ISMA

INTERNET STREAMING MEDIA ALLIANCE

Output Document Number TD00105

November 2007

TITLE: ISMA Ultravox Part 3: Transport of MPEG-4 Codecs

Status: External Provisional Specification

1. Introduction

This specification describes how MPEG-4 Elementary Streams are transported over Ultravox [5]. It is described how decoder configuration information is provided during session setup and how synchronization between multiple Elementary Streams is achieved. In order to provide the configuration information, a cacheable metadata frame is specified. Furthermore, a data message is defined for the transport of the media data, i.e. the Access Units. This data message supports fragmentation, aggregation, synchronization, and random access. The multiplex of several streams (audio, video, data) is done on the Ultravox level.

2. Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this annex are to be interpreted as described in RFC 2119 [6].

Throughout this document, all fields and variables are defined in network byte order.

3. MPEG-4 Configuration Message

MPEG-4 codecs typically require *decoder configuration information* which must be available before the decoding of the media data. This information is usually provided out-of-band during session setup which can be accomplished in Ultravox through the use of cacheable metadata [5].

The following Ultravox message contains the setup for a session with MPEG-4 codecs. A Broadcaster MUST send this message prior to any Data Messages and SHOULD NOT update this setup information during the remaining session. Distribution points MUST transmit this message as the first data sent to each listener (as required for all cacheable metadata messages).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sync Byte   |     Flags     |Class=3|      Type=0xa01       |
|0 1 0 1 1 0 1 0|              E|0 0 1 1|1 0 0 1 0 0 0 0 0 0 1 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |          Metadata ID          |
|             N=6+Nc            |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Metadata Total Fragments    |    Metadata Fragment Index    |
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   MPEG4ConfigBox (Nc Byte)                    |      End      |
|                       ...                     |0 0 0 0 0 0 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Class (4bits) – 0x3 (Cacheable Metadata)
Type (12bits) – 0xa01 (MPEG-4 Configuration)
Length (16bits) – length of metadata specific fields (6 Byte) plus length of the MPEG4ConfigBox (variable size, Nc Byte).
Payload – data which consists of the metadata specific fields and the MPEG4ConfigBox, having together `Length` bytes.

The encryption flag (E) in the flags section of the Ultravox message SHALL be set to zero.

Note that MPEG-4 decoder configuration information is typically compact enough to fit into a single Ultravox frame. Hence, Metadata Total Fragments is typically 0x0001 and Metadata Fragment Index is typically 0x0000.

Besides the metadata specific fields (Metadata ID, Metadata Total Fragments, Metadata Fragment Index) the MPEG-4 Configuration Message contains the MPEG4ConfigBox. The syntax of this data record is derived from the ISO-FileFormat Box class [2] and contains all necessary information for the initialization of the session. The decoder configuration is derived from the MPEG-4 File Format (MP4, [3]).

Syntax

```
aligned(8) class MPEG4ConfigBox extends FullBox('m4co',0,0) {
  MPEG4SessionBox(); // session-level parameters
  for(i=0; i<num_streams; i++) {
    MPEG4MediaBox(); // media-level parameters
  }
}
```

Semantics

The MPEG4ConfigBox is composed of a MPEG4SessionBox which contains all parameters which are relevant on a session level plus a MPEG4MediaBox for the information on the media level of each stream in the Ultravox session. These two data records are defined in the following two subsections.

3.1 MPEG4SessionBox

The MPEG4SessionBox contains all necessary information on the session level, i.e. information that is common to all media streams or required for the session setup. It follows directly after the metadata related fields and has a fixed size.

Syntax

```
aligned(8) class MPEG4SessionBox extends FullBox('m4co',0,0) {
  unsigned int(8) num_streams;
  unsigned int(32) time_scale;
  unsigned int(32) initial_delay;
}
```

Semantics

`num_streams` is an 8 bit unsigned integer, that indicates the number of streams in the Ultravox session with a valid range of 1-255.

`time_scale` is a 32 bit unsigned integer, that specifies the time-scale for this session; this is the number of time units that pass in one second. For example, a time coordinate system that measures time in sixtieths of a second has a time scale of 60. A typical time scale for MPEG-4 video is 90.000 while for audio typically the sampling rate is used (e.g. 48.000). In order to allow easy synchronization, the same time scale is used for all media streams in one session. Hence, if several streams of different media are transmitted, the "native time scale" has to be converted to the "session time scale".

