The Structure of an MXF File:

The Physical view

Paper supplied courtesey of Omneon, Inc.

There are 2 ways of looking at an MXF file: the Physical View, which examines exactly how and where the bits and bytes are stored into the file, and the Logical View, where a more human-understandable representation of the file is presented as a series of timelines along which events occur. Effectively, the Physical View is a literal description of the makeup of the file, whereas the Logical View is more concerned with how the media is actually synchronized and played. This paper will examine the Physical View – a separate paper will cover the Logical View.

There are 2 basic types of MXF clip: The first are those where the essence (media) is actually stored in the same file as the Metadata that refers to it. These files are said to have *Internal Essence*. The second type is that where the essence is stored in separate files to the Metadata, and these files are said to have *External Essence*. In this case, the decoder first reads the Metadata file, and that Metadata points to the files in which the individual pieces of essence are stored. To the outside world, the Metadata file - which is an MXF file in terms of its layout, but simply doesn't have any media - is the clip, the essence often being stored in a separate directory which is not directly seen by the decoder.

KLV Wrapping

All data in an MXF file is encapsulated in KLV packets, so an MXF file is actually just a collection of KLV triplets stored one after the other. Basically, the principle behind KLV wrapping is to provide each piece of data with a unique *Key*, which can be examined by a decoder to ascertain whether that specific piece of data is important to the decoder (or if it can even decode it). If not, the decoder can jump the data item and move on to examine the next one. This becomes an important technique for quickly locating specific items in an MXF file, as will be seen later in this paper.

File Partitioning

Media files can be extremely large, and parsing a multi-Gigabyte file as a single entity can place significant strain on a decoder. As such, it is often useful to break the file up into multiple partitions. This is a commonly used technique when dealing with media files, and offers several advantages, easing the memory requirements for a decoder, for example (as it now only needs enough memory to store a full partition as its worst case, rather than the entire file). Partitions are also useful in streaming applications, where it is not possible to determine when any individual decoder will start to ingest the streaming data. Partitions, in this case, can be used giving the decoder clear locations to begin decoding the stream, rather like bus stops give passengers clear locations of where to get on a bus. For these reasons (and many others), MXF files are often broken up into partitions.

The basic structure of an MXF file is shown in Fig. 1. By definition, all MXF files contain exactly one Header Partition, 0 or more Body Partitions, and 0 or 1 Footer Partition. Header Partitions and Body Partitions can contain essence (the actual media), but a Footer Partition cannot – it can only contain Metadata. Since the Body and Footer Partitions are optional, then the simplest possible MXF file simply comprises the Header Partition. Operational constraints mean that a single partition file is quite restricted in duration, so in practice there are usually more partitions than that, with the excess essence being stored in one or more Body Partitions.

The Footer Partition, if present, can contain a repeat of the Metadata that is contained in the Header Partition (often referred to as the "Header Metadata"). The reason for this is simple: often, it is not possible to know

Figure 1 . Basic structure of an MXF file

the value for all Metadata items when the clip starts to record – you may be performing an open-ended record, for example, so the duration of the recording is unknown when the file is created. By putting a copy of the Header Metadata in the Footer Partition, these issues can easily be addressed - there are mechanisms that allow you to know which copy you should look at when you read the files, so there is no ambiguity on where to read the data from

Figure 2 is a diagram of an MXF file with all of its optional components. As you can see, the file structure can grow in complexity, but all of the general principles still apply.

