Internet Engineering Task Force
Internet Draft

Avaro-France Telecom
Basso-AT&T
Casner-Packet Design
Civanlar-AT&T
Gentric-Philips
Herpel-Thomson
Lifshitz-Optibase
Lim-mp4cast
Perkins-ISI
van der Meer-Philips

April 2001
Expires Oct. 2001

Document: draft-gentric-avt-mpeg4-multisl-03.txt

RTP Payload Format for MPEG-4 Streams

Status of this Memo

Abstract

This document describes a payload format for transporting MPEG-4 encoded data using RTP. MPEG-4 is a recent standard from ISO/IEC for the coding of natural and synthetic audio-visual data. Several services provided by RTP are beneficial for MPEG-4 encoded data transport over the Internet. Additionally, the use of RTP makes it possible to synchronize MPEG-4 data with other real-time data types.

This specification is a product of the Audio/Video Transport working group within the Internet Engineering Task Force and ISO/IEC MPEG-4 ad hoc group on MPEG-4 over Internet. Comments are solicited and should be addressed to the working group's mailing list at rem-

conf@es.net and/or the authors.

1. Introduction

MPEG-4 is a recent standard from ISO/IEC for the coding of natural and synthetic audio-visual data in the form of audiovisual objects that are arranged into an audiovisual scene by means of a scene description [1][2][3][4]. This draft specifies an RTP [5] payload format for transporting MPEG-4 encoded data streams.

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

The benefits of using RTP for MPEG-4 data stream transport include:

i. Ability to synchronize MPEG-4 streams with other RTP payloads

ii. Monitoring MPEG-4 delivery performance through RTCP

iii. Combining MPEG-4 and other real-time data streams received from multiple end-systems into a set of consolidated streams through RTP mixers

iv. Converting data types, etc. through the use of RTP translators.

1.1 Overview of MPEG-4 End-System Architecture

Fig. 1 below shows the general layered architecture of MPEG-4 terminals. The Compression Layer processes individual audio-visual media streams. The MPEG-4 compression schemes are defined in the ISO/IEC specifications 14496-2 [2] and 14496-3 [3]. The compression schemes in MPEG-4 achieve efficient encoding over a bandwidth ranging from several kbps to many Mbps. The audio-visual content compressed by this layer is organized into Elementary Streams (ESs). The MPEG-4 standard specifies MPEG-4 compliant streams. Within the constraint of this compliance the compression layer is unaware of a specific delivery technology, but it can be made to react to the characteristics of a particular delivery layer such as the path-MTU or loss characteristics. Also, some compressors can be designed to be delivery specific for implementation efficiency. In such cases the compressor may work in a non-optimal fashion with delivery technologies that are different than the one it is specifically designed to operate with.

The hierarchical relations, location and properties of ESs in a

presentation are described by a dynamic set of Object Descriptors (ODs). Each OD groups one or more ES Descriptors referring to a single content item (audio-visual object). Hence, multiple alternative or hierarchical representations of each content item are possible.

ODs are themselves conveyed through one or more ESs. A complete set of ODs can be seen as an MPEG-4 resource or session description at a

stream level. The resource description may itself be hierarchical, i.e. an ES conveying an OD may describe other ESs conveying other ODs.

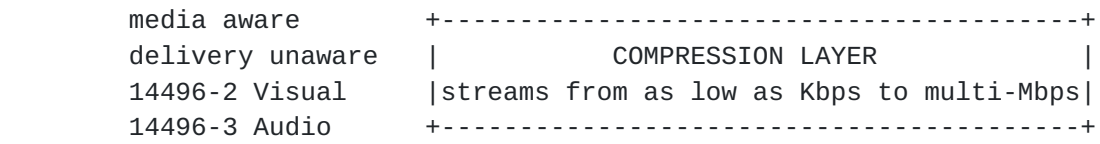
The session description is accompanied by a dynamic scene description, Binary Format for Scene (BIFS), again conveyed through one or more ESs. At this level, content is identified in terms of audio-visual objects. The spatio-temporal location of each object is defined by BIFS. The audio-visual content of those objects that are synthetic and static are described by BIFS also. Natural and animated synthetic objects may refer to an OD that points to one or more ESs that carries the coded representation of the object or its animation data.

By conveying the session (or resource) description as well as the scene (or content composition) description through their own ESs, it is made possible to change portions of the content composition and the number and properties of media streams that carry the audio-visual content separately and dynamically at well known instants in time.

One or more initial Scene Description streams and the corresponding OD stream have to be pointed to by an initial object descriptor (IOD). The IOD needs to be made available to the receivers through some out-of-band means that are out of scope of this payload specification. However in the context of transport on IP networks it is defined in a separate document [9].

A homogeneous encapsulation of ESs carrying media or control (ODs, BIFS) data is defined by the Sync Layer (SL) that primarily provides the synchronization between streams. The Compression Layer organizes the ESs in Access Units (AU), the smallest elements that can be attributed individual timestamps. Integer or fractional AUs are then encapsulated in SL packets. All consecutive data from one stream is called an SL-packetized stream at this layer. The interface between the compression layer and the SL is called the Elementary Stream Interface (ESI). The ESI is informative.

The Delivery Layer in MPEG-4 consists of the Delivery Multimedia Integration Framework defined in ISO/IEC 14496-6 [4]. This layer is media unaware but delivery technology aware. It provides transparent access to and delivery of content irrespective of the technologies used. The interface between the SL and DMIF is called the DMIF Application Interface (DAI). It offers content location independent procedures for establishing MPEG-4 sessions and access to transport channels. The specification of this payload format is considered as a part of the MPEG-4 Delivery Layer.



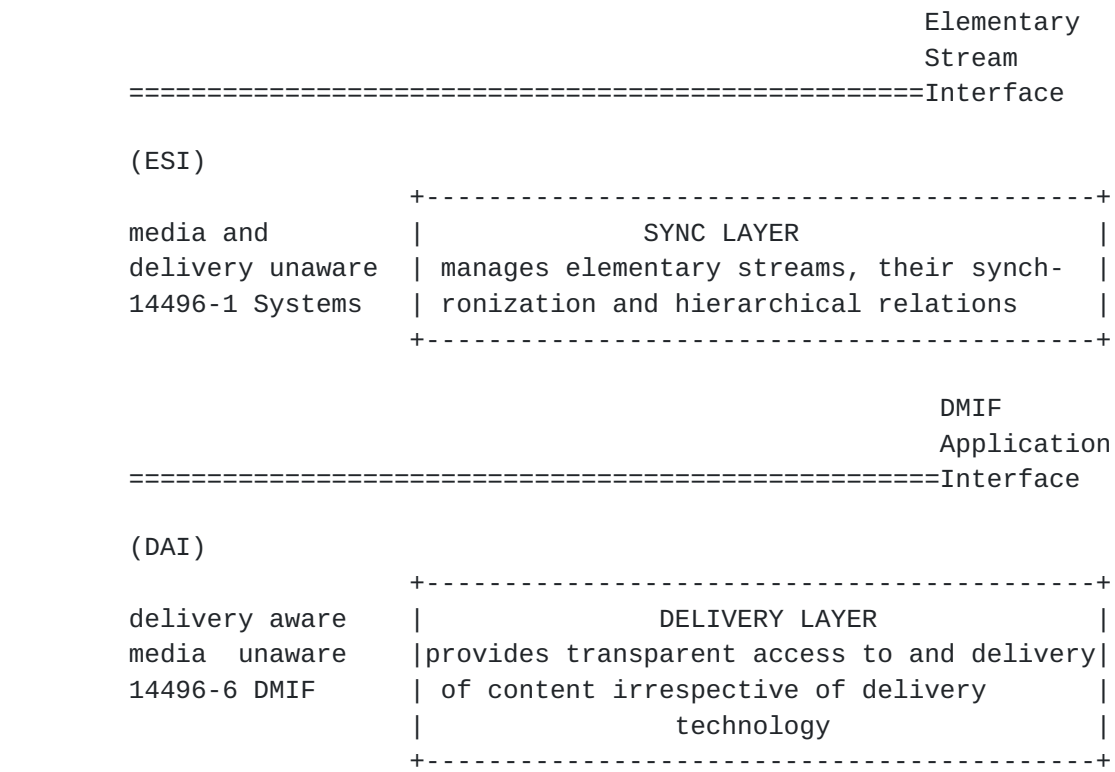


Figure 1: General MPEG-4 terminal architecture

1.2 MPEG-4 Elementary Stream Data Packetization

The ESs from the encoders are fed into the SL with indications of AU boundaries, random access points, desired composition time and the

current time.

The Sync Layer fragments the ESs into SL packets, each containing a header that encodes information conveyed through the ESI. If the AU is larger than a SL packet, subsequent packets containing remaining parts of the AU are generated with subset headers until the complete AU is packetized.

The syntax of the Sync Layer is configurable and can be adapted to the needs of the stream to be transported. This includes the possibility to select the presence or absence of individual syntax elements as well as configuration of their length in bits. The configuration for each individual stream is conveyed in a SLConfigDescriptor, which is an integral part of the ES Descriptor for this stream. The MPEG-4 SLConfigDescriptor, being configuration information, is not carried by the media stream itself but is rather transported via an ObjectDescriptor Stream encoded using the MPEG-4 Object Description framework. This can be done in a separate stream using this payload format (see section 4.2 for details). The SLConfigDescriptor MAY be transported by other means (for example as a a=fmtp parameter, see section 5).

2. Analysis of the carriage of MPEG-4 over IP

When transporting MPEG-4 audio and video, applications may or may not require the use of MPEG-4 systems. To achieve the highest level of interoperability between all MPEG-4 applications, it is desirable that (a) in both cases the same MPEG-4 transport format can be used and that (b) receivers that have no MPEG-4 system knowledge can easily skip the MPEG-4 system specific information, if any.

RTP is perfectly suitable to transport MPEG-4 audio and MPEG-4 video, but when using MPEG-4 systems a problem arises from the fact that both RTP and MPEG-4 systems contain a synchronization layer. In particular, the RTP header duplicates some of the information provided in SL packet headers such as the composition timestamps (CTSs) and the marker bit that signals the end of access units.

To avoid unnecessary overhead and potential interoperability risks when transporting MPEG-4 systems, it is desirable to remove the redundancy between the SL packet header and the RTP packet header. To be independent on the use of MPEG-4 systems, synchronization can rely on the parameters provided in the RTP header.

In case SL headers are used, the redundant fields are removed from

the SL header, producing "reduced SL headers".
The remaining information from the SL header, if any, is contained inside the RTP packet payload, together with the SL packet payload. The combination of RTP packet headers and reduced SL packet headers can be used to logically map the RTP packets to complete SL packets.

Some of the information contained in the reduced SL headers is also useful for transport over RTP when MPEG-4 systems is not used.

