

SMPTE ENGINEERING GUIDELINE

Material Exchange Format (MXF) — Engineering Guideline (Informative)



Page 1 of 47 pages

Warning: This Engineering Guideline is purely informative and meant to provide tutorial information to the industry about a Standard or Recommended Practice. Do not rely on this for interoperability, but rather the Standards and Recommended Practices referenced herein.

Table of Contents	Page
Foreword.....	2
Introduction	2
1 Scope.....	4
2 MXF Document Suite Structure	4
2.1 About this Document	5
3 MXF Concepts	5
3.1 MXF Background	6
3.2 MXF Design Criteria	6
4 MXF File Structure	7
4.1 What are the File Header, the File Body and the File Footer?	7
4.2 MXF Partitions	7
4.3 KLV Coding in MXF	9
4.4 MXF Metadata	11
4.5 MXF Essence.....	13
4.6 MXF File Tools.....	14
5 MXF Interoperability.....	16
5.1 Working with Stream Interfaces.....	16
5.2 MXF Interoperability with Other Files	17
6 A Guide to the Wording of the MXF Dstandards	17
6.1 Conformance Text	17
6.2 Functional Descriptions — Encoder Required, etc.....	18
6.3 Encoding and Decoding Concepts	18
6.4 Essential Terms	19
7 Metadata Classifications and Placement.....	21
7.1 Embedded Metadata Location.....	22
7.2 Linked Metadata Location.....	22
7.3 Attached Metadata Location	23
7.4 Server Metadata Location.....	23

8	MXF Header Metadata Concepts.....	23
8.1	What are Objects?	23
8.2	What is a Class?	24
8.3	What is a Package?	24
8.4	How are Packages Coded?	26
8.5	Using ULs and UUIDs	29
8.6	Coding Objects as Sets.....	31
8.7	Using Text	32
8.8	Tracking Changes with Generation Numbers	32
9	MXF in Detail.....	33
9.1	General Overview	33
9.2	Content Package Model	34
9.3	MXF Essence Containers	35
9.4	Mapping Essence and Metadata into the MXF Generic Container.....	39
9.5	How MXF Header Metadata Relates to the Essence Container.....	40
9.6	Operational Patterns	42
9.7	MXF Decoder Design.....	45
Annex A	Bibliography — Further Reading	47

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.

SMPTE EG 377-3 was prepared by Technology Committee 31FS.

Introduction

The Material Exchange Format (MXF) was developed by the industry to provide a means of exchanging video, audio, data essence and metadata content in a unified data format. The MXF file structure provides a means of exchanging metadata using a formal data structure that permits interoperability together with an essence container that can accommodate many different essence formats both compressed and uncompressed and including video streams, individual pictures, sound and essence data (such as sub-titles, VBI data etc.).

The starting point of the work was the report from the Task Force for Harmonized Standards for the Exchange of Program Material as Bitstreams as identified in the Bibliography. The work of this report was used as the basis for an exchange file format by the Pro-MPEG Forum which acted as a catalyst to encourage industry-wide participation to create a unified file format that could be used for distribution, exchange and cut-editing. The results of this initial development were then brought to SMPTE to create an internationally recognized sets of standards, recommended practices and engineering guidelines that form the MXF document suite.

The other key aspect of the development of MXF was the adoption of the SMPTE Key-Length-Value approach to coding the whole of the MXF file structure using both individual and group coding for a consistent approach within the whole file format. The evolution of XML coding can be seen as attractive, but in many cases, such coding is not suitable for the high-speed and wide-bandwidth requirements of high definition digital television and digital cinema. Instead, there are methods that can be used to transform the metadata parts of MXF files to XML and vice-versa (though only practical for the metadata part of MXF files).

1 Scope

This guideline gives an introduction to and the background for the Material eXchange Format (MXF). This guideline describes the technology involved in the format, the names of the various elements within the format, and the way in which the format can be used within the real world applications.

Some parts of the descriptions within this guideline are generic to file formats, while other parts are specific to the material exchange format. There are descriptions of the object-oriented technology used within the MXF format, as well as a discussion of the metadata that can be used within the file. There are worked examples within this guideline to guide users and help improve the interoperability of applications using different MXF files.

