Contents for Subpart 5

Figures

Tables

Subpart 5: Structured audio

5.1 Scope

5.1.1 Overview of subpart

5.1.1. Purpose

The Structured Audio toolset enables the transmission and decoding of synthetic sound effects and music by standardising several different components. Using Structured Audio, high-quality sound can be created at extremely low bandwidth. Typical synthetic music may be coded in this format at bitrates ranging from 0 kbps (no continuous cost) to 2 or 3 kbps for extremely subtle coding of expressive performance using multiple instruments.

MPEG-4 does not standardise a particular set of synthesis methods, but a method for describing synthesis methods. Any current or future sound-synthesis method may be described in the MPEG-4 Structured Audio format.

5.1.1.2 Introduction to major elements

There are five major elements to the Structured Audio toolset:

1. The Structured Audio Orchestra Language, or SAOL. SAOL is a digital-signal processing language which allows for the description of arbitrary synthesis and control algorithms as part of the content bitstream. The syntax and semantics of SAOL are standardised here in a normative fashion.

2. The Structured Audio Score Language, or SASL. SASL is a simple score and control language which is used in certain object types (see subclause 5.6) to describe the manner in which sound-generation algorithms described in SAOL are used to produce sound.

3. The Structured Audio Sample Bank Format, or SASBF. The Sample Bank format allows for the transmission of banks of audio samples to be used in wavetable synthesis and the description of simple processing algorithms to use with them.

4. A normative scheduler description. The scheduler is the supervisory run-time element of the Structured Audio decoding process. It maps structural sound control, specified in SASL or MIDI, to real-time events dispatched using the normative sound-generation algorithms.

5. Normative reference to the MIDI standards, standardised externally by the MIDI Manufacturers Association. MIDI is an alternate means of structural control which can be used in conjunction with or instead of SASL. Although less powerful and flexible than SASL, MIDI support in this standard provides important backward-compatibility with existing content and authoring tools. MIDI support in this standard consists of a list of recognised MIDI messages and normative semantics for each.

5.2 Normative references

[DLS] (c) 1997 MIDI Manufacturers Association, *The MIDI Downloadable Sounds Specification, v. 97.1.*

[DLS2] (c) 1998 MIDI Manufacturers Association, *The MIDI Downloadable Sounds Specification, v. 98.2*.

[MIDI] (c) 1996 MIDI Manufacturers Association, *The Complete MIDI 1.0 Detailed Specification v. 96.2.*

5.3 Definitions

5.3.1 Absolute time: The time at which sound corresponding to a particular event is really created; time in the real-world. Contrast score time.

5.3.2 Actual parameter: The expression which, upon evaluation, is passed to an opcode as a parameter value.

5.3.3 A-cycle: See **audio cycle**.

5.3.4 A-rate: See **audio rate**.

5.3.5 asig: The lexical tag indicating an a-rate variable.

5.3.6 Audio cycle: The sequence of processing which computes new values for all a-rate expressions in a particular code block.

5.3.7 Audio rate: The rate type associated with a variable, expression or statement which may generate new values as often as the sampling rate.

5.3.8 Audio sample: A short snippet or clip of digitally represented sound. Typically used in wavetable synthesis.

5.3.9 Authoring: In Structured Audio, the combined processes of creatively composing music and sound control scripts, creating instruments which generate and alter sound, and encoding the instruments, control scripts, and audio samples in MPEG-4 Structured Audio format.

5.3.10 Backus-Naur Format: (**BNF**) A format for describing the syntax of programming languages, used here to specify the SAOL and SASL syntax. See subclause 5.4.2.2.

5.3.11 Bank: A set of samples used together to define a particular sound or class of sounds with wavetable synthesis.

5.3.12 Beat: The unit in which score time is measured.

5.3.13 BNF: See Backus-Naur Format**.**

5.3.14 Bus: An area in memory which is used to pass the output of one instrument into the input of another.

5.3.15 Context: See state space.

5.3.16 Control: An instruction used to describe how to use a particular synthesis method to produce sound.

EXAMPLES

"Using the piano instrument, play middle C at medium volume for 2 seconds."
"Glissando the violin instrument up to middle C."
"Turn off the reverberation for 8 seconds."

