SMPTE STANDARD

SMPTE 381-1-2005
Previously numbered SMPTE 381M-2005

for Television — Material Exchange Format (MXF) — Mapping MPEG Streams into the MXF Generic Container

Table of contents

1 Scope

This standard specifies the mapping of MPEG streams, as identified by an ISO 13818-1 stream_id value, into the MXF generic container. This includes, but is not limited to, ISO 13818-2 MPEG video stream, ISO 11172-2 MPEG video streams, ISO 13818-3 MPEG audio streams, and ISO 11172-3 MPEG audio streams. Provision is made for mapping the content with or without the ISO 13818-1 PES layer. Provision is also made for single stream mapping and synchronized multi-stream mapping. The document outlines synchronization requirements for the MXF encapsulation of co-timed MPEG essence streams. This standard defines SMPTE universal labels to be used to uniquely identify specific MPEG implementations.

This standard also provides for clip wrapping a single program MPEG program stream or transport stream multiplex. Support for multi-program MPEG multiplexes is not provided in this standard. Support for transport streams where some of the transport stream packets have been removed is not provided in this standard. This standard does not map MPEG PSI or DVB-SI or ATSC-PSIP to any metadata scheme.

In order to achieve interoperability within any given operational pattern, restrictions may be placed on the way in which this essence container can be implemented. The reader is advised to carefully study the appropriate operational pattern document before implementation.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision and

Approved
February 15, 2005

parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

SMPTE 336M-2001, Television — Data Encoding Protocol using Key-Length-Value

SMPTE 377M-2004, Television — Material Exchange Format (MXF) — File Format Specification

SMPTE 379M-2004, Television — Material Exchange Format (MXF) — Generic Container

SMPTE 385M-2004, Television — Material Exchange Format (MXF) — Mapping SDTI-CP Essence and Metadata into the MXF Generic Container

SMPTE RP 210, Metadata Dictionary Registry of Metadata Element Descriptions

SMPTE RP 224, SMPTE Labels Registry

ISO 13818 – Parts 1, 2, 3, 7, Information Technology — Generic Coding of Moving Pictures and Associated Audio Information: Systems, Video, Audio, Advanced Audio Coding (AAC)

3 Glossary of acronyms, terms and data types

The full glossary of terms and acronyms used in the MXF specification is given in the MXF file format specification. Additional terms are given in the glossaries in SMPTE 379M and ISO 13818 (the MPEG2 specification). Terms are not repeated here to avoid any divergence of meaning.

AFD: Active format descriptor, defined by ETSI ETR154 – a parameter which defines the aspect ratio of the active content within the MPEG coded frame; e.g., 4:3 content within a coded 16:9 rectangle where black bars left and right are part of the MPEG coded signal.

BWF: Broadcast wave audio format (see SMPTE 382M in annex E)

ES: Elementary stream

GOP: Group of pictures

SPTS: Single program transport stream

PES: Packetized elementary stream

PMT: Program map table

PSM: Program stream map

VCD: Video CD

VOB: Video object (DVD term)

4 Introduction

In this introduction, it is assumed that a general understanding of MPEG video compression and the meaning of acronyms such as GOP, B-frame, and PES are understood. These acronyms are well defined in the normative references and will not be repeated here.

4.1 Basics of wrapping: Frames and access units (informative)

The MPEG video specifications enable the compression of picture-based video. These pictures may be frames or fields. The MPEG specification provides signaling for the number of pixels in the picture, the field / frame rate and other parameters including the aspect ratio of the pixels. Many of these parameters will be

copied into the MXF essence descriptor. In order to ensure interoperability, it is important to ensure that MPEG header information can be found when decoding a KLV wrapped MPEG file. For this reason, the text in section 5 refers to access units (AUs) rather than pictures. The full definition of an access unit (AU) is given in ISO 13818-1 and is summarized below for MPEG video and MPEG audio:

Audio: 1 AU = The coded representation of an audio frame

Video: 1 AU = The coded data for a picture and any stuffing that follows it, up to but not including the start of the next access unit. If a picture is *not* preceded by a group_start_code or a sequence_header_code, the access unit begins with a picture_start_code. If a picture is preceded by a group_start_code or a sequence_header_code, the access unit begins with the first byte of these start codes. If it is the last picture preceding a sequence_end_code in the bit stream all bytes between the last bytes of the coded picture and the sequence_end_code (including the sequence_end_code) belong to the access unit.

4.2 Timeline of the MXF tracks representing MPEG content (informative)

Many MXF files containing MPEG content will contain a picture track for the MPEG video data and sound tracks for any audio data. Many professional applications will use uncompressed audio which will have a simple timing relationship with the MPEG video. When an MPEG transmission or distribution feed is captured, the MXF representation may be more complicated. There is no intention within this mapping document to be able to exactly recreate a program stream or transport stream from its demultiplexed MXF components.

4.2.1 Creating an MXF file from an MPEG transport stream or an MPEG program stream

Consider an MPEG transport stream which contains compressed audio tracks and compressed video with other PES streams such as subtitles. It is likely that these PES streams may be bursty in nature; i.e., there may be many minutes of picture and sound with no other PES content present in the stream. These same issues also apply to program streams.

Capture option 1: Capture the entire transport stream or program stream and wrap in a single KLV packet (as outlined in section 6.3.5). This may be optimal when the stream is likely to be processed or used as a transport stream or program stream, respectively.

Capture option 2: Capture a partial transport stream (i.e., a stream where elements within the selected duration of the original stream have been removed) or a partial stream program stream (i.e., where elements within the selected duration of the original stream have been removed). These scenarios are outside the scope of this standard.

Capture option 3: Demultiplex the components of the captured transport stream or program stream into its elementary streams or PES streams and maintain the relative timing information by creating a valid MXF header on the fly. There are a few issues here:

1. In general the desired start frame is not the first presentation frame in a closed GOP sequence. The first "I" frame or sufficient frames before the desired frame (in the case where there are no I frames) must be captured to pre-charge the decoder.

2. The compressed audio AU duration is rarely the same as the video AU duration. To capture without recoding the audio requires calculating the temporal audio-video offset by inspection of PTS values and using these to create the MXF tracks with appropriate origin values.

3. PES data elements may have AU durations unrelated to the audio or video AUs. For example data such as ETSI EN300374 subtitle bitmaps may be delivered in the stream well before they are required to be displayed. A capture device may not know that the data is subtitles, only that the PTS is some time in the distant future. The MXF track which defines the subtitle shall be able to offset the origin of the track by a sufficiently large value to correct the timing.

4. Any PES data elements must be preserved with their PES headers which may contain extra information which is unknown to the capture device.

5. When demultiplexing takes place, it is assumed that preservation of the MPEG timing is sufficient and that preservation of the data in a form which allows streaming with the small MPEG buffers is not required. There is no buffer model for MXF streaming and, as a result, the streaming of the data in MXF form may have a higher latency than the streaming as a transport stream or program stream. It also means that re-creation of the transport stream or program stream will require an MPEG multiplexer with knowledge of the buffer models of all the tracks (possibly even the private PES ones) in the stream. The exact relative timing relationship between the ES and PES components of the program stream or transport stream depends on the original time base of the video stream, the sampling rate of the audio, the compression scheme used by the audio, and the PES strategy employed by the original multiplexer. These factors allow the relative timing relationships between the MXF tracks to be defined.

4.3 **MXF tracks** (normative)

In order to be able to represent the captured MPEG data in an MXF file, it must be possible to describe the start point of the track data, as well as the "in" and "out" points of the desired picture, sound, and PES data. This is shown in figure 1.

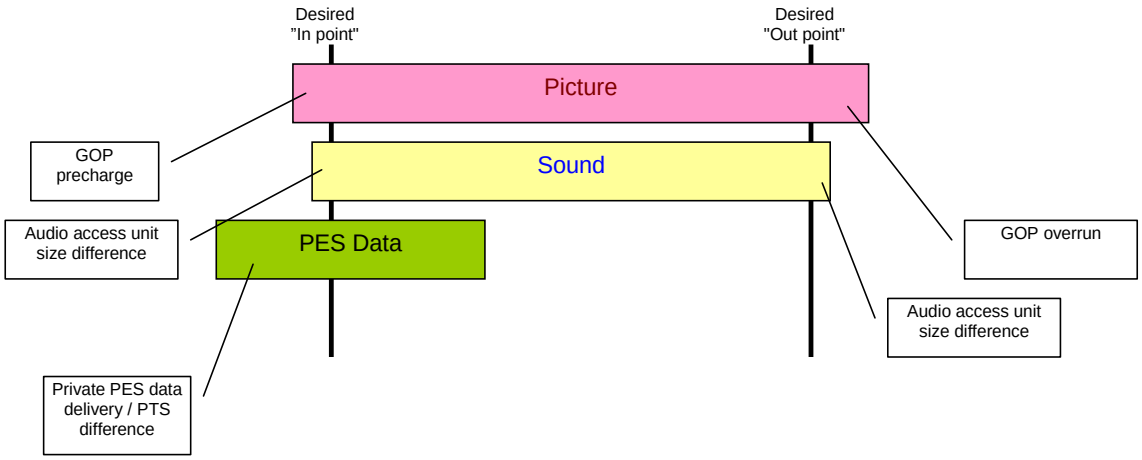

Figure 1 – MXF track timing for MPEG generic containers

The example shown in figure 1 shows the relationship between the desired start and end points of a stored generalized MPEG segment and the actual data start and end points which would be found in the stored essence. The track and sequence items within the header must describe accurately the synchronization scenario above.

The picture track shows that the desired start frame does not coincide with the first frame of an MPEG random access point. There is picture data between the first byte of stored data and the first byte of the desired picture information. In the MXF picture track, there shall be a positive **origin** value which shall correspond to the difference in **position** of the first picture stored in the generic container and the **position** of the desired start picture of the essence.