The Header Partition

The Header Partition is the first partition of the file, and is comprised of a Header Partition Pack (which is used to tell the decoder that this partition is the Header Partition), followed by the Header Metadata itself. After the Metadata, we see an Index Table, or more correctly, an Index Table Segment, as it only relates to the part of the file stored in its associated partition – if you add all the Index Table Segments together, you get the Index Table for the file. Index Table Segments, which are really only used for MPEG files, turn a timecode number into a byte offset. This byte offset tells the decoder that timecode value hh:mm:ss:ff is N bytes from the start of the Essence Container (the Essence Container is the data construct which encapsulates the actual essence – more on that later). One thing worth noting is that the Index Table can reference the essence that comes immediately after it in a partition, or reference the material in the partition that comes immediately before it. Both modes are perfectly legal. Note also that in some cases, the index tables are placed in their own separate partitions, so that one may make a rule that only one entity – Metadata, Index Tables or Essence - is placed in any single partition. Index Tables are not needed for essence which is defined as *Constant Bytes per Entry* (CBE). In these cases, since each frame is exactly the same length, you can use simple mathematics to determine the offset from the start of the Essence Container for any Timecode value. Finally, the actual essence is then encoded into the Essence Container itself. Placing some essence into the Header Partition is totally legal according to the SMPTE specifications, but it is optional, and up to the manufacturer.

Body Partitions

Following the Header Partition, we have the first Body Partition, which is used to store additional essence. The Partition Pack (now called the "Body Partition Pack") tells the decoder that this partition is a Body Partition, and that it contains essence which is a continuation of the previous essence. This may seem unnecessary, but MXF allows for multiple Essence Containers to be multiplexed together. This option is beyond the scope of this paper, but it does mean that it is important to indicate which Essence Container this partition is a continuation of. After the Partition Pack, there is the option to repeat of the Header Metadata. While the MXF specification allows for this option, it is very rarely adopted, and has therefore been left off the diagram for the sake of legibility. Next comes the essence, again with an optional Index Table Segment either before or after it (or in a separate partition of its own) In our example, there follows

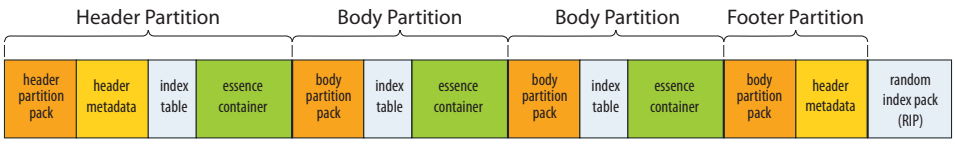

Figure 2 Structure of a typical MXF file (including options)

a second Body Partition, which completes the storage of the essence.

The Footer Partition and the RIP

The Footer Partition is the final partition in most MXF files, although in reality the presence of a Footer Partition is optional. Once again, the Partition Pack is used to indicate to the decoder that this is a Footer Partition, and in our case, the Footer Partition contains a copy of the Header Metadata (this time, all items should be complete). The presence of this repeated Metadata is also optional, but this option is often adopted – many files do indeed contain a copy of the Header Metadata in the Footer.

Most files terminate with something called a *Random Index Pack* (RIP). This is another optional table, which tells the decoder where the start of each Body Partition is located. This can aid in non-linear access to the contents of the file. The RIP, if it exists, is stored at the end of the file, so that a decoder always knows where to go to find it.

Metadata Repetition

All partitions have the option to repeat the Header Metadata. This could be very important in some streaming operations, ensuring that a decoder may start to receive data at any point in the transfer, and have the opportunity to quickly have access to Metadata values. In practice, in the broadcast market, it is very rare for a file to have copies of the Header Metadata in all partitions –usually only the Footer partition has such a copy.

Complete and Incomplete Partitions

To examine this topic, Let's consider the Header Partition in slightly more detail – the same information holds true for all partitions, but the amount of Metadata will vary, depending on whether or not the Header Metadata has been replicated in any specific partition.

As mentioned earlier, the Header Partition begins with the *Header Partition Pack*. The term "Pack" in MXF parlance simply denotes a grouping of KLV items which must appear in a specific order. The Header Partition Pack is a simple data construct, which indicates that this is an MXF Header Partition, along with a status flag which indicates how complete the Metadata is. MXF uses 4 parameters to indicate the status of a partition, which are encoded into a single byte in the Header Partition Pack. It should be noted that all partitions contain some Metadata. The Partition Pack itself has a couple of Metadata items to provide information about the location of the partition which precedes it.