2 MXF Document Suite Structure

The MXF specification is split into a number of categories in order to create a document structure that allows new applications to be covered in the future. These categories are:

Category 1 – MXF File Format Specification – Normative (SMPTE ST 377-1)

Category 2 – MXF File Format Extensions – Normative (e.g. SMPTE ST 377-2 is the KLV Extension Syntax, KXS)

Category 3 – Engineering Guidelines – Informative (including this document)

Category 4 – Operational Patterns – Normative (e.g., OP1a is SMPTE ST 378)

Category 5 – MXF Descriptive Metadata Schemes – Normative (e.g., DMS-1 is SMPTE ST 380)

Category 6 – Essence Containers – Normative (e.g., the MXF Generic Containers: SMPTE ST 379-1 and SMPTE ST 379-2)

Category 6a – Mapping Essence and Metadata into the Essence Container – Normative (e.g., Mapping MPEG Streams into the Generic Container is SMPTE ST 381 (all parts))

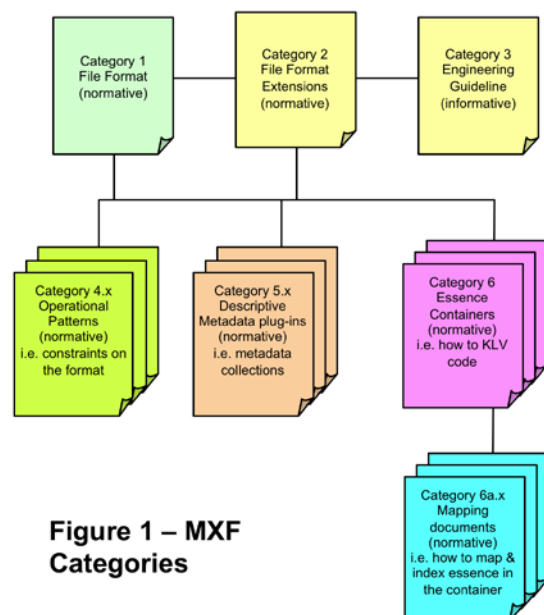


Figure 1 – MXF Categories

When creating an MXF application or system, you need to ensure that you have the latest versions of all of these documents.

Note that in some cases, the MXF specifications have been revised by splitting into multiple parts (e.g. SMPTE ST 381 parts 1~n) or providing different variants (e.g. SMPTE ST 379-1 and SMPTE ST 379-2).

The MXF categories are summarized below.

Category 1 is a normative definition of the format of an MXF file. It defines the toolbox from which different file interchange tools can be chosen to fulfill the requirements of different applications. The MXF file format defines the syntax and semantics of MXF files. Since MXF is an extensible format, new standards can be added that define extensions to the format defined by SMPTE ST 377-1. Category 1 also contains formal definitions of the many acronyms, terms and data types used in the MXF specifications, including this engineering guideline.

Category 2 provides for extensions to the file format to provide tools not present in the basic file format. These extensions provide new tools to extend the capabilities on MXF to provide extra functionality as needed for specialist applications,

Category 3 contains informative documents that help users to understand various aspects of MXF. This document is MXF engineering guideline, which is an introduction to MXF. This document introduces many of the concepts and explains what problems MXF is intended to solve. Part 3 also includes other engineering guidelines, including a descriptive metadata engineering guideline (SMPTE EG 42), which explains the concepts behind the use of descriptive metadata in MXF files. Other engineering guidelines can be added to the MXF document suite as and when there is a requirement.

Category 4 documents describe the operational patterns of the MXF format. In order to create an application to solve a particular interchange problem, some constraints and structural metadata definitions are required before SMPTE ST 377-1 can be used. An operational pattern defines those restrictions of the format that allow interoperability between applications of defined levels of complexity. Applications that use the MXF format need to adhere to one of the operational patterns in order to achieve interchange.

Category 5 documents define MXF descriptive metadata sets that can be plugged in to an MXF file. Different application environments will require different metadata sets to be carried by MXF. These collections of metadata sets are described in the part 5 document(s).

Category 6 documents define the essence container of the MXF format for containing picture and sound essence. There can be limitations to the essence container that are needed for a particular operational pattern. The reader is advised to cross-reference at least categories 1 and 4 both before and during implementation. The MXF generic container is a standardized essence container providing an encapsulation mechanism that allows many formats to be mapped into MXF.

Category 6a comprises a number of documents for mapping many of the essence and metadata formats used in the content creation industry into the defined MXF essence container.

The MXF document suite makes reference to other documents that contain information required for an MXF system. One such document is SMPTE ST 336 that provides the basic coding rules used throughout MXF. Other documents include the SMPTE metadata dictionary, SMPTE RP 210, which contains definitions of metadata parameters and the SMPTE Labels Register, SMPTE RP 224, which contains a list of defined 16-byte labels for use in MXF sets.

SMPTE ST 377-1 provides a full list of all MXF documents for user guidance.

The Bibliography of this document provides a list of additional documents and publications for background reading to aid the understanding of MXF.

2.1 About this Document

The information in this document is ordered to aid readability. Concepts are introduced gradually and repeated in more detail later in the document. This is done to make the document easier to read; however, it does make the document somewhat less useful as a guide. The table of contents [see pages 1 and 2] is provided at the start of the document to allow easy access to each heading within the document.