Caution is advised, however, when higher operational patterns are used with the MPEG generic container mapping. In an OP1a file, the material package shall have the same start position and duration as the top-level file package. Unwanted frames which were captured to allow decoding of the desired frames are, therefore, hidden using the mechanism described above. This is still a valid OP1a file.

In an operational pattern 3x file, editing is allowed; i.e., the material package may reference any portion of a top-level file package track. In this case, the desired start point of the file package track may be the very first image in the container, even when it is not decodable. The desired start position of the material package track may then be offset relative to the start of the top-level file package track so that a decodable portion of the essence is referenced.

If an MXF encoder captures an MPEG long GOP stream and is *unable* to determine the correct value of **origin**, then a decoder shall assume that the desired start point of the track is the first frame of the track. The first frame of the track may not be decodable and/or displayable. An MXF decoder should exhibit some consistent behavior under these circumstances; e.g., by displaying black frames until a decodable frame is reached.

The audio track shows that the decoded audio sample which should start the captured clip does not coincide with the first sample of the MPEG coded audio frame in which it lies. Compressed audio frames shall be decoded in order to obtain the desired audio sample within them. In the MXF sound track, there shall be a positive **origin** value which shall correspond to the difference in **position** of the first stored audio sample in the compressed audio frame and the **position** of the desired audio start sample. To achieve this, the **edit rate** of the sound track (e.g., 1/48 kHz) may be different to the **edit rate** of the picture track (e.g., 1/29.97 Hz). It is important to note that in order to construct an index table, all the indexed tracks shall have the same edit rate as described in the MXF format document.

The PES data track shows that there may be PES data in the file which is delivered a long time before it needs to be applied. The origin and edit rate of the data track shall be set to ensure that the data is delivered synchronously with the picture and sound information which had the same PTS values in the original MPEG stream.

It is the intention of this standard that the above scenarios can be calculated by using the PTS/DTS values in each captured stream.

4.4 External audio (informative)

In many applications involving MPEG video MXF files, the audio may be located in an external file as described in SMPTE EG 41. When the audio is organized in large chunks / frames, then the timing guidelines above must still be respected. Note that when the audio is external, there will be at least two file packages.

4.5 Wrapping options (normative)

4.5.1 Overview

The MXF generic container consists of one or more contiguous KLV wrapped content packages. The goal is that each content package should have the same duration and should contain the picture and sound data for that portion of the timeline. When long GOP MPEG video is used, this becomes difficult to achieve.

If we choose to wrap every access unit of long GOP MPEG video then we discover that the frames are being wrapped in transmission order rather than display order. If we interleave uncompressed audio sample in the same content package, we will not be re-ordering the audio samples to match the video. In addition, if the video is at a frame rate of 29.97 fps and the audio is sampled at 48 kHz then there will be a varying number of audio samples in the content packages as a result of the complex relationship of the clocks.

In general, wrapping modes are defined where one of the elements is deemed to be the "master" and all the other elements in the generic container are “slaves”. The most common “master” will be the picture element.

NOTE – It is anticipated that frame wrapped long GOP MPEG ES with the picture element as “master” will be the most used wrapping mode.

4.5.2 Missing and variable elements (informative)

There will be occasions when a wrapping algorithm will not require data from a particular element. This may occur when, for example, there are no private PES packets for this content package. Under these circumstances, the element is missing from the content package.

SMPTE 379M states that each content package in an essence container should have a constant number of elements.

Elements within a content package may vary in size from one content package to another. These variations may be quite large and in the limit, the data for an element may not be present at all — the element may have zero length.

4.5.3 Converting a TS / PS multiplex to an MXF KLV multiplex

MPEG TS and PS that contain audio packetized at a different rate from video have already made a determination of how to interleave audio and video ES packets. Altering this interleaving should only be done if the resulting stream still maintains the integrity of the original MPEG buffers.

When encoding an MPEG PS or other previously multiplexed stream of PES or ES packets in an MXF generic container, packets shall be assigned to generic container content packages in the order they are encountered. A new content package shall commence with each video picture. For each other stream type, all packets encountered in the MPEG stream before the next video picture shall be grouped into a single GC element of that type, whose duration shall be the sum of the durations of the individual packets.

If no packets of a given stream type are encountered, a zero duration GC element shall be encoded in order to meet the MXF encoder guidelines in the generic container document.

4.5.4 Finding the content (informative)

When an application needs to find MPEG content on the timeline, it needs to search through the various header metadata items to locate the essence. The normative relationships between the parameters are defined in the MXF format document and the text here is an informative algorithm to locate the content:

- The MXF application has a material package picture track with a sequence of SourceClips;
- The desired content occurs during one of these SourceClips;
- From the SourceClip, it knows which position on which track of which file package it is looking for;
- From the file package, it gets the PackageID;
- From the MultipleDescriptor in the FilePackage, it gets the EssenceContainerUL, which it finds to be the UL of a generic container;
- From the track, it selects the FileDescriptor within the MultipleDescriptor for the track of interest, and it gets the EssenceContainerUL — which will be used by the essence codec;
- The FileDescriptor may have TextLocators or NetworkLocators which may indicate external essence;
- It then searches the ContentStorage::EssenceContainerData property for an EssenceContainerData set with the correct PackageID;
- From the EssenceContainerData set, it gets the IndexSID and BodySID;
- It then scans the partition packs in the file, to select partitions with the correct IndexSID;
- It then searches the index table for the desired position, then the slice and delta for the desired track, and obtains a byte offset into the essence data (in partitions of the correct BodySID);
- It then scans the partition packs in the file, to select partitions with the correct BodySID;
- It uses the BodyOffset in the partition pack to find the correct body partition;
- Within these partitions, it looks only in KLV packets with the correct essence element key;
- Bytes 13,14,15 and 16 of the key will match the EssenceTrackNumber item in the track ID;
- The contents of the KLV packets are then parsed to find the right content start point (depending on the wrapping method used).

4.5.5 MPEG in the generic container (informative)

From the generic container document: *"The premise for the MXF generic container format is that of a general purpose essence data and metadata container for the containment of many different kinds of essence and metadata elements into a single entity by interleaving the data streams in a defined and time-synchronous manner (typically over a 1-frame duration)."*

The figure below shows the use of a generic container to contain an MPEG stream. In general, the generic container was designed to create a streamable format where each repetition of the interleaved elements represents a single field or frame. In many MPEG applications this works well. These are usually the ones where there is information in each item every frame.

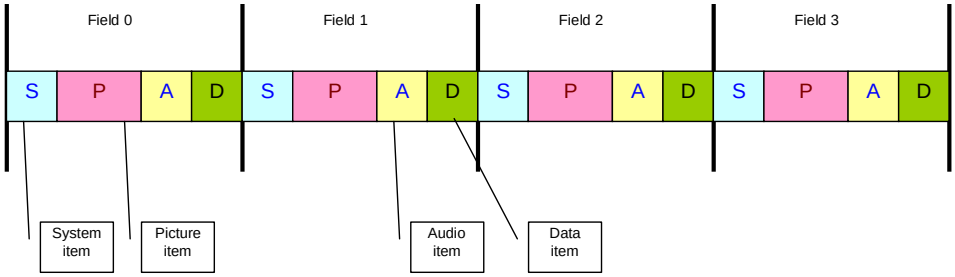

Figure 2 – MPEG in a generic container

Figure 2 shows an example of the interleave obtained by using the generic container. The picture implies that there is one field of picture in every content package, and that is associated with a system item, some audio information and some data. This follows the generic container policy that the audio and data are co-timed with the video information in the same content package.

Guidelines are required to ensure interoperability when the audio AU duration is not the same as the video AU duration. For example, when 48-kHz audio is coded with the MPEG-1 layer II algorithm, the AU size is 24 ms. The audio and video AUs will be synchronized every:

25 Hz:	every 3 picture frames (5 audio AU)
30 Hz:	every 18 picture frames (25 audio AU)
29.97 Hz:	every 32 seconds approx.

This gives rise to the situation where different content packages will contain different durations or different numbers of sample of the audio and data information. It is assumed that the video information is used as the “master” for creating content packages. This will be true for most cases, but it is important that MXF decoders follow MXF and KLV rules rather than make assumptions on how an MXF encoder might have behaved.

4.5.6 Interleaving guidelines (informative)

In order to give MXF encoders guidance for creating consistent generic containers between applications, interleaving guidelines are defined in section 5.1.2. The principle of creating the interleaving guidelines is to allow the creation of MXF files containing a mixture of unspecified components. For example:

- Long GOP MPEG ES + BWF sound;
- Long GOP MPEG PES + BWF sound;
- Uncompressed pictures + long GOP MPEG proxy + AES sound.

Different mapping mechanisms are listed in this and other mapping documents. Specific interleaving guidelines for implementing those mapping schemes are provided to improve interoperability. It should be

remembered that in addition to these low level interleaving guidelines, there may be higher level partition multiplexing rules operating which may be determined by the operational pattern of the file.

4.5.7 Multi-channel audio

There will frequently be multi-channel audio associated with long GOP MPEG pictures. This multi-channel audio may be coded as a single stream (e.g., 5.1 AC-3 encoding) or may be coded as 2 or more separate streams.

It is recommended that each separable audio stream be mapped into its own GC sound element, each of which follows the appropriate mapping and interleaving rules given elsewhere in this standard.

4.5.8 Picture identification

The MXF index table mechanism provides a deterministic method of identifying the key frames and dependencies of each indexed frame within the file. The user recommendations for the design of the MXF specification stated that there should always be index tables present in the file. The index tables are the primary mechanism for determining key frames and dependencies. There have been additional requests to identify the picture type within the KLV wrapping. This would allow easy building of index tables on the fly, all capture devices shall be able to parse the video stream to create this signaling. An MPEG video descriptor is defined in this document which gives parameters which allow fixed picture patterns to be identified.