"You will find that the MXF specifications leave a lot of options open to manufacturers, which has led to some confusion and interoperability problems. Many of these are now being ironed out as the format matures"

The first two parameters are *Closed* and *Open*. If a partition is marked as "Closed", it means that all Metadata which must be present (as per the MXF specification suite) have been filled in correctly. Conversely, if a partition is marked as "Open", it means that some of the Metadata has not been filled in, or is just plain wrong. The location of the Footer Partition (as an offset from the Header Partition) would be one such parameter – it won't truly be known until the file is closed. As such, decoders may read the Metadata in an Open partition, but it shouldn't trust the information absolutely. The remaining two parameters are *Complete* and *Incomplete*. These can sometimes seem confusing, but the terms have very specific meaning. If a Partition is marked as "Complete", then it means

Figure 3: KLV recursion (the "value" is made up of KLV packets

that either it doesn't contain any Header Metadata, or that all the required properties/values in the Header Metadata are correctly filled in. Marking a partition as "Incomplete" flags that some of the Metadata values were unknown at the time the partition was completed. Clip duration is one such parameter (don't confuse duration with Footer Partition location – they are most definitely not one and the same thing). So a file can be "Open and Incomplete" – meaning that the location of the footer cannot be determined from this data, and some of the parameter values are default or guesses (and therefore likely to be wrong), or "Closed and incomplete" – meaning that you can determine where the Footer partition is, but that some of the Metadata values are defaults or guesses, or "Open and Complete" – meaning that you can't locate the footer from this data, but the Metadata values are all correct, or finally "Closed and Complete" – you can locate the Footer partition, and all the Metadata values are correct. Obviously, Header Metadata should only be completely trusted if it came from a "Closed and Complete" partition – which is usually in the Footer. The MXF spec does allow file writers to go back into other partitions, and update their Header Metadata so that they are also "Closed and Complete", but this very rarely happens in real-time systems.

Metadata

The Metadata itself is simply a number of sets which contain Structural Metadata. The topic of sets is examined in the Omneon white paper entitled "Encoding Data Into MXF Files", so it will not be covered here. Metadata is mandatory in the Header Partition, but optional in all other partitions. Metadata falls into 2 distinct classes – Structural and Descriptive.

Structural Metadata is largely intended for machine consumption – it details the encoding used to compress the video and audio (if compression is applied), the aspect ratio of the video, how the video and audio should be synchronized, the physical arrangement of the bytes that make up the file. After the Structural Metadata Sets come the Descriptive Metadata Sets.

Descriptive Metadata is for humans to read: Title, cameraman, producer, series title, reel number, camera ID, script version etc. etc. There is a predefined scheme called DMS-1 (for Descriptive Metadata Scheme – 1) which attempts to detail all of the possible Metadata items that may be important to a specific use case – there will undoubtedly be more as we move forward

All of the above can seem pretty complex and overwhelming, but in a nutshell, the Header Partition tells a decoder what kind of file its reading, tells it how valid its information is likely to be (and potentially where to go to get more accurate data), gives it information on how the file goes together technically so that the decoder can play it (or flag that it can't play it for some reason), and allows the decoder to interact with human-readable information about the file or present it to the user (Obviously, this is all data that the decoder needs before it can begin to decode the material which is contained in the rest of the file).

The Footer Partition, on the other hand, largely gives us a place to store another copy of the Header Metadata, where some values may be more up-to-date than those in the Header Partition. Body Partitions are used to store additional essence, although in theory they could also contain replications of the Header Metadata.

Encoding the Essence into Partitions

Essence can be placed into the Header Partition or into Body Partitions, but never into the Footer Partition. When the specifications were written, it was recognized that new compression technologies were bound to be developed over time, and that if MXF didn't allow for these new compression technologies to be "plugged in", then it would have a very short lifespan indeed. To overcome this, the main MXF specification document (SMPTE 377M) details something called an Essence Container. The idea of the Essence Container is to define a simple mechanism for essence to be wrapped in KLV, to optionally associate it with an Index Table, to allow for interleaving of essence within it, to once again identify the decoding capabilities required to play it, and to uniquely identify which kind of Essence Container it is. The concept of the Essence Container is (as is always the case with MXF) a very flexible idea, designed to allow the addition of new Essence Containers as time progresses. Having said that, there is currently only one Essence Container specified in the MXF documentation, and that instance is called the *Generic Container*. It was designed to carry all of the major essence types in use at the time that the spec was written, and there are individual SMPTE specifications on exactly how you put each essence type into the Generic Container.