This document is a guideline and does not specify conformance points. Nor does this document contain conformance terms except where present in a quote from normative documents for the purpose of illustration.

3 MXF Concepts

The concepts in MXF are introduced in a way that gives an overall view of the specification and the concepts embodied within it. Once the introduction is understood, the requirements of the file format are discussed. Some specific words and phrases used in the specification are then defined and finally

the material exchange format is introduced in much more detail. Although this entire document is informative, its aim is to give sufficient information for technical and non-technical readers to have a basic understanding of MXF.

3.1 MXF Background

The MXF specification is designed to interchange multimedia information as a file. The design allows users to take advantage of non-real time transfers and to package together essence and metadata for effective interchange between servers and between businesses. MXF is not a panacea, but is an aid to automation and machine-to-machine communication. It allows essence and metadata transfer without the metadata elements having to be manually re-entered.

The MXF specification is intended to allow the storage and interchange of video, audio, data essence and metadata as a file. MXF files can be efficiently stored on a variety of physical media as well as transported over communications links.

Finally, the MXF specification is intended to be extensible. Considerable effort was put into ensuring that SMPTE ST 377-1 is compression codec independent, resolution independent and can be constrained where needed to suit a large number of application environments. The document structure has been created to allow new applications to extend MXF in a backwards-compatible way.

3.2 MXF Design Criteria

The MXF file format was designed for use in high throughput hardware or software devices. This translates into the need for well-defined design parameters for buffer size, latency, and the need for algorithmic simplicity. MXF is also intended to cover a very large application space, and not all the requirements apply to all the applications.

3.2.1 Representing multi-byte numbers

There are two distinct ways in which multi-byte numbers are stored in computer systems - big-endian and little-endian. Big-endian systems place higher value bytes in the lower value addresses (i.e. MSB first), whereas little-endian systems do the reverse. Because of the two storage formats, MXF defines the storage format as follows:

- MXF header metadata uses the big-endian data format only. The primary reason behind this decision was to address the handling of dark metadata where the endian-ism cannot be known (because the metadata is dark).
- Essence containers use the byte order defined by the individual specifications of each essence container where needed.

3.2.2 Requirements for application diversity

Some applications can require metadata to be processed separately from the essence data. Other applications can require metadata to be stored with the essence data. This requires efficient insertion and extraction of the metadata from the essence data container(s) of the file.

Indexing of the file content is desirable to provide for rapid access to the file contents and this desire is served by the provision of index tables. Some applications prefer such index tables to be accessed separately from the essence; others can require the two to be accessed together. In some cases, the index tables are most naturally stored at the start of the file; however, while recording, the most natural location is at the end of the file. This diversity requires efficient insertion, extraction and relocation of index tables within the file.

4 MXF File Structure

The MXF format follows a common theme of many file formats with the following basic structure:

- A file **header** that provides information about the file as a whole, including labels for the early determination of decoder compliance. The header metadata in the file header is a complex data structure that provides a full description of the contents of the essence container.
- A file **body** that comprises picture, sound and data essence stored in essence containers. Essence containers from different tracks can be interleaved or separated. Section 9.6 on operational patterns goes into more detail on how the essence content is structured.
- A file **footer** that terminates the file. The file footer can include some information not available at the time of writing the file header (such as the duration of the file). In certain specialized operational patterns, the file footer can be omitted.

A simple MXF file is illustrated in Figure 2.

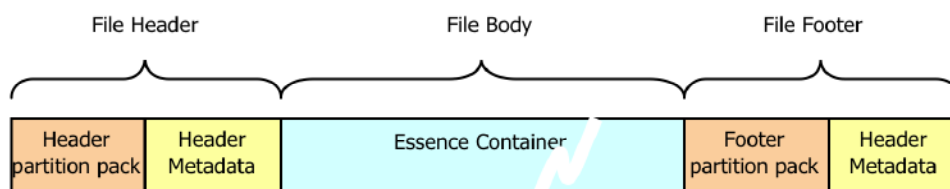


Figure 2 – Simple MXF file

4.1 What are the File Header, the File Body and the File Footer?

The basic file header, file body and file footer are identified in Figure 2. The reason for the split is simple. The file header is designed to be small enough that it can easily be isolated and sent to a microprocessor for parsing. The bulk of the file will usually be the file body — this is the picture, sound and data essence that commonly exceeds 90% of the file data. The file footer provides a means to put the header metadata at the end of the file.

Why is a file footer needed? In certain applications, such as recording an audio-video stream to an MXF file, there will be header metadata values that are not known until the recording is finished. The file footer provides a mechanism for providing such header metadata values. The file footer also provides a clear indication that the file has terminated.

