

MXF::SDK

White Paper

April 2004



This information is subject to change without prior notice.

MXF::SDK developed in co-operation with Institut für Rundfunktechnik GmbH, www.irt.de.

Institut für Rundfunktechnik 

Introduction

The Material eXchange Format (MXF) is a new file format intended to allow the interchange of finished or “almost finished” audiovisual material, taking full advantage of the tight integration of Essence and Metadata.

The MXF specification is a group of SMPTE¹ standards having at its core the S377M specification.

But this is not just another specification that may or may not make it to the marketplace. Last IBC and NAB were the proof of this: as you covered ground on the exhibition premises, you could see that the industry’s major players are seriously backing up this effort.

However, taking full advantage of all MXF features without the proper tools is not an easy task: to catch the MXF train, you will need to commit a considerable amount of resources to fully support MXF in your product line. Or, you can start by supporting only the simplest mechanisms. Nevertheless, as more features are added to your products, you will need more MXF technology to back it up and adding it will become increasingly difficult.

What you need is a fully supported, thoroughly tested and well- thought structure to work with from the start. For this reason, MOG Solutions provides you a **MXF Software Development Kit²**, the **MXF::SDK**.

In the **MXF::SDK**, your developers will find a robust C++ implementation of an easy-to-use API³, with detailed interface documentation. Tutorial documentation along with sample applications is also included. With this tool, you will cut development time and reduce your time to market.

¹ Society of Motion Picture and Television Engineers (www.smpte.org)

² A SDK is a collection of code, documentation and tools that is used as a “building block” to develop other software and possibly other SDK’s.

³ Application Programming Interface – an API defines how the services provided by an SDK must be used and is formally defined using an interface definition mechanism (in this case C++ header files).

But do not confuse simplicity with limited functionality. The interface is easy to use because it uses advanced Object-Oriented Programming software patterns to ensure that, although a full featured Object-Oriented model of MXF is exposed, it is still simple enough for any developer to quickly become familiar with it.

The **MXF::SDK** was designed from the start to optimise memory and processor usage and to simplify portability. It is available for Windows, Linux, Solaris and Apple OS-X and MOG Solutions is ready to port it to your favourite Operating System, including embedded ones.

The **MXF::SDK** is being developed in a partnership between IRT and MOG Solutions. IRT (Institut für Rundfunktechnik) is a private company acting as the central research and development establishment for the public broadcasting organisations in Germany (ARD, ZDF, and DLR), in Austria (ORF), and in Switzerland (SRG/SSR). The main focus of IRT's work is the improvement of existing and development of new digital systems and their standardisation.

IRT and MOG Solutions are actively involved in standardisation activities, including the development of the upcoming MXF SMPTE standard, therefore providing the needed expertise to verify **MXF::SDK** standard compliance. IRT is also a leading organization in equipment compliance testing, and is committed to use **MXF::SDK** in this activity.

The involvement of IRT in the development of the **MXF::SDK** brings not only additional technical expertise, but also an end user view of MXF. This ensures that this SDK is designed from the start to allow end users to take full advantage of MXF features through your renewed MXF enabled product line.

Object Model Overview

The API of the MXF::SDK follows an Object-Oriented model designed to closely follow the MXF specification. This specification is split into a number of separate documents (Figure 1).

