UMID Applications in MXF and Streaming Media

By Yoshiaki Shibata

This paper reports on work in the SMPTE standards community to enhance applications of the Unique Material Identifier (UMID) in the material exchange format (MXF) technology. While the latest SMPTE RP 205 specifying the UMID Application Principles was published at the end of 2014, the UMID Resolution Protocol is still under standardization. This paper demonstrates how various kinds of UMIDs in an MXF file are to be utilized to enable various UMID applications based on the UMID Application Principles. Furthermore, thanks to a type of UMID attached to every frame in an MXF file, there is a way for the UMID to be employed consistently and seamlessly between the worlds of an MXF file and a media stream composed of a sequence of frames. Some guidelines are proposed to maximize the interoperability of UMID applications in MXF and streaming media.

Keywords: UMID, Basic UMID, Extended UMID; UMID Application Principles, UMID Resolution Protocol, MXF, MXF Package UID, MXF Material Package, MXF File Package, MXF Generic Container

INTRODUCTION

The unique material identifier (UMID) is a globally unique audio-visual material identifier standardized by the SMPTE as SMPTE ST 330[1] and RP 205.[2] It is based on the recommendations of the European Broadcasting Union (EBU)/SMPTE Task Force for Harmonized Standards for the Exchange of Programme Material as Bitstreams, which addresses the file-based media production workflow we have today.[3] It was initially standardized by SMPTE in 2000 prior to the recommendation-based standardization of the material exchange format (MXF) in 2004.[4]

While the UMID itself is independent of any material format or packaging, it was a natural consequence from its origin that the UMID was adopted as a mandatory component in an MXF file to uniquely identify it as a material. Thus, as MXF has gained an ever-increasing role within file-based workflows, the UMID has also become more widely used within the media and entertainment (M&E) industry.

Contrary to the original intent upon its introduction, the UMID has been substantially useless in practice. Even the most anticipated use of the UMID as a globally unique material identifier to associate material with its external metadata has seldom been seen thus far.

In 2011, we pointed out[5] that this was because of a lack of UMID Application Principles and a UMID Resolution Protocol, which needed standardization. To address this issue, the SMPTE UMID Application Project[6] was established in April 2012.

At the time of this writing, the project has successfully standardized the UMID Application Principles (UAPs), the fundamental rules for UMID to be treated in a reliable and consistent way over media products from multiple vendors. These were published as SMPTE RP 205[2] at the end of 2014, while the UMID Resolution Protocol, a standard method for converting a UMID into the corresponding uniform resource locator (URL) for material uniquely identified by that UMID, is still in the process of standardization in SMPTE.

Although both the UAPs and the UMID Resolution Protocol are, by nature, agnostic to the format of a media file, the origins of MXF and UMID described above make it natural to expect that MXF files can take full advantage of the UMID applications compared with media files in other file formats.

This paper explores how the UMID in an MXF file can be proactively utilized to enable various UMID applications, including applications in a media stream resulting from the playout of the MXF file. This work can also be seen as a trial on how UAPs are to be embodied specifically in the context of MXF and media streaming.

This has been another major focus of the UMID Application Project. An intensive study was conducted and an internal report was submitted to SMPTE Technology Committee (TC) Metadata and Registers (30MR). In order to seek a wide range of feedback from M&E industry experts, a report titled "Study of UMID Applications in MXF and Streaming Media" was created from the internal report and made publically available as a SMPTE Standard Committee Report.[7] This paper provides the digest version of that report. Those interested in and desiring further study on this topic are invited to review the original Standard Committee Report.

UMID APPLICATION BASICS

UMID Format

An introduction to UMID application basics starts with the UMID format. The UMID is a byte string of either 32 or 64 bytes, which are called basic UMID and extended UMID, respectively. The ex-

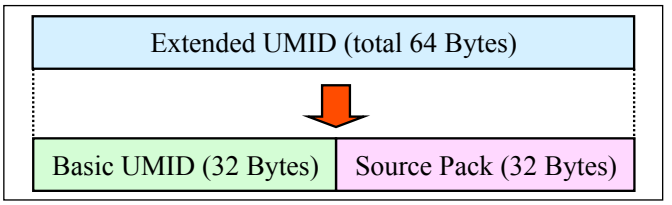

Figure 1. Top-level structure of UMID.