For that reason the information in the "reduced" SL headers is split into "general useful information" and "MPEG-4 systems only information".

The "general useful information" hereinafter called Mapped SL Packet Header (MSLH) is carried by a number of fields configurable using parameters defined in section 5.1; all receivers can parse these fields.

The "MPEG-4 systems only information", if any, is contained in a reduced SL header, hereinafter called Remaining SL Packet Header (RSLH), also signaled by parameters (see section 5.1)and preceded by a length field, so as to enable easy skipping of this information by non-MPEG-4 system devices.

This is depicted in figure 2.

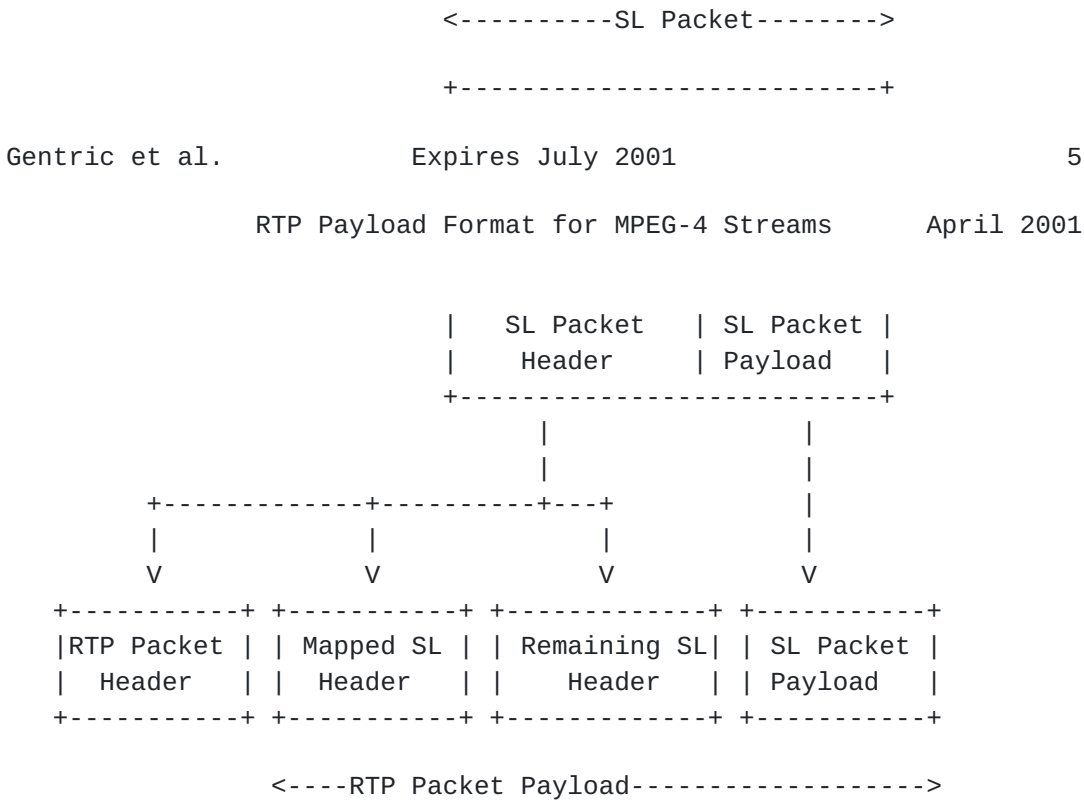


Figure 2: Mapping of SL Packet into RTP packet

This RTP payload format has been designed so that it can be configured (using parameters described in section 5.1) to be identical to RFC 3016 for the recommended MPEG-4 video configurations. Hence receivers that comply with this payload specification can decode such RTP payload.

3. Payload Format

The RTP Payload corresponds to an integer number of SL packets.

SL packets inside RTP packets MUST be in the SL stream order i.e:
i) decodingTimeStamp order, if present
ii) packetSequenceNumber order, if present
iii) Implicit decoding order in all other cases.

The SL Packet Headers are transformed into RSLH with some fields extracted to be mapped in the RTP header and others extracted to be mapped in the corresponding MSLH. The SL Packet Payload is unchanged.

This payload format has two modes. The "SingleSL" mode is a mode where a single SL packet is transported per RTP packet. The "MultipleSL" mode is a mode where more than one SL packet are transported per RTP packet. The default mode is the Single-SL mode. The mode can be set to Multiple-SL by adding a non-zero SLPPSize or SLPPSizeLength parameter (see section 5.1).

RTP Packets SHOULD be sent in the SL stream order (as defined above).

The size (or number) of the SL packet(s) SHOULD be adjusted such that the resulting RTP packet is not larger than the path-MTU. To handle larger packets, this payload format relies on lower layers for fragmentation, which may not be desirable.

3.1 RTP Header Fields Usage

Payload Type (PT): The assignment of an RTP payload type for this new packet format is outside the scope of this document, and will not be specified here. It is expected that the RTP profile for a particular class of applications will assign a payload type for this encoding, or if that is not done then a payload type in the dynamic range shall be chosen.

Marker (M) bit: The M bit is set to 1 when all SL packets in the RTP packet are Access Units ends i.e. the M bit maps to the SL accessUnitEndFlag.

M is set to 1 when the RTP packet contains either:
. a single SL packet containing a full Access Unit
. a single SL packet transporting the last fragment of an Access Unit
. multiple SL packets each containing a full Access Unit
. multiple SL packets each containing the last fragment of an Access Unit
. multiple SL packets each containing either a full Access Unit or the last fragment of an Access Unit

The last 2 cases occur when using specific interleaving schemes. In some interleaving schemes it may not be practical to reshuffle the SL packets so as to group Access Unit ends in the same RTP packet. In that case, Access Unit boundaries SHOULD be transported using one or both of the SL flags accessUnitStartFlag and accessUnitEndFlag.

Extension (X) bit: Defined by the RTP profile used.

Sequence Number: The RTP sequence number should be generated by the sender with a constant random offset and does not have to be correlated to any (optional) MPEG-4 SL sequence numbers.

Timestamp: Set to the value in the compositionTimeStamp field of the first SL packet, if present. If compositionTimeStamp has less than 32 bits length, the MSBs of timestamp MUST be set to zero.

Although it is available from the SL configuration data, the resolution of the timestamp may need to be conveyed explicitly through some out-of-band means to be used by network elements which are not MPEG-4 aware.

If compositionTimeStamp has more than 32 bits length, this payload format cannot be used.

In all cases, the sender SHALL always make sure that RTP time stamps are identical only for RTP packets transporting fragments of the same Access Unit.

In case compositionTimeStamp is not present in the current SL packet, but has been present in a previous SL packet the reason is

that this is the same Access Unit that has been fragmented therefore the same timestamp value MUST be taken as RTP timestamp.

If compositionTimeStamp is never present in SL packets for this stream, the RTP packetizer SHOULD convey a reading of a local clock at the time the RTP packet is created.

According to RFC1889 [5, Section 5.1] timestamps are recommended to start at a random value for security reasons. However then, a receiver is not in the general case able to reconstruct the original MPEG-4 Time Stamps (CTS, DTS, OCR) which can be of use for applications where streams from multiple sources are to be synchronized. Therefore the usage of such a random offset SHOULD be avoided.

Note that since RTP devices may re-stamp the stream, all time stamps inside of the RTP payload (CTS and DTS in MSLH, OCR in RSLH) MUST be expressed as difference to the RTP time stamp. Since this subtraction may lead to negative values, the offset MUST be encoded as a two's complement signed integer in network byte order. Note these offsets (delta) typically require much fewer bits to be encoded than the original length, which is another justification.

SSRC, CC and CSRC fields are used as described in RFC 1889 [5].

RTCP SHOULD be used as defined in RFC 1889 [5].

RTP timestamps in RTCP SR packets: according to the RTP timing model, the RTP timestamp that is carried into an RTCP SR packet is the same as the compositionTimeStamp that would be applied to an RTP packet for data that was sampled at the instant the SR packet is being generated and sent. The RTP timestamp value is calculated from the NTP timestamp for the current time, which also goes in the RTCP SR packet. To perform that calculation, an implementation needs to periodically establish a correspondence between the CTS value of a data packet and the NTP time at which that data was sampled.

3.2 RTP payload structure

The packet payload structure consists of 3 byte-aligned sections.

The first section is the MSLHSection and contains Mapped SL Packet Headers (MSLH). The MSLH structure is described in 3.3. In the Single-SL mode this section is empty by default.

The second section is the RSLHSection and contains Remaining SL Headers (RSLH). The RSLH structure is described in 3.5. By default this section is empty.

The last section (SLPPSection) contains the SL packet payloads. This section is never empty.

The Nth MSLH in the MSLHSection, the Nth RSLH in the RSLHSection and the Nth SL packet payload in the SLPPSection correspond to the Nth SL packet transported by the RTP packet.

