

The Material Exchange Format (MXF) and its Application

By Jim Wilkinson and Bruce Devlin

This paper describes the specification of the Material eXchange Format (MXF) currently under joint development by the Pro-MPEG forum¹ and the AAF (Advanced Authoring Format) Association.² As the name implies, the specification is focused towards the exchange of program material between devices that support the MXF. An MXF file is a “wrapper” for containing audiovisual material in a playable format. Each file contains a comprehensive metadata structure together with component parts that enable MXF files to be written and read directly by AAF-compliant tools. The entire data structure is based on approved SMPTE standards relating to metadata coding using the KLV syntax³ and standardized metadata dictionary items for interchange.⁴ The paper will also describe how the G-FORS project actively contributes to the MXF specification in order to achieve its goals of network file-based TV production. Details of the project and its use of MXF are included to demonstrate applications of MXF.



Jim Wilkinson



Bruce Devlin

AAF Compatibility

The structural metadata defined in the Format document has a class structure compatible with that of the AAF class. This, together with other aspects of the file structure, allows MXF files to be directly read and written by AAF systems without an import/export filter.

The MXF specification is large, so far, some 180 pages for the basic specification with more expected as new parts are added. The first 40 pages address many of the issues of exchanging program material as files and how MXF files resolve them. Figure 1 shows how the various parts of the documents are related.

Part 1 is the Engineering Guideline. If you are new to MXF, this is an essential read. Part 2 is a normative definition of MXF that defines the format data structures. A supplementary document provides a catalog of values for components defined in other parts of the specification. Part 3 defines operational patterns (OPs) that apply coding constraints to aid interoperability between applications at defined levels of complexity. Part 4 comprises the descriptive metadata schemes that may be plugged into an MXF file. Part 5 includes a number of individual documents that describe the essence containers used in MXF. One of these, the MXF Generic Container, is a streamable audiovisual essence data container that requires associated mapping documents to describe how various essence data types can be mapped into it.

Presented at the 143rd SMPTE Technical Conference and Exhibition (paper no. 17) in New York City, November 4-7, 2001. The content was updated on June 21, 2002. Jim Wilkinson is with Sony BPE, Basingstoke, Hants, U.K., and Bruce Devlin is with Snell & Wilcox Ltd., Petersfield, Hants, U.K. An unedited version of this paper appears in *Pixels, Packets, Processing, and Infrastructure*, SMPTE 2001. Copyright © 2002 by SMPTE.

MXF Basics

MXF files are based on SMPTE KLV coding according to SMPTE 336M,³ which allows full flexibility for current operations and extensibility for future applications. The specification also relies heavily on 16-byte Universal Labels (SMPTE 298M) to provide identification of strategic parts of an MXF file; for example, labels for the operational pattern, the essence container(s), and the descriptive metadata scheme(s).

File Structure

The MXF file structure follows a common theme of many files with the following basic components shown in Fig. 2 (note that the individual blocks are not to scale):

- A file Header, which starts the file and provides information about the file as a whole. Its three parts are described below. At least the first part of the header must be consistent for all implementations.
- A file Body, which comprises the essence component(s) in a container. Where there is more than one essence component, interleaving is provided by the container to allow many stream features such as recovery from incomplete transfers, cuts-only editing, etc.
- A file Footer, which terminates the file.