4.2 MXF Partitions

MXF allows files to be divided into file partitions for the file header, the file body and the file footer. This sub-section describes the use of file partitions in MXF files including the reasons for their use.

4.2.1 File partitions

MXF files can include optional partitions that can be used to split up the file components to provide a variety of requirements as follows:

1. Robustness of metadata information by repetition of the header metadata;
2. Multiplexing of different essence containers;
3. Easier location of essence container data when using high-speed tape devices;
4. Optimizing the distribution of the data in a file for storage or transmission.

Notes:

1. Repetition of the header metadata within a body partition is dependent upon the application on a per-application basis. Such applications can be found, for example, in the transfer of an MXF file as a stream over a unidirectional link and in data tape shuttling. One purpose of such header metadata repetition is to support the recovery of critical metadata in applications where the file can be interrupted or where a decoder starts to receive data in mid-transfer.
2. Multiplexing and storage optimization can be a complex subject and is highly dependent on the storage or transmission device used. Hard discs, DVDs, satellite links and tape devices all have different requirements. The MXF structure allows a great deal of flexibility in the positioning of the partitioning information and the use of fillers to allow optimization for different devices. Typically, if storage or transmission optimization is important in an application then the MXF encoder will know which parameters are important. MXF provides the tools, but encoders can make those optimizations that add value to a specific application.

4.2.2 Partition Kinds

A partition is a division of data within an MXF file. There are three different kinds of partition in an MXF file:

Header Partition – this is the first partition of the file and occurs only once in the file header of an MXF file.

Footer Partition – this is the last partition in the file and occurs only once in the file footer of an MXF file.

Body Partitions – one or more partitions located in the file body. Multiple body partitions can be used to divide an essence container into a sequence of parts, provide for multiple essence containers or provide for multiple essence containers each of which are divided into a sequence of parts.

A header or body partition can be open or closed, but the footer partition can only be closed. The normative definition of the terms 'open' and 'closed' is provided in SMPTE ST 377-1 and introductory information is given below:

Open – This marks the information in a partition with a “caution” notice. Any metadata information in the partition was correct at the time of writing, but the application writing the file had not completed the writing process. This means that some of the information can be absent, or can turn out to be plain wrong when the file is ultimately closed. For example, a capture device can have identified a picture and a sound track when it initially started writing the file. During the writing process, a second sound track commenced — this track was not described in the open header metadata.

Closed – This marks any metadata information in the partition as finalized. The application or device creating the file correctly terminated the file and all the properties of the Metadata sets were filled in to the best of the application's ability. In the example above, a repetition of the Header Metadata would be placed in the footer that correctly described the existence and duration of the second Sound Track. All closed partitions in a file must have the same metadata property values without fail. This allows an MXF decoder to determine that the metadata is correct as soon as it finds a closed partition. SMPTE ST 377-1 states that the file footer, if present, will always be a closed partition.

An MXF File can only be called a “closed” file if there is at least one closed partition with metadata. It is important to note that robustness is enhanced when all the partitions in a file are closed. If a file is accidentally truncated during a transfer and the only closed partition in the file was the footer, then the file is no longer a “closed” file. If robustness is desired (and it usually is), application and device developers are urged to close all the partitions of their files. All valid MXF files ought to be closed however certain situations, such as an interrupted file transfer, can leave an “open” file that is still partly usable. The ability of a device to be able to handle “open” MXF files is an application issue.

In an ideal world, the two states of “open” and “closed” would be sufficient to describe all the files in existence. The desire for simple hardware and software, however, means that some capture devices and applications will not be able to parse the wide variety of essence types they might expect to place in an MXF file. To cope with this condition, the states “complete” and “incomplete” have been defined in SMPTE ST 377-1 to mark the status of the essence descriptor(s) in the header metadata of an MXF file.

These states are summarized below:

Complete – Each of the properties in the header metadata with a status of “required” or “best effort” are present in the file and are correct. The status of each of the properties is defined in SMPTE ST 377-1.

Incomplete – One or more properties within in the header metadata with a status of “best effort” has a distinguished value. The distinguished value is used to mark the property as “unknown at the time of writing”. An MXF file can still be a closed file because all the other properties of the file are known. Some of the header metadata can be incomplete due to the absence of an essence parser at the time of file creation. This allows an application to report many of the metadata properties of the file, but certain essence decoders might need to parse portions of the file before it is playable.

Maximum robustness is achieved when applications and devices create Closed and Complete MXF files.