If frame wrapping is used, then it is a simple procedure to determine these values. The value of the KLV triplet starts with the MPEG start_code of the video access unit. These are rapidly parsed to determine the picture type, coded_frame and AFD.

4.5.9 MPEG metadata

The MPEG descriptor detailed in section 8.1 provides a mechanism for mapping some MPEG descriptors to MXF structural metadata.

It is recommended that if time code is present in the MPEG GOP headers or has been inserted in accordance with SMPTE 328M then a time code track should be present in the MXF header metadata which accurately reflects the time code information found in the stream.

5 Mapping MPEG into the generic container

Three types of KLV wrapping are specified for the MPEG body structure:

- Frame wrapping;
- Clip wrapping;
- Custom wrapping.

A sequence of images shall be KLV coded as defined in SMPTE 336M. If this mapping specification is to be used with an MPEG multiplex (e.g., program stream or transport stream), then the multiplex shall be split into its component parts prior to wrapping. The splitting options are as follows:

- Picture Data → Demultiplex to ISO 13818-1 PES stream or demultiplex and remove PES headers to obtain elementary stream as appropriate;
- Sound Data → Demultiplex to ISO 13818-1 PES stream, demultiplex and remove PES headers to obtain elementary stream, B-Wav compatible stream, or other generic container compatible mapping as appropriate;
- Other Data → Demultiplex to ISO 13818-1 PES stream, demultiplex and remove PES headers to obtain elementary stream, or other generic container compatible mapping;
- Once demultiplexing has taken place, one of the KLV wrapping options below may be chosen.

5.1 Frame (picture access unit) wrapping

The "frame wrapping" methods for MPEG are shown in figures 3 and 4.

Figure 3 shows the whole of the image wrapped in a number of content packages with no other generic container elements in the container. Each content package has the duration of one MPEG video access unit. Note that this determines where the MPEG headers will be found (see section 4).

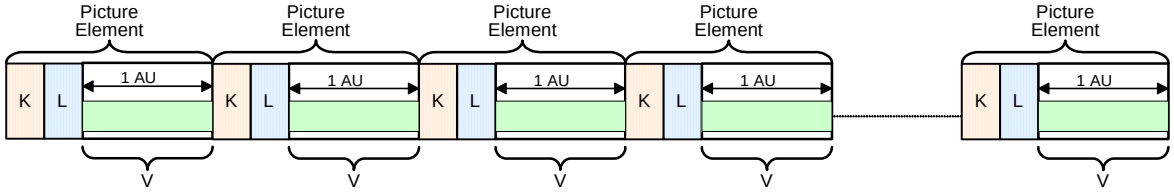

Figure 3 – Simple representation of frame wrapping

The frame wrapping method is intended to enable frame-by-frame access by MXF applications which process at the KLV level. This can be particularly useful for applications which support multiple generic container mapping types. Sufficient information is provided to allow individual frames to be identified at the KLV level without an MXF decoder having to parse or decode the essence data. Each frame of MPEG image data is KLV wrapped using a GC picture element key.

In some applications, the frame wrapped MPEG picture data will exist in content packages with other elements such as sound and data elements. An example generic container is shown in figure 4.

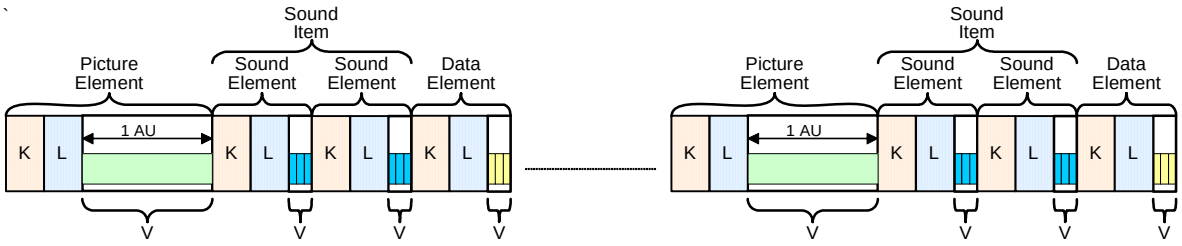

Figure 4 – Frame wrapping with other GC elements

The generic container mapping specifications for the sound and data elements will detail the key values and format of the data within the elements. Note that in this wrapping mode, the sound and data elements shall follow the guidance given in SMPTE 379M, section 5.4, concerning placement of the synchronized samples in essence containers. There are many situations where each wrapped frame of MPEG picture data will not exactly coincide with the sound/data elements within the same content package (e.g., a subtitle PES stream may have a duration of many seconds, but the bytes for the subtitle are included in a single data element in a single content package).

The sound/data elements may be PES wrapped, which will guarantee the synchronization. The MXF picture sound and data descriptors, along with the track definitions shall provide information to ascertain whether the picture/sound/data information is locked together. This specification allows a content package to be constructed from any MXF defined elements providing the generic container rules are respected.

5.1.1 Interleaving guidelines

An MXF encoder has many choices as to where data should be placed in a frame wrapped MPEG multiplex. Consistent rules will improve interoperability, especially when the relationship between the different elements is complex. Section 4.5.3 recommends that when wrapping an existing MPEG multiplex, the existing arrangements should be maintained where possible. When creating an MXF file from raw elementary stream components, the MXF encoder has greater flexibility. There are two different cases which will be considered:

- Creating an MXF file for interchange – easy access to all the components;
- Optimizing an MXF file for streaming – reducing buffer sizes and adding a small increase to the latency.

5.1.2 Interchange MXF file

In this case, it is desirable to group together elements which are synchronized on the timeline. The overall goal is to keep the synchronized video, audio and associated data in the same content package. This leads to the following rules.

In each content package:

- There shall be one video access unit.
- The synchronized sound sample should be in the first sound element in the same content package in accordance with SMPTE 379M, section 5.4. This implies that the start position of the video access unit should be equal to the start position of the sound element or fall within the duration of the first sound element. The unit of audio is defined in the appropriate GC sound element mapping document. Any valid GC sound element may be used. Examples might be an AES sample, a broadcast wave RIFF chunk, audio PES packet or MPEG-2 layer II audio frame.
- Sound elements should be placed in the content package until a sound element is found which may start a later content package. (Note that when the sound element duration is greater than the video access unit, this results in content packages with no or zero length sound elements).
- Any data element should start with the first indivisible unit of data where the start position of the video access unit is equal to the start position of the data element or falls within the duration of the first data element. The unit of data is defined in the appropriate GC data element mapping document. Any valid GC data element may be used.
- Any data element shall end with the unit of data whose position on the timeline is not later than the position of the next video access unit.

An implication of the above guidelines is that for each picture there will always be approximately the same time duration of data in the content package. Sometimes there will be empty elements and sometimes there will be more samples than average. This copes with the relationships between 29.97 frames/s video and its associated audio and data. Note that in many cases the sound and picture elements within the content package will have exactly the same duration.

Note that sound elements can be distinguished by the last 4 bytes of their key (see section 6.2). All sound element relating to a given track shall have the same last 4 bytes of the sound element key. This allows simple demultiplexing at the KLV level without having to parse the header metadata in advance.

5.1.3 Wrapping 3:2 sequences

The MPEG coding of 3:2 sequences and the definition of the AU is defined in the MPEG specifications (and explained in ISO13818-2, annex D). KLV wrapping of the 3:2 access units for interchange is, therefore, defined in section 5.1.2. Ensuring that an MPEG video access unit is wrapped defines the KLV wrapping point regardless of the regularity of the 3:2 sequence.

Construction of index tables for 3:2 sequences requires the index table construction application to calculate the correct byte offset for a given temporal offset regardless of the MPEG encoding used. This may be done

by the inspection of the MPEG headers for the occurrence of repeat_first_field flags or by some equivalent method.

5.2 Clip wrapping

In clip wrapping, KLV encoding wraps the whole of the MXF MPEG stream that may contain a single frame or thousands of frames. Any other elements in the generic container should also be clip wrapped.

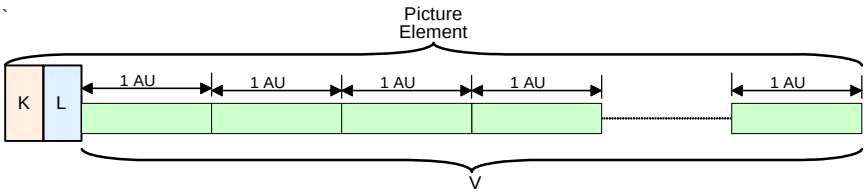

Figure 5 – Simple representation of clip wrapping a video stream

The clip wrapping method is intended for applications which carry the MPEG stream as a single large entity. This can be very useful in applications such as store and forward servers which process whole files and also in applications where it is desired to use the rich metadata structures of MXF as an annotation to MPEG data. The clip of MPEG data is KLV wrapped using an appropriate key as detailed in section 5.3 below. When MPEG data is clip wrapped, there shall be only one clip per generic container. Multiple clips can be concatenated and edited using the operational pattern mechanism detailed in the MXF format document.

In some applications, the clip wrapped MPEG data will exist in a content package with other elements such as sound and data elements. An example generic container is show in figure 6.

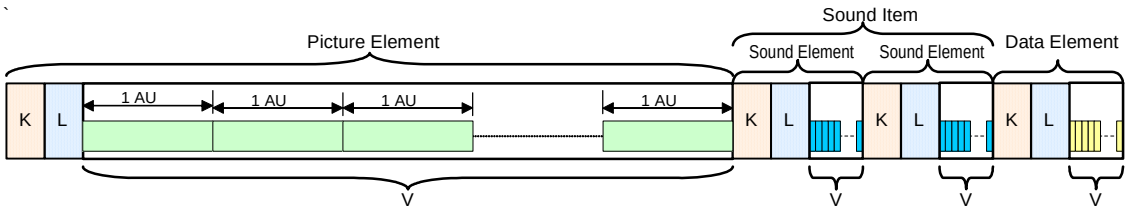

Figure 6 – Clip wrapping with other GC elements

The generic container mapping specifications for the sound and data elements will detail the key values and format of the data within the elements. Note that in this wrapping mode, the sound and data elements are intended to have a duration of the entire clip.