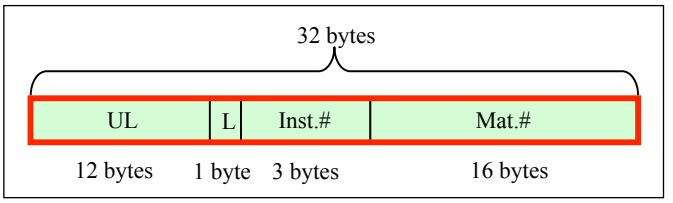

Figure 2. Basic UMID format.

tended UMID is composed of two parts: the 32 byte basic UMID (called "basic part") and the 32 byte source pack, which immediately follows the basic part, as shown in **Fig. 1**. The UMID is used either as the 32 byte basic UMID on its own or as the 64 byte extended UMID.

Figure 2 shows the basic UMID format, which is composed of the following four fields:

(1) *SMPTE universal label (UL):* The first 12 bytes of the basic UMID constitute the SMPTE UL registered to the *SMPTE Metadata Dictionary*[8] for the UMID. The first 10 bytes are fixed values, and the 11th byte indicates the type of material identified by the UMID. The 12th byte is divided into top and bottom nibbles, and it is used to indicate the number generation methods for the material number and instance number, respectively. The bottom nibble is also used to signal the "live stream"; i.e., when specified as "F_h," it indicates that the material attached with this UMID is a direct live signal source from a material creation device, implying that it is non-persistent and thus cannot be uniquely identified.

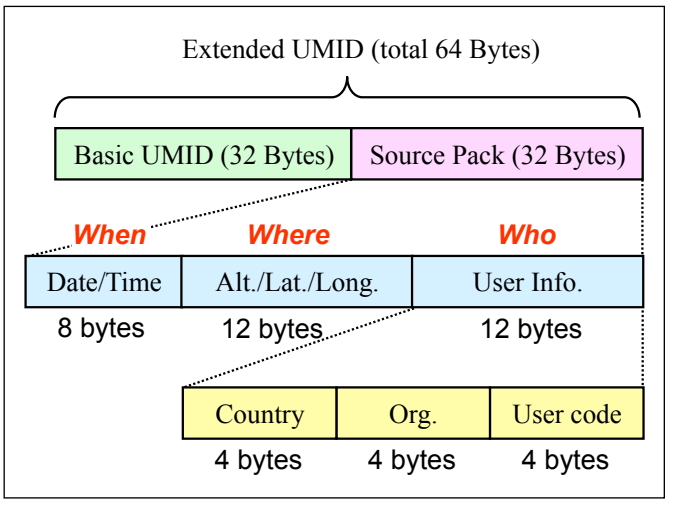

Figure 3. Extended UMID format.

(2) *Length (L):* This 1 byte field specifies the length of the byte string that follows. Since 19 bytes follow in the case of basic UMID, this field is fixed to 13_h, while it is 33_h for the extended UMID because of the additional 32 byte source pack that follows.

(3) *Instance number (Inst.#):* This 3 byte field specifies whether the material number is a newly created value or the inherited one from another UMID existing elsewhere. For a newly created UMID, this field must be zero-filled (00_h 00_h 00_h), indicating that the Mat.# is a newly created value.

(4) *Material number (Mat.#):* This 16 byte field accommodates a globally unique value at its generation, which makes the newly created UMID a globally unique material identifier. An example of such a value generation is given by a combination of the network node number of a device creating material and the time stamp at which the material is created. Because the network node number is globally unique, material with a UMID in this combination can also be globally uniquely identified when only a single material item is created at one time.

When a quantum duration of material such as a frame needs to be further uniquely identified within a material, the source pack is appended to the basic UMID, which forms the extended UMID, as shown in **Fig. 1**. Here, the quantum duration to be uniquely identified by the extended UMID is called material unit. **Figure 3** shows the extended UMID format, where the source pack is composed of the following three fields:

(1) *"When" field (date/time):* This 8 byte field specifies the date and time stamp at which a material unit was initially created. Because the timing granularity for this field is smaller than the cyclic period of material units, each material unit is distinguished with the value for this field.

(2) *"Where" field (alt./lat./long.):* This 12 byte field specifies the spatial coordinate information associated with the location at which a material unit was initially created. This field is further decomposed into three parts: altitude, latitude and longitude, each of which is 4 bytes long.

(3) *"Who" field (user info):* This 12 byte field specifies who initially created a material unit. This field is further decomposed into three parts: country, organization, and user code, each of which is 4 bytes long.

Double-Layered Material Identifications by UMIDs

Figure 4 schematically illustrates how the basic UMID and the extended UMID are utilized to uniquely identify material as a bounded sequence of frames and an individual frame in the material as a material unit, respectively.