Each partition starts with a partition pack that defines what sort of partition it is, followed by the following optional items:

- Header metadata
- Index table segment(s)
- Essence container data

From these and other restrictions, a user can limit an MXF partition to contain only a single “thing”; i.e., a single essence type with its associated index table segments. If different essence containers need to be multiplexed together within the file, then a new partition can be started when the essence container changes.

4.2.2.1 Other Partition Kinds

MXF can be extended to include other kinds of file partitions. For example, SMPTE ST 410 defines a Generic Stream partition to allow large data streams to be carried in MXF files. An example use-case is where there is a need to carry non-KLV coded data such as an XML file. SMPTE RP 2057 defines a means of carrying text-based metadata in a Generic Stream partition.

4.3 KLV Coding in MXF

MXF files use key-length-value (KLV) coding throughout for flexibility and extensibility. KLV coding is defined in SMPTE ST 336. This mechanism is used to encapsulate the individual elements of an MXF file in such a way that devices can ignore information when the key of a KLV triplet is unknown. The length parameter tells the user how much data is to be used or ignored.

Note: A full review was published in the July 2000 edition of the SMPTE Journal (Vol. 109, No 7, Engineering Report).

Every item of metadata and essence in an MXF file has a **key** (as a 16-byte value) and a **length** that defines how long the value of the item is, followed by the **value** of the item. The key is a SMPTE Universal label and follows the rules defined in SMPTE ST 298.

SMPTE ST 336 includes not just the encapsulation of individual data items, but also the encapsulation of collections of individually coded KLV data items into logical data sets and packs. MXF sets are KLV coded

objects that contain a group of individually coded metadata items in a non-ordered sequence whereas MXF packs are KLV coded objects that contain a group of individual metadata items in a specified ordered sequence.

A decoder that does not recognize a key is able to skip over the unknown (dark) value and inspect the next key. This allows extra functionality to be added to the MXF specification at a later date, knowing that older decoders will be able skip over unknown values.

Individual data words within the key are ISO object identifiers (OID) using primitive BER (basic encoding rules: ISO/IEC 8825-1 ASN.1). Descriptions of primitive and constructed encoding of ISO object identifiers can be found in SMPTE ST 298, Annexes D and E.

In summary, BER OID encoding creates an identifier from a series of one or more sub-identifiers, none of which can be zero. The first zero sub-identifier is interpreted as the termination of the identifier. Thus if it is desired to make a sub-identifier from a series of bit flags, at least one bit needs to be a fixed '1' to prevent misinterpretation of the sub-id as a terminator if all the other bits are '0'. Any of bits 6 through 0 can be used as the fixed '1' bit, often using either b0 or b6.

This means that the most significant bit of each 8-bit value is a flag to say that the word is greater than a 7-bit value and therefore needs to be followed by more bits. For example, if the 12-bit value **b** ($b_{11} \dots b_0$) is to be mapped into a KLV key, then Figure 3 illustrates a possible mapping into successive bytes X and Y of a 16-byte key:

Byte No. bit	X 7	X 6	X 5	X 4	X 3	X 2	X 1	X 0	Y 7	Y 6	Y 5	Y 4	Y 3	Y 2	Y 1	Y 0
Value	1	0	b ₁₁	b ₁₀	b ₉	b ₈	b ₇	b ₆	0	1	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀

Figure 3 – Example of BER OID encoding

Figure 3 shows a binary 1 in bit 7 of byte X to indicate that this is a multi-byte value and a binary 0 in bit 7 of byte Y to indicate that this is the last byte of the multi-byte value. A byte value of 0 is often used to terminate a label so, for example, setting bit 6 of byte 2 to a value of 1 can be used to prevent accidental termination from occurring in the case where bits 5~0 are all zero.

Note that the actual mapping of bits into a label key needs to be normatively defined in an appropriate document. Note also that, at the time of writing this guideline, this multi-byte OID technique is not in use in any of the MXF specifications, although it is permitted in the SMPTE UL specification as defined by SMPTE ST 298.

MXF parser writers need to be aware that this technique could be used in the future so that, instead of, for example, using 3 single-byte OIDs, a single 3-byte OID could be used. Note that the number of words in other identifiers could be less than 16, or there could be a requirement for 16 bytes of which the final words are set to 0. As an example, where the length of SMPTE keys is fixed at 16 bytes, any unused bytes at the end of a SMPTE key value need to be set to zero.

The **Length** field is BER (Basic Encoding Rules: ISO/IEC 8825-1 ASN.1) coded and is neither binary nor a fixed number of bytes. BER coding allows the length field to have a variable number of bytes. So how do you know the length of the length field?