5.3 Custom wrapping

In custom wrapping, KLV encoding wraps the MPEG data in an application specific way. Usually there is a very good reason for this and all new wrapping modes shall be defined in the SMPTE registry. Unless otherwise stated, each of the custom wrappings shall treat the MPEG long GOP video stream as the master for determining the start and end points of the content package. In addition, the interleaving rules adapted from above:

In each content package:

- Any sound element shall start with the first unit of audio whose position on the timeline does not precede the first MPEG long GOP video access unit. The unit of audio is defined in the appropriate GC sound element mapping document. Any valid GC sound element may be used. Examples might be an AES sample, a broadcast wave RIFF chunk, audio PES packet or MPEG-2 layer II audio frame.
- Any sound element shall end with the unit of audio whose position on the timeline is not later than the position of the first video access unit in the next content package
- Any data element shall start with the first indivisible unit of data whose position on the timeline does not precede the first MPEG long GOP video access unit. The unit of data is defined in the appropriate GC data element mapping document. Any valid GC data element may be used.
- Any data element shall end with the unit of data whose position on the timeline is not later than the position of the first video access unit in the next content package.

A table of pre-defined custom wrapping modes is given here:

Table 1 – Custom wrapping descriptions

Name	Constraint	Application
Stripe	One KLV per macroblock row	Each Content Package contains an entire row of macroblocks. Note that for some MPEG 1 video streams this mode may not be possible.
PES	One KLV per PES	Each Content Package contains a single PES packet. Note that in some MPEG encoding applications, this may result in Content Packages with durations of less than a frame.
Fixed Audio Size	Fixed Audio Samples	The Sound Element is the Master Element. Wrapping is determined by a fixed number of audio samples. Can be used when the video rate is 1000/1001 * Integer_rate. The Picture Elements are fitted into the closest Content Package to their start time.
Splice	KLV at splice points	Each Content Package starts and ends at a valid MPEG splice point.
Closed GOP	KLV at closed GOPs	Each Content Package starts and ends at a closed GOP.
Slave	Slaved to another constraint	For example a video stream may be PES wrapped, and the associated audio is wrapped in chunks of approximately the same duration as the video. The audio is slave wrapped.
No specific constraint		No specific wrapping constraints were employed

5.3.1 MXF files optimized for streaming

When streaming an MXF file, it is desirable to reduce the size of the buffers needed in the receiver, which in turn reduces the overall latency of the system. The interchange file rule given in section 5.1.2 is streamable, but may require large receiver buffers to synchronize the picture, sound and data. The MPEG specification provides a lot of information on buffering and streaming and creating a system with compatible buffer characteristics is the goal.

When streaming a file, the decoder is intended to display the pictures and recreate the sound, while the file is being sent. The delay through the video and audio decoders is often not the same and, therefore, buffering is required in the decoder to bring the sound and pictures into synchronization. This buffering is often in addition to any buffering required for compression decoding and basic demultiplexing of the streams.

The MPEG specification ISO 13818 gives rules and guidelines for multiplexing the audio and video streams into either a program stream or a transport stream. The guidance given here is that an MXF encoder should create essence segments as though the output were a program stream with the desired properties, but instead of interleaving PES streams, interleaving the essence segments using this generic container mapping should be performed.

6 KLV coding of MPEG elements

When using this specification, all video streams (even when PES wrapping is employed) shall be wrapped using a picture element key. All audio streams (even when PES wrapping is employed) shall be wrapped using a sound element key. All other streams shall be wrapped using a data element key.

6.1 Picture element key

The values of the last four bytes of the essence element key are given below:

Table 2 – Key value for the MPEG picture element

Byte No.	Description	Value (hex)	Meaning
1-12	See MXF Generic Container Specification		
13	Item Type Identifier	15_h	Picture Item
14	Essence Element Count	kk_h	Count of Picture Elements in this Picture Item
15	Essence Element Type	05_h 06_h 07_h	Frame Wrapped Picture Element Clip Wrapped Picture Element Custom Wrapped Picture Element
16	Essence Element Number	nn_h	The Number (used as an Index) of this Picture Element in this Picture Item

6.1.1 Essence element count – Byte 14

This is a count of the number of elements in the picture item of the content package.

6.1.2 Essence element type – Byte 15

For frame wrapped MPEG pictures, this shall be 05_h. For clip wrapped MPEG pictures, this shall be 06_h. For custom wrapped MPEG pictures, this shall be 07_h.

6.1.3 Essence element number – Byte 16

This is a number used as an index to identify this instance of the element type within the item in each content package. The generic container specification requires that this byte be unique within an item. This mapping document recommends that each and every instance of the MPEG picture element should retain a consistent essence element number throughout the entire generic container.

NOTE – The actual value of this number has no meaning; however, its uniqueness is important.

6.1.4 Picture element length

The length field of the KLV coded element shall be 4 bytes BER long-form encoded (i.e., 83_h.xx.yy.zz) for frame wrapping. The length field of the KLV coded element shall be 8 bytes BER long-form encoded (i.e., 87_h.aa.bb.cc.dd.ee.ff.gg) for clip wrapping. For custom wrapping, the length field shall be consistent and shall be either 4 bytes or 8 bytes as appropriate.

6.1.5 Picture element value

The value field of the KLV coded element shall be the video access unit for frame wrapping. This may include PES headers. For clip wrapping, this shall be the entire MPEG video clip.

6.2 Sound element key

The values of the last four bytes of the essence element key are given below:

Table 3 – Key value for the MPEG sound element

Byte No.	Description	Value (hex)	Meaning
1-12	See MXF Generic Container Specification		
13	Item Type Identifier	16h	Sound Item
14	Essence Element Count	kkh	Count of Sound Elements in this Sound Item
15	Essence Element Type	05h 06h 07h	Frame Wrapped Sound Element Clip Wrapped Sound Element Custom Wrapped Sound Element
16	Essence Element Number	nnh	The Number (used as an Index) of this Sound Element in this Sound Item

6.2.1 Essence element count – Byte 14

This is a count of the number of elements in the sound item of the content package.

6.2.2 Essence element type – Byte 15

For frame wrapped MPEG sound, this shall be 05_h. For clip wrapped MPEG sound, this shall be 06_h. For custom wrapped MPEG sound, this shall be 07_h.

6.2.3 Essence element number – Byte 16

This is a number used as an index to identify this instance of the element type within the item in each content package. The generic container specification requires that this byte be unique within an item. This mapping document recommends that each and every instance of the MPEG sound element should retain a consistent essence element number throughout the entire generic container.

NOTE – The actual value of this number has no meaning; however, its uniqueness is important.

6.2.4 Sound element length

The length field of the KLV coded element shall be 4 bytes BER long-form encoded (i.e., 83_h.xx.yy.zz) for frame wrapping. The length field of the KLV coded element shall be 8 bytes BER long-form encoded (i.e., 87_h.aa.bb.cc.dd.ee.ff.gg) for clip wrapping. For custom wrapping the length field shall be consistent and shall be either 4 bytes or 8 bytes as appropriate.

6.2.5 Sound element value

The value field of the KLV coded element shall be the audio access unit(s) for frame wrapping as detailed in the rules in section 5.1.2, this may include PES headers. For clip wrapping, this shall be the entire MPEG audio clip.

6.3 Data element key

The values of the last four bytes of the essence element key are given below:

Table 4 – Key value for the MPEG data element

Byte No.	Description	Value (hex)	Meaning
1-12	See MXF Generic Container Specification		
13	Item Type Identifier	17h	Data Item
14	Essence Element Count	kkh	Count of Data Elements in this Data Item
15	Essence Element Type	05h 06h 07h	Frame Wrapped Data Element Clip Wrapped Data Element Custom Wrapped Data Element
16	Essence Element Number	nnh	The Number (used as an Index) of this Data Element in this Data Item

6.3.1 Essence element count – Byte 14

This is a count of the number of elements in the data item of the content package.

6.3.2 Essence element type – Byte 15

For frame wrapped MPEG data, this shall be 05_h. For clip wrapped MPEG data, this shall be 06_h. For custom wrapped MPEG data, this shall be 07_h.

6.3.3 Essence element number – Byte 16

This is a number used as an index to identify this instance of the element type within the item in each content package. The generic container specification requires that this byte be unique within an item. This mapping document recommends that each and every instance of the MPEG data element should retain a consistent essence element number throughout the entire generic container.

NOTE – The actual value of this number has no meaning; however, its uniqueness is important.

6.3.4 Data element length

The length field of the KLV coded element shall be 4 bytes BER long-form encoded (i.e., 83_h.xx.yy.zz) for frame wrapping. The length field of the KLV coded element shall be 8 bytes BER long-form encoded (i.e., 87_h.aa.bb.cc.dd.ee.ff.gg) for clip wrapping. For custom wrapping, the length field shall be consistent and shall be either 4 bytes or 8 bytes as appropriate.

6.3.5 Data element value

The value field of the KLV coded element shall be the data access unit(s) for frame wrapping as detailed in the rules in section 5.1.2; this may include PES headers. For clip wrapping, this shall be the entire MPEG data clip, for example an entire PS or TS clip.

6.4 Other elements in the generic container

6.4.1 Fill elements

There may be fill information present in the generic container. These are filler metadata items as defined in the SMPTE metadata dictionary (SMPTE RP 210).

6.4.2 Other elements

Other elements may be present in frame wrapped, clip wrapped or custom wrapped MXF generic containers. The mappings of these other elements which may contain information such as extra audio, data, ancillary data, etc. are defined in separate generic container mapping documents.