5.3.17 Control cycle: The sequence of processing which computes new values for all control-rate expressions in a particular code block.

5.3.18 Control period: The length of time (typically measured in audio samples) corresponding to the control rate.

5.3.19 Control rate: (1) The rate at which instantiation and termination of instruments, parametric control of running instrument instances, sharing of global variables, and other non-sample-by-sample computation occurs in a particular orchestra. (2) The rate type of variables, expressions, and statements that can generate new values as often as the control rate.

5.3.20 Decoding: The process of turning an MPEG-4 Structured Audio bitstream into sound.

5.3.21 Duration: The amount of time between instantiation and termination of an instrument instance.

5.3.22 Encoding: The process of creating a legal MPEG-4 bitstream, whether automatically, by hand, or using special authoring tools.

5.3.23 Envelope: A loudness-shaping function applied to a sound, or more generally, any function controlling a parametric aspect of a sound

5.3.24 Event: One control instruction.

5.3.25 Expression: A mathematical or functional combination of variable values, symbolic constants, and opcode calls.

5.3.26 Formal parameter: The syntactic element that gives a name to one of the parameters of an opcode.

5.3.27 Future wavetable: A wavetable that is declared but not defined in the SAOL orchestra; its definition must arrive in the bitstream before it is used.

5.3.28 Global block: The section of the orchestra that describes global variables, route and send statements, sequence rules, and global parameters.

5.3.29 Global context: The state space used to hold values of global variables and wavetables.

5.3.30 Global parameters: The sampling rate, control rate, and number of input and output channels of audio associated with a particular orchestra.

5.3.31 Global variable: A variable that can be accessed and/or changed by several different instruments.

5.3.32 Grammar: A set of rules that describes the set of allowable sequences of lexical elements comprising a particular language.

5.3.33 Guard expression: The expression standing at the front of an if, while, or else statement that determines whether or how many times a particular block of code is executed.

5.3.34 I-cycle: See initialisation cycle**.**

5.3.35 Identifier: A sequence of characters in a textual SAOL program that denotes a symbol.

5.3.36 Informative: Aspects of a standards document that are provided to assist implementers, but are not required to be implemented in order for a particular system to be compliant to the standard.

5.3.37 I-pass: See initialisation pass**.**

5.3.38 I-rate: See initialisation rate.

5.3.39 Initialisation cycle: See initialisation pass**.**

5.3.40 Initialisation rate: The rate type of variables, expressions, and statements that are set once at instrument instantiation and then do not change.

5.3.41 Initialisation pass: The sequence of processing that computes new values for each **i-**rate expression in a particular code block.

5.3.42 Instance: See instrument instantiation.

5.3.43 Instantiation: The process of creating a new instrument instantiation based on an event in the score or statement in the orchestra.

5.3.44 Instrument: An algorithm for parametric sound synthesis, described using SAOL. An instrument encapsulates all of the algorithms needed for one sound-generation element to be controlled with a score.

NOTE - An MPEG-4 Structured Audio instrument does not necessarily correspond to a real-world instrument. A single instrument might be used to represent an entire violin section, or an ambient sound such as the wind. On the other hand, a single real-world instrument that produces many different timbres over its performance range might be represented using several SAOL instruments.

5.3.45 Instrument instantiation: The state space created as the result of executing a note-creation event with respect to a SAOL orchestra.

5.3.46 ivar: The lexical tag indicating an i-rate variable.

5.3.47 K-cycle: See control cycle.

5.3.48 K-rate: See control rate.

5.3.49 ksig: The lexical tag indicating a k-rate variable.

5.3.50 Lexical element: See token.

5.3.51 Looping: A typical method of wavetable synthesis. Loop points in an audio sample are located and the sound between those endpoints is played repeatedly while being simultaneously modified by envelopes, modulators**,** etc.