As shown in the figure, the material is uniquely identified by the basic UMID, "U_1," being attached to all frames in the material. This UMID implementation is helpful especially for a linear recording device such as video tape recorder (VTR) because of its capability to access only a part of the material such as a frame at a certain point of time.

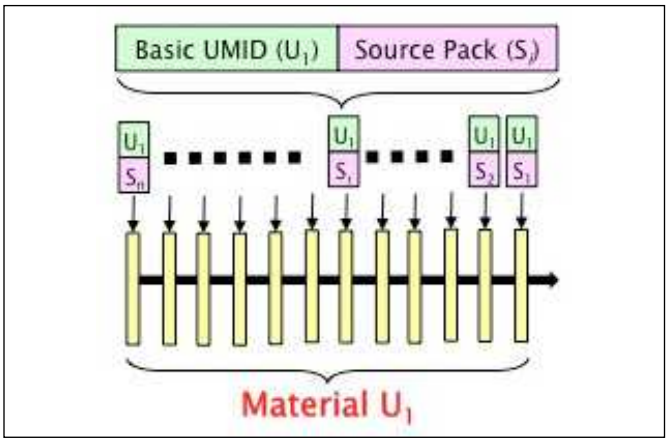

Figure 4. Double layered material identifications by Basic and Extended UMIDs.

An individual frame in the material (as a material unit) is distinguished by the source pack, "S_i" ($i = 1 - n$), which immediately follows the basic UMID as the basic part to form the extended UMID. Thanks to a finer granularity than the reciprocal of the frame rate for the "When" field, the source pack varies at every frame, and, therefore, the extended UMID, composed of the basic part uniquely identifying the material as a whole and the source pack distinguishing each frame in the material, can uniquely identify an individual frame in the material.

Note that while the original intention of the extended UMID was to identify an individual material unit that comprises the material, it also accommodates fundamental information about the creation of each material unit, which is preserved throughout the processes of the media production workflow chain.

Two Distinct Uses of UMID

The primary use of a UMID is as a globally unique material identifier. When new material is created from scratch by, for example, the camera acquisition, a new UMID is created and attached to the material automatically, so that it can be globally uniquely identified by the UMID value. In this case, while the Mat.# for the UMID is a newly created globally unique value (e.g., "FE_h DC_h BA_h ... 32_h 10_h"), the Inst.# must be zero-filled ("00_h 00_h 00_h") as show in **Fig. 5 (a)** to indicate that it is original material with a newly created Mat.# value.

Another use of a UMID is as a linking tool. When material is created from existing source material by, for example, partial retrieval, a UMID, for which the Mat.# is inherited from that of the source material while the Inst.# is set to a nonzero value, is attached to a resulting derived material.

Figure 5 (b) shows the case where the Mat.# value of an original material as a source, or "FE_h DC_h BA_h ... 32_h 10_h," is inherited, and a new nonzero value, or "01_h 98_h 76_h," is created to form a UMID to be attached to the resulting derived material. Consequently, the resulting material is logically associated with its source material via their Mat.#; in other words, the UMID globally uniquely identifying its source material is easily obtained by just masking the Inst.# of the UMID of the resulting material to zero.

It should be noted that while there are two uses of UMID as described above, those two uses are completely exclusive. If a UMID is used as a linking tool as shown in **Fig. 5 (b)**, it cannot be used as a globally unique material identifier for which the global uniqueness is always guaranteed, because the value space of the 3 byte Inst.# is far from sufficient to accommodate a globally unique value.

This leads to a conclusion that when the material at hand has a UMID with a nonzero Inst.#, the material is not original material but derived material for which the source material is globally uniquely identified by the UMID with its Inst.# masked to zero.

UMID as a Globally Unique Material Identifier

When a UMID is used as a globally unique material identifier, the most important role of the UMID is to unambiguously associate the material with metadata describing the material. Assuming an MXF file and an extensible markup language (XML) file are given as the material and its associated metadata, respectively, **Figure 6** schematically illustrates how they are associated via UMID.

In **Fig. 6**, the MXF files, "Material-1.mxf" and "Material-2.mxf," represent certain materials that are globally uniquely identified by the UMID "U_A" and "U_B," respectively. As will be discussed later, an MXF file has a field for the basic UMID in its file header. As a result, "Material-1.mxf" containing the UMID "U_A" value at its header is to be globally uniquely identified by the UMID "U_A."