The length field is always coded MSB (most significant byte) first. If bit 7 of the first byte is a '0' then the 7 least significant bits contains the length value (0 .. 127). If bit 7 of the first byte is a '1' then the 7 least significant bits tell you the number of bytes in the length field; e.g., the value '83h' means that the next 3 bytes contain the length field. The format document gives recommendations for the upper limit of the length field. Decoders need to be able to handle both short form (1 byte) and long form (>1 byte) BER coding.

The examples below show a length value of 64 coded in the three different ways:

40_h short form coded

83.00.00.40 long form coding using 4 bytes overall

87.00.00.00.00.00.40 long form coding using 8 bytes overall

In specialized operational patterns (see Section 9.6.1), the file is allowed to start with a non-KLV run-in. This is to permit synchronization bytes or other bytes to be added at the front of the file for certain (limited) applications. In all other circumstances, there will be no run-in and the entire file must consist only of KLV elements with NO gaps.

4.4 MXF Metadata

MXF files using the data model allow it to represent a file interchange data stream. This means it can represent an output timeline that has video, audio and data essence content. It has a logical metadata structure, a defined physical representation (KLV) and is interoperable with other MXF systems. It has been designed so that specific editing systems can open an MXF file without modification to either the MXF file or the editing system.

In practice, this means that there is a lot of overlap between MXF and editing functionality. MXF is targeted at file interchange throughout the broadcast and content creation chain, whereas editing tools are optimized for the multiple generations used in content creation and post production. As a rough rule of thumb, essence content interchange, cut-edit functionality or simpler is an MXF application whereas editing is more appropriate for more complex functions.

Data modeling arranges the file format in terms of objects within an object model. These objects are chosen and defined to reflect the actual processes and content items that go into the MXF file. An IT engineer would take great care before modifying an object model to remove things that looked like they were not needed - the implications for future enhancement and interoperability might be serious and not always obvious. Likewise for the data model of the MXF header metadata.

4.4.1 Header Metadata Data Model

The structure of the MXF header metadata is defined by an object-oriented data model.

Section 7 of this document gives an outline of the header metadata concepts and how the data model describes the file essence. More details about this data model approach are given in SMPTE EG 42 with a specific focus on descriptive metadata.

4.4.2 Metadata Placement

The MXF model creates “hooks” on which the metadata can be placed. These hooks live in the file header, the body partitions, some essence containers and the file footer.

The header metadata area provides for the structural metadata that describes the essence container parameters, but it can also contain descriptive metadata that allows the content to be described. For example production, clip and scene information is described in the MXF descriptive metadata scheme 1 document (part 5 of the MXF specification).

The essence container can also contain timeline-oriented metadata such as time-code and, potentially, pan-scan metadata. Such timeline-oriented metadata can be present either within the essence data or within the essence container.

There are certain metadata parameters that might live in multiple places. The most obvious of these is time code. This can exist in the header metadata, but might also live embedded within the essence container data; e.g., in the GOP header of an MPEG essence container. This repetition is often important and the handling of any conflict between the different instances of the data is application dependent.

Although only occupying a small fraction of the size of a typical MXF file, the header metadata is often, for those inexperienced in formal data models, the most difficult part to understand. The following sections introduce the metadata categories and the topic of object-oriented coding in a general and easy to understand manner. For a more rigorous explanation, there are many reference books that cover the principles and use in far more detail than given here.

4.4.3 Header Metadata Categories

Broadly speaking, the metadata items can be split into two categories: structural metadata and descriptive metadata. The structural metadata is intended to bind the different elements of the file together and is needed to define the basic file structure. The descriptive metadata is intended to supply extra information about the file, such as a program name or scene description.

There are a large number of metadata elements defined in SMPTE ST 377-1. To understand the restrictions on the use of metadata elements, it is necessary to understand the terminology described in Section 7.

4.4.3.1 Structural Metadata

MXF structural metadata is used to describe different essence types and their relationships along a timeline. The MXF structural metadata defines the way in which the output timeline of the file relates to the one or more stored top-level file packages. It defines the synchronization of different tracks along a timeline. It also defines the picture size, picture rate, aspect ratio, audio sampling and other essence description parameters.

The structural metadata is defined in SMPTE ST 377-1. Most of the parameters are defined in the MXF file format document, but additional descriptors and labels can be defined in individual essence mapping documents. More information on the structural metadata concepts appear later in this document.

4.4.3.2 Descriptive metadata

MXF descriptive metadata comprises information in addition to the structure of the MXF File. This can be intended for human use (as in the majority of the SMPTE ST 380, the MXF DMS-1 specification) or it can be information for machine use, such as a track of information containing depth information for 3D processing.

SMPTE ST 377-1 provides a plug-in mechanism that allows different Metadata sets to be defined and used in an MXF environment. SMPTE ST 377-1 also provides mechanisms for uniquely identifying the metadata scheme(s) present in the file, mechanisms for preventing numerical conflict with existing metadata and a mechanism for determining the version of the descriptive metadata specification used.