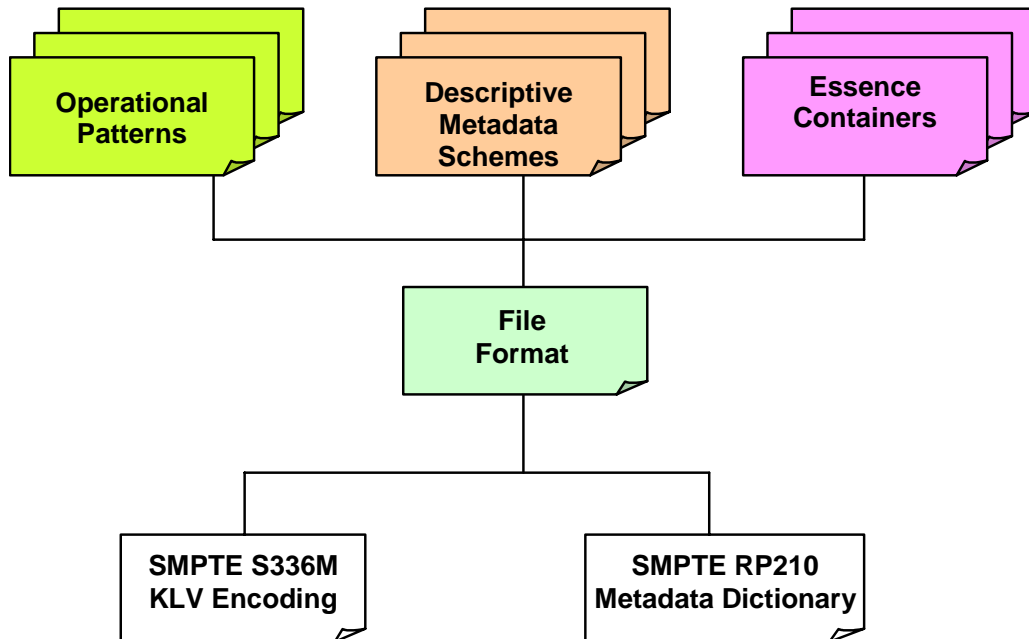


Figure 1 - MXF Specification document structure

The File Format specification document references other documents outside the specification. For this object model overview, the most important of these are SMPTE S336M and SMPTE RP210 (Figure 1).



*At the lowest level abstraction level, the **MXF::SDK** object model provides an interface for manipulation of data in KLV format.*

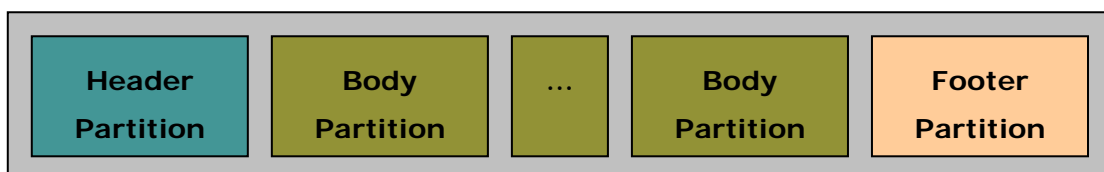


Figure 2 – an MXF file is physically a sequence of Partitions

The File Format document defines that a MXF file is physically divided in Partitions which can hold Essence and/or Metadata.



On top of the KLV interface, the MXF::SDK object model includes a set of objects for the manipulation of Partitions.

The Essence enclosed in Partitions is wrapped in an Essence Container, which is specified in additional documents (Figure 1), one for each type of Essence to be enclosed.



The MXF::SDK object model provides a plug-in mechanism that allows Essence Parsers to be connected. For each Essence type, an Essence Parser compliant with the Essence Container specification for that Essence type may be plugged in.

Each Essence parser is able to:

- *extract technical Metadata from the Essence;*
- *generate index tables; and*
- *KLV encode the Essence.*

The Metadata enclosed in Partitions is encoded as Sets of Metadata Items. Each of these items has a certain data type (e.g. integer, timestamp...).



The MXF::SDK defines a class for each data type in the specification, ensuring type safety on the manipulation of Metadata. E.g. if a Metadata Item is declared to be of type UInt16 (as defined in the specification), the UInt16 class ensures that only positive integers in a range of values that fits in 16 bits are allowed.

Metadata Items are grouped as Metadata Sets. An example is the Identification Metadata Set, which captures information on the author of changes to the file and includes Metadata Items such as “Company Name” and “Modification Date” in its group of attributes.

The MXF file format document specifies a group of Metadata Sets which is called Structural Metadata. While Partitions are related to the physical layout of the file, Structural Metadata Sets describe its logical organization.

Among other things, Structural Metadata defines the concept of Tracks and Source Clips. Therefore, a VERY simple MXF file can gather information as a timeline with tracks of Video and Audio, as shown in Figure 3.

