SMPTE 410-2008

SMPTE STANDARD

Material Exchange Format — Generic Stream Partition

Table of Contents

Approved
July 31, 2008

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in Part XIII of its Administrative Practices. This SMPTE Engineering Document was prepared by Technology Committee W25.

Intellectual Property

At the time of publication no notice had been received by SMPTE claiming patent rights essential to the implementation of this Standard. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

1 Scope

This document defines an extension of the MXF File Format that allows specific classes of data streams to be contained in MXF Body Partitions.

The classes of data streams are either essence that is unevenly distributed along the timeline or large amounts of metadata that cannot suitably be stored in the Header Metadata. Examples for such data streams are time-varying metadata, structured text files and ‘lumpy’ essence, i.e. that is not evenly spread along the timeline.

The Generic Stream Container is not intended to carry metadata that could be placed in MXF Header Metadata or essence or metadata that could suitably be stored in the MXF Generic Container.

This document defines the KLV encoding methods used to carry the data stream.

This document also defines partitions that are used to multiplex Generic Streams into the byte stream of MXF files. These Generic Stream Partitions provide for the carriage of Generic Streams in separate partitions from those partitions carrying Essence Container data or Index Table segments.

Essence and metadata payloads that are carried in Generic Stream Partitions are defined in associated documents. This document defines rules for the documents that specify the Generic Stream Payloads and their application. It also defines the basic linking mechanisms of Generic Streams to the Header Metadata of the MXF file.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:”

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified the order of precedence of the types of normative information in this document shall be as follows. Normative prose shall be the authoritative definition. Tables shall be next, followed by formal languages, then figures, and then any other language forms.

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

SMPTE 336M-2007, Data Encoding Protocol Using Key-Length-Value

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

4 Glossary of Acronyms, Terms and Data Types

The general glossary of acronyms, terms and data types used in the MXF specification is given in SMPTE 377M. They are not repeated here to avoid any divergence of meaning.

AU: Access Unit

Access Unit: The smallest temporal section of a stream of data that is intended to be accessed independently. For video essence, an Access Unit is normally defined to be one complete picture (usually a frame, but occasionally a field). For audio essence, an Access Unit is normally defined to be either a fixed number of uncompressed samples, or a single block of compressed data. (Note: This may also be called an Audio Frame.)

Descriptive Metadata: Metadata strongly referenced, either directly or indirectly, from a DM Framework or a subset of a DM Framework.

Generic Stream: A stream of data bytes with a defined format. Generic streams are linear in nature, but may be divided into structured items or even temporally into Access Units

Generic Stream Data: The data bytes within a generic stream partition. This includes all bytes from the start of the first key after the partition pack, excluding the filler immediately following the partition pack if one exists, until the last byte before the start of the following partition pack key.

Generic Stream Partition: The partition defined by this document, including all bytes from the start of the partition pack key to the byte before the start of the following partition pack key.

Generic Stream Payload: Those bytes of Generic Stream Data that constitute the Generic Stream. If the Generic Stream is Intrinsically KLV Wrapped (see Section 6.3.1), all keys and length fields as well as any KLV Fill items are regarded as part of the payload, otherwise these are excluded.

Lumpy Essence: Essence that is not continuous along a timeline. For example pre-rendered subtitles may be stored as still images that are associated with particular points along the timeline.

Payload: Used to refer to the Generic Stream Payload.

SID: Stream ID.

TLV: Tag-Length-Value local set coding as used inside SMPTE 336M KLV local sets.

UL: SMPTE 298M Universal Label with a value given in the Registry defined by SMPTE 400M (RP 224)

5 Introduction (Informative)

The MXF Format specification defines partitions that can contain Header Metadata repetitions, Essence Container data, and Index Table segments either in separate partitions or within the same partition. However, there is no provision in SMPTE 377M for carrying large streams of data other than continuous essence that is regularly distributed along the timeline. This document defines the Generic Stream Partition for carrying generic data streams within the MXF body.

From the point of view of an MXF encoder or decoder, a Generic Stream may be regarded simply as an unstructured sequence of data bytes, or a sequence of multi-byte items. This document defines how this data is wrapped in KLV packets, how the Generic Stream Data is multiplexed into the MXF byte stream, and includes information that allows some manipulation of these streams independent of the semantics of the particular Generic Stream Payload. Further details of each Generic Stream are given by a mapping specification, described in Section 10, which defines the wrapping schemes to be used and the format of the payload data.

For an MXF application that is programmed with knowledge of a particular Generic Stream payload mapping, the payload can be encoded, decoded and manipulated as required.

In some MXF decoder applications, the precise nature of the stream data will be unknown or “dark”. It is important that these applications be able to extract the payload from a Generic Stream Partition and forward it to another application or codec device for processing. Similarly, it is important that an MXF encoder be able to receive a Generic Stream Payload and write it to a valid Generic Stream Partition without detailed knowledge of its format; in practice all that is required are the correct values of the bits specified in Table 4 and Table 5.

As much as possible, the application's interaction with the Generic Stream Data needs to be independent of the actual way in which any access units are divided and partitioned on the underlying KLV storage layer.

In an MXF file, a Generic Stream is enclosed in KLV packets, and is then stored in the body of one or more Partitions.

This document defines the following two component parts:

1. Generic Stream encapsulation

2. A Generic Stream Partition Pack which identifies the start of a Generic Stream Partition.

The relationship between these components is illustrated in Figure 1. The linkage between the Header Metadata and the Generic Stream Data is provided by a StreamID and the precise details of this linkage mechanism is defined in the appropriate payload mapping specification.

