INTERNATIONAL STANDARD

ISO/IEC 14496-11

First edition
2005-12-15

AMENDMENT 5
2007-04-15

Information technology — Coding of audio-visual objects —

Part 11: Scene description and application engine

AMENDMENT 5: Support for Symbolic Music Notation

Technologies de l'information — Codage des objets audiovisuels —

Partie 11: Description de scène et moteur d'application

AMENDEMENT 5: Support pour la notation musicale symbolique

Reference number
ISO/IEC 14496-11:2005/Amd.5:2007(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published 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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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 document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 5 to ISO/IEC 14496-11:2005 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This Amendment contains new node specifications with syntax and semantics as well as a short description of SMR and its role inside MPEG-4.

Information technology — Coding of audio-visual objects —

Part 11:
Scene description and application engine

AMENDMENT 5: Support for Symbolic Music Notation

In clause 4, add the following definition:

Symbolic Music Representation (SMR)

A method of describing a logical structure consisting of: symbolic elements that represent audiovisual events; the relationship between those events; and aspects of rendering those events as defined by ISO/IEC 14496-23.

In subclause 7.2, Node Semantics, add the following subclauses:

7.2.2.87 MusicScore

7.2.2.87.1 Node interface

MusicScore {

eventIn	SFBool	**executeCommand**	
eventIn	SFString	**gotoLabel**	
eventIn	SFInt32	**gotoMeasure**	
eventIn	SFTime	**highlightTimePosition**	
eventIn	SFVec3f	**mousePosition**	
exposedField	MFString	**argumentsOnExecute**	[]
exposedField	SFString	**commandOnExecute**	[]
exposedField	SFInt32	**firstVisibleMeasure**	0
exposedField	SFBool	**hyperlinkEnable**	TRUE
exposedField	SFBool	**loop**	FALSE
exposedField	MFString	**partsLyrics**	[]
exposedField	MFInt32	**partsShown**	[]
exposedField	SFTime	**scoreOffset**	0.0
exposedField	SFVec2f	**size**	-1, -1
exposedField	SFFloat	**speed**	1.0
exposedField	SFTime	**startTime**	0.0
exposedField	SFTime	**stopTime**	
exposedField	SFFloat	**transpose**	0.0
exposedField	MFURL	**url**	[]
exposedField	MFURL	**urlSA**	[]
exposedField	SFString	**viewType**	[]
eventOut	SFString	**activatedLink**	
eventOut	MFString	**availableCommands**	
eventOut	MFString	**availableLabels**	
eventOut	MFString	**availableLyricLanguages**	
eventOut	MFString	**availableViewTypes**	
eventOut	SFBool	**isActive**	

```
    eventOut        SFVec3f         highlightPosition
    eventOut        SFInt32         lastVisibleMeasure
    eventOut        SFInt32         numMeasures
    eventOut        MFString        partNames
}
```

NOTE For the binary encoding of this node see node coding tables in electronic attachment.

7.2.2.87.2 Functionality and semantics

Rendering of Symbolic Music allows different solutions ranging from bitmap to vector graphics. To minimize the impact on some widespread existing solutions, including the SMR reference software, two new nodes are defined: **ScoreShape**, similarly to Shape, is used to map a **MusicScore** on a geometry, and a **MusicScore** as a child node. In such a way different solutions are allowed, including vector graphics and bitmaps.

The **loop**, **startTime**, and **stopTime** exposedFields and the **isActive** eventOut, and their effects on the **MusicScore** node, are described in 7.1.1.1.6.2. A **MusicScore** node is inactive before **startTime** is reached.

The **MusicScore** node displays the score at SMR stream time t=0 until it is activated, and keep the last composed image available when it is deactivated. Please note that the internal SMR decoder has also its own time representation and it may continue to run after **stopTime** with SMR stream being processed. However, a **loop** field set to TRUE may infer a restart of a certain portion of the score rendering.

The **executeCommand** eventIn is an input event indicating that when the **hyperlinkEnable** field is FALSE the command set in **commandOnExecute** has to be performed considering the values of **argumentsOnExecute** and of **mousePosition** while if **hyperlinkEnable** is TRUE the value of **mousePosition** is used to see if in that position is present something with an associated link, if this is the case the **activatedLink** eventOut is generated with the value of the link.

The **gotoLabel** eventIn positions the score on the page containing the specified label (one of the *availableLabels*).

The **gotoMeasure** eventIn positions the score on the page containing the specified measure.

The **highlightTimePosition** eventIn highlights the time position indicated relative to the scoreOffset field.

The **mousePosition** eventIn is used to indicate the point where the user has clicked; the position will be taken into account when the next **executeCommand** eventIn will be issued.

The **argumentsOnExecute** exposedField indicates arguments for the **commandOnExecute** command.

The **commandOnExecute** exposedField indicates the command to be executed when the user clicks on the score (via *executeCommand* eventIn).