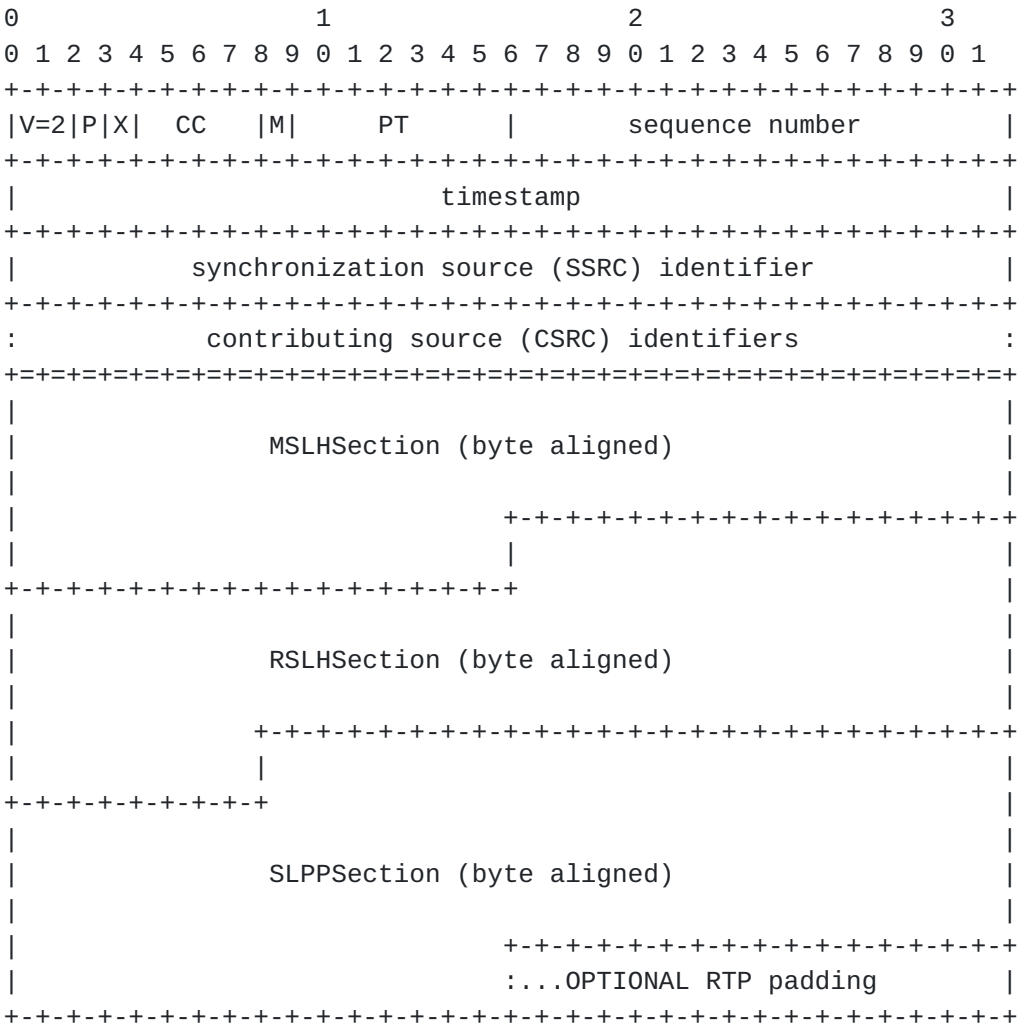


Figure 3: An RTP packet for MPEG-4

3.3 MSLHSection structure

If the MSLHSection consumes a non-integer number of bytes, up to 7 zero-valued padding bits MUST be inserted at the end in order to achieve byte-alignment.

In the Single-SL mode the MSLHSection consists of a single MSLH.

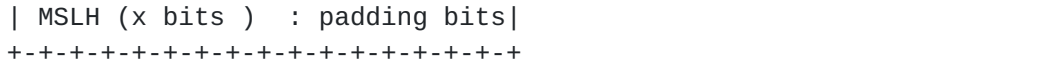


Figure 4: MSLHSection structure in Single-SL mode

In the Multiple-SL mode this section consist of a 2 bytes field giving the size in bits (in network byte order) of the following block of bit-wise concatenated MSLHs.

This size field is absent in the Single-SL mode not because it is not needed (which would be a minor gain) but for compatibility with RFC 3016.

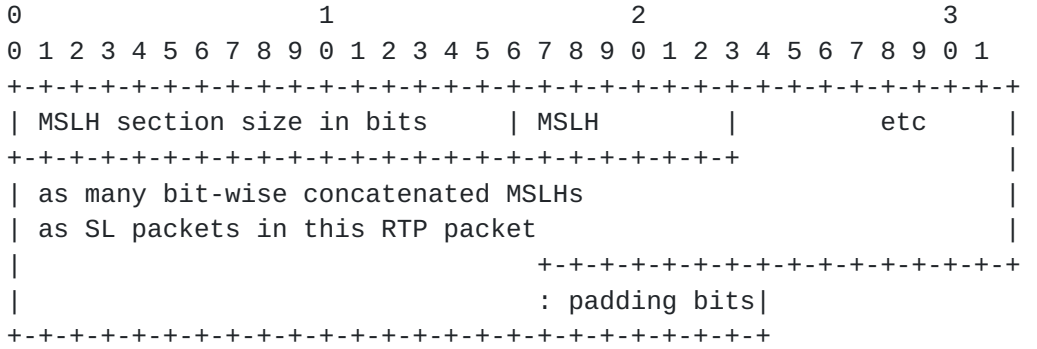
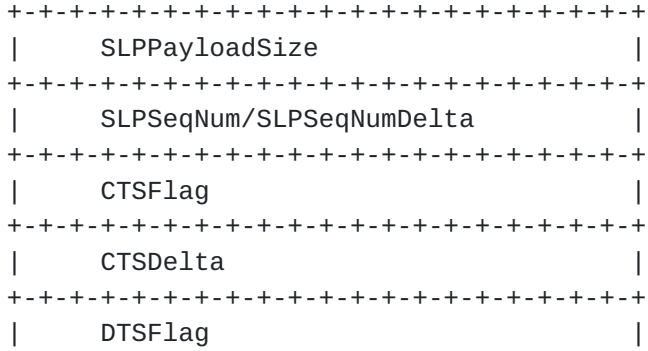


Figure 5: MSLHSection structure in Multiple-SL mode

3.4 MSLH structure

The Mapped SL Packet Header content depends on parameters (as described in section 5.1); by default it is empty for the Single-SL mode and contains only the SLPPayloadSize (SL Packet Payload Size) field in the Multiple-SL mode.

When all options are used the MSLH structure is given in figure 6.



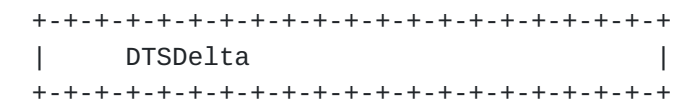


Figure 6: Mapped SL Packet Header (MSLH) structure

In the general case a receiver can only discover the size of a MSLH by parsing it since for example the presence of CTSDelta is signaled by the value of CTSFlag.

3.4.1 Fields of MSLH

SLPPayloadSize (SL Packet Payload Size): Indicates the size in bytes of the associated SL Packet Payload, which can be found in the SLPPSection of the RTP packet. The length in bits of this field is signaled by the SLPPSizeLength parameter (see section 5.1).

SLPSeqNum/SLPSeqNumDelta: Encodes the packetSequenceNumber (serial number) of the SL Packet.

SLPSeqNum is found only for the first SL packet. SLPSeqNumDelta is optional and -if present- appears for subsequent (non-first) SL packets.

The length in bits of the SLPSeqNum field is defined by the SLPSeqNumLength parameter (see section 5.1).

The length in bits of the SLPSeqNumDelta field is defined by the SLPSeqNumDeltaLength parameter (see section 5.1).

If the parameter SLPSeqNumDeltaLength is defined, non-first SL packets have their packetSequenceNumber encoded as a difference named SLPSeqNumDelta. This difference is relative to the previous SL packet in the RTP packet according to (with i>=0):
packetSequenceNumber(0) = SLPSeqNum(0)
packetSequenceNumber(i+1) = packetSequenceNumber(i) + SLPSeqNumDelta(i+1) + 1

If the parameter SLPSeqNumDeltaLength is not defined the default value is zero i.e. the SLPSeqNumDelta field is not present for non-first SL packets. Furthermore receivers SHALL then apply the above formula with SLPSeqNumDelta equal to zero i.e. by default packetSequenceNumber is incremented by 1 for each SL packet in one RTP packet. This means that for streams that use packetSequenceNumber and are not interleaved the transport of

packetSequenceNumber in the Multiple-SL mode is "almost free".

CTSFlag (1 bit): Indicates whether the CTSDelta field is present. A value of 1 indicates that the field is present, a value of 0 that it is not present.

If CTSDeltaLength is not zero this field is present in all MSLH since the receiver needs it to reconstruct the compositionTimeStampFlag of SL Headers.

CTSDelta: Specifies the value of the CTS as a 2-complement offset (delta) from the timestamp in the RTP header of this RTP packet. The length in bits of each CTSDelta field is specified by the CTSDeltaLength parameter (see section 5.1).

This field is present if CTSFlag is 1 except for the first MSLH since the composition time stamp of the first SL packet is mapped to the RTP time stamp, regardless of whether CTSFlag is 1. In all cases the sender MUST remove the compositionTimeStamp from the RSLH.

DTSFlag (1 bit): Indicates whether the DTSDelta field is present. A value of 1 indicates that DTSDelta is present, a value of 0 that it is not present.

If DTSDeltaLength is not zero this field is present in all MSLH since the receiver needs it to reconstruct the decodingTimeStampFlag of SL Headers.

DTSDelta: Specifies the value of the decodingTimeStamp as a 2 complement offset (delta) from the timestamp in the RTP header of this packet. The length in bits of each DTSDelta field is specified by the DTSDeltaLength parameter (see section 5.1).

This field appears when DTSFlag is 1. The sender MUST always remove the decodingTimeStamp from the RSLH.

3.4.2 Relationship between sizes of MSLH fields and parameters

The relationship between a Mapped SL Packet Header and the related parameters is as follows:

Fields of MSLPH	Number of bits (parameters)
SLPPayloadSize	SLPPSizeLength

SLPSeqNum	SLPSeqNumLength
SLPSeqNumDelta	SLPSeqNumDeltaLength
CTSFlag	1 If (CTSDeltaLength > 0)
CTSDelta	CTSDeltaLength If(CTSFlag==1)
DTSFlag	1 If (DTSDeltaLength > 0)
DTSDelta	DTSDeltaLength If(DTSFlag==1)

Table 1: Relationship between MSLH fieldsÆ size and parameters

3.5 RSLHSection structure

This section consists of a field (RSLHSectionSize) giving the size in bits of the following block of bit-wise concatenated RSLHs.

If the section consumes a non-integer number of bytes, up to 7 zero padding bits MUST be inserted at the end in order to achieve byte-alignment.

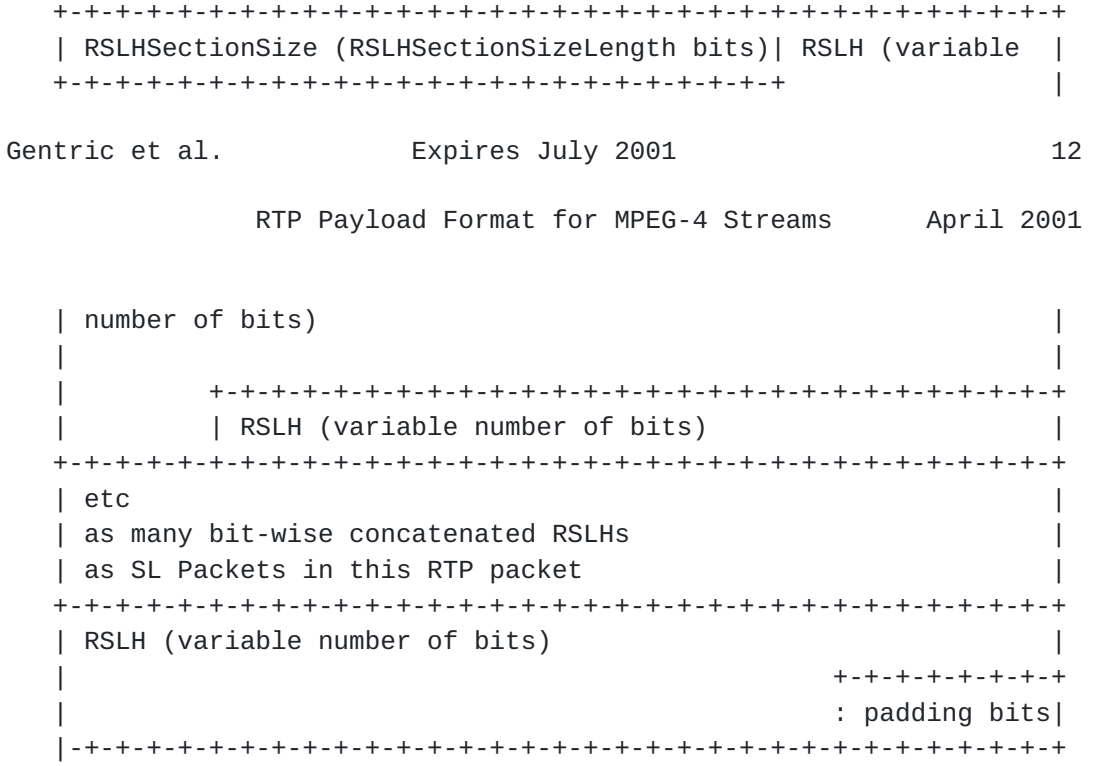


Figure 7: RSLHSection structure

The length in bits of the RSLHSectionSize field is RSLHSectionSizeLength and is specified with a default value of zero indicating that the whole RSLHSection is absent.