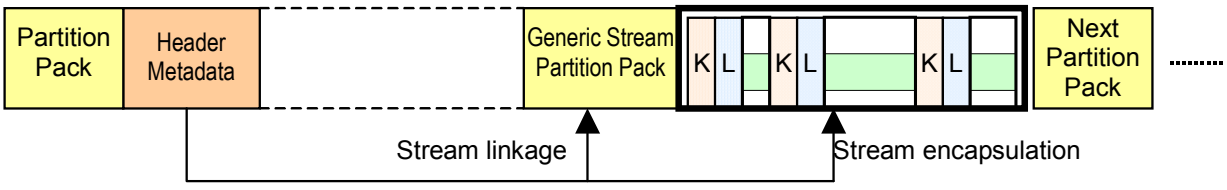

Figure 1 – Relationship between Partition Components

6 Generic Stream Partition Specification

6.1 Overview

The Generic Stream Partition is defined in the following sections; an informative summary is shown below:

1. A Generic Stream Partition is a special kind of Body Partition.
2. A Generic Stream Partition includes one or more KLV-wrapped Generic Data Elements.
3. A Generic Stream Partition contains data from a single Generic Stream.
4. A Generic Stream Partition does not include any essence container data, Header Metadata repetition or Index Table Segments.
5. The order of the Generic Stream Data Elements is significant and it is important that this is not altered by any application treating the Generic Stream Data Elements as dark.
6. The Generic Stream Data Elements may be placed in a single partition, or distributed over two or more partitions.
7. Each unique Generic Stream is assigned a Stream ID value that is unique within the file. Different Generic Streams can thus be uniquely identified even if there are several Generic Streams distributed throughout the file.
8. Generic Stream Partitions are included in the Random Index Pack.

6.2 Generic Stream Partition Pack

The Generic Stream Partition Pack shall comprise a Generic Stream Partition Pack Key, a Length and a Value as defined below. In accordance with SMPTE 377M, the Generic Stream Partition Pack shall be a KLV-coded, fixed-length pack as defined in SMPTE 336M.

6.2.1 Generic Stream Partition Pack Key

The Key of the Generic Stream Partition Pack shall be as defined in Table 1.

Table 1 – Partition Pack Key Value

Byte No.	Description	Value (hex)	Meaning
1~7	See SMPTE 377M, Table 1	—	As defined by MXF File Format Specification
8	Version Number	01h	Registry Version at the point of registration of this Key
9~13	See SMPTE 377M, Table 1	—	As defined by MXF File Format Specification
14	Set / Pack Kind	03h	Body Partition Pack
15	Partition Status	11h	Generic Stream Partition
16	Reserved	00h	

Byte 14 of the SMPTE Key defines that the partition is a Body Partition.

Byte 15 of the SMPTE Key defines that the partition is a Generic Stream Body Partition

Note: The Partition Pack keys defined in SMPTE 377M include flags for open or closed and complete or incomplete, which relate to the Header Metadata in that partition. These flags are not used for Generic Stream Partitions as according to Section 7.1.2 any metadata contained in a Generic Stream Payload is regarded as closed and complete.

6.2.2 Generic Stream Partition Pack Length

In accordance with SMPTE 377M, it is preferred that the Length field of this KLV fixed length pack be BER long-form encoded using four bytes.

MXF decoders shall be SMPTE 336M compliant and must respond to short or long-form BER encoding as received. Decoders shall not rely on a fixed number of bytes for length fields.

6.2.3 Generic Stream Partition Pack Value

The value of the Generic Stream Partition Pack shall be as defined in SMPTE 377M, with the following settings:

- The value of the **HeaderByteCount** shall be set to zero indicating that there is no Header Metadata in this partition.
- The value of the **IndexByteCount** shall be set to zero indicating that there are no Index Table segments in this partition.
- The value of the **IndexSID** shall be set to zero.
- **BodyOffset** specifies the byte offset of the Generic Stream Partition segment in this Partition relative to the start of the Generic Stream with the specified Stream ID. This parameter is used to mark repetitions of the Generic Stream data as indicated in Section 7.3. See Section 6.7 for details of byte counting.
- **BodySID** specifies the Stream ID of the Generic Stream data in this Partition. See Section 6.6 for details of Stream ID.
- **EssenceContainers** is a Batch of ULs that identifies the different Essence Container types used in or referenced by this MXF file. This property is included to provide consistency with other MXF partitions and may be incomplete; therefore its value shall be ignored by decoders.

6.3 Generic Stream Data Element Coding

The Generic Stream Partition shall contain one or more KLV triplets wrapping the data. The precise wrapping scheme will vary depending on the payload data and the requirements of the application.

The Generic Stream data may be distributed in one or more Generic Stream Partitions but each Generic Stream Partition shall only contain data from a single Generic Stream.

Signaling is provided in the Generic Stream Data Element key to distinguish a number of features of the Generic Stream (see Section 6.3.2). Table 2 is an informative summary of the features signaled.