A precautionary note: MXF experts often use the terms *Essence Container* and *Generic Container* interchangeably – even in the same paper. Don't get confused – at the moment, they are one and the same thing (that would change if ever another Essence Container type was developed). The Generic Container is defined in SMPTE 379M.

Fundamentally, the Generic Container (or *GC* as it's often called) breaks the essence up into sections, which are called Content Packages, and these packages are laid end-to-end until all of the essence has been stored (see Figure 3). It is entirely possible that the Generic Container be so large as to not fit into a single partition, but this does not cause a problem. The specification allows for GCs that extend across several partitions: a SMPTE Unique Label (UL) in the Partition Pack of each partition identifies the GC that any individual partition belongs to.

In practice, clips have Content packages of only two types. The first is a single Content Package containing all of the essence for the file. Each individual essence (video, audio, timecode, etc) is presented in its entirety, followed by the next essence type. So all of the video may be encoded first, followed by all of the audio etc. Termed *Clip Wrapping*, this may not be too convenient for streaming files, as you have to wait until all of the video is delivered before you get any of the audio. The second choice solves that problem. *Frame Wrapping*, means that each Content Package contains all of the data for a single frame of the file. So video and audio for any specific frame are delivered in the same Content Package. It is important to note that this does not mean that all of the Content Packages are the same size, particularly in the case of long–GOP MPEG. In most cases, the essence of a clip will be Frame Wrapped.

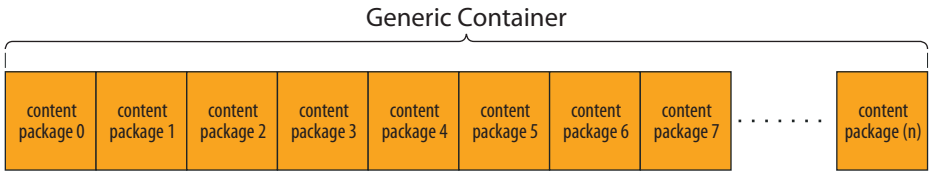

Figure 3. The Generic Container is a contiguous sequence of Content Packages

Figure 4 shows the basic layout of a Content Package. As you can see, the Content Package is further divided into *Essence Items*, each of which represents one type of material in the clip. The MXF specifications state that the Content Package can contain any combination of the essence items, but you can only have one Essence Item of each type per Content Package (i.e. you can't have two Data Items in a single content Package), they must be in the same order in every Content Package, and they must always be present. If

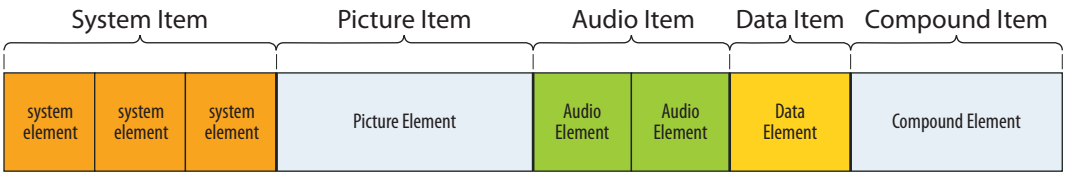

Figure 4. Structure of a Content Package

you have no data for a specific item in a specific Content Package, you still have to have the item; you just don't put any data into it.

The Content Packages can be presented in any order (as long as it's consistent), with the exception of the System Item. If a Content Package has more than one Item, then the System Item should be present (there are those loose specs again – it is possible to not have the System Item in a multi-item Content Package and still have a legal file, in the strictest sense). If the System Item is present, it must be the first item in the Content Package.