Fields of RSLHSection	Number of bits
RSLHSectionSize	RSLHSectionSizeLength
all bit-wise concatenated RSLHs	RSLHSectionSize

Table 2: Sizes in bits inside RSLHSection

Parsing of the bit-wise concatenated RSLHs requires MPEG-4 system awareness, specifically it requires to understand the MPEG-4 Synchronization Layer (SL) syntax and the modifications to this syntax described in the next section.

However thanks to the RSLHSectionSize field non-MPEG-4-system receivers MAY skip this part by rounding up RSLPHSize/8 to the next integer number of bytes.

3.6 RSLH structure

A Remaining SL Packet Header (RSLH) is what remains of an SL header after modifications for mapping into this payload format.

The following modifications of the SL packet header MUST be applied. The other fields of the SL packet header MUST remain unchanged but are bit-shifted to fill in the gaps left by the operations specified below.

3.6.1 Removal of fields

The following SL Packet Header fields -if present- are removed since they are mapped either in the RTP header or in the corresponding MSLH:

. compositionTimeStampFlag
. compositionTimeStamp
. decodingTimeStampFlag
. decodingTimeStamp
. packetSequenceNumber
. AccessUnitEndFlag (in Single-SL mode only)

The AccessUnitEndFlag, when present for a given stream, MUST be removed from every RSLH when using the Single-SL mode since it has

the same meaning as the Marker bit (and for compatibility with RFC 3016). However when using the Multiple-SL mode, AccessUnitEndFlag MUST NOT be removed since it is useful to signal individual AU ends.

3.6.2 Mapping of OCR

Furthermore if the SL Packet header contains an OCR, then this field is encoded in the RSLH as a 2-complement difference (delta) exactly like a compositionTimeStamp or a decodingTimeStamp in the MSLH. The length in bit of this difference is indicated by the OCRDeltaLength parameter (see section 5.1).

With this payload format OCRs MUST have the same clock resolution as Time Stamps.

If compositionTimeStamp is not present for a SL packet that has OCR then the OCR SHALL be encoded as a difference to the RTP time stamp.

3.6.3 Degradation Priority

For streams that use the optional degradationPriority field in the SL Packet Headers, only SL packets with the same degradation priority SHALL be transported by one RTP packet so that components may dispatch the RTP packets according to appropriate QOS or protection schemes. Furthermore only the first RSLH of one RTP packet SHALL contain the degradationPriority field since it would be otherwise redundant.

3.7 SLPPSection structure

The SLPPSection (SL Packet Payload Section) contains the concatenated SL Packet Payloads. By definition SL Packet Payloads are byte aligned.

For efficiency SL packets do not carry their own payload size. This is not an issue for RTP packets that contain a single SL Packet.

However in the Multiple-SL mode the size of each SL packet payload MUST be available to the receiver.

If the SL packet payload size is constant for a stream, the size information SHOULD NOT be transported in the RTP packet. However in that case it MUST be signaled using the SLPPSize parameter (see section 5.1).

If the SL packet payload size is variable then the size of each SL packet payload MUST be indicated in the corresponding MSLH. In order to do so the MSLH MUST contain a SLPPayloadSize field. The number of bits on which this SLPPayloadSize field is encoded MUST be indicated using the SLPPSizeLength parameter (see section 5.1).

The absence of either SLPPSize or SLPPSizeLength indicates the Single-SL mode i.e. that a single SL packet is transported in each RTP packet for that stream.

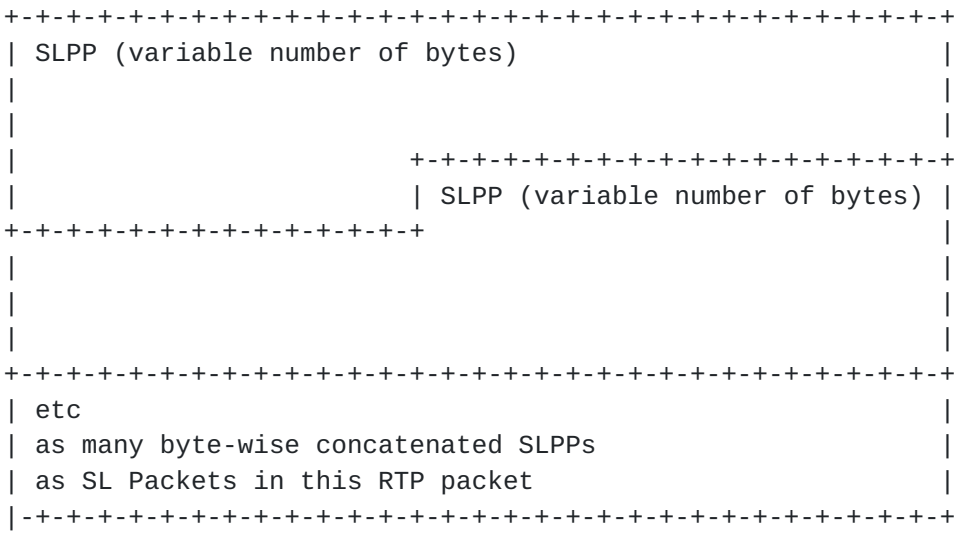


Figure 8: SLPPSection structure

3.8 Interleaving

SL Packets MAY be interleaved. Senders MAY perform interleaving. Receivers MUST support interleaving.

When interleaving of SL packets is used it SHALL be implemented using the SLPSeqNum field of MSLH.

The AUSequenceNumber field of the SL header MUST NOT be used for interleaving since firstly it may collide with the Scene Description Carousel usage described in section 5.1 and secondly it is not visible to non-MPEG-4 system receivers.

The conjunction of RTP sequence number and SLPSeqNum can produce a quasi-unique identifier for each SL packet so that a receiver can unambiguously reconstruct the original order even in case of out-of-order packets, packet loss or duplication.

3.9 Fragmentation Rules

This section specifies rules for senders in order to prevent media decoding difficulties at the receiver end.

MPEG-4 Access Units are the default fragments for MPEG-4 bitstreams and SHOULD be mapped one-to-one to RTP packets of this format with two exceptions:
- Access Units larger than the MTU,
- When using interleaving for better packet loss resilience.

In all cases Access Unit start MUST be aligned with SL packet start.

This section gives rules to apply when performing Access Unit fragmentation.

Some MPEG-4 codecs define optional syntax for Access Units sub-entities (fragments) that are independently decodable for error resilience purposes. Examples are Video Packets for video and Error Sensitivity Categories (ESC) for audio. This always corresponds to specific bitstream syntax, which is signaled in the DecoderSpecificInfo inside the DecoderConfig in SLConfig, and/or using the corresponding parameters as described in section 5.1. Therefore encoders and decoders are both aware whether they are operating in such a mode or not (however since this codec configuration is an opaque data block this is not explicitly signaled by this payload format).

If not operating in such a mode it is obvious that the decoder has to skip packets after a loss until an Access Unit start is received. Similarly decoder implementations that do not implement robust decoding of Access Units fragments have to discard all packets after a packet loss until an Access Unit start is received. In the same way decoder implementations that do not implement re-synchronization at any Access Units start have to discard all packets after a packet loss until a Random Access Point Access Unit is received.

One problem would arise however for decoder implementations that try to restart decoding after a packet loss if independently decodable fragments are signaled (in the decoder configuration) but the fragments actually received are not independently decodable because the RTP sender has made RTP packets on different boundaries than the fragments provided by the encoder (so this issue applies to the interface between the encoder and the RTP sender and to the RTP sender component itself).

For this reason the following rules must apply to SL streams that are specifically made for transport with this payload format:

SL packets SHOULD be codec-semantic entities in the spirit of ALF i.e. either complete Access Units or fragments of Access Units that

are independently decodable. Specifically when a given codec has an independently decodable Access Unit fragments optional syntax this option SHOULD be used.

Furthermore when streams are generated using independently decodable Access Units fragments these Access Units fragments MUST be mapped one-to-one into SL packets. Consequently independently decodable Access Units fragments MUST NOT be split across several SL packets and therefore MUST NOT be split across several RTP packets.

For example an MPEG-4 audio stream encoded using the ESC syntax MUST NOT split one ESC across 2 RTP packets.

This rule is relaxed when using MPEG-4 Video Packets for two reasons: firstly Video Packets can be much larger than typical MTU and secondly all Video Packets start with a specific resynchronization marker that can be unambiguously detected. Therefore for video streams using the Video Packet syntax Video Packets MAY be split across several SL packets although it is strongly RECOMMENDED to always adapt the Video Packet size to fit the MTU. A Video Packet start MUST always be aligned with a SL packet start, except when a GOV is present, in which case the GOV and the first Video Packet of the following VOP MUST be included in the same SL packet.

4. Other issues

4.1 SL packetized stream reconstruction

The MPEG-4 over IP framework [9] requires that the way a receiver can reconstruct a valid SL packetized stream shall be documented, this is the purpose of this section.

Since this format directly transports SL packets this reconstruction is trivial with the following rules:

- SLPacketHeader.packetSequenceNumber is restored from MSLH.SLPSeqNum for the first SL packet in the RTP packet (i= 0):
SLPacketHeader.packetSequenceNumber(0) = MSLH.SLPSeqNum(0)
and for subsequent packets using (for i>=0) :
SLPacketHeader.packetSequenceNumber(i+1) =
SLPacketHeader.packetSequenceNumber(i) + MSLH.SLPSeqNumDelta(i+1) +1

- All time stamps (CTS, DTS, OCR), when present, are restored from the delta values.
- Time stamps flags (CTSFlag, DTSFlag) in MSLH are used to

reconstruct respectively the compositionTimeStampFlag and decodingTimeStampFlag of SLPacketHeader.