Table 2 – Signaling provided in the Generic Stream Data Element Key

Feature	Options		
Byte Order of Data The byte order of any multi-byte structures within the data wrapped in this element. Where the data consists entirely of single-byte items it is signaled as big-endian.	Little-endian	Big-endian	Unknown
KLV Type If the Generic Stream Payload is itself a valid SMPTE-336M KLV stream it can be classed as *intrinsic* and the data can be stored in the Generic Stream Partition without further wrapping. All other data streams need to be wrapped in KLV triplets before storing. A stream cannot be classed as intrinsic if any of the KLV keys in that stream match an MXF partition pack key.	Intrinsic	Wrapped	
Data Wrapped by Access Unit Is the stream divided into Access Units AND wrapped such that the first byte of each KLV value is the first byte of an Access Unit?	Yes	No	
Multi-KLV Is every Access Unit wrapped in exactly one KLV?	Yes	No	
Wrapping Synchronized to Essence If the essence is wrapped such that each element KLV is synchronized with exactly one KLV of an associated frame-wrapped Essence Container it is signaled as “Frame”, otherwise it is signaled as “Other”	Frame	Other	

6.3.1 Generic Stream Coding of Existing KLVs

When the Generic Stream Payload is a valid SMPTE 336M KLV wrapped stream, such as the example called Type A in Annex A, the KL of the KLV coding is an intrinsic part of the data. In this case the key and length will be defined by the underlying data type.

If an intrinsically KLV-Wrapped payload may contain keys that match any MXF Partition Pack key it shall not be written in a Generic Stream Partition without further wrapping in KLVs using the key defined in Section 6.3.2.

Note: This prevents MXF decoders reading the Payload KLV and treating it as an actual partition pack of the MXF file.

6.3.2 Default Generic Stream Data Element Key

For all other cases of Generic Stream Data, the following key shall be used. This key includes a signal bit for Intrinsic Wrapping; if this bit is set the key and length shall be treated as part of the Generic Stream Payload. This means that an MXF decoder that normally unwraps each KLV before passing the payload data to another device or codec application shall pass on the complete KLVs rather than just their values. The default value of the Generic Stream Data Element Key for KLV wrapping of the Generic Stream is defined in Table 3.

Table 3 – Generic Stream Data Element Key Value

Byte No.	Description	Value (hex)	Meaning
1	Object Identifier	06h	
2	Label size	0Eh	
3	Designator	2Bh	ISO, ORG
4	Designator	34h	SMPTE
5	Registry Category Designator	01h	Dictionaries
6	Registry Designator	01h	Metadata Dictionaries
7	Structure	01h	Dictionary Structure (SMPTE RP 210)
8	Version Number	0Ch	Version of the registry in which this key was registered
9	Item Designator	0Dh	Organizationally registered
10	Organization	01h	AAF Association
11	Application	05h	Generic Stream Keys
12	Data signaling	Table 4	Bit pattern signaling data arrangement
13	Wrapping signaling	Table 5	Bit pattern signaling wrapping strategy
14	Reserved	00	
15	Reserved	00	
16	Reserved	00	

6.3.2.1 Data Arrangement

This is a bit pattern signaling the arrangement of the underlying data as given in Table 4.

Table 4 – Data Arrangement Byte 12

Bit	Value	Meaning
0	1	Marker bit to prevent termination of key
1	1	The KLV is an intrinsic part of the underlying data stream and shall be kept with the payload data (e.g. Type ‘A’ data)
	0	The KLV is not a part of the data and should be removed before processing
3,2	00	Reserved
	01	the Generic Stream is Little-Endian
	10	the Generic Stream is Big-Endian or is a byte-stream
	11	the Endian-ness of the Generic Stream is unknown
7-4	xxxx	Reserved – set to 0

Note: Type ‘A’ data is discussed in Annex A.

6.3.2.2 Wrapping Signaling

This is a bit pattern signalling the wrapping strategy applied to the underlying data as given in Table 2. If Bit 1 is set to zero, then Bit 2 shall be set to zero.

Table 5 – Wrapping Signaling Byte 13

Bit	value	Meaning
0	1	Marker bit to prevent termination of key
1	1 0	The first byte of the Value of each KLV triplet starts an access unit of the Generic Stream Data (e.g. Types A, B1, B2, B3) The first byte of a KLV triplet has no special importance (e.g. Types C1, C2)
2	1 0	The Generic Stream is divided into Access Units and there is exactly one KLV enclosing each Generic Stream Data Access Unit (e.g. Types 'A', B1, B2) The above condition is not true
3	1 0	The Generic Stream Data is Frame Wrapped **and** each KLV is synchronized to the Frame Wrapped Essence The above condition is not true
7-4	xxxx	Reserved – set to 0

Note: Types A, B1, B2 etc are discussed in Annex A.

6.3.3 Generic Stream Data Element Length

The length field should be 4-byte BER long-form encoded (i.e., 83h.xx.yy.zz), unless the length value exceeds 00FFFFFFh; in this event, 8-byte BER long-form encoding should be used (i.e., 87h.aa.bb.cc.dd.ee.ff.gg).

MXF decoders shall be SMPTE 336M compliant and must respond to short or long-form BER encoding as received. Decoders shall not rely on a fixed number of bytes for length fields.

6.3.4 Generic Stream Data Element Value

The value shall be a portion or the entirety of the Generic Stream Payload.

6.4 Use of KLV Fill

Each Generic Stream KLV Triplet may be followed by a KLV Fill Item for the purpose of KAG alignment. There may also be a KLV Fill item between the Generic Stream Partition Pack and the first Generic Stream Data Element

See Section 6.7 for details of how Generic Stream Data bytes are counted when fill packets are used.

6.5 Use of the RIP

Generic Stream Partitions shall be included in the RIP, if one exists in the file.