The first byte of the key for the first Essence Item can be used to indicate the start of the Content Package, for synchronization purposes (the MXF spec allows for other ways of determining the start of the Content Package, but doesn't define what those other ways are). This is made easier by the fact that the first 12 bytes of the keys for Essence Items are all the same, and that combination of values is not used for any other key, so when you see that specific sequence of 12 bytes, you know you are looking at the start of the key for an Essence Item. The start of the GC is indicated by the first occurrence of the key of the first Essence Item in the Content Packages.

Note that even the individual Essence Items are then broken down into Essence Elements. In simple terms, each element is an individually KLV wrapped entity, containing some amount of data. Each item can contain a maximum of 127 Elements.

Picture Item

In the case of the Picture Item, it is unlikely that you will have more than one Picture Element, although you certainly could have two — one for video, and one for associated key, for example.

Audio Item

You are very likely to have more than one Audio Element — the audio could be a stereo pair, in which case you would have 2 audio elements — one for the left channel, and one for the right. Or you could have 6, for discrete 5.1 audio, or you may have 8: discrete 5.1 plus a stereo mixdown. Remember, though, that you can only have a single audio item.

System Item

System Items are used to store Essence Item-related Metadata. An example of this would be frame-by-frame timecode values. It is not really practical to embed this into the Header Metadata, due to the complexity of relating that Metadata to a specific timecode value. Most elemental video streams include Timecode encoded somewhere in the essence itself. Extracting that Timecode and putting it into the System Item makes it easier to decode — you don't have to parse the essence for the Timecode data. That is one of the reasons that the System Item (if it exists) is always the first item in the Content Package — you can check for valid timecode values before decoding the essence.

Data Item

The Data Item, if it is present, is used to store continuous data that are neither picture nor audio – subtitles, teletext and other VBI data are examples of the essence types stored in the Data Item.

Compound Item

The Compound Item, if it is present, is used to store essence streams that are intrinsically interleaved and difficult to extract to individual Items. An example of this would be DV essence. DV essence has video and audio interleaved in a complex manner, so it was deemed easier to define a separate Item for this than force manufacturers to de-interleave the data. In almost all cases, if a Compound item is present in a Content Package, the Picture Item will not be — there may be additional Audio Items, though, as DV essence only has 2 channels of audio.

Conclusion

The physical layout of an MXF file can seem quite intimidating at first glance, but the principles themselves are relatively straightforward. Most of the complexities come from the large number of options a programmer may choose from – all of which are perfectly legal. This abundance of options can make file interchange difficult, but by constraining some of the options to pre-defined choices via a document called an "Application Specification", interoperability can be much simplified.

To reduce the information provided in this paper into simple blocks, MXF files are made up of:

One, and only one Header partition, which contains important Metadata that a decoder will need before it starts to decode the file, and may include essence (most do)

Zero or more Body Partitions which store additional essence as required, and may include a repetition of the Header Metadata (most don't). The frequency and size of these partitions is left up to the designer

Zero or one Footer Partition, which may include a repetition of the Header Metadata (most do). This is the logical place to start when looking for accurate Metadata in a file.

Zero or one Random Index Pack. If present, this is always the last KLV in the file, and indicates where each o f the partitions are located.

The essence itself is stored in the Generic Container, and segmented up into Content Packages of various sizes. In most cases, the Content Packages contain all of the essence associated with an individual frame of the clip, but it is possible to encapsulate all of a specific essence type in a single Content Package, in which case that essence is said to be *Clip Wrapped*.

The adoption of all of these concepts into the structure of an MXF file results in format independence, both in the spatial and temporal domains. Coupled with the ability to include or exclude essence types at will, it is extensible into multiple use cases and market segments, with the promise to allow seamless interchange of Media and the vital Metadata that goes with it.

This white paper was supplied to the AMWA by Omneon, Inc.

Further white papers on MXF, AAF, XML and SOA applied to advanced media workflow can be downloaded from the AMWA website at www.amwa.tv. 3/2010