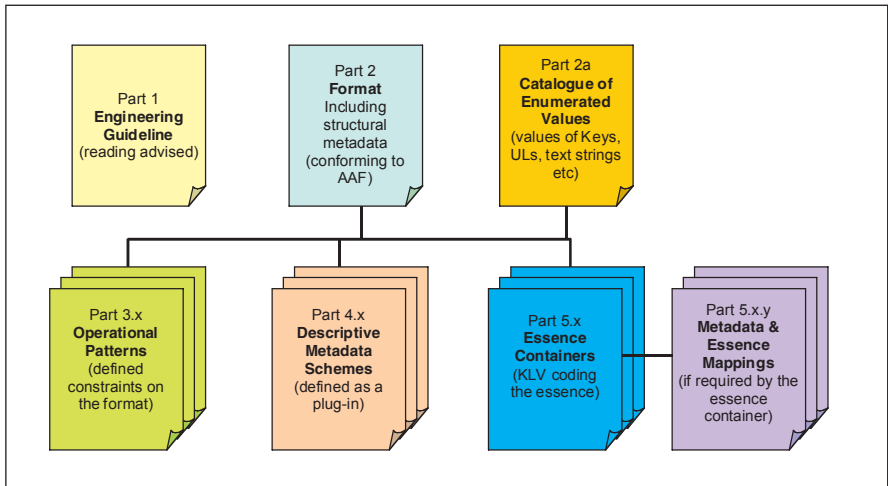


Figure 1. MXF document structure.

File Header

The file Header entails a number of basic components including an optional Run-in, a Header Partition Pack, Header Metadata, and an optional Index Table.

Run-in. In some specialist applications, there may be a need to recognize a run-in sequence before the start of the Header Partition. The run-in is allowed only to support specialist applications.

The Header Partition Pack is provided to “kick-off” the file with a KLV-coded Partition Pack. It provides some critical metadata about the common structure of the file and also some simple indexing of the Header components for rapid decoder access.

Header Metadata. Although occupying only a small fraction of the size of a typical MXF file, the Header Metadata is, for traditional video engineers, the most difficult part to understand. It is based extensively on object-orientated coding techniques that are beyond the scope of this paper, although the Engineering Guideline document does attempt to describe some of the basics. The Header Metadata has two parts: (1) Structural Metadata, a defined object structure compatible with the AAF classes. The object structure is defined, although the presence of any object is dependent on the operational pattern and the essence containers. (2) Descriptive Metadata, defined as a “plug-in” to provide enhancements to the file description. The generic descriptive metadata is that defined by the Geneva Scheme, although other schemes may also be used as required by the application.

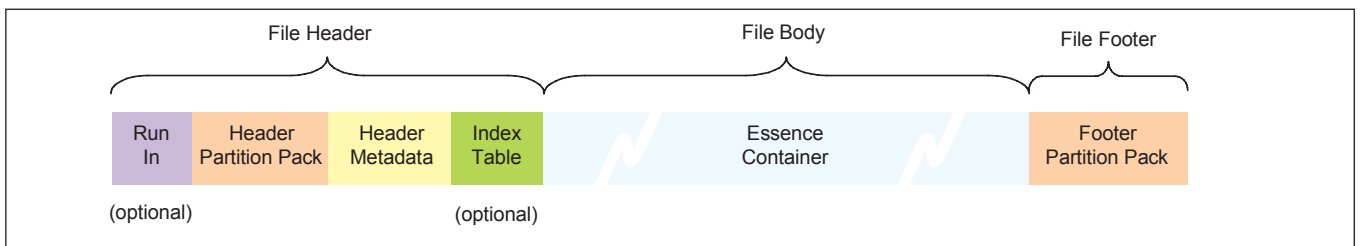


Figure 2. Core components of a simple MXF file.

Index Table

An Index Table is an optional component that allows rapid access to any component in an essence container through a byte offset value from a defined address. Although optional, it is strongly recommended to increase the functionality of an MXF file. For essence containers with a defined frame length, the index table is very small and simple.

File Body

The MXF file Body comprises zero or more defined essence containers. A container may include a single essence component or an interleave of multiple

essence components. Essence container specifications are written as a plug-in to allow the inclusion of any standardized container. The Generic Container is the recommended MXF essence container, based on the work of the EBU/SMPTE Task Force report.⁵

The MXF Format specification defines only the constraints that an essence container should meet in order for it to become an acceptable MXF container plug-in. These constraints are defined in order to ensure that all MXF files meet the criteria laid out in the user requirements and are summarized as follows:

- Must encapsulate essence components with KLV coding using publicly registered keys.
- Must provide interleaving of the essence components over a defined duration.
- Should be a publicly available specification from an internationally accredited standards body.