Specifically the reconstruction depends on the parameters as follows:

If CTSDeltaLength is absent or equals 0:
The SL stream reconstruction rules are:
. for the first (or only) SL packet:
 . if SLConfig.useTimeStamps == true, then:

  . SLPacketHeader.compositionTimeStampFlag = true
  . SLPacketHeader.compositionTimeStamp = RTP TimeStamp
 . if SLConfig.useTimeStamps == false, then:
  . SLPacketHeader.compositionTimeStampFlag is not defined
. for the following SL packets:
 . SLPacketHeader.compositionTimeStampFlag = false

If CTSDeltaLength is not zero:
 . SLPacketHeader.compositionTimeStampFlag = MSLH.CTSFlag
 . SLPacketHeader.compositionTimeStamp = RTP TimeStamp + MSLH.CTSDelta

- The other SL packet header fields SHALL remain as found in RSLH.

It is obvious that in the general case the reconstruction of the original SL packetized stream requires SL-awareness. However this payload format allows in all cases a receiver that does not know about the SL syntax to reconstruct the semantic of SL for the following very useful features:
- Packet order (decoding order)
- Access Unit boundaries (using the M bit)
- Access Unit fragments (i.e. SL packet boundaries using MSLH.SLPPayloadSize)
- Composition Time Stamps (using the RTP Time Stamp and MSLH.CTSDelta)
- Decoding Time Stamps (using the RTP Time Stamp and MSLH.DTSDelta)
- Packet sequence number (using the RTP Time Sequence number and MSLH.SLPSeqNum)

4.2 Handling of scene description streams

MPEG-4 introduces new stream types as described in section 1 namely Object Descriptors and BIFS. In the following both OD and BIFS are discussed on the same basis i.e. as "scene description".

Considering Scene description as a "stream-able" type of content is

a rather new concept and for that reasons some specific comments are needed.

Typically scene descriptions are encoded in such a way that information loss would in the general case cripple the presentation beyond any hope of repair by the receiver. Still this is well suited for a number of multimedia applications were the scene is first made available via reliable channels to the client and then played. This payload format is not intended for this type of applications for which download of MPEG-4 interchange (.mp4) files is typical. However it can also be used if the RTP packets are transported using TCP or any other reliable protocol.

On the other hand MPEG-4 has introduced the possibility to dynamically change the scene description by sending animation information (changes in parameters) and structural change information (updates). Since this information has to be sent in a

timely fashion MPEG-4 has defined a number of techniques in order to encode the scene description in a manner that makes it behave similarly to other temporal encoding schemes such as audio and video. This payload format is intended for this usage.

Note that in many cases the application will consist of first the reliable transmission of a static initial scene followed by the streaming of animations and updates. For this reason the usage of this payload format is attractive since it offers a unique solution.

Senders must be aware that suitable schemes should be used when scene description streams transport sensitive configuration information. For example in case the RTP packet transporting an OD-update command would be lost, the corresponding media stream would not be accessible by the receiver.

Redundancy is a possibility and may either be added by tools hierarchically higher than this payload format, e.g. by packet based FEC, re-transmission, or similar tools. In such a case, the general congestion control principles have to be observed.

Since BIFS and OD streams may be modified during the session with update commands, there is a need to send both update commands and full BIFS/OD refresh. For that reason MPEG-4 defines Random Access Points (RAP) for scene description streams (OD and BIFS) where by definition a decoder can restart decoding i.e. receives a "full update" of the scene. This mechanism is called Scene and Object Description Carrousel. The AU Sequence Number field of SL Packet Header is used to support this behavior at the Synchronization

Layer. When two access units are sent consecutively with the same AU Sequence Number, the second one is assumed to be a semantic repetition of the first. If a receiver starts to listen in the middle of a session or has detected losses, it can skip all received Access Units until such a RAP. The periodicity of transmission of these RAPs should be chosen/adjusted depending on the application and the network it is deployed on; i.e. exactly like Intra-coded frames for video, it is the responsibility of the sender to make sure the periodicity of RAPs is suitable.

4.3 Multiplexing

An advanced MPEG-4 session may involve a large number of objects that may be as many as a few hundred, transporting each ES as an individual RTP stream may not always be practical. Allocating and controlling hundreds of destination addresses for each MPEG-4 session may pose insurmountable session administration problems. The input/output processing overhead at the end-points will be extremely high also. Additionally, low delay transmission of low bitrate data streams, e.g. facial animation parameters, results in extremely high header overheads.

To solve these problems, MPEG-4 data transport requires a multiplexing scheme that allows selective bundling of several ESs. This is beyond the scope of the payload format defined here.

The MPEG-4's Flexmux multiplexing scheme may be used for this purpose and a specific RTP payload format is being developed [11].

Another approach may be to develop a generic RTP multiplexing scheme usable for MPEG-4 data. The multiplexing scheme reported in [8] may be a candidate for this approach.

For MPEG-4 applications, the multiplexing technique needs to address the following requirements:

i. The ESs multiplexed in one stream can change frequently during a session. Consequently, the coding type, individual packet size and temporal relationships between the multiplexed data units must be handled dynamically.

ii. The multiplexing scheme should have a mechanism to determine the ES identifier (ES_ID) for each of the multiplexed packets. ES_ID is not a part of the SL header.

iii. In general, an SL packet does not contain information about its size. The multiplexing scheme should be able to delineate the multiplexed packets whose lengths may vary from a few bytes to close to the path-MTU.

4.5 Overlap with RFC 3016

This payload format has been designed to have an overlap with RFC 3016 [7]. The conditions for this overlap are:
Conditions for RFC 3016:
i. MPEG-4 video elementary streams only
ii. Maximum one VOP or Video Packet per RTP packet
Conditions for this payload format:
i. No structural parameters defined (or all set to zero), i.e. Single-SL mode with empty MSLH and empty RSLH.
ii. Receivers MUST be ready to accept (ignore) video configuration headers (e.g. VOSH, VO and VOL) and visual-object-sequence-end-code transported in-band.

5. Types and Names

This section describes the MIME types and names associated with this payload format. Section 5.1 is intended for registration with IANA in RFC 2048.

This format may require additional information about the mapping to be made available to the receiver. This is done using parameters described in the next section. The absence of any of these fields is equivalent to a field set to the default value, which is always

zero. The absence of any such parameters resolves into a default "basic" configuration.

In the MPEG-4 framework the SL stream configuration information is carried using the Object Descriptor. For compatibility with receivers that do not implement the full MPEG-4 system specification this information MAY also be signaled using parameters described here. When such information is present both in an Object Descriptor and as a parameter of this payload format it MUST be exactly the same.

For transport of MPEG-4 audio and video without the use of MPEG-4 systems, as well as to support non-MPEG-4 system receivers, it is also possible to transport information on the profile and level of the stream and on the decoder configuration. This is also described

in the next section.

5.1 MIME type registration

MIME media type name: "video" or ôaudio" or "application"

"video" SHOULD be used for MPEG-4 Video streams (ISO/IEC 14496-2) or MPEG-4 Systems streams that convey information needed for an audio/visual presentation.

"audio" SHOULD be used for MPEG-4 Audio streams (ISO/IEC 14496-3) or MPEG-4 Systems streams that convey information needed for an audio only presentation.

"application" SHOULD be used for MPEG-4 Systems streams (ISO/IEC14496-1) that serve other purposes than audio/visual presentation, e.g. in some cases when MPEG-J streams are transmitted.

MIME subtype name: mpeg4-sl

Required parameters: none

Optional parameters:

DTSDeltaLength:
The number of bits on which the DTSDelta field is encoded in MSLH. The default value is zero and indicates the absence of DTSFlag and DTSDelta in MSLH (the stream does not transport decodingTimeStamps). A value larger than zero indicates that there is a DTSFlag in each MSLH. Since decodingTimeStamp -if present- must be encoded as a difference to the RTP time stamp, the DTSDeltaLength parameter MUST be present in order to transport decodingTimeStamps with this payload format.

CTSDeltaLength:
The number of bits on which the CTSDelta field is encoded in (non-first) MSLH. The default value is zero and indicates the absence of

the CTSFlag and CTSDelta fields in MSLH. Non-zero values MOST NOT be signaled in the Single-SL mode. Since compositionTimeStamps -if present- must be encoded as a difference to the RTP time stamp, the CTSDeltaLength parameter MUST be present in order to transport compositionTimeStamps using this payload format (in the Multiple-SL mode). However CTSDeltaLength SHOULD be set to zero (or not signaled) for streams that have a constant Access Unit duration (which can be explicitly signaled using the DurationFlag and

AccessUnitDuration field of SLConfigDescriptor).

OCRDeltaLength:
The number of bits on which the OCRDelta field is encoded in RSLH. The default value is zero and indicates the absence of OCR for this stream. Since objectClockReference -if present- must be encoded as a difference to the RTP time stamp, the OCRDeltaLength parameter MUST be present in order to transport objectClockReferences with this payload format.

SLPPSizeLength:
The number of bits on which the SLPPayloadSize field of MSLH is encoded. The default value is zero and indicates the Single-SL mode (unless SLPPSize is present). Simultaneous presence of this parameter and SLPPSize is illegal. Either the SLPPSizeLength or SLPPSize parameter MUST be present in order to signal the Multiple-SL mode of this payload format.

SLPPSize:
The constant size in bytes of each SL Packet Payload for this stream. The default value is zero and indicates variable SL Packet Payload size (or the Single-SL mode if SLPPSizeLength is absent). Simultaneous presence of this parameter and SLPPSizeLength is illegal. Either the SLPPSizeLength or SLPPSize parameter MUST be present in order to signal the Multiple-SL mode of this payload format. When SLPPSize is present the SLPPayloadSize of MSLH in the RTP packets MUST NOT be present.

SLPSeqNumLength:
The number of bits on which the SLPSeqNum is encoded in the first MSLH. The default value is zero and indicates the absence of SLPSeqNum and SLPSeqNumDelta for all MSLHs. Since packetSequenceNumber -if present- must be mapped in MSLH, the SLPSeqNumLength parameter MUST be present in order to transport packetSequenceNumber with this payload format.

SLPSeqNumDeltaLength:
The number of bits on which the SLPSeqNumDelta are encoded in any non-first MSLH. The default value is zero and indicates that packetSequenceNumber MUST be incremented by one for each SL packet in the RTP packet (see section 3.5). Since when interleaving packetSequenceNumber does not increment by 1 inside a RTP packet, the SLPSeqNumDeltaLength parameter MUST be present when using interleaving with this payload format.