`initial_delay` is a 32 bit unsigned integer, that indicates the initial delay that a Listener SHOULD wait and buffer incoming data before starting the playout of the session. The delay is given in time-scale ticks of the session and corresponds to the amount of buffered media time. Note that the sender side has to consider the constraints for each media stream (e.g. re-ordering of B-frames) plus the resulting multiplexing delay which results from interleaving audio and video (or other media) frames in the Ultravox stream (see section 5 below). The Listener may observe the buffered media time in any stream to decide on when to start the playout. It is the responsibility of the sender to comply with the signaled value by appropriate multiplexing.

3.2 MPEG4MediaBox

The MPEG4MediaBox contains all necessary information for the initialization of MPEG-4 decoders. It is based on the MPEG-4 File Format (MP4, [3]).

Syntax

```
aligned(8) class MPEG4MediaBox extends FullBox('m4co',0,0) {
  unsigned int(8) stream_ID;
  unsigned int(32) handler_type;
  SampleDescriptionBox();
}
```

Semantics

`stream_ID` is an 8 bit unsigned integer, that corresponds to the stream_ID used in Ultravox data frames (see section 4 below). It is used to associate the content of Ultravox data frames with the media format as described in the `SampleDescripionBox`. For each stream within the same Ultravox session a different unique number (0-255) must be selected. It is RECOMMENDED to use increasing stream_IDs for the different streams, starting with 0.

`handler_type` is a 32 bit unsigned integer describing the nature of the media and the necessary presentation unit. It contains typically one of the following values:

`'vide'`	Video track
`'soun'`	Audio track

Its value is mainly required as a switch value within the `SampleDescriptionBox`.

`SampleDescriptionBox` is a SampleDescription, as defined in [2]. Sample Description boxes ('stsd') MUST contain exactly one sample entry (the `entry_count` field MUST be equal to 1). Typically, this box is transmitted without modifications from the MP4 file [3] or generated based on information necessary for the decoder and presentation unit. It may also include information on the encryption of the media data based on ISMACryp [7].

4. MPEG-4 Data Message

The basic decoding unit in MPEG-4 is an Access Unit (AU), also referred to as a sample [1]. In this section it is described how AUs are transported in Ultravox Data Messages and how AUs from multiple Elementary Streams are multiplexed in one Ultravox session. To allow synchronization, fragmentation, and aggregation, an additional layer between Ultravox and the raw Access Unit is necessary which is defined below. For this purpose the Data Message Class=0xa is reserved for the use of MPEG-4 transport. The structure of the MPEG-4 Data Message is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sync Byte    |    Flags       | Class |       Type            |
|0 1 0 1 1 0 1 0|R R R R R R K E|1 0 1 0|Format |   stream_ID    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |    Payload     |      End       |
|               N               |    (N Byte)    |0 0 0 0 0 0 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Class (4bits) – 0xa (MPEG-4 Data Message)
Type (12bits) – 4 bits for `Payload_Format` and 8 bits for `stream_ID`. The `Payload_Format` describes the structure of the `Payload`. Currently only a single `Payload_Format` is defined (0x1, see below). The `stream_ID` MUST be one of the `stream_IDs` signaled in the MPEG-4 Configuration Message during session setup. It indicates the type and format of the media data contained in the `Payload`.
Length (16bits) – length of the `Payload`.
Payload – media data consisting of MPEG-4 Access Units, which are formatted according to the `Payload_Format`. The `Payload` may contain several AUs and/or fragmented AUs as defined in more detail below.

When AUs are encrypted with ISMACryp [7], the Encryption flag (E) in the flags section of the Ultravox frame MUST be set.

If the Payload contains an AU, or the first fragment of an AU, which can be decoded independently from other AUs, then the Key-Frame flag (K) in the flags section of the Ultravox frame MUST be set.

4.1 Multiplex

Note that each MPEG-4 Data Message contains only one type of media, as indicated by the `stream_ID`. This allows to multiplex audio and video (and possibly other data) on the Ultravox level and avoids the need to introduce an additional multiplex format. Messages with different `stream_ID`s MUST be sent interleaved using an appropriate interleaving-depth. I.e., one or several consecutive MPEG-4 Data Messages of the same `stream_ID` may not represent more media time than the interleaving-depth. Note that the interleaving-depth is not explicitly signaled but must necessarily be smaller than the `initial_delay` as signaled in the MPEG-4 Configuration Message. A recommended value for the interleaving-depth is one second.