6.4.3 Use of generic container system elements

There are applications where the wrapping mechanism does not readily identify the start of a frame, random access unit or other important mark within a stream. In SDTI-CP interchange applications, it is recommended that the system item defined in SMPTE 385M is used. Other system elements may be designed and used for this purpose.

6.4.4 Use of KAG

There are no specific KAG requirements for this mapping. MXF encoders and decoders shall comply with the KAG rules in the MXF format document.

7 SMPTE label for essence container identification

The values for the essence container UL are give in table 5.

Table 5 – Specification of the MPEG essence container label

Byte No.	Description	Value (hex)	Meaning
1-12	Defined by Generic Container		
13	Essence Container Kind	02_h	MXF Generic Container
14	Mapping Kind	04_h 07_h 08_h 09_h	MPEG ES as listed in SMPTE RP224 MPEG PES as listed in SMPTE RP224 MPEG PS MPEG TS
15	Locally defined	xx_h	ISO13818-1 stream_id bits 6..0
16	Locally defined	yy_h	Wrapping scheme Table 6

This SMPTE label is the individual “essence container” property used in the partition pack, in the preface set and in the appropriate file descriptor.

Byte 14 identifies the container as the MPEG mapping into the generic container.

Byte 15 enumerates the content of the element within the essence container. It is the value of bits 6..0 of the stream_id as specified by ISO 138181-1. If the content to be wrapped is ES, then this byte shall have the correct value of stream_id that would be used if the content were PES. If byte 14 is set to 08_h or 09_h then this byte shall be set to the same value as byte 14.

NOTE – According to SMPTE 377M, “an EssenceContainer is a UL that identifies the different essence container types used in or referenced by this MXF file.” To uniquely determine what kind of essence is within any container, an application uses the essence descriptor. The stream_id value used here is intended to give a rough guide as to the MPEG content. The stream_id may indicate private data or the stream_id extension mechanism. As is required by SMPTE 377M, full details of the precise signaling will be found in the essence descriptor. Specifications which map new, extended or private MPEG data types may use this specification and an extension to one of the MXF essence descriptors to create the mapping.

Byte 16 is intended to carry basic information about the containment of the MPEG data. It is set according to table 6.

Table 6 – Specification of the MPEG essence container label, byte 16

Byte 16	Meaning
00_h	Not used
01_h	Frame Wrapping
02_h	Clip Wrapping
03_h	Custom: Stripe Wrapping
04_h	Custom: PES
05_h	Custom: Fixed Audio Size
06_h	Custom: Splice
07_h	Custom: Closed GOP
08_h	Custom: Slave
09_h - $7E_h$	Reserved
$7F_h$	Custom: No specific wrapping constraints

It is important to note that this specification may create several elements in a generic container. This UL is intended to identify the wrappings of the elements. If there are N elements, then there will be N ULs in the collection of ULs in the partition pack and header metadata.

If Byte 14 is set to 08_h or 09_h then this byte shall be set 02_h indicating clip wrapping.

8 Essence descriptors for MPEG mappings

Most MXF files containing long GOP streams will also contain sound and data elements in the generic container. This will create the need for a multiple descriptor in the file package which describes the essence. The multiple descriptor will, in turn, reference a descriptor for each of the elements in the generic container.

NOTE – In accordance with SMPTE 377M, the precise coding of the content is identified by one of:

- Generic picture essence descriptor “picture essence coding” property (SMPTE 377M, section D2.1)
- Generic sound essence descriptor “sound essence compression” property (SMPTE 377M, section D3)
- Generic data essence descriptor “data essence coding” property (SMPTE 377M, section D4)

8.1 MPEG video descriptor

For MPEG-2 long GOP video, the descriptor below which extends the CDCI descriptor shall be used. Local tags shall be allocated using the primer pack mechanism defined in the MXF file format descriptor.

Note – This descriptor is called the MPEG video descriptor because it carries many of the video properties which were originally standardized in the MPEG1 and MPEG2 standards. The properties defined here are all optional and can be used with any essence picture essence which can be mapped by this document, for example MPEG4-2.

Table 7 – MPEG video descriptor

Item Name	Type	Len	UL Designator	Req ?	Meaning	Default
MPEG Video Descriptor	Set UL	16	See table below	Req	Defines the MPEG Video Descriptor Set	
Length	BER Length	4		Req	Set length	
All items from the MXF Format CDCI Descriptor						
SingleSequence	Boolean	1	04.01.06.02.01.02	Opt	TRUE if the essence consists of a single MPEG sequence. False if there are a number of sequences. This flag implies that the sequence header information is not varying in the essence stream.	Unknown
ConstantBframes	Boolean	1	04.01.06.02.01.03	Opt	TRUE if the number of B frames is always constant	FALSE
CodedContentType	Enum	1	04.01.06.02.01.04	Opt	0= "Unknown" 1= "Progressive" 2= "Interlaced" 3= "Mixed" an enumerated value which tells if the underlying content which was MPEG coded was of a known type	Unknown
LowDelay	Boolean	1	04.01.06.02.01.05	Opt	TRUE if low delay mode was used in the sequence	FALSE
ClosedGOP	Boolean	1	04.01.06.02.01.06	Opt	TRUE if closed_gop is set in all GOP Headers, per ISO/IEC 13818-1 IBP descriptor.	FALSE
IdenticalGOP	Boolean	1	04.01.06.02.01.07	Opt	TRUE if every GOP in the sequence is constructed the same, per ISO/IEC 13818-1 IBP descriptor.	FALSE
MaxGOP	Uint16	2	04.01.06.02.01.08	Opt	Specifies the maximum occurring spacing between I frames, per ISO/IEC 13818-1 IBP descriptor. A value of 0 or the absence of this property implies no limit to the maximum GOP	no limit
BPictureCount	Uint16	2	04.01.06.02.01.09	Opt	Specifies the maximum number of B pictures between P or I frames, equivalent to ISO/IEC 13818-2 annex D (M-1)	
BitRate	UInt32	4	04.01.06.02.01.0B	Opt	Maximum bit rate of MPEG video elementary stream in bit/s as defined in ISO/IEC 13818-2 bit_rate property.	
ProfileAndLevel	Uint8	1	04.01.06.02.01.0A	Opt	Specifies the MPEG-2 video profile and level. The value is taken directly from the profile_and_level_indication in the MPEG-2 sequence header extension. For main profile @ main level, the value is 0x48. For 4:2:2 profile @ main level, the value is 0x85	

NOTES

1 Many of these properties in table 7 appear in the ISO/IEC 13818-1 IBP descriptor. They may be specified even if the descriptor is not present in the MPEG stream.

2 The properties in table 7 should be used only when the values are constant throughout the essence described. Dynamically varying descriptor properties should be specified separately.

8.1.1 Key for the MPEG 2 video descriptor

The key (UL) for this local set is defined below:

Table 8 – Key for MPEG 2 video descriptor

Byte No.	Description	Value (hex)	Meaning
1-13	Defined in the Structural Header Metadata Implementation section of SMPTE377M (File Format Specification)		
14	Set Kind (1)	01_h	MPEG Video Descriptor
15	Set Kind (2)	51_h	
16	Reserved	00_h	Reserved

8.1.2 MPEG defined descriptors and their relationship to MXF

When demultiplexing a single program MPEG PS or MPEG TS into an MXF file, an MXF multiple descriptor is required which references an MXF file descriptor for each of the tracks placed in the MXF file. The ES_info and elementary_stream_info descriptors may contain useful information. It is not a requirement that this information be present in the MXF files. Where there is an equivalent MXF descriptor, these values should be copied from the corresponding MPEG descriptor.

Annex A (informative)
Index tables for MPEG mappings

This annex gives some worked examples for the use of index tables in MXF. Entries in index tables should point to start of the **synchronized** data for each element. Note that due to frame reordering in long GOP MPEG and the fact that audio access units may not be the same duration as video access units, this may result in the synchronized data being physically located in different content packages. Refer to the MXF file format specification for details of the index table specification, especially the reordering look up mechanism.

A.1 Edit units and index tables

The MXF file format specification allows each track in a package to have a different edit rate. Although flexible, this makes indexing of the content difficult. For interleaved essence, the file format specification constrains the edit rates of all indexed tracks to be identical. This mapping document further constrains the edit rate of the indexed tracks so the temporal offset and key frame offset properties shall be integer numbers.

NOTE – Most commonly, the edit rate of a track will be an MPEG video frame, but sometimes greater flexibility is achieved if the edit rate is a video field.

A.2 Setting the properties in an index entry

There are several properties in the IndexEntry which have specific meanings for a long GOP MPEG index table. These flags shall be correctly set according to the text below table A.1. This section describes the conditions for setting the flags. Annex B describes how these parameters can be used to correctly identify the MPEG picture type from these properties once they have been correctly set.

Table A.1 – Index table entry properties

Parameter	Type	Meaning	Use
Temporal Offset	Int8	Offset in edit units from Display Order to Coded Order (see SMPTE 377M for usage)	Used to find the IndexEntry for a stored Picture given its Display Temporal position.
Key Frame Offset	Int8	Offset in edit units to previous Key Frame (i.e. key frame or I frame). The value is zero if this is a Key frame.	The offset to the Key frame
Flags	EditUnitFlag	Flags for this Edit Unit Bit 7: Random Access Bit 6: Sequence Header Bit 5: Forward prediction flag Bit 4: Backward prediction flag Bit 3: Offsets out of range Bit 2: Not used by MPEG Bits 0,1: MPEG Frame type	e.g. naïve settings for Bits 5, 4: 00== I frame 10== P frame 11== B frame is identical to the MPEG frame type bits 1,0 00== I frame 10== P frame 11== B frame

Temporal offset: This is an offset used to allow lookups in the index table. It corresponds to the difference between the display order and transmission order for the indexed picture measured in edit units, not in frames. This is explained in the text near figures 22 and 23 of the MXF file format specification.