Currently, the following essence container specifications have been defined:

- A generic container with intrinsic interleaving of components that offers direct compatibility with SDTI-CP.⁶ This is a pure essence and metadata container specification and does not provide specific essence mappings. For that, associated documents are required to describe the mapping of specific essence components into the elements of the Generic Container.
- Mapping the SDTIL-CP System and essence elements into the Generic Container.

- SMPTE Type-D10 (MPEG-2 50-Mbit/sec I-frame) and multichannel AES-3 data mapping.
- DV-DIF (both IEC DV and DV-based) mapping.
- SMPTE Type 11 (HDCAM) and multichannel AES-3 data mapping.
- Individual mappings of uncompressed pictures and Wave audio.
- A mapping of MPEG elementary and systems streams, now in the final phase of development.

File Footer

The file Footer comprises at least a Footer Partition Pack. It is optionally followed by a repetition of the Header Metadata and an optional Index Table. For files with multiple partitions, a random index pack (RIP) is provided as an optional KLV pack, which, if present, must be the last item in the file.

Partitions

The MXF specification offers considerable extra flexibility over the simple file so far described. There is more detail in the engineering guideline document, but the next section indicates some possibilities.

Body Partitions

The File Body can be split into two or more partitions. Partitions allow an MXF file to be logically divided to aid parsing, help streaming, and simplify the creation of index tables (which, in turn, eases random access in a storage system). The first Body Partition is always part of the Header Partition, hence, Fig. 2 shows only a Header Partition and a Footer Partition. Second and subsequent Body Partitions are partitions in their own right.

The mechanism for achieving this functionality is the Partition Pack that precedes every partition. It has already been described as the first data pack of the Header Partition and the Footer Partition. A file with a Header Partition, Body Partition, and Footer Partition is shown in Fig. 3. Note the distinction between the parts that include the File Header/Body/Footer components and Header/Body/Footer Partitions.

Partitions allow an MXF file to be constructed with almost arbitrary complexity summarized by the following:

- A file with a single partition and a single essence container that can carry a single essence stream or a multiple of interleaved essence streams. This latter dimension is

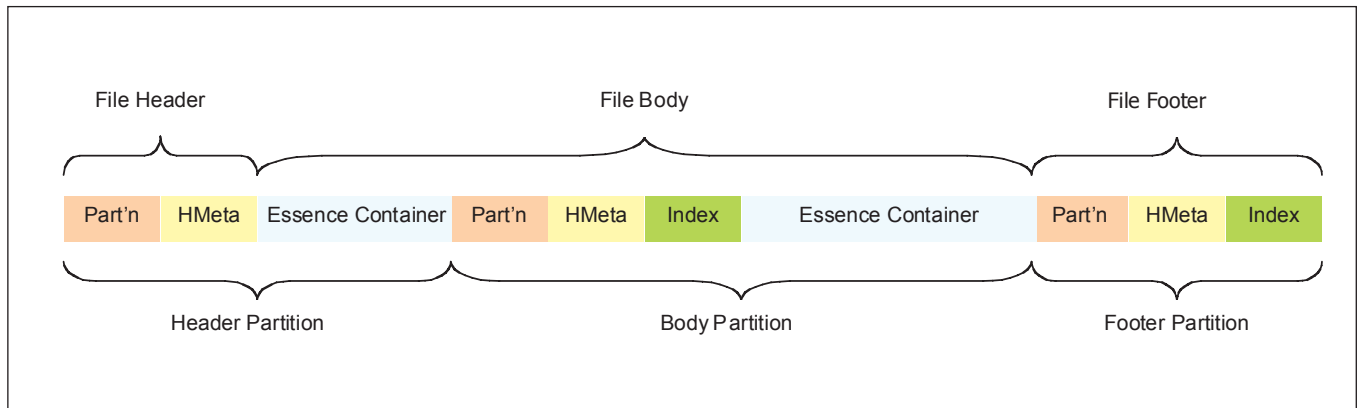


Figure 3. Partitions in a simple MXF file.