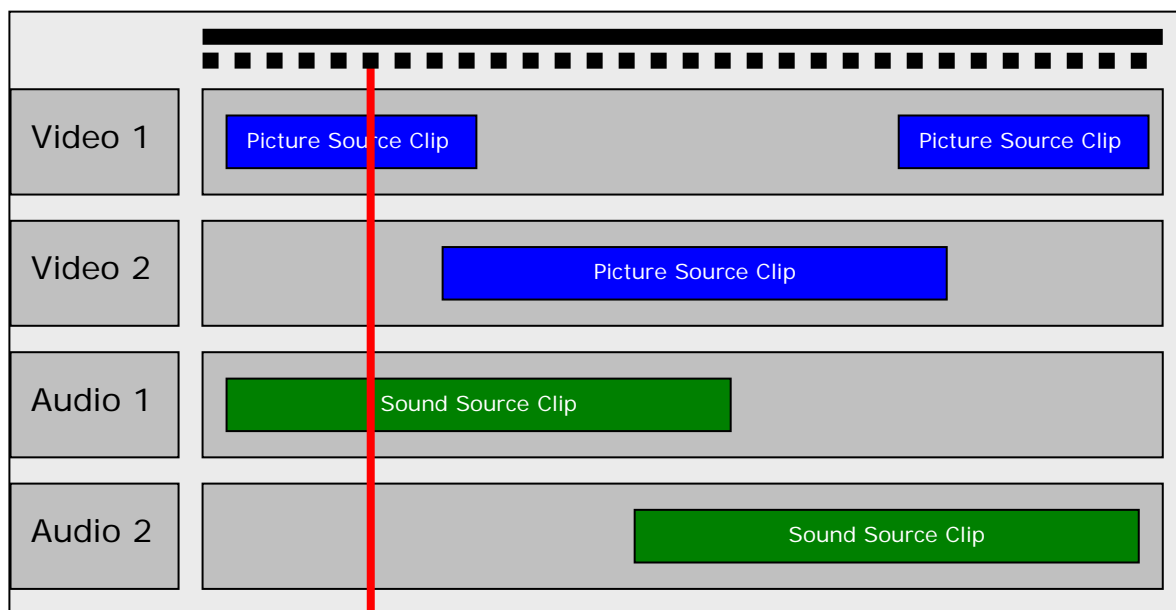


Figure 3 - A timeline holding tracks of Video and Audio where Source Clips may be placed

Track and SourceClip are examples of Metadata Sets included in Structural Metadata.



For each Structural Metadata Set defined in the specification, the **MXF::SDK** defines a class that provides access to the attributes of that specific Set. E.g. you can access the “Edit Rate” attribute on an object of class Track.

These classes also gather the rules imposed by the semantics defined in the specification. E.g. an object of type Track that is configured to hold Sound Source Clips will not allow you to add a Picture Source Clip to it.

Obviously, MXF files can gather a lot more information than the simple timeline in the example above. You can see the MXF File Format specification as a set of mechanisms and in your application you may not need to use them all. Therefore, MXF defines the concept of Operational Pattern, which is basically a set of constraints on the File Format document that defines a specific level of complexity.

This enables you to state “my MXF files comply with Operational Pattern X” and this means that you are using only the subset of MXF mechanisms allowed by the document specifying that Operational Pattern.



*The **MXF::SDK** object model includes a mechanism to plug-in classes (OPBuilders) that verify the MXF file structure, both physical and logical, and guarantee that it is according to the Operational Pattern that was selected for that specific file.*

Since an OPBuilder has some knowledge of the logical structure (Structural Metadata organization) that must be followed, it helps you build your MXF file by generating an initial skeleton of the file structure.

All standardized Operational Patterns are already plugged into the SDK.

Now, suppose that the timeline in **Figure 3** is ready and that it represents an interview. Assume we want to add some description about the person being interviewed, some information on the location where the interview took place or the subject of the interview. That kind of information is called Descriptive Metadata.

MXF not only allows you to add this kind of information to MXF files but also provides you mechanisms that enable you to synchronise this Metadata with the Essence. **Figure 4** illustrates how this is done.