4.2 Payload Format

If the `Payload_Format` in the MPEG-4 Data Message is set to 0x1 then the payload contains a single or a concatenation of several `MPEG4MediaData` records as defined below. For the case of two records the resulting MPEG-4 Data Message is illustrated below. Other `Payload_Format` values and corresponding payload formats are reserved for future use.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sync Byte   |   Flags       | Class |Format |   stream_ID   |
|0 1 0 1 1 0 1 0|R R R R R R K E|1 0 1 0|0 0 0 1|               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             | MPEG4MediaData #1 (N1 Byte)  |
|            N=N1+N2            |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
:            ...                                                :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   MPEG4MediaData #2 (N2 Byte)                                 |
:            ...                                                :
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |     End       |
|                               |0 0 0 0 0 0 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Each `MPEG4MediaData` contains an Access Unit header followed by the Access Unit data. The AU header contains flags which are required for fragmentation and random access as well as a timestamp which is required for synchronization. The AU data is copied directly from the encoder or file. The definition of the `MPEG4MediaData` is as follows:

Syntax

```
aligned(8) class MPEG4MediaData {
  bit(1) start;
  bit(1) end;
  bit(1) key;
  bit(5) reserved;
  unsigned int (32) timestamp;
  unsigned int (16) length;     // of following AU data
  AUData();
}
```

Semantics

`start` is a flag that is set to 1 to indicate, that the following AUData is the first fragment of a fragmented AU or a complete AU.
`end` is a flag that is set to 1 to indicate, that the following AUData is the final fragment of a fragmented AU or a complete AU.
`key` is a flag that is set when the AUData is part of, or a complete reference, key or random access frame.
`reserved` are 5 reserved bits which SHOULD be set to zero.
`timestamp` is a 32 bit unsigned integer that specifies the composition time of the `AUData` in time-scale ticks. It is based on the `time_scale` as signaled during session setup in the `MPEG4ConfigBox`. Note that the timestamp may wrap-around after an overflow.
`length` is a 16 bit unsigned integer, that defines the length of the following `AUData`. This value MUST be in the range 1-65528.
`AUData` is either a complete AU or a fragment of an AU.

Note that the number of `MPEG4MediaData` records is derived from the total length of the payload in the MPEG-4 Data Message. I.e., if the end of the Ultravox payload is not reached after parsing one `MPEG4MediaData` record, another record follows. The entire size of an Access Unit is not known instantly if the Access Unit has been fragmented (start and end bit are not both set). It can be calculated by accumulating the length fields of all MPEG4MediaData records, that form an Access Unit.

All Access Units and fragmented Access Units MUST be transmitted in decoding order, so that fragments can be reassembled by concatenating all AUData units with the same timestamp in the order they were received by the Listener. For AVC, each Access Unit is conveyed as stored in file and hence contains the NALU length field. The size of the NALU length field is available from the `SampleDescriptionBox` during session setup (see [2] and [4] for further information on the NALU length field).

5. Synchronization

Since the payload format for MPEG-4 Data Message includes a timestamp for each AU and all media streams use a common `time_scale` as indicated in the MPEG-4 Configuration Message, a Listener can easily align the media streams in the playout time-line after tuning into an Ultravox session.

The resolution of the `time_scale` determines how accurately audio and video (and other media) can be aligned. If the time scale is too coarse (e.g. 1 time unit/second) then no appropriate synchronization (e.g. lip-synch) is possible. On the other hand, at most one wrap-around of the time stamp SHALL occur during the `initial_delay` which sets an upper limit on the resolution. Typical values are in the range 8.000 – 90.000.

6. References

[1] ISO/IEC 14496-1:2003: “Information technology – Coding of audio-visual objects – Part 1: Systems”.

[2] ISO/IEC 14496-12:2003: “Information technology – Coding of audio-visual objects – Part 12: ISO base media file format

[3] ISO/IEC 14496-14:2003: “Information technology – Coding of audio-visual objects – Part 14: MP4 file format”.

[4] ISO/IEC 14496-15: 2004: “Information technology – Coding of audio-visual objects – Part 15: Advanced Video Coding (AVC) file format”.

[5] Internet Streaming Media Alliance: ”Ultravox”, TBD.

[6] IETF RFC 2119: “Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997.

[7] Internet Streaming Media Alliance: “ISMA 1.0 Encryption and Authentication”, Version 1.0, January 2004

Modification History

Date	Revision	Author	Description of changes
02/06/05	Draft 0.1	Stefan Döhla, Nikolaus Färber, Jochen Issing	Initial version for 17th Forum
23/08/05	Draft 0.2	Stefan Döhla, Nikolaus Färber, Jochen Issing	- Use common time scale for all streams (which eliminates the need for Synch-Message). - Restructure MPEG-4 Configuration Message into session-level and media-level parameters - Introduce initial_delay parameter