5.3.52 MIDI: The Musical Instrument Digital Interface standards, see **[MIDI]** in subclause 5.2. MIDI is one method for specifying control of synthesis in MPEG-4 Structured Audio.

5.3.53 Natural Sound: A sound created through recording from a real acoustic space. Contrasted with synthetic sound.

5.3.54 Normative: Those aspects of a standard that must be implemented in order for a particular system to be compliant to the standard.

5.3.55 Opcode: A parametric signal-processing function that encapsulates a certain functionality so that it may be used by several instruments.

5.3.56 Orchestra: The set of sound-generation and sound-processing algorithms included in an MPEG-4 bitstream. Includes instruments, opcodes, routing, and global parameters.

5.3.57 Orchestra cycle: A complete pass through the orchestra, during which new instrument instantiations are created, expired ones are terminated, each instance receives one k-cycle and one control period worth of a-cycles, and output is produced.

5.3.58 Parameter fields: The names given to the parameters to an instrument.

5.3.59 P-fields: See parameter fields.

5.3.60 Production rule: In Backus-Naur Form grammars, a rule that describes how one syntactic element may be expressed in terms of other lexical and syntactic elements.

5.3.61 Rate-mismatch error: The condition that results when the **r**ate semantics rules are violated in a particular SAOL construction. A type of syntax error.

5.3.62 Rate semantics: The set of rules describing how rate types are assigned to variables, expressions, statements, and opcodes, and the normative restrictions that apply to a bitstream regarding combining these elements based on their rate types.

5.3.63 Rate type: The "speed of execution" associated with a particular variable, expression, statement, or opcode.

5.3.64 Route statement: A statement in the global block that describes how to place the output of a certain set of instruments onto a bus.

5.3.65 Run-time error: The condition that results from improper calculations or memory accesses during execution of a SAOL orchestra.

5.3.66 SASBF: See Sample Bank Format

5.3.67 SAOL: The Structured Audio Orchestra Language, pronounced like the English word "sail." SAOL is a digital-signal processing language that allows for the description of arbitrary synthesis and control algorithms as part of the content bitstream.

5.3.68 SAOL orchestra: See orchestra.

5.3.69 SASL: The Structured Audio Score Language. SASL is a simple format that allows for powerful and flexible control of music and sound synthesis.

5.3.70 Sample: See Audio sample.

5.3.71 Sample Bank Format: A component format of MPEG-4 Structured Audio that allows the description of a set of samples for use in wavetable synthesis and processing methods to apply to them.

5.3.72 Scheduler: The component of MPEG-4 Structured Audio that describes the mapping from control instructions to sound synthesis using the specified synthesis techniques.The scheduler description provides normative bounds on event-dispatch times and responses.

5.3.73 Scope : The code within which access to a particular variable name is allowed.

5.3.74 Score: A description in some format of the sequence of control parameters needed to generate a desired music composition or sound scene. In MPEG-4 Structured Audio, scores are described in SASL and/or MIDI.

5.3.75 Score time: The time at which an event happens in the score, measured in beats. Score time is mapped to absolute time by the current tempo.

5.3.76 Send statement: A statement in the global block that describes how to pass a bus on to an effect instrument for post-processing.

5.3.77 Semantics: The rules describing what a particular instruction or bitstream element should do. Most aspects of bitstream and SAOL semantics are normative in MPEG-4.

5.3.78 Sequence rules: The set of rules, both default and explicit, given in the global block that define in what order to execute instrument instantiations during an orchestra cycle.

5.3.79 Signal variable: A unit of memory, labelled with a name, that holds intermediate processing results. Each signal variable in MPEG-4 Structured Audio is instantaneously representable by a 32-bit floating point value.

5.3.80 Spatialisation: The process of creating special sounds that a listener perceives as emanating from a particular direction.

5.3.81 State space: A set of variable-value associations that define the current computational state of an instrument instantiation or opcode call. All the “current values” of the variables in an instrument or opcode call.