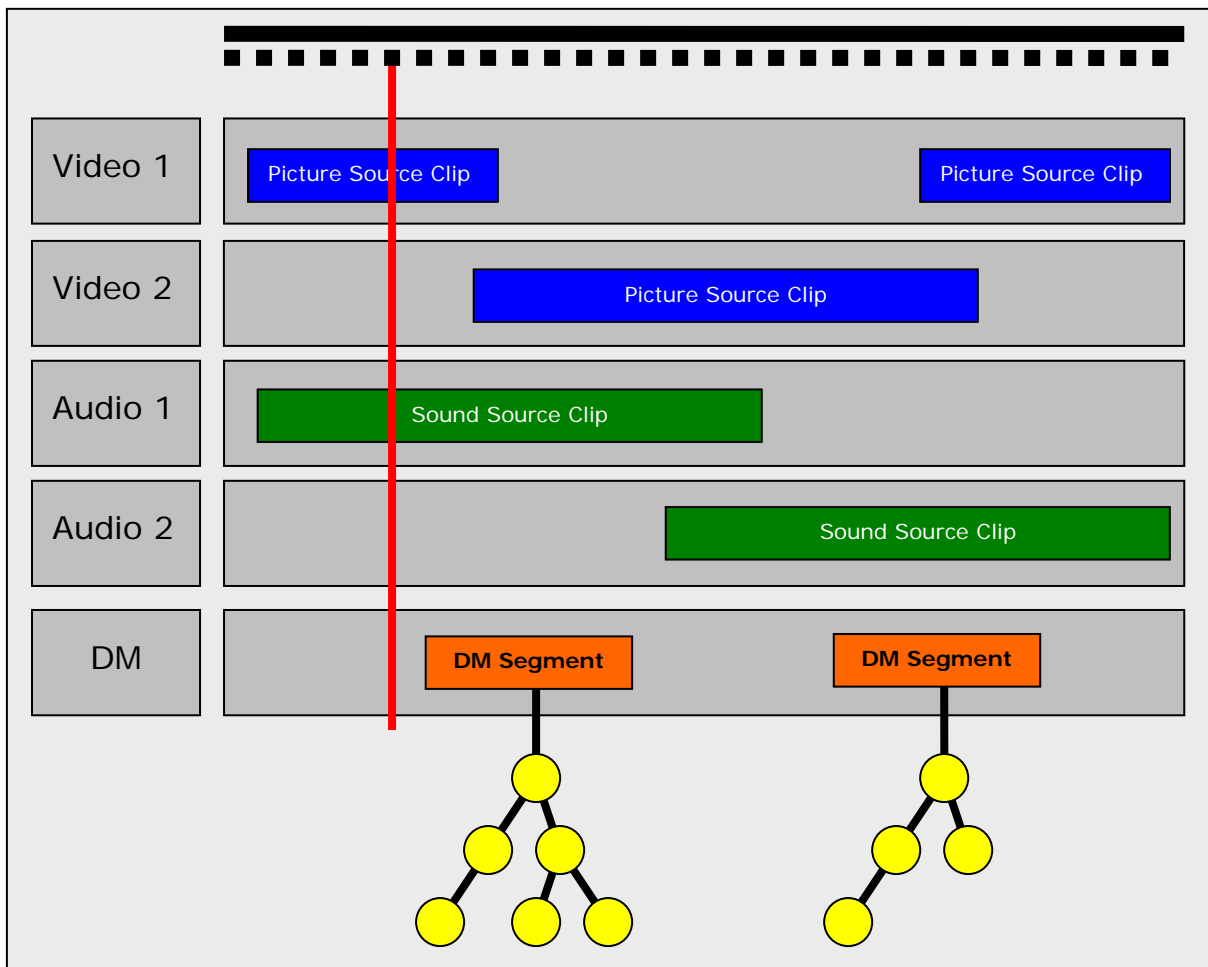


Figure 4 – Adding Descriptive Metadata

To add Descriptive Metadata you just add one or more Descriptive Metadata (DM) Tracks and add “DM Segments” to the Track(s). By placing a DM Segment on the Track, you are synchronising it with a specific clip of your Essence timeline.

Following the example above, one of the segments could be synchronised with the footage of the interview and you could add information like the person’s name and organisation to the DM Segment. The DM Segment’s origin and duration could then match the part of the footage with a close up on the person.

There are several Descriptive Metadata Schemes (DMS’s) out there so, which one can you use?

The MXF specification includes a DMS definition (Figure 1) named DMS-1, S380M. But you can use the DMS you want (you can even define your own!), provided you can express it in terms of Metadata Sets, with attributes and associations between Metadata Sets as defined in the File Format document.

Whenever you define a new DMS you must define a Metadata Set that is the entry point to your DMS and is to be pointed to from a DM Segment. You can then extend your DMS by adding more Metadata Sets and connect them using the Metadata Set association mechanism defined in the MXF File Format document. This concept is represented by the tree structures in **Figure 4**, where each circle is a Metadata Set for which you define any attributes you want.



The *MXF::SDK*'s object model includes a *DMSegment* class that captures the semantics just defined. Just as you can obtain *SourceClips* and *DMSegments* from a *Track* object, you can also navigate to your DMS entry *Metadata Set* from the *DMSegment*.

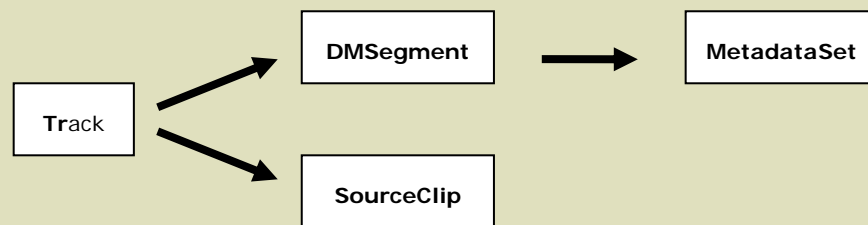


Figure 5 – Navigating from Structural to Descriptive

When you navigate from the *DMSegment* into the *Descriptive Metadata* direction, the *MXF::SDK* object model becomes *Description Scheme independent*. From here on the *Metadata Sets* are represented as objects of the generic type "*MetadataSet*".