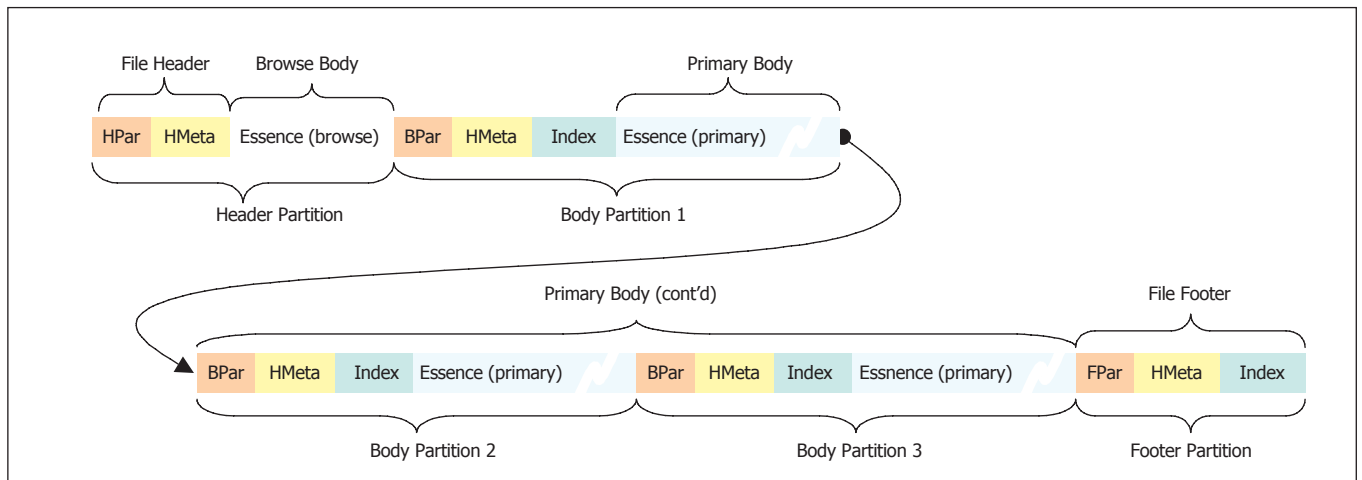


Figure 4. An MXF File with a Browse Container and a Primary Partitioned Container.

relatively commonplace in broadcast applications (e.g., most tape formats and SDI with embedded audio). This file has only a Header Partition and a Footer Partition.

- A file with a single essence container that contains one or more essence streams, but that container is divided into separate Body Partitions (e.g., for partial restore after reception failure).
- A file with multiple Body Partitions may also carry multiple essence containers in each partition, each of which carries a single essence stream. This can be used to provide sequences of essence data where receiver buffering is sufficiently large to accommodate the partition length.
- A file with multiple Body Partitions may also carry multiple essence containers each of which carries a multiple of essence streams. This allows the most complex operational patterns described later.

Multiple Partitions with a Single Essence Container

A file with a single essence container can be divided into partitions as determined by the application. Figure 3 shows an example file containing a single body together with an associated Index Table. If reception of the file starts midway, then the Body Partition can provide a level of recovery including all the file structural and descriptive metadata.

Multiple Partitions with Multiple Essence Containers

In more complex MXF files, there may be multiple essence containers (either of the same kind or different kinds) each with an optional Index Table. Each essence container is identified by a unique BodySID, and each index table with a unique IndexSID. Each partition may contain data only from the one essence container that it identifies through the BodySID, allowing an index table to index the essence container from any partition (often the previous partition).