If the numerical range of this property is exceeded, then bit 3 of the flags shall be set, and this property shall be clipped to the maximum value which can be represented. The actual temporal offset can be determined by inspecting IndexEntries for neighboring pictures until a valid temporal offset is found. The value of temporal offset can be determined by the number of index entries inspected, their picture types and the valid temporal offset value found.

Key frame offset: This is the offset measured in edit units (not frames) to the key frame (I frame) required for decoding the indexed picture. If there are no key frames used in the MPEG coding, then this parameter shall be set to zero. If the numerical range of this property is exceeded, then bit 3 of the flags shall be set, and this property shall be clipped to the maximum value which can be represented. The actual key frame offset can be determined by inspecting IndexEntries for neighboring pictures until a valid key frame offset is found. The value of key frame offset can be determined by the number of IndexEntries inspected, their picture types and the valid key frame offset value found.

If there are no I pictures in the stream, then this parameter shall be set to a constant value which allows the current indexed frame to be decoded. This will depend on the percentage of I macroblocks and their distribution throughout the image.

Key frame offset is always negative for MPEG-2 streams. The offset is measured in edit units and is the transmission order offset of the key frame index entry. This keeps the meaning of the sign of key frame offset consistent with the sign of temporal offset. It also means that the only display order process is that of using the temporal offset to find the appropriate index entry. Index entries are always stored in transmission order.

Flag bit 7: This is the random access bit and shall be set when a random access point in the long GOP MPEG stream is encountered. A random access point is one where decoding can commence at that point in the stream. A sequence_header and a closed_GOP shall be present at the indexed picture.

Flag bit 6: This bit marks a sequence_header in the long GOP MPEG stream. This bit shall be set if the random access bit is set. If the indexed picture has a sequence_header, but occurs at an open_GOP, then this bit shall be set and the random access bit shall not be set.

NOTE – Decoding may commence at this point, but there may be some invalid content decoded until a key frame is displayed, or all the macroblocks on the screen have been refreshed.

Flag bits 5 and 4: These bits indicate the temporal dependence of the indexed frame. These bits may be set naively or may be set strictly depending on picture type. The strict settings are used by applications which require higher levels of random access performance. The naïve settings are used when strict setting is not possible. The setting of these bits is as follows:

Naïve setting of bits 5 and 4:

I frame	00	(no prediction)
P frame	10	(forward prediction from previous frame)
B frame	11	(forward and backward prediction)

Strict setting of bits 5 and 4 shall be used to improve performance of the system. The MPEG bitstream must be parsed and each of the motion vectors for each macroblock inspected. They should then be categorized into “no prediction”, “forward prediction” and “backward prediction”. The following settings are possible:

00 No prediction. This is always the case for I frames. P frames and B frames consisting only of I macroblocks should also set these flags to 0.
10 Forward prediction from previous frame. This is the case for P frames which have non zero motion vectors, or B frames which have only forward prediction motion vectors.
01 Backward prediction to future frame. This is the case for B frames which commence a closed GOP.
11 Forwards and backwards prediction. This is the general case for bi-directionally predicted B frames.

Flag bit 3: This bit indicates that the numerical range of the temporal offset or key frame fields has been exceeded. In this case, the offset and/or key frame must be determined by inspecting other index entries to determine the precise values.

Flag bit 2: Not used by the MPEG mapping — reserved for future use or use by other mappings.

Flag bits 1 and 0: These bits indicate MPEG picture type. The setting of these bits is as follows:

I frame	00	(no prediction)
P frame	10	(forward prediction from previous frame)
B frame	11	(forward and backward prediction)

A.3 Frame wrapping with extra Items in the generic container

This is a case where the index table points to the first byte of the MPEG picture element generic container key. The other generic container elements should be indexed by correct use of the delta entries and index entries. This example assumes that the sound elements which are indexed require the use of the fractional position mechanism defined in the MXF format specification. Figure A.1 represents a typical content package being indexed. This figure is based on figure 21 in the MXF file format specification.

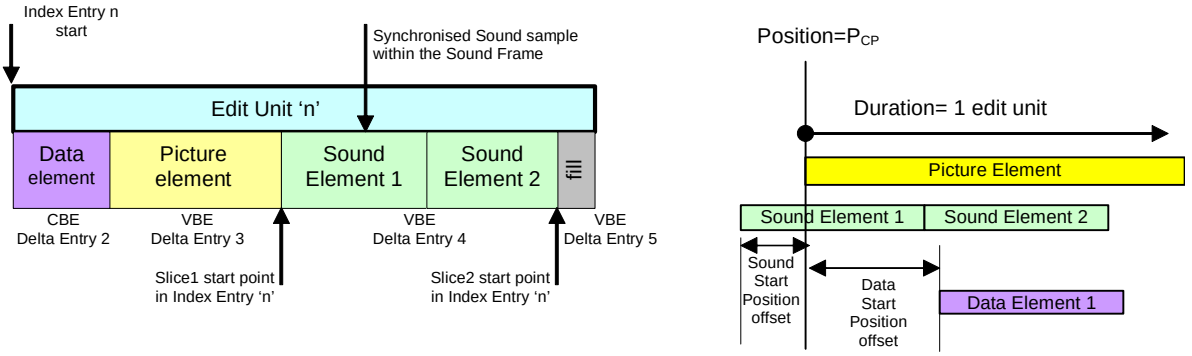

Figure A.1 – Content package for index table example

In this example, the picture, sound and fill are all VBE. The fill is indexed so that it can be eliminated from any essence byte counting based solely on calculations in the index table.

Table A.2 – Frame wrapped index table segment set example for figure A.1

Item Name	Req ?	Meaning	Use
Index Table Segment	Req	An Index Table Segment set	See MXF Format Specification
Length	Req	Set Length	See MXF Format Specification
Instance ID	Req	Unique ID of this instance	See MXF Format Specification
Edit Rate	Req	Edit Rate copied from the tracks of the Essence Container	See MXF Format Specification
Start position	Req	The first editable unit indexed by this Index Table segment measured in File Package Edit Units	This sets the temporal start point for an Index Table segment
Duration	Req	Time duration of this table segment measured in Edit Units of the referenced Package	Combined with start position, allows an application to determine if this Index Table Segment spans a particular Position value
Edit Unit Byte Count	D/Req	Defines the byte count of each and every Edit Unit. A value of 0 defines the byte count of Edit Units is only given in the Index Entry Array	0 unless the total length of all the GC Elements are of constant size. In this example for Figure 4 we assume that only the Data Elements are VBR so the value is 0.
IndexSID	D/Req	Stream Identifier (SID) of Index Table	See MXF Format Specification
BodySID	Req	Stream Identifier (SID) of the indexed essence container	See MXF Format Specification
Slice Count	D/Req	Number of slices minus 1 (NSL)	2
PosTableCount	Opt	Number of PosTable Entries minus 1 NPE	1
Delta Entry Array	Opt	Map Elements onto Slices	Table A.3
Index Entry Array	D/Req	Index from Edit Unit number to stream offset	Table A.4

The delta entry array contains an entry for every indexed element in the generic container. The order of the elements in the delta entry array matches the order of the elements in the generic container. The example below is a delta entry array designed to match the example in figure A.1. Implementations should construct a delta entry array according to the properties of the actual essence in the file.

In frame wrapping mode, the element delta values shall include the lengths of the "KL" for each element. The result is that each element delta shall point to the first byte of the key in the KLV which wraps an element. If the overall length of all the elements in each frame is constant, then a delta entry array and an "edit unit byte count" item are sufficient to define the index table segment. In this example, we have several variable length elements so that an index entry array is required.

Note also that the delta entry array does not distinguish which element is which in the index table. To know which element is indexed, the following rules shall apply when an MPEG long GOP stream is indexed:

- Each content package shall start with the same number and order of elements and the previous content package;
- If new elements are introduced for whatever reason, they shall be appended to the end of the existing content package elements;
- If elements in the content package have no data, then an index entry for a 0 length VBE element shall be created;
- Index tables shall have the same number of delta entries as the maximum number of elements in any content package;
- If there is no data in an element in a content package, then its index entry shall be zero;
- The data type of an index entry can be determined by inspecting the key which is pointed to by a **non-zero length** index entry.

Table A.3 – Frame wrapped delta entry array example for figure A.1

	Field Name	Type	Meaning	Use
	NDE	UInt32	Number of delta entries	4
	Length	UInt32	Length of each delta entry	6
Data Delta Entry	PosTableIndex	Int8	0= No reordering +ve = PosTable Index	1 (1st PosTable Entry)
	Slice	UInt8	Slice number in IndexEntry	0
	Element Delta	UInt32	Delta from start of slice to this Element	0
Picture Delta Entry	PosTableIndex	Int8	-ve - reordered	-1 (reordered Long GOP content)
	Slice	UInt8	Slice number in IndexEntry	0
	Element Delta	UInt32	Delta from start of slice to this Element	sizeof(KL) + sizeof(Data)
Sound Delta Entry	PosTableIndex	Int8	0= No reordering +ve = PosTable Index	2 (2nd PosTable Entry)
	Slice	UInt8	Slice number in IndexEntry	1
	Element Delta	UInt32	Delta from start of slice to this Element	0
Fill Delta Entry	PosTableIndex	Int8	0= No reordering +ve = PosTable Index	0
	Slice	UInt8	Slice number in IndexEntry	2
	Element Delta	UInt32	Delta from start of slice to this Element	0