5.3.82 Statement: “One line” of a SAOL orchestra.

5.3.83 Structured audio: Sound-description methods that make use of high-level models of sound generation and control. Typically involving synthesis description, structured audio techniques allow for ultra-low bitrate description of complex, high-quality sounds. See **[SAUD]**.

5.3.84 Symbol: A sequence of characters in a SAOL program, or a symbol token in a MPEG-4 Structured Audio bitstream, that represents a variable name, instrument name, opcode name, table name, bus name, etc.

5.3.85 Symbol table: In an MPEG-4 Structured Audio bitstream, a sequence of data that allows the tokenised representation of SAOL and SASL code to be converted back to a readable textual representation.The symbol table is an optional component.

5.3.86 Symbolic constant: A floating-point value explicitly represented as a sequence of characters in a textual SAOL orchestra, or as a token in a bitstream.

5.3.87 Syntax: The rules describing what a particular instruction or bitstream element should look like. All aspects of bitstream and SAOL syntax are normative in MPEG-4.

5.3.88 Syntax error: The condition that results when a bitstream element does not comply with its governing rules of syntax.

5.3.89 Synthesis: The process of creating sound based on algorithmic descriptions.

5.3.90 Synthetic Sound: Sound created through synthesis.

5.3.91 Tempo: The scaling parameter that specifies the relationship between score time and absolute time. A tempo of 60 beats per minute means that the score time measured in beats is equivalent to the absolute time measured in seconds; higher numbers correspond to faster tempi, so that 120 beats per minute is twice as fast.

5.3.92 Terminal: The “client side” of an MPEG transaction; whatever hardware and software are necessary in a particular implementation to allow the capabilities described in this document.

5.3.93 Termination: The process of destroying an instrument instantiation when it is no longer needed.

5.3.94 Timbre: The combined features of a sound that allow a listener to recognise such aspects as the type of instrument, manner of performance, manner of sound generation, etc. Those aspects of sound that distinguish sounds equivalent in pitch and loudness.

5.3.95 Token: A lexical element of a SAOL orchestra: a keyword, punctuation mark, symbol name, or symbolic constant.

5.3.96 Tokenisation: The process of converting a orchestra in textual SAOL format into a bitstream representation consisting of a stream of tokens.

5.3.97 Variable: See signal variable.

5.3.98 Wavetable synthesis: A synthesis method in which sound is created by simple manipulation of audio samples, such as looping, pitch-shifting, enveloping, etc.

5.3.99 Width: The number of channels of data that an expression represents.

5.4 Symbols and abbreviations

5.4.1 Mathematical operations

The mathematical operators used to describe this part of ISO/IEC 14496 are similar to those used in the C programming language.

+	addition
-	subtraction
x or *	multiplication
/	division
exp	exponential function (base *e*)
log	natural logarithm
log10	base-10 logarithm
abs	absolute value
floor(x)	greatest integer less than or equal to x
ceil(x)	least integer greater than or equal to x
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
<> or !=	not equal to

5.4.2 Description methods

5.4.2.1 Bitstream syntax

The Structured Audio bitstream syntax is described using SDL, the MPEG-4 Syntactic Description Language. See ISO/IEC 14496-1 clause 12.

5.4.2.2 SAOL syntax

The textual SAOL syntax (in subclause 5.8) is described using extended Backus-Naur format (BNF) notation [see **DRAG**]. BNF is a description for context-free grammars of programming languages.

BNF grammars are composed of terminals, also called tokens, and production rules. Terminals represent syntactic elements of the language, such as keywords and punctuation; production rules describe the composition of these elements into structural groups.

Terminals will be represented in **boldface**; production rules will be represented in <angle brackets>.

The rewrite rules, which map productions into sequences of other productions and terminals, are represented with the -> symbol.

EXAMPLE

<letter>	-> **a**
<letter>	-> **b**
<sequence>	-> <letter>
<sequence>	-> <letter> <sequence>

This grammar (starting from the *sequence* token) describes, using a recursive rewrite rule and a two-symbol alphabet, all strings containing at least one letter that are made up of ‘a’ and ‘b’ characters.