Figure 4 shows a file containing a browse container in the first partition followed by a primary container divided into three body partitions. The Header Metadata describes both the browse and primary essence containers and the essence streams that they each contain.

Operational Patterns

Different applications produce and consume material of various degrees of complexity and structure, from a single clip to a multitude of clips and effects. Applications requiring only the simplest files should not be burdened with

support of the most complex. To maximize interoperability, the application levels are classified into OPs, each defined by a separate document.

During the development of MXF there were many different attempts at defining the functionality of an OP. The goal was to create one or more axes that allowed software and hardware developers to create products with different levels of functionality (and hence cost). These different axes had to correspond to real-world ways of working and had to provide mechanisms for a file to be “flattened” from a complex to a simple OP in a way that makes sense to someone working with the audiovisual content.

Operational Pattern “Axes”

When trying to constrain the complexity of an MXF file, there are several orthogonal axes that can be defined. Most OPs will be constrained by the axes described here, however, certain specialized applications may define their own OP, which constrains the specification differently. Regardless of the OP, any MXF decoder should be able to read the file Header and report the contents of the file and why it can or cannot process the file.

Item Complexity

This defines the temporal relationship between different essence container items within an MXF file. In principle, there are three levels of constraint:

Single item. The file contains only one item. The single Material Package source clip has the same duration as the File Package.

Playlist Items. The file contains

several items that are butted one against the other. Each Material Package source clip is the same duration as an entire File Package.

Edit Items. The file contains several items with one or more cut edits. Any Material Package source clip may define any part of any appropriate File Package.

Package Complexity

Single package. The Material Package can only access a single File Package at a time.

Ganged packages. The Material Package can access one or more File Packages at a time.

Alternate Packages. There are two or more alternative Material Packages, each of which can access one or more File Packages at a time. These may be used to provide different language versions or special edits destined for different censorship zones (Fig. 5).

“The essence container is tracked and related through the use of UMIDs and essence locator objects. Many MXF files will have the audiovideo essence embedded in the file so that the material is instantly available when an MXF file is opened.”

MXF Application

Descriptive Metadata

Whereas the structural metadata was concerned with how the different material in the file interrelated, descriptive metadata is intended to annotate the file in a way that adds value for the user. The default MXF descriptive metadata scheme is called the Geneva Scheme. Geneva Scheme Metadata is concerned with describing the production, a clip of material, and any scene in the audiovisual content. There are, however, other ways in which a production item, clip, or scene might be described.

Descriptive Metadata Plug-ins

The MXF format provides a plug-in mechanism for adding new descriptive metadata schemes to the format, with the understanding that one size does not fit all. Any new schemes will need to be standardized in order to achieve interchange.

SDKs and Metadata Exchange

The EBU is currently encouraging developers to bring MXF codec software into the public domain. As a part of this work, the header metadata is being defined in the eXtended Markup Language (XML), the current language *du jour* for data exchange between computer systems.

Internal and External Essence Containers

In an MXF file, the essence can be located within the file, or externally referenced. The essence container is tracked and related through the use of UMIDs and essence locator objects. Many MXF files will have the audiovideo essence embedded in the file so that the material is instantly available when an MXF file is opened. Other files have the reference to the UMID within the file, but the actual essence is located externally, such as in an exter-

nal file server. This is useful in shared storage networks where copying large files leads to wasted time, overloaded networks, and reduced storage capacity. However, it is unsuitable for program exchange where embedding the essence for reliability is preferred.