6.6 Use of Stream ID

The following rules and recommendations apply to the use of Stream IDs used by encoders writing MXF files containing Generic Stream Partitions:

1. TheStream ID values used for identifying Generic Stream Partitions in any file shall not be used for identifying Index Streams or Essence Containers in the same file.

2. Each and every partition of an MXF file should be associated with a single Stream ID value; i.e., partitions should contain only EssenceContainer data, IndexTable segments, or Generic Stream data. Combinations of these streams in a single partition are not recommended.

3. Every Generic Stream in an MXF file shall be linked from the Header Metadata using the value of the stream's SID. See Section 7.1 for details of this linkage.

6.7 Byte Counting within Generic Stream Data

Where it is necessary to calculate a byte offset within Generic Stream Data, such as for the value of BodyOffset in the Generic Stream Partition Pack, the byte counting method is analogous to the method of counting bytes in Essence Container data. The method is as follows:

- The bytes of the Generic Stream Partition Pack shall not be counted as part of the Generic Stream Data.
- The bytes of any KLV Fill item immediately following a Generic Stream Partition Pack shall not be counted.
- All bytes of any KLVs in the Generic Stream Partition other than the two listed above shall be counted (this includes the bytes that make up the KLV key and length).
- If the last KLV in a Generic Stream Partition is a KLV Fill item, its bytes shall be counted.
- MXF decoders should use the value of BodyOffset to reset any internal stream offset count for the current Generic Stream to this value when reading the Partition Pack.

The portion of a Generic Stream Partition that is included in byte counts is shown in Figure 2.

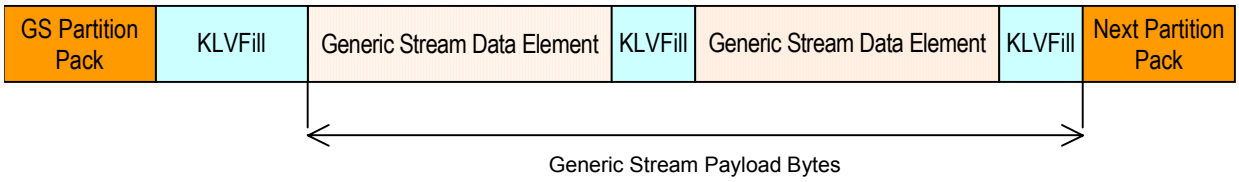

Figure 2 – A Generic Stream Partition with Leading, Internal and Trailing Filler

6.8 Blind Repartitioning of Generic Streams

Any application treating the Generic Stream data as dark shall not alter the KLV coding of the Payload. Blind repartitioning that preserves KLV Fill Items shall be permitted. Such repartitioning shall not add, remove, move or change the size of any KLV Fill items in the Generic Stream Payload Bytes.

Note: The requirement to preserve KLV Fill items exactly as they are is so that any indexing of byte-offsets into a Generic Stream Payload shall still index the exact same part of the payload after repartitioning.

7 Generic Stream Payload Mapping Specifications

The format of each Generic Stream Payload shall be defined by a mapping specification. This specification should be a SMPTE Engineering Document.

Each payload mapping specification shall define the format and semantics of the Payload bytes. This definition may be included in the mapping specification, or in another document referenced from the mapping specification.

Each payload shall be identified by a UL in either the EssenceContainers batch, the DMSchemes batch or the GenericPayloads batch (see Section 7.1.2). Each payload mapping specification shall specify the label to use, and which batch shall hold that label.

Where a payload mapping specification defines a group of payload types that are used together, a different UL should be defined for each stream type.

The payload mapping specification shall clearly state if the stream is Intrinsically KLV-wrapped in such a way that no further wrapping needs to be performed for use in a Generic Stream Partition. For all other cases the signaling values for bytes 12 and 13 of the Generic Stream Element Keys (Table 4 and Table 5) shall be specified. These may be specific values that are always used, or a selection of values, along with a description of when each value shall be used.

A payload mapping specification may also add restrictions to the use of Generic Stream repetitions defined in Section 7.3. For example repetitions could be forbidden for that stream, or repetitions may be permitted, but without additions.

7.1 Stream ID Linkage Mechanism

Each payload mapping specification shall define the mechanism used to link from the Header Metadata to the Generic Stream. In all cases the basic link shall be made by use of the Generic Stream ID. Sub-streams or other sub-sections within the Generic Container Payload may be identified by properties within the header metadata in addition to the Stream ID linkage; in these cases the mechanism shall be defined in the mapping specification.

The linkage by Stream ID is regarded as an ownership of the Generic Stream by the header metadata property making the link and is analogous to a strong reference. There shall only be one such link from any copy of the header metadata to each Generic Stream.

Any closed header metadata instance shall link by Stream ID every Generic Stream that exists in the file.

A payload mapping specification may define a secondary linkage mechanism to refer to a Generic Stream without conferring ownership — such properties are analogous to weak references.

7.1.1 Generic Streams with Essence Payload

If the Generic Stream Payload is essence, it shall be linked to one Top-Level File Package via the Essence Container Data set BodySID property. This Top-Level File Package shall describe the Generic Stream Payload. The mapping specification shall specify a label to be added to the Essence Containers batch in the Preface, and in the copy of that property in each partition pack. The common framework for essence payload identification labels is defined in Section 7.3.

Where a Generic Stream Payload contains more than one essence stream, these streams shall each be classified as either active or passive. Each active essence stream within a payload shall be described by a separate Essence Descriptor in the Top-Level File Package and linked from a separate essence track using a different value of Track Number. Passive streams shall not be described by Essence Descriptors and shall not be linked from essence tracks.