In addition, rewrite rules using optional elements will be described using the [ ] symbols. Using this notation does not increase the power of the syntax description (in terms of the languages it can represent), but makes certain constructs simpler.

EXAMPLE

<head>	-> **c**
<seqhead>	-> [<head>] <sequence>

This grammar (starting from the seqhead token) describes, in addition to the set above, all strings beginning with a ‘c’ character and followed by a sequence of ‘a’s and ‘b’s.

The **NULL** token may be used to indicate that a sequence of no characters (the empty string) is a permissble rewrite for a particular production.

Other symbols such as the ellipsis (...) will be used occasionally when their meaning is clear from the context.

Normative aspects of the relationship between the BNF grammar, other grammar representation methods, the bitstream syntax, and the textual description format are described in subclause 5.8.1.

5.4.2.3 SASL Syntax

The SASL syntax is specified using extended BNF grammars, as described in subclause 5.4.2.2.

5.5 Bitstream syntax and semantics

5.5.1 Introduction to bitstream syntax

This subclause describes the bitstream format defining an MPEG-4 Structured Audio bitstream.

Each group of classes is notated with normative semantics, which define the meaning of the data represented by those classes.

5.5.2 Bitstream syntax

```
/***************************
      symbol table definitions
```

```
*****************************/

class symbol {
  unsigned int(16) sym;         // no more than 65535 symbols/orch + score
}

class sym_name {                // one name in a symbol table
  unsigned int(4) length;       // names up to 15 chars long
  unsigned int(8) name[length];
}

class symtable {                // a whole symbol table
  unsigned int(16) length;      // no more than 65535 symbols/orch+score
  sym_name name[length];
}
```

A bitstream may contain a symbol table, but this is not required. The symbol table allows textual SAOL and SASL code to be recovered from the tokenised bitstream representation.The inclusion or exclusion of a symbol table does not affect the decoding process.

If a symbol table is included, then all or some of the symbols in the orchestra and score shall be associated with a textual name in the following way: each symbol (a symbol is just an integer) shall be associated with the character string paired with that symbol in a `sym_name` object. There shall be no more than one name associated with a given symbol, otherwise the bitstream is invalid. It is permissible for the symbol table to be incomplete and contain names associated with some, but not all, symbols used in the orchestra and score.

SAOL and SASL implementations that require textual input, rather than tokenised input, are permissible in a compliant decoder, in which case the decoder would detokenise the bitstream before it can be processed. In such a case, any symbols without associated names are suggested to be associated with a default name of the form `_sym_x`, where x is the symbol value. Names of this form are reserved in SAOL for this purpose, and so following this suggestion guarantees that names will not clash with symbol-table-defined symbol names.

```
/********************************
    orchestra file definitions
***********************************/

class orch_token {              // a token in an orchestra
  int done;

  unsigned int(8) token;         // see standard token table, Annex 5.A
  switch (token) {
  case 0xF0 :                   // a symbol
    symbol sym;                 // the symbol name
    break;
  case 0xF1 :                   // a constant value
    float(32) val;              // the floating-point value
    break;
  case 0xF2 :                    // a constant int value
    unsigned int(32) val;       // the integer value
    break;
  case 0xF3 :                   // a string constant
    int(8) length;
    unsigned int(8) str[length]; // strings no more than 255 chars
    break;
  case 0xF4 :                   // a one-byte constant
    int(8) val;
    break;
  case 0xFF : // end of orch
    done = 1;
    break;
  }
}

class orc_file {                // a whole orch file
  unsigned int(16) length;
  orch_token data[length];
}
```

An orchestra file is a string of tokens. These tokens represent syntactic elements such as reserved words, core opcode names, and punctuation marks as given in the table in Annex 5.A; in addition, there are five special tokens. Token 0xF0 is the symbol token; when it is encountered, the next 16 bits in the bitstream shall be a symbol number. Token 0xF1 is the value token; when it is encountered, the next 32 bits in the bitstream shall be a floating-point