Some commands that shall be supported by the *commandOnExecute,* according to the profile, are:

- "ADD_TEXT_ANNOTATION"
  the first value in *argumentsOnExecute* contains the text to be added to the score in the position indicated by the last *mousePosition* eventIn (that is the position where the user clicked)
- "ADD_LABEL"
  the first value in *argumentsOnExecute* contains the label text to be added to the measure indicated by the last *mousePosition* eventIn, if the measure already has a label the label is substituted
- "ADD_NOTE"
  the first value in *argumentsOnExecute* contains the note duration: "D1", "D1_2", "D1_4", "D1_8", "D1_16", "D1_32", "D1_64"; the second value indicates the notehead type: "CLASSIC", "X", "DSHARP", "DIAMOND", "RYTHMIC", "DIDAPTIC", etc. (see Table 11 in ISO/IEC 14496-23) the note is inserted where the user clicks or it is added to a chord if sufficiently near to another note/chord.

- "ADD_REST"
  the first value in *argumentsOnExecute* contains the rest duration: "D1", "D1_2", "D1_4", "D1_8", "D1_16", "D1_32", "D1_64"; the rest is inserted in the position indicated by the last *mousePosition* eventIn.
- "SET_ALTERATION"
  the first value in *argumentsOnExecute* contains the alteration to be set on the note, it can be: "SHARP","DSHARP","FLAT","DFLAT","NATURAL". The alteration is set to the note indicated by the last *mousePosition* eventIn.
- "SET_DOTS"
  the first value in *argumentsOnExecute* contains the number of dots to be set on the note, it can be: "0","1","2". The dots are set to the note indicated by the last mousePosition eventIn.
- "ADD_SYMBOL"
  the first value in *argumentsOnExecute* contains the symbol to be added on the note/rest/measure, it can be: "STACCATO", "TENUTO" or any symbol defined using the formatting language (see Table 116 in ISO/IEC 14496-23). The symbol is added in the position indicated by the last *mousePosition* eventIn.
- "ADD_MEASURE"
  adds a measure to the score, the first value in *argumentsOnExecute* can be: "BEFORE", "AFTER" or "APPEND", the second value in *argumentsOnExecute* indicates the measure number with respect to the new measure is added. If the second value is not present or empty the last *mousePosition* eventIn is used to identify the reference measure. Note that adding a measure means add a measure to all the parts
- "DEL_MEASURE"
  removes a measure of the score; the first value in *argumentsOnExecute* indicates the measure number to be removed. If the first value is not present or empty the last *mousePosition* eventIn is used to identify the measure to be delete. Note that deleting a measure means delete a measure from all the parts.
- "CHANGE_CLEF"
  changes the clef of a measure and for all the following until another clef change or to the end. The first value in *argumentsOnExecute* contains the clef type, it can be: "TREBLE", "SOPRANO", "BASS", "TENOR" etc. (see Table 9 in ISO/IEC 14496-23) The clef change applies to the measure indicated by the last *mousePosition* eventIn.
- "CHANGE_KEYSIGNATURE"
  changes the key signature of a measure and for all the following until another key signature change or to the end. The first value in *argumentsOnExecute* contains the key signature type, it can be: "DOdM", "FAdM", "SIM", etc. (see Table 10 in ISO/IEC 14496-23) The key signature change applies to the measure indicated by the last *mousePosition* eventIn.
- "CHANGE_TIME"
  changes the time of a measure and for all the following until another time change or to the end. The first value in *argumentsOnExecute* contains the time, it can be: "4/4", "3/4", "2/4", "C" or "C/". The time change applies to the measure indicated by the last *mousePosition* eventIn.
- "SET_METRONOME"
  sets the metronome for the whole piece. The first value in *argumentsOnExecute* contains the reference note duration (D1, D1_2, D1_4,…) the second value contains "TRUE" if the reference note is with augmentation dot ("FALSE" or empty otherwise), the third value indicates the number of reference notes in one minute. For example ["D1_4", "TRUE", "100"] sets a metronome with 100 dotted quarters in one minute. The metronome is set using the *executeCommand* eventIn.
- "DELETE"
  allows deleting any symbol, note, rest, alteration, label and annotation added by the user in the position indicated by the last *mousePosition* eventIn..
- "TRANSPOSE"
  allows transposing the score. The first value in *argumentsOnExecute* contains the part to be transposed (0 for the whole main score, 1 for the first upper part, 2 the second part, …), the second value indicates the measure from which to start the transposition, the third value indicates the measure where to end transposition (the measure is included) a value of 0 or negative indicates to transpose until the last measure, the fourth value indicates the amount of transposition in half tones (e.g. 1 to increase of a half tone, 2 to increase of a tone, -1 to decrease of a half tone). This command does not depend on the mouse position and it is executed when the *executeCommand* eventIn is issued.

The **firstVisibleMeasure** exposedField is the first measure currently visible.