Note: Passive essence streams are intended to be ignored by decoders. They cannot be referenced from Material Package tracks. An example might be where picture essence is captured with associated data essence, yet only the picture essence is required; here the data essence would be passive and so would not be described by an Essence Descriptor or an essence track.

The value of the Track Number property of the essence tracks used to link to the Generic Stream Payload shall be specified in the payload mapping specification.

Note: MXF Decoders cannot make assumptions about the meaning of Track Number for Generic Stream Payloads that are not specifically supported by that decoder.

7.1.2 Generic Streams with Metadata Payload

Generic Stream Data that consists of Metadata shall be regarded as closed and complete.

The property that links from the Header Metadata to a metadata Generic Stream shall be specified in the associated payload mapping specification. This linkage shall be made by Stream ID.

If the Generic Stream Payload is Descriptive Metadata, it shall be identified with a UL in the DMSchemes batch of the Preface set. All other metadata payloads shall be identified by a UL in the GenericPayloads batch of the Preface set. The appropriate label shall be given by the payload mapping specification.
The GenericPayloads batch is shown in Table 6.

Table 6 – GenericPayloads Property of the Preface Set

Item Name	Type	Len	Local Tag	Item UL	Req ?	Meaning	Default
GenericPayloads	Batch of UL (GenericPayloads)	8+ 16n	dyn	06.0E.2b.34.01.01.01.0c 06.01.01.04.03.06.00.00	Opt	A Batch of Universal Labels of all the Generic Payloads used in this file. [**RP 210** An unordered Batch of Universal Labels of all the generic payloads used in this file.]	

The common framework for metadata payload identification labels for use in the GenericPayloads batch is defined in Section 7.4.

7.2 Indexing Generic Stream Data

Some Generic Stream Payload data can vary along the timeline and may have structural metadata associated with it which describes temporal offsets. In order to identify which portion of the data is to be used, an indexing mechanism must be provided to associate the temporal offset with a byte offset within the stream.

Each Generic Stream payload mapping specification shall specify whether an index table is forbidden, optional or mandatory for the associated Generic Streams. It shall also specify the format of the index table. The index table mechanism described in SMPTE 377M shall be used where suitable.

As Generic Streams are generally not continuous, any SMPTE 377M index table is likely to be sparse; i.e. it may be a sequence of Index Segments each including as little as one entry.

If the Generic Stream is indexed using SMPTE 377M index tables, the link between Generic Stream ID and IndexSID shall be given via the Essence Container Data set. For all other indexed Generic Streams, the payload mapping specification shall define the linkage mechanism.

Some Generic Stream Payloads may be linked to a timeline by information within the Payload bytes, for example there may be timecodes embedded in the stream. In these cases the mechanisms shall be documented in the payload mapping specification. This shall not be regarded as indexing for the purpose of this standard.

7.3 SMPTE Label for Essence Identification

The common framework for a SMPTE Label that identifies the essence payload shall be as defined in Table 7.

Table 7 – Specification of the Essence Identification Label

Byte No.	Description	Value (hex)	Meaning
1	Object Identifier	06h	
2	Label size	0Eh	
3	Designator	2Bh	ISO, ORG
4	Designator	34h	SMPTE
5	Registry Category Designator	04h	Labels
6	Registry Designator	01h	Labels Registry
7	Structure Designator	01h	Labels Structure
8	Version Number	vvh	Version of the Registry
9	Item Designator	0Dh	Organizationally Registered
10	Organization	01h	AAF Association
11	Application	03h	Essence Containers
12	Structure Version	01h	Version 1
13	Essence Container Kind	03h	MXF Generic Stream
14	Payload Kind	xxh	Defines the kind of payload
15~16	Locally defined	yyh	Defined by the payload specification

Notes:

1. Byte 8 will be defined by the SMPTE Labels Register when each essence label is registered. See the appropriate payload specification.

2. Byte 14 is defined by the appropriate payload mapping document and will have a value in the range '01'h – '7F'h.

This SMPTE Label is the individual ‘Essence Container’ property used in the Partition Pack, in the Preface Set and in the appropriate File Descriptor.

7.4 SMPTE Label for Metadata Identification

The common framework for a SMPTE Label that identifies the metadata payload for use in the GenericPayloads batch shall be as defined in Table 8.

Table 8 – Specification of the Metadata Identification Label

Byte No.	Description	Value (hex)	Meaning
1	Object Identifier	06h	
2	Label size	0Eh	
3	Designator	2Bh	ISO, ORG
4	Designator	34h	SMPTE
5	Registry Category Designator	04h	Labels
6	Registry Designator	01h	Labels Registry
7	Structure Designator	01h	Labels Structure
8	Version Number	vvh	Version of the Registry
9	Item Designator	0Dh	Organizationally Registered
10	Organization	01h	AAF Association
11	Application	05h	Metadata Stream Application
12	Structure Version	01h	Version 1
13	Payload Kind	xxh	Defines the payload kind
14~16	Locally defined	yyh	Defined by the payload specification

Notes:

1. Byte 8 will be defined by the SMPTE Labels Register when each metadata label is registered. See the appropriate payload specification.

2. Byte 13 is defined by the appropriate payload mapping document and will have a value in the range '01'h - '7F'h.

8 Repetition

The Generic Stream Payload may be repeated in a file, however, the following rules shall be respected:

1. Each repetition shall start with a new Generic Stream Partition Pack.

2. At the start of each and every repetition, the BodyOffset property shall be set to 0.

3. The Generic Stream Data shall be byte-for-byte identical between all copies (including any filler), with the exception that new data may be added to the end of repetitions that occur later in the file if the payload mapping specification specifically allows.