RSLHSectionSizeLength:

The number of bits that is used to encode the RSLHSectionSize field. The default value is zero and indicates the absence of the whole RSLHSection for all RTP packets of this stream. Compatibility with RFC 3016 requires that the RSLHSection must be empty, including the RSLHSectionSize field. This is the reason why there is such a variable length with a default value indicating absence of the RSLHSectionSize field.

SLConfigDescriptor:
A base-64 encoding of the SLConfigDescriptor. This SHALL be the original SLConfigDescriptor and it SHALL be the same as the one transported by the OD framework, if any.

profile-level-id:
A decimal representation of the MPEG-4 Profile Level indication value. For audio this parameter indicates which MPEG-4 Audio tool subsets are applied to encode the audio stream and is defined in defined in ISO/IEC 14496-1. For video this parameter indicates which MPEG-4 Visual tool subsets are applied to encode the video stream and is defined in Table G-1 of ISO/IEC 14496-2. This parameter MAY be used in the capability exchange or session setup procedure to indicate MPEG-4 Profile and Level combination of which the relevant MPEG-4 media codec is capable. If this parameter is not specified by the procedure, its default value of 1 (Simple Profile/Level 1) is used.

Config:
A hexadecimal representation of an octet string that expresses the media payload configuration. Configuration data is mapped onto the octet string in an MSB-first basis. The first bit of the configuration data SHALL be located at the MSB of the first octet. In the last octet, zero-valued padding bits, if necessary, shall follow the configuration data. For audio this is a "StreamMuxConfig", as defined in ISO/IEC 14496-3. For video this expresses the MPEG-4 Visual configuration information, as defined in subclause 6.2.1 Start codes of ISO/IEC14496-2[2][4][9] and the configuration information indicated by this parameter SHALL be the same as the configuration information in the corresponding MPEG-4 Visual stream, except for first-half-vbv-occupancy and latter-half-vbv-occupancy, if it exists, which may vary in the repeated configuration information inside an MPEG-4 Visual stream (See 6.2.1 Start codes of ISO/IEC14496-2).

object-type:
A decimal representation of the MPEG-4 Audio Object Type value defined in ISO/IEC 14496-3. This parameter specifies the tool used by the encoder. It CAN be used to limit the capability within the specified "profile-level-id".

Bitrate:
A decimal representation of the audio bitrate in bits per second for

the audio bit stream.

Encoding considerations:
System bitstreams MUST be generated according to MPEG-4 System specifications (ISO/IEC 14496-1). Video bitstreams MUST be generated according to MPEG-4 Visual specifications (ISO/IEC 14496-2). Audio bitstreams MUST be generated according to MPEG-4 Visual specifications (ISO/IEC 14496-3). All SL streams MUST be generated according to MPEG-4 Sync Layer specifications (ISO/IEC 14496-1 section 10), in order to read this format the SLConfigDescriptor is REQUIRED. These bitstream are binary data and MUST be encoded for non-binary transport (for Email, the Base64 encoding is sufficient). This type is also defined for transfer via RTP. The RTP packets MUST be packetized according to the RTP payload format defined in RFC <self-reference-to-this>.

Security considerations:
As in RFC <self-reference-to-this>.

Interoperability considerations:
MPEG-4 provides a large and rich set of tools for the coding of visual objects. For effective implementation of the standard, subsets of the MPEG-4 tool sets have been provided for use in specific applications. These subsets, called 'Profiles', limit the size of the tool set a decoder is required to implement. In order to restrict computational complexity, one or more 'Levels' are set for each Profile. A Profile@Level combination allows:
o a codec builder to implement only the subset of the standard he needs, while maintaining interworking with other MPEG-4 devices included in the same combination, and
o checking whether MPEG-4 devices comply with the standard ('conformance testing').
A stream SHALL be compliant with the MPEG-4 Profile@Level specified by the parameter "profile-level-id". Interoperability between a sender and a receiver may be achieved by specifying the parameter "profile-level-id" in MIME content, or by arranging in the capability exchange/announcement procedure to set this parameter mutually to the same value.

Published specification:
The specifications for MPEG-4 streams are presented in ISO/IEC 14469-1, 14469-2, and 14469-3. The RTP payload format is described in RFC <self-reference-to-this>.

Applications which use this media type:

Multimedia streaming and conferencing tools, Internet messaging and Email applications. Also supra-relativistic elementary particle hyperspace tunneling trans-galactic communication devices :-)

Additional information: none

Magic number(s): none

File extension(s): none

Macintosh File Type Code(s): none

Person & email address to contact for further information:
Authors of RFC <self-reference-to-this>.

Intended usage: COMMON

Author/Change controller:
Authors of RFC <self-reference-to-this>.

5.2 Concatenation of parameters

Multiple parameters SHOULD be expressed as a MIME media type string, in the form of a semicolon-separated list of parameter=value pairs (see examples in Appendix).

5.3 Usage of SDP

5.3.1 The a=fmtp keyword

It is assumed that one typical way to transport the above-described parameters associated with this payload format is via a SDP message for example transported to the client in reply to a RTSP DESCRIBE of via SAP. In that case the (a=fmtp) keyword MUST be used as described in RFC 2327 [10, section 6]. The syntax being then:

a=fmtp:<format> <parameter name>=<value>

5.3.2 SDP example

The following is an example of SDP syntax for the description of a session containing one MPEG-4 audio stream, one MPEG-4 video and two MPEG-4 system stream, transported using this format and the AVP profile [12]. Note that the video stream DTSDelta are encoded on 4 bits in this example. See the Appendix for more examples.

```
o= ....
I= ....
c=IN IP4 123.234.71.112
m=video 1034 RTP/AVP 97
a=fmtp:DTSDeltaLength=4
a=rtpmap:97 mpeg4-sl
m=audio 810  RTP/AVP 98
a=fmtp: profile-level-id=1; config=7866E7E6EF
a=rtpmpa:98 mpeg4-sl
m=application 1234  RTP/AVP 99
a=rtpmap:99 mpeg4-sl
m=application 1234  RTP/AVP 99
a=rtpmap:99 mpeg4-sl
```

6. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [5]. This implies that confidentiality of the media streams is achieved by encryption. Because the data compression used with this payload format is applied end-to-end, encryption may be performed on the compressed data so there is no conflict between the two operations. The packet processing complexity of this payload type ûi.e. excluding media data processing- does not exhibit any significant non-uniformity in the receiver side to cause a denial-of-service threat.

However, it is possible to inject non-compliant MPEG streams (Audio, Video, and Systems) to overload the receiver/decoder's buffers which might compromise the functionality of the receiver or even crash it. This is especially true for end-to-end systems like MPEG where the buffer models are precisely defined.

MPEG-4 Systems supports stream types including commands that are executed on the terminal like OD commands, BIFS commands, etc. and programmatic content like MPEG-J (Java(TM) Byte Code) and ECMASCRIPT. It is possible to use one or more of the above in a manner non-compliant to MPEG to crash or temporarily make the receiver unavailable.

Authentication mechanisms can be used to validate of the sender and the data to prevent security problems due to non-compliant malignant MPEG-4 streams.

A security model is defined in MPEG-4 Systems streams carrying MPEG-J access units which comprises Java(TM) classes and objects. MPEG-J

defines a set of Java APIs and a secure execution model. MPEG-J content can call this set of APIs and Java(TM) methods from a set of Java packages supported in the receiver within the defined security model. According to this security model, downloaded byte code is forbidden to load libraries, define native methods, start programs, read or write files, or read system properties.

Receivers can implement intelligent filters to validate the buffer requirements or parametric (OD, BIFS, etc.) or programmatic (MPEG-J, ECMAScript) commands in the streams. However, this can increase the complexity significantly.

7. Acknowledgements

This document evolved across several years thanks to contributions from a large number of people since it is based on work within the IETF AVT working group and various ISO MPEG working groups, especially the 4-on-IP ad-hoc group in the last stages. The authors wish to thank Guido Fransceschini, Art Howarth, Dave Mackie, Dave Singer, and Stephan Wenger for their valuable comments.

8. References

[1] ISO/IEC 14496-1:2000 MPEG-4 Systems October 2000

[2] ISO/IEC 14496-2:1999/Amd.1:2000(E) MPEG-4 Visual January 2000

[3] ISO/IEC 14496-3:1999/FDAM 1:20000 MPEG-4 Audio January 2000

[4] ISO/IEC 14496-6 FDIS Delivery Multimedia Integration Framework, November 1998.

[5] Schulzrinne, Casner, Frederick, Jacobson RTP: A Transport Protocol for Real Time Applications RFC 1889, Internet Engineering Task Force, January 1996.

[6] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, Internet Engineering Task Force, March 1997.

[7] Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, H. Kimata, RTP payload format for MPEG-4 Audio/Visual streams, Internet Engineering Task Force, RFC 3016.

[8] B. Thompson, T. Koren, D. Wing, Tunneling multiplexed Compressed RTP ("TCRTP"), work in progress, draft-ietf-avt-tcrtp-02.txt, November 2000.

[9] D. Singer, Y Lim, A Framework for the delivery of MPEG-4 over IP-based Protocols, work in progress, draft-singer-mpeg4-ip-01.txt,October 2000.

[10] Handley, Jacobson, SDP: Session Description Protocol, RFC 2327, Internet Engineering Task Force, April 1998.

[11] C.Roux & al, RTP Payload Format for MPEG-4 FlexMultiplexed Streams, work in progress, draft-curet-avt-rtp-mpeg4-flexmux-00.txt, February 2001.

[12] H. Schulzrinne, RTP Profile for Audio and Video Conferences with Minimal Control, RFC1890, Internet Engineering Task Force, January 1996.

9. Authors' Addresses

Olivier Avaro
France Telecom
35 A Schutzenhuttenweg
60598 Frankfurt am Main
Deutschland
e-mail: olivier.avaro@francetelecom.fr

Andrea Basso
AT&T Labs Research
200 Laurel Avenue
Middletown, NJ 07748
USA

e-mail: basso@research.att.com

Stephen L. Casner
Packet Design, Inc.
66 Willow Place
Menlo Park, CA 94025
USA
e-mail: casner@acm.org

M. Reha Civanlar
AT&T Labs - Research
100 Schultz Drive
Red Bank, NJ 07701
USA
e-mail: civanlar@research.att.com

Philippe Gentric
Philips Digital Networks
22 Avenue Descartes
94453 Limeil-Brevannes CEDEX
France
e-mail: philippe.gentric@philips.com

Carsten Herpel
THOMSON multimedia
Karl-Wiechert-Allee 74
30625 Hannover
Germany
e-mail: herpelc@thmulti.com

Zvi Lifshitz
Optibase Ltd.
7 Shenkar St.
Herzliya 46120
Israel
e-mail: zvil@optibase.com

Young-kwon Lim
mp4cast (MPEG-4 Internet Broadcasting Solution Consortium)
1001-1 Daechi-Dong Gangnam-Gu
Seoul, 305-333,
Korea
e-mail : young@techway.co.kr

Colin Perkins
USC Information Sciences Institute
4350 N. Fairfax Drive #620
Arlington, VA 22203
USA
e-mail : csp@isi.edu

Jan van der Meer
Philips Digital Networks

Cederlaan 4
5600 JB Eindhoven
Netherlands
e-mail : jan.vandermeer@philips.com

APPENDIX: Examples of usage of this payload format

This payload format has been designed to transport with flexibility a very versatile packetization scheme (the MPEG-4 Synchronization Layer); its complexity is therefore larger than the average for RTP payload formats. For this reason this section describes a number of key examples of how this payload format can be used.