When the **hyperlinkEnable** exposedField is set to TRUE hyperlinks are shown; when the user clicks (via **executeCommand** eventIn) on a link an eventOut **activatedLink** is generated.

The **partsLyrics** exposedField is an array of strings indicating for which part to view the lyrics and in which language (e.g. ["it", "en", ""] to view lyrics for part 1 in Italian and for part 2 in English).

The **partsShown** exposedField is an array of integers indicating which parts have to be shown; the number is the position in the array of parts names; if **partShown** is empty all parts will be visible (e.g. [ ] to view main score with all parts, [2] to view single part number 2, [1,3] view main score with parts 1 and 3, etc.).

The **scoreOffset** exposedField indicates the initial (or point 0) offset from the beginning of the score; it may be used to change page or move inside the score before starting it, or in pause etc. **scoreOffset** is indicated in seconds from the beginning of the score. **scoreOffset** can be used only if synchronization information is provided or a metronome indication is present in the score.

The **size** exposedField parameter expresses the width and height of the music score in the units of the local coordinate system. A size of -1 in either coordinate means that the **MusicScore** node is not specified in size in that dimension, and that the size is adjusted to the size of the parent node.

The **speed** exposedField indicates how fast the score shall be played. It shall be a strictly positive (>0) tempo multiplier, so a speed of 2 indicates the score plays twice as fast the tempo metronomic indication.

The **transpose** exposedField defines the transposition in half tones (e.g. 1 to increase of a half tone, 2 to increase of a tone, -1 to decrease of a half tone) to be applied to the whole score (all parts and all measures). For a more fine grained transposition the "TRANSPOSE" command can be used.

The **url** exposedField defines the SMR data stream; the stream may be composed by different data chunks for parts, lyrics, score, and synchronization info as described in ISO/IEC 14496-23.

The **urlSA** exposedField defines a possibly associated SA (i.e. MIDI) data stream.

The **viewType** exposedField indicates the kind of view to be used (one of the *availableViewTypes*).

The **activatedLink** eventOut is generated when the user clicks on a link via **executeCommand** when **hyperlinkEnable** is TRUE; it has associated the link value.

The **availableCommands** eventOut is an array of commands that can be performed on the score by the user when the user clicks on the score (e.g. ["ADD_LABEL", "ADD_TEXT_ANNOTATION", "DELETE"]) some commands will be normative other may be decoder dependent, see in the following for details.

The **availableLabels** eventOut is an array of strings with labels (e.g. ["A", "B", "SEGNO", "CODA"]).

The **availableLyricLanguages** eventOut is an array of strings where for each part there is the list of languages (using the ISO 639-2 standard), separated with ";", for which the lyric is available (e.g. ["en;it", "en;it", ""]) (this field may or may not be filled by the scene author, which is supposed to know the SMR content and thus languages that are available).

The **availableViewTypes** eventOut is an array of strings describing which view types are available for the score and for the decoder (e.g. ["CWMN", "braille", "neumes"]).

The **highlightPosition** eventOut outputs the highlight position in local coordinates.

The **lastVisibleMeasure** eventOut is the last measure currently visible.

The **numMeasures** eventOut is the number of measures in the score.

The **partNames** eventOut is an array of strings with part names (instruments, e.g. ["soprano", "baritone", "piano"]).

7.2.2.114 ScoreShape

7.2.2.114.1 Node interface

ScoreShape {

exposedField	SFMusicScoreNode	**score**	NULL
exposedField	SFNode	**geometry**	NULL

}

NOTE For the binary encoding of this node see node coding tables in electronic attachment.

7.2.2.114.2 Functionality and semantics

The **score** field allows the connection of a **MusicScore** node containing the Symbolic Music Representation formatted content.

The semantics of the **geometry** field are the same of the field with equivalent name in the Shape node specification, and are specified in ISO/IEC 14772-1:1998, subclause 6.41.

When the **score** field is NULL nothing shall be done.

When the **geometry** field is NULL the node shall be rendered as if Bitmap node is specified as geometry

and:

Add subclause 7.9:

7.9 Informative: The SMR Decoder and MPEG-4

Symbolic representations of music have a logical structure consisting of: symbolic elements that represent audiovisual events; the relationship between those events; and aspects of rendering those events.

SMR is enabled in MPEG-4 by:

- defining an XML format for a text based symbolic music representation, to be used for interoperability with other symbolic music representation/notation formats and as a source for the production of an equivalent binary information that may be stored in files and/or streamed by a suitable transport layer; a format and decoding process for Symbolic Music Representation is specified in ISO/IEC 14496-23.
- specifying a binary stream containing Symbolic Music Representation, it basic formatting and synchronization information; the associated decoder will allow to manage the music notation model and to add the necessary “musical intelligence” for the interaction with humans;
- specifying the interface and the behavior for the symbolic music representation decoder and its relationship with the MPEG-4 Scene Representation, i.e. this specification.