No single metadata definition and structure will be appropriate for everyone, so a mechanism was added to allow the integration of other metadata schemes without redeveloping applications and equipment. The MXF plug-in mechanism is lightweight and allows versatility and extensibility for the users.

When descriptive metadata is added using the plug-in mechanism, many of the features of MXF are achieved automatically. The ability to create multiple tracks and synchronize them against each other, the ability to add metadata events synchronized with the video / audio or other tracks and the ability to use metadata in the output timeline that was available in the source file are all part of the standard MXF feature set. This document will outline only the basics of a descriptive metadata scheme. A fuller treatment of the subject can be found in the descriptive metadata engineering guideline, SMPTE EG 42. It is worth noting that descriptive metadata can be for both human and machine use. Much of the machine-descriptive metadata relates to special properties of the essence and has an intimate spatio-temporal relationship to the essence.

4.4.3.3 Application Specific Metadata

SMPTE ST 377-1 provides a consistent method for users to formally add extensions by enabling application specific metadata to be added to the header metadata. This method also provides for the simple removal of such application specific metadata if that is needed. This provision has some parallels with the addition of descriptive metadata in that there are MXF sets that provide the hooks to add (or remove) application specific metadata in the same way as for descriptive metadata.

An application specific plug-in can provide MXF files with additional information not present in the MXF specification that describes some new feature not present in regular MXF files. This additional metadata is provided in the form of additional properties to existing MXF sets and additional sets to the MXF structural metadata. When application specific metadata is added, there are corresponding identifiers that provide decoders with information about the application specific metadata that has been added. This differs from the simple addition of optional properties to existing sets in that a decoder can simply ignore optional properties on the basis that they add no value to the header metadata.

As per descriptive metadata, application specific metadata is always defined by another standard and it provides identification in the form of a SMPTE universal label that uniquely identifies the application specific application metadata scheme together with an application metadata specification that acts as the common gateway for all application metadata schemes.

4.4.4 Dark metadata

Dark metadata is the term given to metadata that is unknown by an application. This metadata can be additional properties added to SMPTE ST 377-1 or metadata that is part of the MXF standard, but not relevant to the application processing the MXF file. It can also be privately defined and generated metadata. It is important that there are some rules on the use of dark metadata to prevent numerical or namespace clashes when private metadata is added to a file that already contains dark metadata. Rules for handling dark metadata are defined in SMPTE ST 377-1.

4.5 MXF Essence

4.5.1 Representing Time and Timelines with Tracks

Time and timelines are features of the logical representation of an MXF file. The concept of time within the file is independent of the arrangement of bytes within the file, although constraints can be applied in order to get certain functionality from the file stream. Time is used to measure the duration of the content as well as to synchronize the content. In MXF a “track” is used to represent the passage of time. A track has units to represent time and has an associated duration property. Some tracks have segments that butt against each other to form a continuous sequence of video (timeline tracks) whereas others can have overlapping events that refer to the point at which descriptive metadata is valid (event tracks). The mechanism for adding descriptive metadata definitions to an MXF file is to add new tracks on which to “hang” the metadata items.

To synchronize two or more tracks, they need to be somehow related. This is done by placing multiple tracks within a package (a container for tracks) which synchronizes the start and duration of each track. Note, however, that tracks can have different time measurement units within the package. Time is normalized within a track by its “edit rate” property. This in turn gives us an edit unit, of 1/edit rate.

4.5.1.1 What are the units of time?

There are two main units of time used with an MXF file. These are:

Edit Units = 1/Edit Rate – used to mark time along a track

Sample Units = 1/Sample Rate – used to describe the underlying sample rate of the essence

Edit units can be chosen for the convenience of the file writer, whereas sample units define the sampling structure of the essence. A sequence of audio samples, for example can have a sample rate

of 48 kHz, whereas the track that describes the sequence can have an edit rate of 50 Hz so that synchronization with parallel tracks is numerically simplified. For video streams, the sample rate is usually defined as the field or frame rate of the content and not the sampling clock.

4.5.2 Essence containers

An essence container defines the encapsulation of a particular type of essence. Its purpose is to allow the essence content to be wrapped in KLV and to have associated with it an optional index table to allow rapid access to a given time offset within the essence. The essence container is structured to allow easy multiplexing with other essence containers and to allow identification of the decoding requirements needed to display, listen to, play, or execute the essence content.

An essence container specification defines a unique SMPTE label for identification as well as a method for encapsulating the essence in a KLV structure. Different essence containers can place restrictions on the interleaving of the essence data to be compatible with existing applications. The SMPTE label allows decoders to make a fast go/no-go check of the essence type at the very beginning of the file.