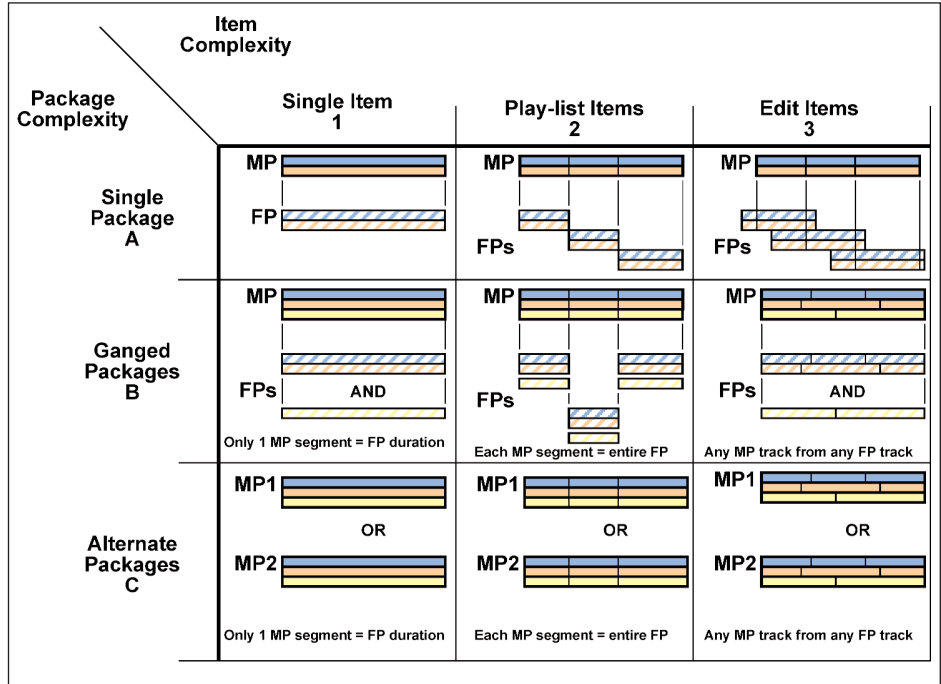


Figure 5. Item and sequence complexity.

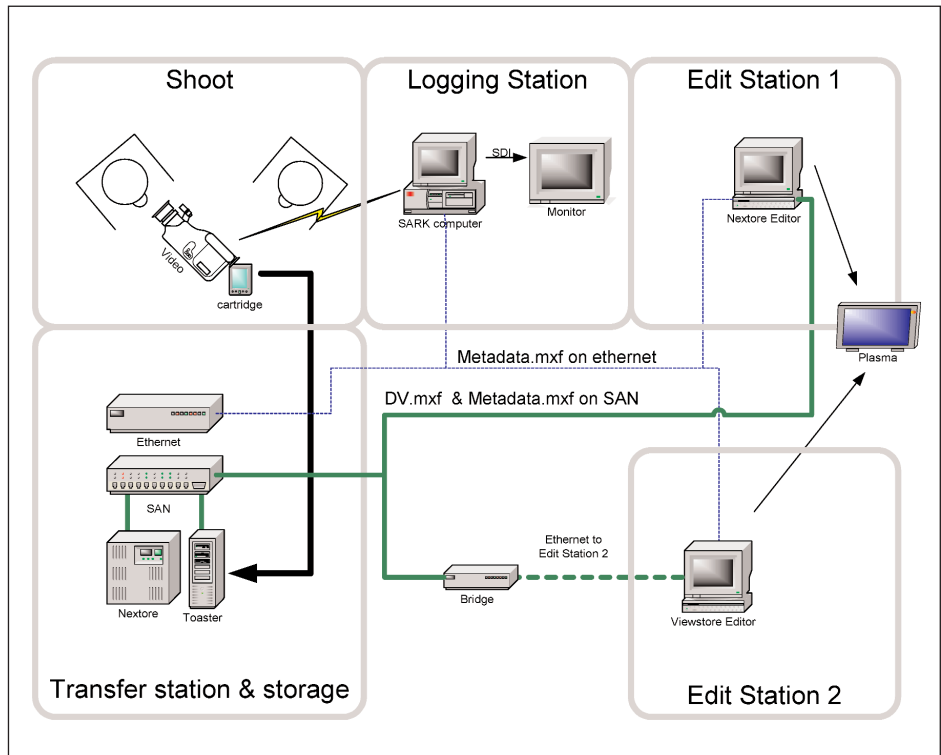


Figure 6. G-FORS demonstration network.