Table A.4 – Frame wrapped index entry array description for figure A.1

	N	Field Name	Type	Meaning	Use
	1	NIE	UInt32	Number of index entries	=number of frames
	1	Length	UInt32	Length of each index array entry	varies depending on the number of slices and Position offsets
One Index Entry for every frame	NIE	Temporal Offset	Int8	Offset in edit units from Display Order to Coded Order	According to picture type
		Key Offset	Int8	Offset in edit units to previous Key Frame. The value is zero if this is an Key frame.	According to picture type
		Flags	EditUnitFlag	Flags for this Edit Unit	Note that some B frames may be forward only or backward only, or entirely Intra. P frames may be entirely Intra. If the precise values are not known, then the flags shall be set naively
		Stream Offset	UInt64	Offset in bytes from the first KLV element in this Edit Unit within the essence container Stream	Offset from the first byte of the key of the KLV for the first frame to the first byte of the Key of the KLV for the Data Element in this frame as shown in figure A.1
		SliceOffset	NSL x UInt32	The offset in bytes from the Stream Offset to the start of this slice.	Optional depending on the complexity of the VBR items. In this case there are 3 slices and NSL is set to 2
		PosTable	NPE *Rational	The fractional position offset from the start of the Content Package to the synchronized sample in the Content Package	This should be calculated for each wrapped element to ensure precise synchronization is maintained. There are 2 elements requiring offsets in this example so NPE=2

The table above shows the descriptions of the various elements required in the index entry. Below, the table shows entries for the first 6 frames of a long GOP sequence. The following example values have been used in creating the table:

- The GOP display sequence for frames 0-5 is $B_0I_1B_2P_3B_4P_5$. This is the indexed order of the frames when looking up the temporal offset;
- The GOP transmission order for frames 0-5 is $I_1B_0P_3B_2P_5B_4$. This is the stored order of the frames and the stored order of the index table data;
- The GOP is closed (i.e. the first B frame contains predictions only from the I frame);
- All lengths include the length of the key and length fields (16 bytes + 4 bytes);
- The data element length is fixed at 700 bytes and temporally offset by -0.25 edit units;
- The I frames are 48000 bytes, P frames are 9000 bytes and B frames 1000 bytes;
- In the 6 content packages, there are 8 sound elements;
- The number of sound elements are multiplexed in the content packages as follows: (1)(1)(2)(1)(1)(2);
- Each sound elements is 1000 bytes;
- Each fill element is 300 bytes

Table A.5 – Index entry samples

	N	Field Name	Type	Value	Note
		NIE	UInt32	6	6 Entries in this Index Table Segment
		Length	UInt32	35	Sizeof(IndexEntry) including the SliceOffsets & PosTable
	Index Entry[0] contains Index data for I_1, and a Temporal offset from B_0 to Index Entry[1]				
0	Index Entry[0] – "I_1"	Temporal Offset	Int8	1	B_0 index data is stored in Index Entry[1]
		Key Frame Offset	Int8	0	The Key frame is IndexEntry[0]
		Flags	EditUnitFlag	00h	I frame – no reference
		Stream Offset	UInt64	0	Offset of the 1st Stored Conent Package in the Stream – I_1
		SliceOffset[0]	UInt32	48700	sizeof(data) + sizeof(I_1 frame)
		SliceOffset[1]	UInt32	49700	sizeof(data) + sizeof(I_1 frame) + sizeof(sound)
		PosTable[0]	Rational	-1/4	Temporal offset from start of data to start of video
		PosTable[1]	Rational	-1/3	Temporal offset from start of sound to start of video
	Index Entry[1] contains Index data for B_0, and a Temporal offset from I_1 to Index Entry[0]				
1	Index Entry[1] – "B_0"	Temporal Offset	Int8	-1	I_1 index data is stored in Index Entry[0]
		Key Frame Offset	Int8	1	The Key frame is IndexEntry[0] and 1-0=1
		Flags	EditUnitFlag	D0h	Closed GOP B frame – backward reference, sequence_header & random access
		Stream Offset	UInt64	50 000	Offset of the 2nd Stored Content Package in the Stream – B_0
		SliceOffset[0]	UInt32	1700	sizeof(data) + sizeof(B_0 frame)
		SliceOffset[1]	UInt32	2700	sizeof(data) + sizeof(B_0 frame) + sizeof(sound)
		PosTable[0]	Rational	0	Temporal offset from start of data to start of video
		PosTable[1]	Rational	0	Temporal offset from start of sound to start of video
	Index Entry[2] contains Index data for P_3, and a Temporal offset from B_2 to Index Entry[3]				
2	Index Entry[2] – "P_3"	Temporal Offset	Int8	1	B_2 index data is stored in Index Entry[3]
		Key Frame Offset	Int8	-2	The Key frame is IndexEntry[0] and 0-2= -2
		Flags	EditUnitFlag	40h	P frame – forward reference
		Stream Offset	UInt64	53 000	Offset of the 3rd Stored Content Package in the Stream – P_3
		SliceOffset[0]	UInt32	9700	sizeof(data) + sizeof(P_3 frame)
		SliceOffset[1]	UInt32	11700	sizeof(data) + sizeof(P_3 frame) + 2*sizeof(sound)
		PosTable[0]	Rational	-1/4	Temporal offset from start of data to start of video
		PosTable[1]	Rational	0	Temporal offset from start of sound to start of video
	Index Entry[3] contains Index data for B_2, and a Temporal offset from P_3 to Index Entry[2]				
3	Index Entry[3] – "B_2"	Temporal Offset	Int8	-1	P_3 index data is stored in Index Entry[2]
		Key Frame Offset	Int8	-3	The Key frame is IndexEntry[0] and 0-3= -3
		Flags	EditUnitFlag	30h	B frame – bidirectional reference
		Stream Offset	UInt64	65 000	Offset of the 4th Stored Content Package in the Stream – B_2
		SliceOffset[0]	UInt32	1700	sizeof(data) + sizeof(B_2 frame)
		SliceOffset[1]	UInt32	2700	sizeof(data) + sizeof(B_2 frame) + sizeof(sound)
		PosTable[0]	Rational	-1/4	Temporal offset from start of data to start of video
		PosTable[1]	Rational	-2/3	Temporal offset from start of sound to start of video

	N		Field Name	Type	Value	Note
	Index Entry[4] contains Index data for P_5, and a Temporal offset from B_4 to Index Entry[5]					
4	Index Entry[4] – "P_5"		Temporal Offset	Int8	1	B_4 index data is stored in Index Entry[5]
			Key Frame Offset	Int8	-4	The Key frame is IndexEntry[0] and 0-4= -4
			Flags	EditUnitFlag	40h	P frame – forward reference
			Stream Offset	UInt64	68 000	Offset of the 5th Stored Content Package in the Stream – P_5
			SliceOffset[0]	UInt32	9700	sizeof(data) + sizeof(P_5 frame)
			SliceOffset[1]	UInt32	10700	sizeof(data) + sizeof(P_5 frame) + sizeof(sound)
			PosTable[0]	Rational	-1/4	Temporal offset from start of data to start of video
			PosTable[1]	Rational	-2/3	Temporal offset from start of sound to start of video
	Index Entry[5] contains Index data for B_4, and a Temporal offset from P_5 to Index Entry[4]					
5	IE[5] – "B_4"		Temporal Offset	Int8	-1	P_5 index data is stored in Index Entry[4]
			Key Frame Offset	Int8	-5	The Key frame is IndexEntry[0] and 0-5= -5
			Flags	EditUnitFlag	30h	B frame – bidirectional reference
			Stream Offset	UInt64	79 000	Offset of the 6th Stored Content Package in the Stream – B_4
			SliceOffset[0]	UInt32	1700	sizeof(data) + sizeof(B_4 frame)
			SliceOffset[1]	UInt32	3700	sizeof(data) + sizeof(B_4 frame) + 2*sizeof(sound)
			PosTable[0]	Rational	-1/4	Temporal offset from start of data to start of video
			PosTable[1]	Rational	-1/3	Temporal offset from start of sound to start of video

A.4 Clip wrapping with extra items in the generic container

This is the most complex case where the index table points to the first byte of the payload of each element in the generic container. The other generic container elements should be indexed by correct use of the delta entries and index entries. The delta entry values are limited to a UInt32 number range and will, therefore, be incapable of specifying the offset from the picture item of frame 1 to the sound item of frame 1.

To overcome this, the index table segment is sliced, even if the offset to the sound item is constant so that there is, in effect, a 33-bit variable offset from the MPEG frame to the corresponding sound item (i.e., a 32-bit variable slice offset + 32-bit fixed delta entry of, for example, 2^{32}-1). In clip wrapping where an index table is required, but 33-bit offsets are insufficient, the clip wrapped generic container must be split into several smaller generic containers, each with a single clip.

Table A.6 – Clip wrapped index table segment set example for figure 6

Item Name	Req ?	Meaning	Use
Index Table Segment	Req	An Index Table Segment set	See MXF Format Specification
Length	Req	Set Length	See MXF Format Specification
Instance ID	Req	Unique ID of this instance	See MXF Format Specification
Edit Rate	Req	Edit Rate copied from the tracks of the Essence Container	See MXF Format Specification
Start position	Req	The first editable unit indexed by this Index Table segment measured in File Package Edit Units	This defines the temporal start point for this Index Table Segment
Duration	Req	Time duration of this table segment measured in Edit Units of the referenced Package	This, combined with the Start Position values allows an application to determine whether or not this Index Table Segment spans a given Position value.
Edit Unit Byte Count	D/Req	Defines the byte count of each and every Edit Unit. A value of 0 defines the byte count of Edit Units is only given in the Index Entry Array	0 unless the total length of all the GC Elements is of constant size. In this example we assume the Data Elements are VBR so the value is 0.
IndexSID	D/Req	Stream Identifier (SID) of Index Table	See MXF Format Specification
BodySID	Req	Stream Identifier (SID) of the indexed Essence Container	See MXF Format Specification
Slice Count	D/Req	Number of slices minus 1 (NSL)	3 (one slice per element)
PosTableCount	Opt	Number of PosTable Entries minus 1 NPE	In this example, there is only a single Content Package and the audio and video start synchronously. Hence NPE=0
Delta Entry Array	Opt	Map Elements onto Slices	Table A.7
Index Entry Array	D/Req	Index from Edit Unit number to stream offset	Table A.8

The delta entry array shall contain an entry for every indexed element in the generic container. The order of the elements in the delta entry array shall match the order of the elements in the generic container. The example below is a delta entry array designed to match the example in figure 6. Implementations should construct a delta entry array according to the properties of the actual essence in the file.