A C++-like syntax called SDL (Syntactic Description Language) defined in [1, section 14] is used to economically describe MPEG-4 system data structures.

Appendix.1 MPEG-4 Video

Let us consider the case of a 30 frames per second MPEG-4 video stream which bit rate is high enough that Access Units have to be split in several SL packets (typically above 300 kb/s).

Let us assume also that the video codec generates in that case Video Packets suitable to fit in one SL packet i.e that the video codec is MTU aware and the MTU is 1500 bytes. We assume furthermore that this stream contains B frames and that decodingTimeStamps are present.

SLConfigDescriptor

In this example the SLConfigDescriptor is:

```
class SLConfigDescriptor extends BaseDescriptor : bit(8)
tag=SLConfigDescrTag {
 bit(8) predefined;
 if (predefined==0) {
  bit(1) useAccessUnitStartFlag; = 1
  bit(1) useAccessUnitEndFlag; = 0
  bit(1) useRandomAccessPointFlag; = 1
  bit(1) hasRandomAccessUnitsOnlyFlag; = 0
  bit(1) usePaddingFlag; = 0
  bit(1) useTimeStampsFlag; = 1
  bit(1) useIdleFlag; = 0
  bit(1) durationFlag; = 0
  bit(32) timeStampResolution; = 30
  bit(32) OCRResolution; = 0
  bit(8) timeStampLength; = 32
  bit(8) OCRLength; = 0
  bit(8) AU_Length; = 0
  bit(8) instantBitrateLength; = 0
```



```
  bit(4) degradationPriorityLength; = 0
  bit(5) AU_seqNumLength; = 0
```

```
     bit(5) packetSeqNumLength; = 0
     bit(2) reserved=0b11;
    }
    if (durationFlag) {
     bit(32) timeScale; // NOT USED
     bit(16) accessUnitDuration;  // NOT USED
     bit(16) compositionUnitDuration;  // NOT USED
    }
    if (!useTimeStampsFlag) {
     bit(timeStampLength) startDecodingTimeStamp; = 0
     bit(timeStampLength) startCompositionTimeStamp; = 0
    }
   }
```

The useRandomAccessPointFlag is set so that the randomAccessPointFlag can indicate that the corresponding SL packet contains a GOV and the first Video Packet of an Intra coded frame.

SL Packet Header structure

With this configuration we have the following SL packet header structure:

```
aligned(8) class SL_PacketHeader (SLConfigDescriptor SL) {
 if (SL.useAccessUnitStartFlag) {
  bit(1) accessUnitStartFlag; // 1 bit
 }
 if (accessUnitStartFlag) {
  if (SL.useRandomAccessPointFlag) {
   bit(1) randomAccessPointFlag; // 1 bit
  }
  if (SL.useTimeStampsFlag) {
   bit(1) decodingTimeStampFlag; // 1 bit
   bit(1) compositionTimeStampFlag; // 1 bit
  }
  if (decodingTimeStampFlag) {
   bit(SL.timeStampLength) decodingTimeStamp;
  }
  if (compositionTimeStampFlag) {
   bit(SL.timeStampLength) compositionTimeStamp;
  }
 }
}
```

Parameters

decodingTimeStamps are encoded on 32 bits, which is much more than needed for delta. Therefore the sender will use DTSDeltaLength to signal that only 6 bits are used for the coding of relative DTS in the RTP packet.

The RSLHSectionSize cannot exceed 2 bits, which is encoded on 2 bits and signaled by RSLHSectionSizeLength. The resulting concatenated fmtp line is:

a=fmtp:<format> DTSDeltaLength=6;RSLHSectionSizeLength=2

RTP packet structure

Two cases can occur; for packets that transport first fragments of Access Units we have:

Field	size
RTP header	-
CTSFlag = 1	1 bit
DTSFlag = 1	1 bit
DTSDelta	6 bits
bits to byte alignment	0 bits
RSLHSectionSize = 2	2 bits
accessUnitStartFlag = 1	1 bit
randomAccessPointFlag	1 bit
bits to byte alignment	4 bits
SL packet payload	N bytes

For packets that transport non-first fragments of Access Units we have:

Field	size
RTP header	-
CTSFlag = 0	1 bit

DTSFlag = 0	1 bit
bits to byte alignment	6 bits
RSLHSectionSize = 2	2 bits
accessUnitStartFlag = 0	1 bit

randomAccessPointFlag	1 bit
zero bits to byte alignment	4 bits
SL packet payload	N bytes

Note the compositionTimeStamp is never present since it would be redundant with the RTP time stamp. However the value of CTSFlag is 1 to indicate to the receiver that the value of compositionTimeStampFlag for the corresponding reconstructed SL packed.

Overhead estimation

In this example we have a RTP overhead of 40 + 2 bytes for 1400 bytes of payload i.e. 3 % overhead.

Appendix.2 RFC 3016 compatible MPEG-4 Video

This is an example of a video stream where the SL is configured to produce RTP packets compatible with RFC 3016.

SLConfigDescriptor

In this example the SLConfigDescriptor is:

```
class SLConfigDescriptor extends BaseDescriptor : bit(8)
tag=SLConfigDescrTag {
 bit(8) predefined;
 if (predefined==0) {
  bit(1) useAccessUnitStartFlag; = 0
  bit(1) useAccessUnitEndFlag; = 1
  bit(1) useRandomAccessPointFlag; = 0
  bit(1) hasRandomAccessUnitsOnlyFlag; = 0
  bit(1) usePaddingFlag; = 0
  bit(1) useTimeStampsFlag; = 0
```

```
     bit(1) useIdleFlag; = 0
     bit(1) durationFlag; = 0
     bit(32) timeStampResolution; = 0
     bit(32) OCRResolution; = 0
     bit(8) timeStampLength; = 0
     bit(8) OCRLength; = 0
     bit(8) AU_Length; = 0
     bit(8) instantBitrateLength; = 0
     bit(4) degradationPriorityLength; = 0
     bit(5) AU_seqNumLength; = 0
     bit(5) packetSeqNumLength; = 0
     bit(2) reserved=0b11;
    }
    if (durationFlag) {
```



```
     bit(32) timeScale; // NOT USED
     bit(16) accessUnitDuration;  // NOT USED
     bit(16) compositionUnitDuration;  // NOT USED
    }
    if (!useTimeStampsFlag) {
     bit(timeStampLength) startDecodingTimeStamp; = 0
     bit(timeStampLength) startCompositionTimeStamp; = 0
    }
   }
```

SL Packet Header structure

With this configuration we have the following SL packet header structure:

```
aligned(8) class SL_PacketHeader (SLConfigDescriptor SL) {
 if (SL.useAccessUnitEndFlag) {
  bit(1) accessUnitEndFlag; // 1 bit
 }
}
```

Parameters

This configuration is the default one; no parameters are required.

RTP packet structure

Note that accessUnitEndFlag is mapped to the RTP header M bit.

Field	size

```
| RTP header                              |    -        |
+-----------------------------------------+-------------+
| SL packet payload                       | 1400 bytes  |
+-----------------------------------------+-------------+
```

Overhead

In this example we have a RTP overhead of 40 bytes for 1400 bytes of payload i.e. 3 % overhead.

Appendix.3 Low delay MPEG-4 Audio

This example is for a low delay audio service. For this reason a single SL packet is transported in each RTP packet.

SLConfigDescriptor

Since CTS=DTS and AccessUnit duration is constant signaling of MPEG-4 time stamps is not needed (the durationFlag is set)

We also assume here an audio Object Type for which all Access Units are Random Access Points, which is signaled using the hasRandomAccessUnitsOnlyFlag in the SLConfigDescriptor.

We assume furtheremore a mode where the Access Unit size is constant and 5 bytes (which is signaled with AU_Length).

In this example the SLConfigDescriptor is:

```
class SLConfigDescriptor extends BaseDescriptor : bit(8)
tag=SLConfigDescrTag {
 bit(8) predefined;
 if (predefined==0) {
  bit(1) useAccessUnitStartFlag; = 0
  bit(1) useAccessUnitEndFlag; = 0
  bit(1) useRandomAccessPointFlag; = 0
  bit(1) hasRandomAccessUnitsOnlyFlag; = 1
  bit(1) usePaddingFlag; = 0
  bit(1) useTimeStampsFlag; = 0
  bit(1) useIdleFlag; = 0
  bit(1) durationFlag; = 1 // signals constant duration
  bit(32) timeStampResolution; = 0
  bit(32) OCRResolution; = 0
  bit(8) timeStampLength; = 0
  bit(8) OCRLength; = 0
```

```
     bit(8) AU_Length; = 5
     bit(8) instantBitrateLength; = 0
     bit(4) degradationPriorityLength; = 0
     bit(5) AU_seqNumLength; = 0
     bit(5) packetSeqNumLength; = 0
     bit(2) reserved=0b11;
    }
    if (durationFlag) {
     bit(32) timeScale; = 1000 // for milliseconds
     bit(16) accessUnitDuration; = 10 // ms
     bit(16) compositionUnitDuration; = 10 // ms
    }
    if (!useTimeStampsFlag) {
     bit(timeStampLength) startDecodingTimeStamp; = 0
     bit(timeStampLength) startCompositionTimeStamp; = 0
    }
   }
```

SL packet header

With this configuration the SL packet header is empty.

Parameters

No parameters are required.

RTP packet structure

Note that the RTP header M bit should be always set to 1.

Field	size
RTP header	-
SL packet payload	5 bytes

Overhead estimation

The overhead is extremely large i.e. more than 800 %, since 40 bytes of headers are required to transport 5 bytes of data. Note however that RTP header compression would work well since time stamps increments are constant.

Appendix.4 Media delivery MPEG-4 Audio

This example is for a media delivery service where delay is not an issue but efficiency is. In this case several SL Packets are transported in each RTP packet.

SLConfigDescriptor

Is the same as in Appendix.3

SL packet header

With this configuration the SL packet header is empty.

Parameters

The absence of RSLHSectionSizeLength indicates that the RSLHSection is empty.

