error detection

Table 1.A.8 - Syntax of adts_error_check

Syntax	No. of bits	Mnemonic
adts_error_check()		
{		
if (protection_absent == '0')		
crc_check	**16**	**Rpchof**
}		

1.A.2.2.4 MPEG-4 Audio Transport Stream (MATS)

Table 1.A.9 - Syntax of mats_sequence()

Syntax	No. of bits	Mnemonic
mats_sequence()		
{		
while (nextbits()==mats_syncword) {		
mats_frame()		
}		
}		

Table 1.A.10 - Syntax of mats_frame()

Syntax	No. of bits	Mnemonic
mats_frame()		
{		
bitstream_type	**1**	**bslbf**
mats_syncword	**7**	**bslbf**
profile_level	**4**	**uimsbf**
number_of_object	**5**	**uimsbf**
for (n=0; n<number_of_object+1; n++) {		
AudioObjectType	**5**	**uimsbf**
if (AudioObjectType==1 || AudioObjectType==2 ||		
AudioObjectType==3 || AudioObjectType==4 ||		
AudioObjectType==6 || AudioObjectType ==7) {		
original/copy	**1**	**bslbf**
Home	**1**	**bslbf**
copyright_identification_bit	**1**	**bslbf**
copyright_identification_start	**1**	**bslbf**
frameLength /* 1024 or 960 */	**1**	**uimsbf**
sampling_frequency_index	**4**	**uimsbf**
if(samplingFrequencyIndex==0xf)		
SamplingFrequency	**24**	**uimsbf**
}		
if (AudioObjectType==1 || AudioObjectType==2 ||		
AudioObjectType==3 || AudioObjectType==4) {		
channel_configuration	**3**	**uimsbf**
protection_absent	**1**	**bslbf**
aac_frame_length	**13**	**bslbf**
adts_buffer_fullness	**11**	**bslbf**
no_raw_data_blocks_in_frame	**2**	**uimsbf**
adts_error_check()		
}		
if (AudioObjectType==6) { /* AAC LC scalable */		
op_mode	**4**	**bslbf**
if(op_mode>=2 && op_mode<=13) {		
blc_select	**3**	**bslbf**
If(blc_select != 0 && blc_select != 7)		

CoreCoderDelay	**12**	**bslbf**
} else		
blc_select = 0		
bitrate_index	**4**	**bslbf**
padding_bit	**1**	**bslbf**
main_data_begin	**10**	**bslbf**
}		
if (AudioObjectType==7) { /* TwinVQ */		
stereo_flag	**1**	**uimsbf**
tvq_no_extension_layers	**3**	**bslbf**
for (lyr=0 ; lyr<tvq_no_extension_layers+1; lyr++) {		
tvq_bitrate_index(lyr)	**8**	**bslbf**
}		
original/copy	**1**	**bslbf**
Home	**1**	**bslbf**
number_of _frame	**4**	**uimsbf**
adts_error_check()		
}		
if (AudioObjectType==8) { /* CELP */		
number_of_frame	**4**	**uimsbf**
CelpHeader()		
if (BandwidthScalabilityMode==ON) {		
CelpBWSenhHeader()		
}		
}		
if (AudioObjectType==9) { /* HVXC */		
number_of_frame	**4**	**uimsbf**
HVXCconfig()		
}		
byte_alignment()		
if (AudioObjectType==1 || AudioObjectType==2 ||		
AudioObjectType==3 || AudioObjectType==4) {		
for (i=0; I<no_raw_data_blocks_in_frame+1; i++) {		
raw_data_block()		
}		
}		
if (AudioObjectType==6) { /* GA LC scalable */		
core_coder_element(0)		
aac_scalable_main_element()		
for(lay=0; lay<aac_ext_layer; lay++) {		
aac_scalable_extension_element()		
}		
core_coder_element(1)		
aac_ scalable_ main_element()		
for(lay=0; lay<aac_ext_layer; lay++) {		
aac_ scalable_extension_element()		
}		
core_coder_element(2)		
aac_ scalable_ main_element()		
for(lay=0; lay<aac_ext_layer; lay++) {		
aac_ scalable_extension_element()		
}		
}		
if (AudioObjectType==7) { /* TwinVQ */		
for(i=0 ; i<number_of_frame+1;i++){		
tvq_scalable_main_element()		
for(lyr=1; lyr<tvq_no_extension_layers+1; lyr++){		
tvq_scalable_extension_element(lyr)		
}		