This *MetadataSet* class still provides an interface that allows you to access its attributes but, if you intend to build an application that is *DMS independent*, you can also use this class's interface to dynamically discover its attributes and associations to other *MetadataSets*.

All Structural Metadata defined in the MXF specification, and all Descriptive Metadata defined in DMS-1, uses the concept of Metadata Sets as groupings of Metadata Items defined in the SMPTE RP210 Metadata Dictionary.

Both the Metadata Sets and the Metadata Items have associated keys, for KLV encoding/decoding, type information and documentation.



MXF::SDK's object model provides detailed access to this information as well.

With **MXF::SDK**, Descriptive Metadata Schemes are specified using the latest technology in schema definition, XML Schema. The **MXF::SDK** already includes a XML Schema representation of the DMS-1 Metadata Sets and the subset of RP210 Metadata Items used in these Sets. When you define your own DMS, you must formally specify it in XML Schema as well. Then you can just feed it into the SDK.

When you load your DMS XML Schema into the **MXF::SDK**, the schema information is exposed to you as an Object-Oriented API that allows you to navigate in your DMS structure.

MetadataSet and related classes hold the actual data, as described above, and they are designed to model the MXF Metadata Set mechanism. The API for DMS navigation provides information on how the Model is organized, so it is called MetaModel.

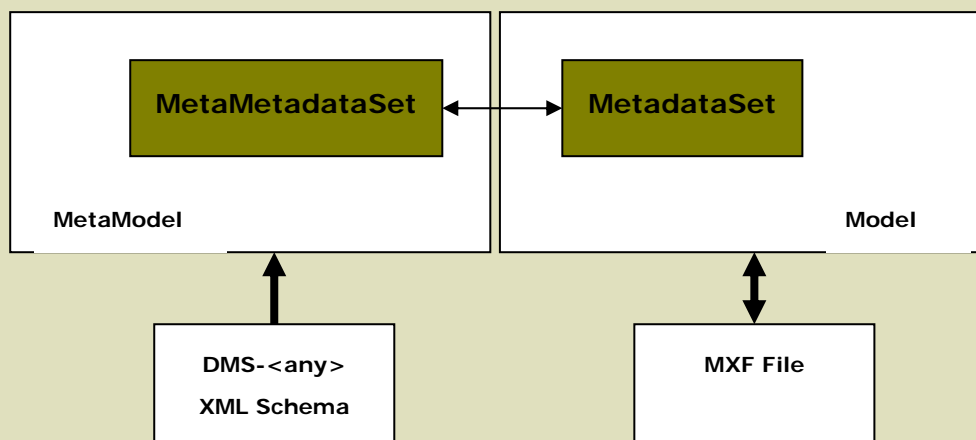


Figure 6 – The MetaModel describing the Model

Not only can you get and set attribute values on a MetadataSet class, but you can also navigate to its MetaMetadataSet to answer questions like “what attributes does Metadata Set ‘X’ have?” or “what is the documentation for this Set?”

With this object model, the **MXF::SDK** provides an object-oriented API that allows you to manipulate Descriptive Metadata in such a way that is Descriptive Metadata Scheme independent!

(Please note that the object model includes a lot more classes to allow you to manipulate DMS information, but a thorough description is out of the scope of this document)

What is in the package?

The **MXF::SDK** provides you with a C++ implementation of an Object-Oriented API. It is designed to be easy to use while still allowing you to take full advantage of all MXF mechanisms to build any kind of MXF-enabled product or to upgrade existing ones.

The **MXF::SDK** API is fully documented and accompanied by sample applications and tutorial kind of documentation that will bring your developers quickly up to speed on MXF technology.

The **MXF::SDK** API gives you different levels of abstraction, allowing you to manipulate MXF files at different levels, from KLV encoding to physical or logical layout.

With **MXF::SDK**, a single tool can be used to integrate any Descriptive Metadata Scheme you want, or even keep your products data model independent.

The **MXF::SDK** provides extensibility interfaces that allow future MXF developments to be easily integrated without breaking your products: plug-in new Operational Pattern structure builders, new Essence analysers or new Descriptive Metadata Schemes.

The **MXF::SDK** is backed by an experienced team that will support the SDK and ensure it will track future MXF developments.

And in the end, if you still have any doubts on how your product line can be upgraded to MXF, or the way MXF can be integrated as part of the services you provide, MOG Solutions is at your service to help you catch the MXF train!