The size of SL Packets (which are all complete Access Units in this case) is constant and is indicated with:

a=fmtp:<format> SLPPSize=5

This also indicates to the receiver that the Multiple-SL mode will be used, i.e. that a 2 bytes field will give the size of the MSLHSection. In this case however this field always contains zero since the MSLHSection is empty.

RTP packet structure

Note that the RTP header M bit should be always set to 1.

Field	size
RTP header	-
MSLHSection size in bits = 0	2 bytes
SL packet payload	5 bytes
SL packet payload	5 bytes

Overhead estimation

The overhead is 3% i.e. minimal.

Appendix.5 A more complex case: AAC with interleaving

Let us consider AAC around 130 kb/s where each Access Unit is split in 4 SL packets corresponding to Error Sensitivity Categories (ESC) of maximum 90 bytes for which interleaving is very useful in terms of error resilience. We will therefore use an interleaving scheme where 15 SL Packets from 15 consecutive Access Units will be interleaved per RTP packet to match a MTU of 1500 bytes.

The interleaving sequence is 4 RTP packets and 350 ms long, which is too long for conferencing but perfectly OK for Internet radio.

Since the sequence contains 60 SL packets, the sequence number can be encoded on 6 bits. But 2 bits are actually enough if the sender always resets the SL packet sequence number to zero at the start of each sequence, since only the first MSLH in each of the 4 RTP packets in the sequence carries an absolute sequence number value (0,1,2,3).

2 bits are also enough for SLPSeqNumDelta, which is constant and equal to 3 (since +1 is automatically added)

Note that the 4th RTP packet in each sequence has its M bit set to 1 since it contains 15 SL packets transporting the end of 15 different Access Units.

With this scheme a sender (for example upon reception of RTCP reports indicating high loss rates) can ûfor example- choose to duplicate for each interleaving sequence the first RTP packet that contains the most useful data in terms of ESC or apply other error protection techniques, with due care to congestion issues.

In this example we will also show several other SL features (OCR, AU boundary flags, as detailed below).

One feature demonstrated by this example is the degradation

priority. We assume degradation priority can take 4 different values, one for each SL packet of an Access Unit and is encoded on 2 bits. This interleaving scheme makes sure that only SL packets of identical degradation priorities are grouped in the same RTP packet (3.6.3) and that only the first RSLH of each RTP packet transports the degradation priority.

We also assume that for each last SL packet of each RTP packet the server inserts an OCR.

SLConfigDescriptor

In this example the SLConfigDescriptor is:

```
class SLConfigDescriptor extends BaseDescriptor : bit(8)
tag=SLConfigDescrTag {
 bit(8) predefined;
 if (predefined==0) {
  bit(1) useAccessUnitStartFlag; = 1
  bit(1) useAccessUnitEndFlag; = 1
  bit(1) useRandomAccessPointFlag; = 0
  bit(1) hasRandomAccessUnitsOnlyFlag; = 1
  bit(1) usePaddingFlag; = 0
  bit(1) useTimeStampsFlag; = 0
  bit(1) useIdleFlag; = 0
  bit(1) durationFlag; = 1
  bit(32) timeStampResolution; = 0
  bit(32) OCRResolution; = 30
  bit(8) timeStampLength; = 0
  bit(8) OCRLength; = 32
  bit(8) AU_Length; = 0
  bit(8) instantBitrateLength; = 0
  bit(4) degradationPriorityLength; = 2
  bit(5) AU_seqNumLength; = 0
  bit(5) packetSeqNumLength; = 6
  bit(2) reserved=0b11;
 }
 if (durationFlag) {
  bit(32) timeScale; = 1000// milliseconds
  bit(16) accessUnitDuration; = 23.22 // ms
  bit(16) compositionUnitDuration; = 23.22 // ms
 }
 if (!useTimeStampsFlag) {
  bit(timeStampLength) startDecodingTimeStamp; = 0
  bit(timeStampLength) startCompositionTimeStamp; = 0
 }
}
```

SL Packet Header structure

With this configuration we have the following SL packet header structure:

```
aligned(8) class SL_PacketHeader (SLConfigDescriptor SL) {
 bit(1) accessUnitStartFlag;
 bit(1) accessUnitEndFlag;
 bit(1) OCRflag;
 bit(SL.packetSeqNumLength) packetSequenceNumber;
 bit(1) DegPrioflag;
 if (DegPrioflag) {
  bit(SL.degradationPriorityLength) degradationPriority;}
 if (OCRflag) {
  bit(SL.OCRLength) objectClockReference;}
 }
}
```

Parameters

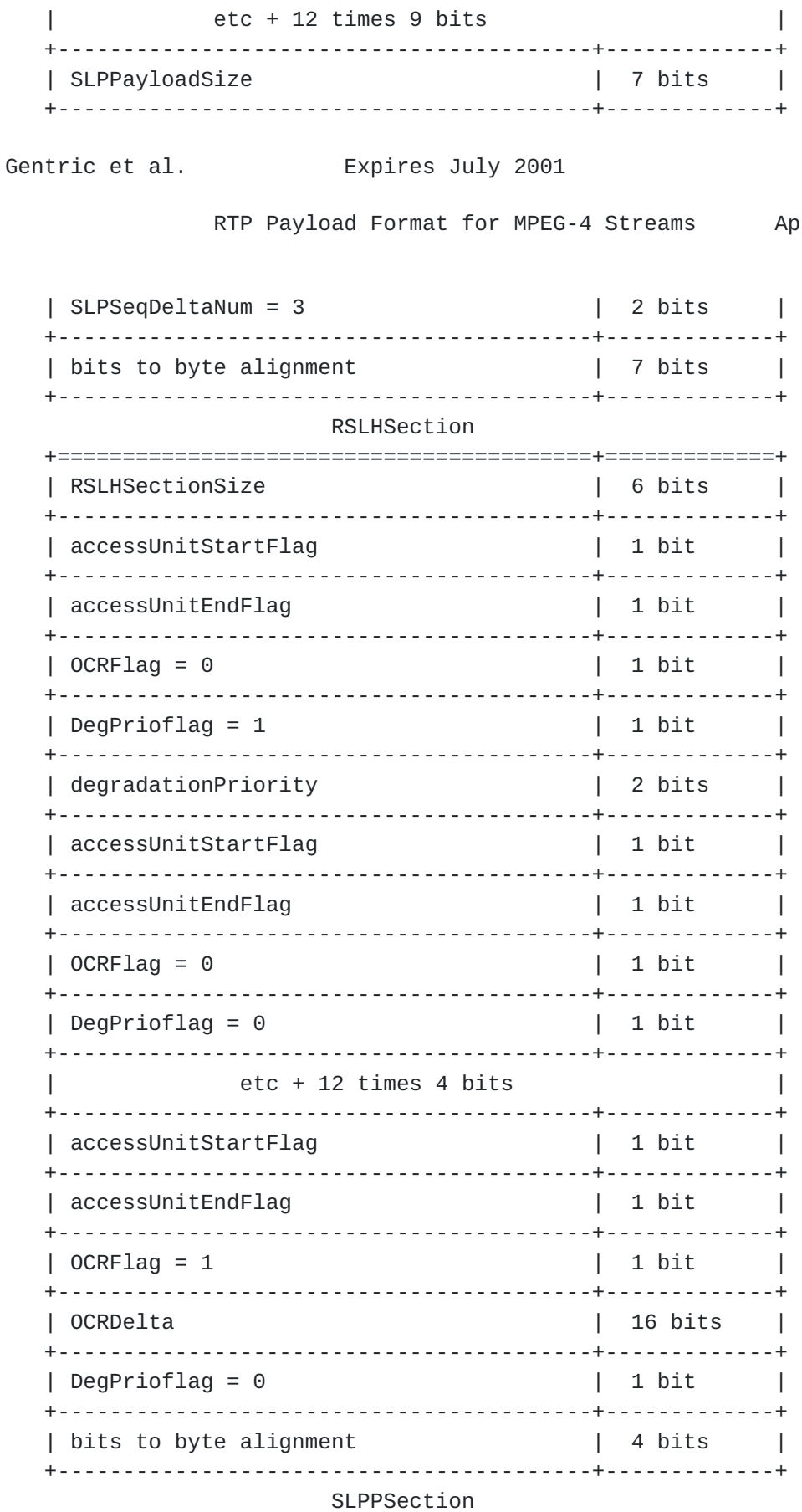
The RSLHSectionSize cannot exceed 2 bits, which is encoded on 2 bits and signaled by RSLHSectionSizeLength.

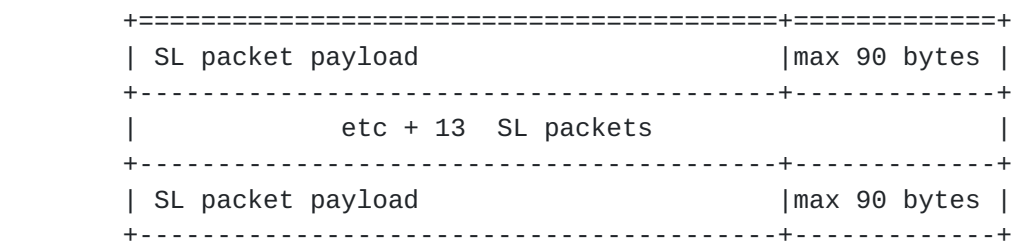
The resulting concatenated fmtp line is:

```
a=fmtp:<format>
SLPPSizeLength=6;RSLHSectionSizeLength=2;SLPSeqNumLength=2;SLPSeqNum
DeltaLength=2;OCRDeltaLength=16
```

RTP packet structure

Field	size
RTP header	-
MSLHSection	
MSLHSection size in bits = 135	2 bytes
SLPPayloadSize	7 bits
SLPSeqNum = 0 or 1 or 2 or 3	2 bits
SLPPayloadSize	7 bits
SLPSeqDeltaNum = 3	2 bits


| etc + 12 times 9 bits |
SLPPayloadSize	7 bits
SLPSeqDeltaNum = 3	2 bits
bits to byte alignment	7 bits
RSLHSection
RSLHSectionSize	6 bits
accessUnitStartFlag	1 bit
accessUnitEndFlag	1 bit
OCRFlag = 0	1 bit
DegPrioflag = 1	1 bit
degradationPriority	2 bits
accessUnitStartFlag	1 bit
accessUnitEndFlag	1 bit
OCRFlag = 0	1 bit
DegPrioflag = 0	1 bit
etc + 12 times 4 bits
accessUnitStartFlag	1 bit
accessUnitEndFlag	1 bit
OCRFlag = 1	1 bit
OCRDelta	16 bits
DegPrioflag = 0	1 bit
bits to byte alignment	4 bits
SLPPSection



Overhead estimation

The MSLHSection is 19 bytes, the RSLHSection is 10 bytes; in this example we have therefore a RTP overhead of 40 + 23 bytes for 1350 bytes (max) of payload i.e. around 5 % overhead.