A Generic Stream payload mapping specification may impose further restrictions on the use of repetitions of the Generic Stream defined by that document.

8.1 Repetition Example (Informative)

The following example shows a Generic Stream whose payload is an XML document and which is fragmented through the file. This may be a common situation where the contents of the payload are being constructed from same source that is supplying the essence.

First GS Partition:

```
<OuterTag>
      <Item1 attribute=”one”>
            <SubItem1/>
            <SubItem2/>
      </Item1>
```

Second GS Partition:

```
      <Item2 attribute=”two”>
            <SubItem1/>
      </Item2>
```

Third GS Partition:

```
      <Item3 attribute=”three”>
            <SubItem2/>
            <SubItem3 subAttr=”sub”/>
      </Item3>
```

Fourth GS Partition:

```
</OuterTag>
```

Notice that until the fourth partition is read that the XML is not ‘valid’ as OuterTag has not been closed. This means that a decoder may need to be written with care as a standard XML parser may not be able to extract useful information until the entire file has been read.

It may be useful for the whole XML document to be available in one partition. In this case a fifth partition could repeat the entire stream. Ideally this repetition would be located just before the footer.

Fifth GS Partition:

```
<OuterTag>
      <Item1 attribute=”one”>
            <SubItem1/>
            <SubItem2/>
      </Item1>
      <Item2 attribute=”two”>
            <SubItem1/>
      </Item2>
      <Item3 attribute=”three”>
            <SubItem2/>
            <SubItem3 subAttr=”sub”/>
      </Item3>
</OuterTag>
```

The Partition Pack of this fifth partition would have its BodyOffset property equal to zero to flag the start of a repetition.

Note that the KLVs holding each XML fragment are maintained so that the byte offsets within the stream are kept constant and the same indexing can be used for either copy.

If the same example stream had been written with KLVFill items following each fragment, to align the following Partition Pack with a KAG boundary, the result would be as follows:

First GS Partition:

```
<OuterTag>
      <Item1 attribute="one">
            <SubItem1/>
            <SubItem2/>
      </Item1>
KLVFill
```

Second GS Partition:

```
      <Item2 attribute="two">
            <SubItem1/>
      </Item2>
KLVFill
```

Third GS Partition:

```
      <Item3 attribute="three">
            <SubItem2/>
            <SubItem3 subAttr="sub"/>
      </Item3>
KLVFill
```

Fourth GS Partition:

```
</OuterTag>
KLVFill
```

Fifth GS Partition:

```
<OuterTag>
      <Item1 attribute="one">
            <SubItem1/>
            <SubItem2/>
      </Item1>
KLVFill
      <Item2 attribute="two">
            <SubItem1/>
      </Item2>
KLVFill
      <Item3 attribute="three">
            <SubItem2/>
            <SubItem3 subAttr="sub"/>
      </Item3>
KLVFill
</OuterTag>
KLVFill
```

Annex A (Informative)
Discussion of Stream Types

This section describes a number of generic stream types and recommended behavior of an encoder when writing these data streams to one or more Generic Stream Partition. The following text is written on the assumption that an MXF encoding application receives the stream data and passes formatted data to the Application Program Interface (API) of an MXF library package, however the basic principles remain valid for different MXF encoder architectures.

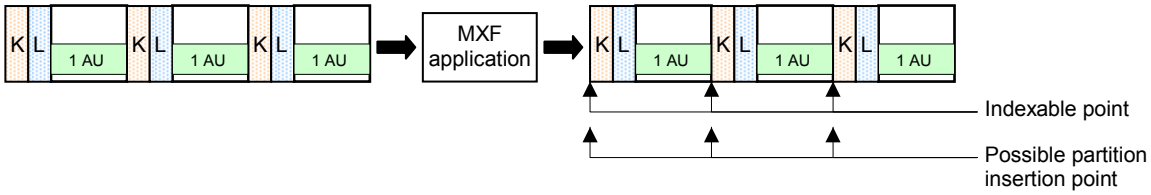

Figure A.1 – Type A — Data which is already wrapped as KLV

When the data is made up of identifiable access units, and when the access units are already encapsulated in KLV (for example, by a codec), no further packetization is required. The encoder application writes individual KLV packets to the Stream API. When reading such a file the Stream API returns unmodified KLV packets to the decoder application. Per MXF rules, the access units are indexed at the start of each K. The Stream may be divided into Partitions at the start of any K; and partitions may be recombined and split at will (called “blind repartitioning”). Note that for file integrity, the Stream API must verify the construction of the KLV packets.

When the data is made up of identifiable access units, but is not inherently packetized, there are the following choices:

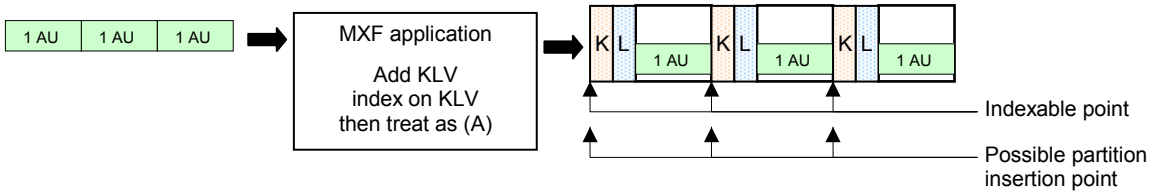

Figure A.2 – Type B1 — Data has access units — layer as though (A)

The application or codec extracts access units, encapsulates them in KLV, and writes them to the Stream API. Indexes must be created on the packetized data. This case allows indexing and arbitrary partitioning in the same way as (A) above. An encoder puts the new Keys in the stream. When reading such a file the Stream API returns unmodified KLV packets to the decoder application, which will remove the keys and lengths before passing the data to the generic stream codec.