In clip wrapping mode, the element delta values shall **not** include the lengths of the "KL" for each element. The result is that each element delta shall point to the first byte of the value in the KLV which wraps an element.

Table A.7 – Clip wrapped delta entry array example for figure 6

	Field Name	Type	Meaning	Use
	NDE	UInt32	Number of delta entries	4
	Length	UInt32	Length of each delta entry	6
Picture Delta Entry	PosTableIndex	Int8	-ve - reordered	-1
	Slice	UInt8	Slice number in IndexEntry	0
	Element Delta	UInt32	Delta from start of slice to this Element	0
Sound Delta Entry	PosTableIndex	Int8	0= No reordering	0
	Slice	UInt8	Slice number in IndexEntry	1
	Element Delta	UInt32	Delta from start of slice to this Element	0
Sound Delta Entry	PosTableIndex	Int8	0= No reordering	0
	Slice	UInt8	Slice number in IndexEntry	2
	Element Delta	UInt32	Delta from start of slice to this Element	0
Data Delta Entry	PosTableIndex	Int8	0= No reordering	0
	Slice	UInt8	Slice number in IndexEntry	3
	Element Delta	UInt32	Delta from start of slice to this Element	0

Table A.8 – Clip wrapped index entry array example for figure 6

	N	Field Name	Type	Meaning	Use
	1	NIE	UInt32	Number of index entries	=number of frames
	1	Length	UInt32	Length of each index array entry	23 in this example
One Index Entry for every frame	N I E	Temporal Offset	Int8	Offset in edit units from Display Order to Coded Order	According to picture type
		Key Frame Offset	Int8	Offset in edit units to previous Key Frame. The value is zero if this is a Key frame.	According to picture type
		Flags	EditUnitFlag	Flags for this Edit Unit	According to picture type
		Stream Offset	UInt64	Offset in bytes from the first KLV element in this Edit Unit within the Essence Container Stream	Offset from the first byte of the value of the MPEG KLV for the first frame to the first byte of the stored data for this MPEG frame in Figure 6
		SliceOffset	UInt32	The offset in bytes from the Stream Offset to the start of slice 1	Offset from the first byte of the stored data for this MPEG frame to the first byte of the corresponding audio sample in the first Sound Element in Figure 6
		SliceOffset	UInt32	The offset in bytes from the Stream Offset to the start of slice 2	Offset from the first byte of the stored data for this MPEG frame to the first byte of the corresponding audio sample in the second Sound Element in Figure 6
		SliceOffset	UInt32	The offset in bytes from the Stream Offset to the start of slice 3	Offset from the first byte of the stored data for this MPEG frame to the first byte of the corresponding audio sample in the Data Element in Figure 6
		PosTable	NPE *Rational	The fractional position offset from the start of the Content Package to the synchronized sample in the Content Package	In this example, there is only a single Content Package and the audio and video start synchronously. NPE is zero so there is no data here

A.5 Indexing of fill items

Indexing of fill should be performed in the same way as any other variable length element (whose length may sometimes / often be zero). Inspection of the element key when the index entry is non-zero will reveal a fill key.

Annex B (informative)
Identifying MPEG picture types

This section is intended to give a mechanism by which the picture type of an MPEG picture can be identified. This has been split into two cases: in-stream for frame wrapping, and index table based for random access.

B.1 Identifying picture type when frame wrapping is used

Figures 3 and 4 show how an MPEG access unit can be frame wrapped. The bytes following the key will be the start of an access unit. For a video elementary stream, this will be either a sequence_start_code, a GOP_start_code or a picture_start_code. For a PES stream, this will be a PES header. In all these cases, there is a simple algorithm which will allow the picture type to be determined as follows:

Table B.1 – MPEG picture type determination

Sequence header	This must be skipped, along with its sequence_header_extensions and look for picture_start_code
Group of Pictures Header	This must be skipped (59 bits) and look for picture_start_code
Picture header	This contains the picture_coding_type (3 bits offset 42 bits from start of header)
PES Header	Skip the PES header and find a picture_start_code

B.2 Using index tables to identify frame type

When MPEG essence is in an external file, or random access to long GOP material is required, it is often useful to be able to know the MPEG picture type so that sufficient frames can be fetched to ensure an MPEG key frame and all the key frames are available for decoding any frame within the MPEG GOP sequence. The index tables provide enough information for this to be calculated.

The goal is to be able to re-create the GOP sequence based on the structural metadata within the file. This example would be simplified if the MPEG descriptor indicated that the number of B frames was constant, and that all the GOPs were identical. We will not make this assumption here.

In table B.2, a partial IndexEntry table is created to show which parameters can be used for picture identification.

Table B.2 – Index table entry properties to identify picture type

Parameter	Type	Meaning	Use
Temporal Offset	Int8	Offset in edit units from Display Order to Coded Order	See A.2
Key Frame Offset	Int8	Offset in edit units to previous Key Frame. The value is zero if this is a Key frame.	See A.2
Flags	EditUnitFlag	Flags for this Edit Unit Bit 7: Random Access Bit 6: Sequence Header Bit 5: Forward prediction flag Bit 4: Backward prediction flag Bit 3: Offsets out of range Bit 2: Not used by MPEG Bits 0,1: MPEG Frame type	See A.2

Picture types are easily identified from bit 1 and bit 0 of the flags.

Annex C (informative)
Requirements for mapping MPEG into MXF

In the creation of the MXF MPEG mapping, there were a number of requirements which had to be met. These are listed below.

- Preserve timing in any MPEG system stream.
- Preserve timing when de-multiplexing a system stream and converting to MPEG ES wrapping.
- Preserve the MPEG buffer model so that when the MPEG is removed, the buffer model is preserved *when this is required*.
- Encapsulate MPEG PES without knowledge of the contents.
- Provide a mechanism for key frame identification.
- No change of original MPEG data order, this will facilitate recording an MPEG stream in MXF and playing out an MPEG stream from MXF.
- Small CPU and memory needs for both recording and playing out.
- Covers all possible types of MPEG data: MPEG-2 TS, PS, MPEG-1 system, MPEG-1 and MPEG-2 PES, video ES, audio ES, private streams and other derived format such as DVD-VOB, VCD, etc.
- Facilitate access to individual access units.
- Possibility to wrap the MPEG stream based on main level elements for storage critical environments.

Annex D (informative)
Guidance for mapping new or non-ISO defined essence in MPEG streams into MXF

This standard follows the SMPTE 377M and SMPTE 379M rules for incorporating essence into MXF. This annex gives guidance on how new essence types can use this specification, and where the appropriate information can be found.

Some essence types need more than a `stream_id` in the essence container UL to describe the essence. In fact, the intention of the essence container UL is a "fast fail" mechanism to give you a quick look at what is in the file. Certain ISO-MPEG and `private_stream` data essence types will require much more information for interoperable carriage in MXF. The correct place for this information is the essence descriptor. There may be required information for private streams, such as the `registration_descriptor` which indicates the provenance of private stream types. To quote from the SMPTE-RA website:

"The registration descriptor of MPEG-2 transport is provided by ISO 13818-1 in order to enable users of the standard to unambiguously carry data when its format is not necessarily a recognized international standard. This provision will permit the MPEG-2 transport standard to carry all types of data while providing for a method of unambiguous identification of the characteristics of the underlying private data."

Carriage of this information should be performed by creating a new essence descriptor with each property, new type, new group and new enumerated value or enumerated value extension correctly registered in the appropriate SMPTE registry.

For decoding devices, the reverse is true. The `stream_id` field in the essence container UL property indicates that a variant of the SMPTE381M mapping is in use. The essence descriptor will indicate the precise variant of essence which is mapped. Specifically, the following points should be noted:

GC Element Key

This is defined in section 6.1 (picture), section 6.2 (sound), and section 6.3 (data). The GC element key depends only on the wrapping type.

SMPTE label for essence container identification

This is defined in section 7. The mapping of an MPEG stream is based on the `stream_id` in the transport stream binding; e.g., from FDAM3 of ISO 13818-1.

Stream_id	Type
110x xxxx	MPEG1 audio stream number x xxxx MPEG2 audio stream number x xxxx ISO/IEC 14496-3 audio stream number x xxxx
1110 xxxx	MPEG1 video stream number xxxx MPEG2 video stream number xxxx ITU-T Rec. H.264 | ISO/IEC 14496-10 video stream number xxxx

So byte 15 can be set identically to the setting for MPEG4 video, MPEG2 video or MPEG1 video when FDAM3 is approved.

How do you tell it's H.264 and not MPEG1 or MPEG 2?

The picture essence coding parameter of the generic picture essence descriptor will be set to a value corresponding to H.264 | ISO/IEC 14496-10. This value will be listed in RP224 (SMPTE Labels Registry) when a proponent wishes to map this standard into MXF. The appropriate essence descriptor can be created when a proponent needs this functionality and creates a SMPTE standard for the descriptor.

Annex E (informative)
Bibliography

ANSI/SMPTE 298M-1997, Television — Universal Labels for Unique Identification of Digital Data

SMPTE 312M-2001, Television — Splice Points for MPEG-2 Transport Streams

SMPTE 328M -2000, Television — MPEG-2 Video Elementary Stream Editing Information

SMPTE 382M, Television — Material Exchange Format (MXF) — Mapping AES3 and Broadcast Wave Audio into the MXF Generic Container

SMPTE EG 41-2004, Material Exchange Format (MXF) — Engineering Guideline

Advanced Authoring Format (AAF): http://www.AAFassociation.org

ETSI EN300743 – DVB Subtitling Specification

ISO/IEC 11172-2:1993, Information Technology — Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s — Part 2: Video

ITU-R BS.1196 (1995) (Annex 2), Audio Coding for Digital Terrestrial Television Broadcasting