An MXF file can have more than one essence container. The precise number of essence containers and their relationships is constrained by the operational pattern with which the file complies.

The “generic container” is an MXF essence container defined as a SMPTE standard. This is intended to carry all the mainstream essence types in existence at the time of creating SMPTE ST 377-1. It is very simple in operation, yet flexible enough to carry uncompressed material as well as reordered MPEG compressed material. Associated with the generic container are a number of SMPTE container mapping standards that define how the actual essence byte stream is placed in the generic container.

4.5.2.1 Essence container specification

There are several specific questions that need to be asked when putting an essence container into an MXF file. These are notably:

1. What limitations are placed on the essence container when it is in an MXF file?
2. Are there interleaved variants of the essence container?
3. How are the contents of the essence container KLV coded?
4. How are the essence containers padded to fit the chosen KAG size?
5. What is done with metadata embedded within the essence container?
6. How are index tables used with the essence container?

The essence container and mapping specifications provide the answers to these questions. It is the intention of the essence container and mapping documents to restrict the choices of an essence container implementation sufficiently to allow interoperability between devices, yet with enough flexibility to solve real world problems.

4.6 MXF File Tools

This sub-section identifies the tools provided by MXF. In some cases, the description provided here is complete. Other cases describe only the overview and leave details to a later section.

4.6.1 Index tables

MXF files can include an index table that provides rapid conversion from sample-based indexes (e.g., time code) into byte offsets within an essence container. The index table can be segmented and can be stored before, after or multiplexed with the essence data segments. Index table segments can be placed in file partitions (see Section 4.2.1) to limit memory requirements or to provide per-stream index tables that are position independent within a file.

An index table provides for random access within an MXF file. Specifically, it allows random access by a time index. This means that if you want to access the picture, sound or data essence located at, for

example, 10 seconds into the essence container, then an index table will provide the translation between the time value and the byte offset within the essence container. MXF index tables are quite complex because the format is designed to cope with interleaved essence containers that can be constant or variable bit rate and that also can be temporally reordered on the disk compared to the presentation order (e.g., long GOP MPEG-2 files).

SMPTE 377M-2004 limited the length of index tables to 64KB with a 2-byte length field and this required some files to employ index table segments. SMPTE ST 377-1:2011 provides for the use of a 4-byte index table length field that provided for large index tables without the need to provide index table segments.

4.6.2 Random index pack

The random index pack (RIP) is optional but, where present, is always found at the end of an MXF file where it provides a list of the positions of all the partitions within the file and contains a mechanism for quickly determining its own existence.

This differs from an index table, which provides the byte offsets of the essence content within a partition. The difference can be clearly seen when two different essence containers are multiplexed together. There will be two separate index tables, each of which contains conversions between temporal offsets and byte offsets within each essence container. In contrast, the RIP gives absolute positions of all the partitions within a file, which can then be used to locate index tables without having to parse the entire file.

4.6.3 Operational patterns overview

An operational pattern (OP) is used to constrain MXF complexity and defined in a SMPTE standard. The generalized operational patterns are intended to split the complexity depending on the complexity of processing required by an MXF decoder. Specialized operational patterns are likely to be created in order to constrain MXF for a particular 'application space'. Usually an MXF file is interchanged for a purpose. This can be the exchange of an ingested clip, a camera output, a finished program or the interchange of a partially edited program. These requirements have different implications for the structure of the file. Different operational patterns define the structural metadata that is required to satisfy a particular application. In general, the higher the number of the operational pattern, the more complex the file and the more functionality is required in the decoder. Simple operational patterns such as OP1a can be used with both linear and non-linear access devices. Some more complex operational patterns require non-linear access devices.

Each operational pattern has an assigned SMPTE label value that allows MXF decoders to quickly establish the complexity of an MXF file.

More detailed information about operational patterns is given in Section 9.6.

4.6.4 KLV alignment grid

The KLV alignment grid (KAG) can be thought of as gridlines spaced on uniform byte boundaries in each partition. To achieve good performance, all the important KLV items within the file (header metadata, content packages of the essence, etc.) must line up on the grid. This means that the first byte of the key needs to be on a grid boundary. Note that a device or application must always be able to read an MXF file regardless of whether the elements within the file fell on rigid byte boundaries within the file.

The reference point for a KAG is the first byte of the key of a partition pack, and the KAG value is valid within the partition. SMPTE ST 377-1 states:

"The first gridline in any partition shall be the first byte of the key of the partition pack that defines that partition."

In order to have a global KAG value, each and every partition pack needs to have the same KAG value. Additionally, to maintain this global KAG value, the first byte of each and every partition pack