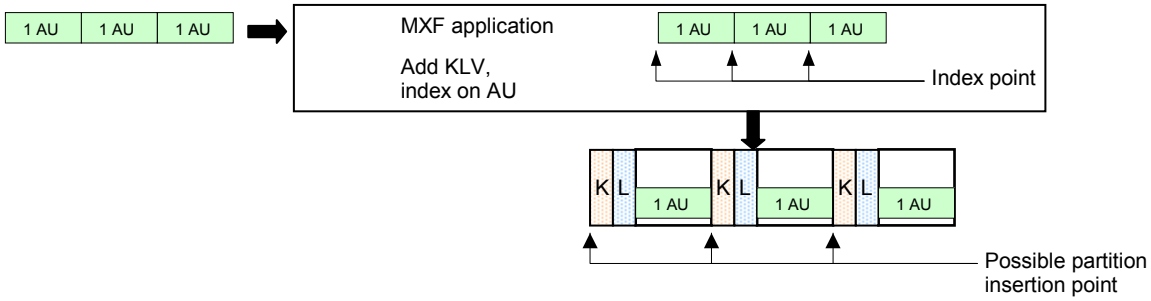

Figure A.3 – Type B2 — Data has access units — indexed as though continuous stream

The application addresses the access units and writes them one by one to the Stream API. The Stream API creates the KLV packets before further processing. When reading such a file the Stream API removes the KL before returning access units to the decoder application. Indexing is possible in the same way as (A) and (B1) above; but in this case, the index offsets are calculated for the unwrapped data stream, possibly by the Stream API. Note that this case is less likely to occur than B1 or B3.

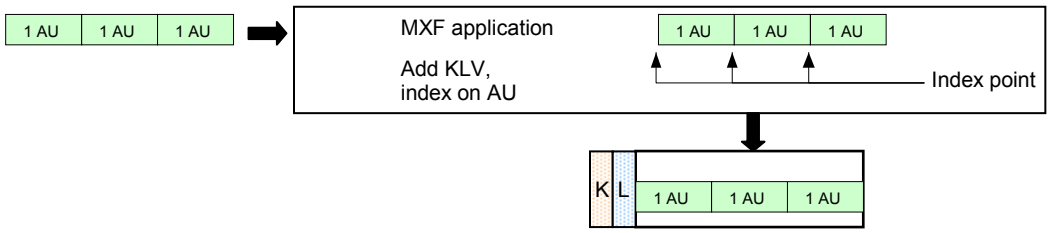

Figure A.4 – Type B3 — Data has access units — indexed as though continuous stream "clip" wrap

The application addresses the access units, indexes them and writes them to the Stream API. The Stream API places them all in a single Partition enclosed in a single KLV packet. The index offsets are calculated for the unwrapped data stream, but in this case the indexing is most likely performed by the encoder application.

There is one further possible case, similar to B3, where the application addresses the access units, indexes them and writes them one by one to the Stream API. The Stream API places them in partitions as desired.

In this case, since the Stream API must create a KLV for each Partition, there is not a clear relationship between the index table entries and the body offset; the algorithm for translating an index entry is not simple and there is no freedom to blindly repartition the Stream.

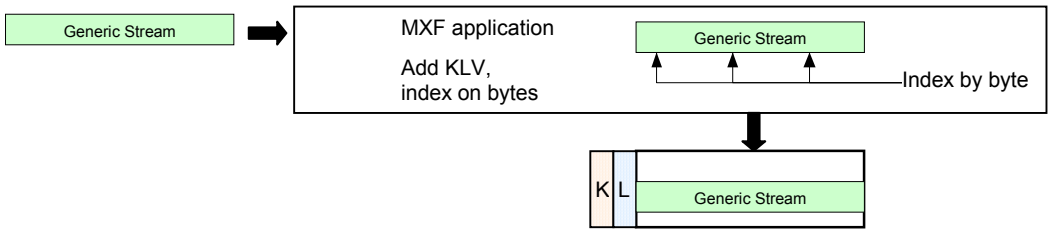

Figure A.5 – Type C1 — Data has no identifiable access units — indexed as though continuous stream "clip" wrap

When there are no identifiable access units in the data, the application writes the data items sequentially in arbitrary numbers to the Stream API. The Stream API places them in a single Partition. Random access is easy.

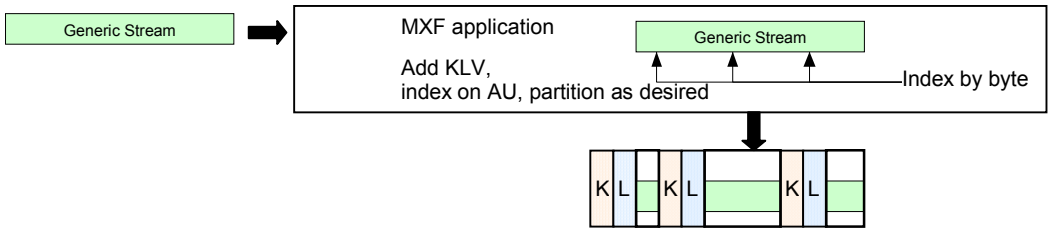

Figure A.6 – Type C2 — Data has no identifiable access units — partition as desired

When there are no identifiable access units in the data, the application writes the data items sequentially in arbitrary numbers to the Stream API. The Stream API places them in partitions as desired. Sequential access to the data is easy. Random access is not easy. Blind repartitioning is not possible.