G-FORS Application

The overall goal of the G-FORS project⁷ was to reduce the cost of European program production. The project demonstrated handling of content in the form of a “Generic Media File” throughout the broadcast chain. In order to ensure interoperability, G-FORS has adopted MXF as the underlying file format in the project after full active participation in the development of MXF.

Demonstrator Network

The demonstration network was presented to an audience of around 60 broadcasters and manufacturers in February 2002, and a short clip was successfully created, shot-logged, edited, and replayed using MXF with the metadata remaining intimately attached to the essence throughout the production process (Fig. 6).

The goal of the demonstrator was to show end-to-end program production using MXF wherever possible. This work provided valuable insight into the use of MXF in real life situations, which, in turn, helped to identify areas within the specification that could do with improvement and/or modification.

As shown in the figure, the camera shot a scene and stored MXF files with DV-based video and audio on its disk cartridge. Simultaneously, the camera created an MPEG transport stream with MPEG-compressed video and audio with the MXF headers transmitted on a separate PID. The MPEG stream was transmitted via a COFDM-modulated digital radio camera to the shot-logging station, which decoded and displayed the video so that an operator could construct a live MXF shot log. This small file contained only UMID, time code, and shot-log information; the browse-quality MPEG essence was not stored.

When shooting was complete, the cartridge from the camera was placed in the transfer station and the MXF file with the high-quality essence transferred to the storage area network (SAN). During this process, an automated MXF “glue task” located the correct shot-log file on the SAN by checking the UMID and time code. The shot log was then appended to the high-quality MXF DV file to ensure that it would never be lost.

The MXF DV file was edited in MXF format and an EDL created. This was exported to the Nextore Edit station where playout of the edit material took place. This task shows import of MXF essence to the native internal format of the Nextore with no loss of video, audio, or metadata.

Software Development Kit

In order to develop all the MXF devices needed for the

demonstration, a software development kit (SDK) was created. The G-FORS SDK is a software library meant to make the creation of realtime applications with MXF easier. It is intended that the code will become available as Open Source towards the end of the project. The descriptive metadata add-ins to the software developers kit are done simply with an XML dictionary. This means that new values can be added to software written using the SDK without recompiling. Once a new dictionary is loaded, the new Metadata items become “known” to the SDK. Functions such as searching, e.g., “Find clips where focal length values = 50,” are likely to take advantage of this no-recompile functionality. A specialized code will always be required for software to understand what the metadata actually means. There are mechanisms for doing this without recompilation, but often new major items will require new code. Derived searching would require this built-in knowledge, e.g., “Find clips with a zoom-in,” would have to be translated into code that looked for metadata where the rate of change of focal length with time was positive (increasing focal length of lens). The SDK was handed to the EBU where development has continued as an Open Source Project. Access to the SDK is available by e-mailing the SDK project coordinator at hofmann@ebu.ch, or alternatively by following the link on the G-FORS website.

“A specialized code will always be required for software to understand what the metadata actually means. There are mechanisms for doing this without recompilation, but often new major items will require new code.”

The Project End

The G-FORS project has been very successful. The overall objective was to reduce the cost of TV production; the partners believe that creation and promotion of the MXF format has provided such an environment. The secondary objectives of standardizing the file format and producing the demonstrator were also successfully accomplished. Predicting that a standard will be created is a very inexact science: a good standard needs the right technology solving the right problem with the right players at the right time. A combination of good luck, technical skill, and user-driven requirements has allowed the G-FORS project to make significant contributions to the MXF format.