Annex B (Informative)
Application in an MXF File

This section illustrates example placements of Generic Stream Partitions in an MXF file. In these examples it can be assumed that the Generic Stream consists of metadata that arrives with, or is extracted from, the essence data stream.

According to SMPTE 377M, Body Partitions shall be placed in the MXF File Body. This means that a Body Partition must not precede the Header Partition and must not follow the Footer Partition.

Within the MXF File Body, Body Partitions may generally be placed in any sequence with the caveat that any segmented Essence Container, Index Table or Generic Stream shall be presented in time sequence.

B.1 A Simple MXF File with a Non-Segmented Essence Container

The Essence Container is in the Header Partition, hence a Generic Stream Partition can only be placed between the end of the Essence Container and the Footer Partition as illustrated in Figure B.1.

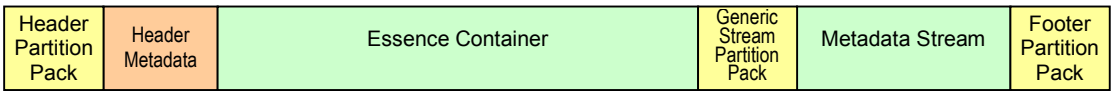

Figure B.1 – Example Placement of a Generic Stream Partition in an Unsegmented MXF File

B.2 A Simple MXF File with a Segmented Essence Container

The Essence Container is segmented within the MXF File Body and each Generic Stream Partition is placed at the end of the Essence Container segment as illustrated in Figure B.2. Each Generic Stream segment has the stream data from the immediately preceding Essence Container segment.

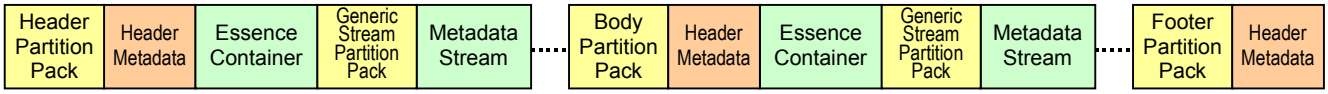

Figure B.2 – Example Placement of Generic Stream Partitions in a Segmented MXF File

B.3 A Multiplexed MXF File with Segmented Essence Containers

The Essence Containers are segmented within the MXF File Body and each Generic Stream Partition is placed at the end of an Essence Container segment as illustrated in Figure B.3. Each Generic Stream segment has the stream data from the immediately preceding Essence Container segment.

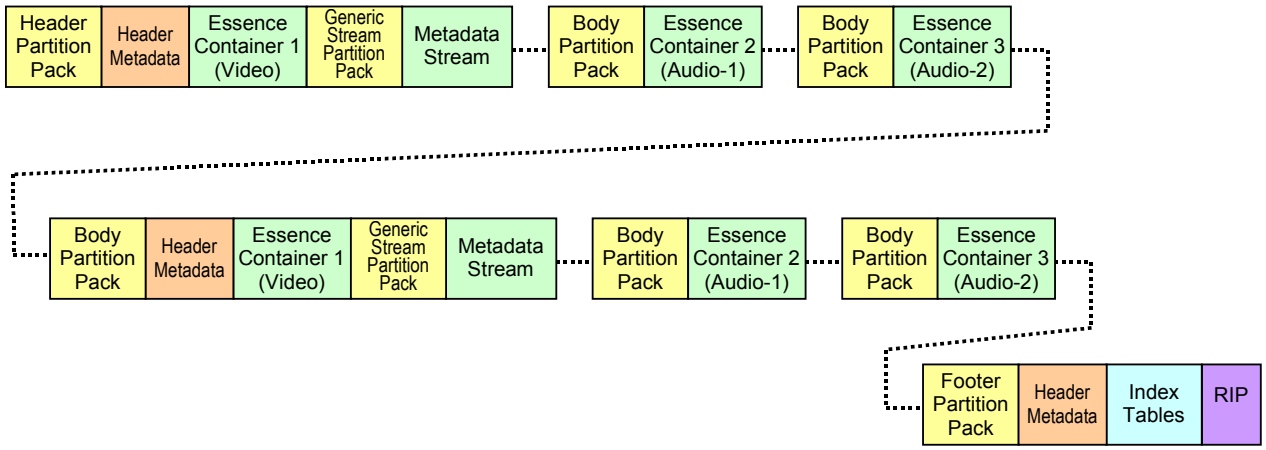

Figure B.3 – Example Placement of Data Partitions in a Multiplexed and Segmented MXF File

Annex C (Informative)
Bibliography

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

SMPTE 335M-2001, Television — Metadata Dictionary Structure

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

SMPTE 395M-2003, Television — Metadata Groups Registry Structure

SMPTE 400M-2004, Television — SMPTE Labels Structure

SMPTE RP 210, Metadata Dictionary Registry of Metadata Element Descriptions

SMPTE RP 224, SMPTE Labels Register

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

AAF Association: www.aafassociation.org