Further Information

With the project now finished, information about its achievements can be found on the G-FORS website.⁷ The use of MXF in shared networks, however, is being investigated in a follow-up project called NUGGETS (Networks Used in Globally Generic TV Systems). Its goal is to enable TV production on LAN, MAN, and WAN networks using MXF. Further information is provided on the website.⁸

Conclusion

The MXF specification as developed by the Pro-MPEG Forum and the AAF Association meets key criteria for a program interchange file format:

- Application access to interleaved essence container streams as files via a common wrapper designed for maximum interoperability and extensibility.
- A compression-format-independent wrapper, allowing production metadata to be preserved regardless of the compression format used (if any).
- Support for random access and partial file transfers, e.g., moving a sports or news highlight without transferring the whole file.
- Support for access and use before completion of transfers, e.g., beginning to play or edit a file before it is completely transferred.
- Support for cuts-only edits for versioning.
- Suitability of the format for library and archive applications.

Different applications in broadcast and related industries have varying metadata requirements. For example, file complexity needed by an editor is very different from that required for distribution or archiving. For this reason, OPs are provided to specify the type of body pattern together with the level of support in the Metadata Header.

Beyond file interchange formats and application-specific OPs, many other network issues remain to be addressed, including multicasting, streaming over IP networks, and other subjects related to network protocols and quality of service. The goal is to provide broadcasters with access to both compressed and uncompressed systems and ensure that equipment from a wide selection of manufacturers will allow files to be successfully exchanged.

A recently updated Frequently Asked Questions (FAQ) on MXF is available from the Pro-MPEG website, and details of further MXF implementations and applications will become public in due course.

Acknowledgments

The authors would like to thank all the contributors to this work, in particular Oliver Morgan (MetaGlue Corp.) to whom we are indebted.

Bruce Devlin would like to thank the partners of the G-FORS and NUGGETS projects for permission to publish that part of the paper concerned with those projects.

References

1. Pro-MPEG Forum: www.pro-mpeg.org.
2. AAF (Advanced Authoring Format) Association: www.AAFassociation.org.

3. SMPTE 336M-2000, Television—Data Encoding Protocol Using Key-Length-Value.
4. SMPTE RP210-2001, Metadata Dictionary Contents.
5. EBU/SMPTE Task Force for Harmonized Standards for the Exchange of Program Material as Bit-streams, Final Report—Sept. 1998: www.smpte.org and www.ebu.ch.
6. SMPTE 326M-2000, Television—SDTI Content Package Format (SDTI-CP).
7. G-FORS: www.g-fors.com.
8. NUGGETS: www.ist-nuggets.tv.

THE AUTHORS

Jim Wilkinson first worked in the area of broadcasting on DICE, video compression using DPCM at 34 Mbits/sec, and other video projects at the IBA's Engineering Headquarters in 1974. In 1979, he joined the newly created Advanced Development Laboratories of Sony Broadcast where he continues to work. Among his many activities and projects at Sony have been digital video recording, video bit rate reduction, digital image processing, digital audio, and metadata and file/stream formats.

Wilkinson has participated in many standards activities in the SMPTE, AES, and EBU. He was a prolific member of the EBU/SMPTE Task Force and is very active within SMPTE engineering committees on the follow-up work now being undertaken, with particular emphasis on metadata and file/stream formats. He is chairman of the File interchange working group of Pro-MPEG (recently joined by members of the AAF Association and the EBU).

Wilkinson is a Fellow of the SMPTE and member of the IEE and AES. In 1995, he was awarded the Alexander M. Poniatoff Gold Medal for Technical Excellence by the Society.

Bruce Devlin is the principal research and innovation engineer at Snell & Wilcox. He graduated from Queens' College, Cambridge, in 1986 and has been working in the broadcast industry ever since. He joined the BBC Research Department working on radio-camera systems before moving to France where his work on sub-band and MPEG coding led to the design of a satellite newsgathering chipset.

In 1993, Devlin joined Snell & Wilcox where he initiated the company's work on compression coding. He holds several patents in the field of compression, has written international standards on MPEG interoperability, and leads collaborative projects towards the creation of an international standard for broadcast file interoperability. He is editor and document controller of the MXF File Format Specification.