When metadata associated with the material are given, they need to unambiguously specify the material they are associated with, and

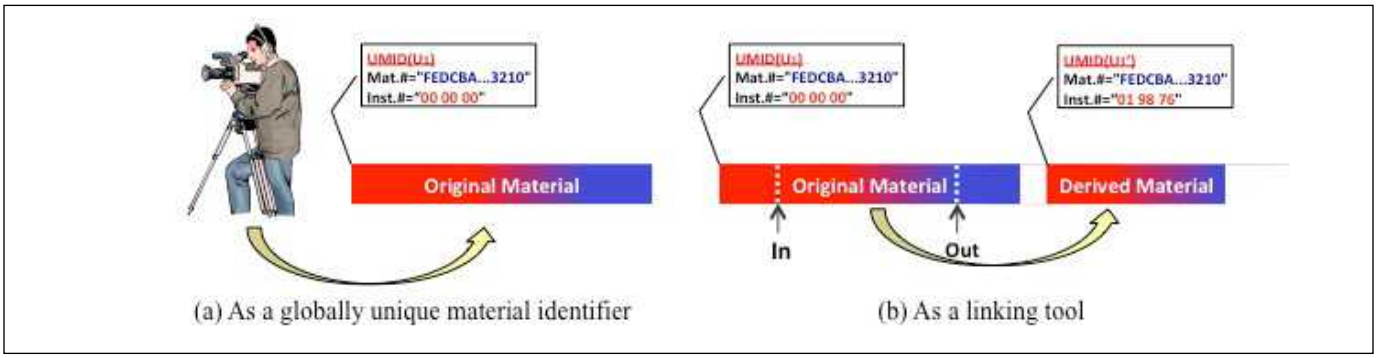

Figure 5. Two distinct uses of UMID. (a) As a globally unique material identifier (left). (b) As a linking tool (right).

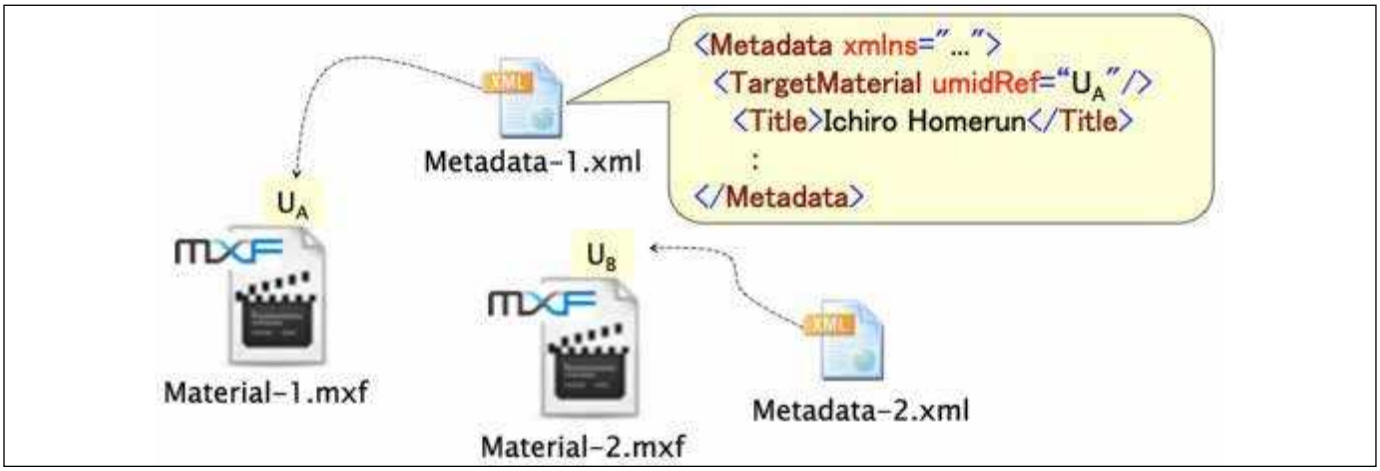

Figure 6. Material associated with metadata via UMID.

it is the UMID that works as a *hook* to hang the metadata. In the case of **Fig. 6**, "Metadata-1.xml" describing "Ichiro Homerun" for its title is associated with "Material-1.mxf" via the UMID "U_A" as represented by the value of the umidRef attribute of the TargetMaterial element in "Metadata-1.xml."

Note that such an association has been conventionally implemented by using a URL such as "http://server.example.com/materials/Material-1.mxf," which directly indicates where the material as an MXF file is located. The use of UMID, on the other hand, is regarded as an indirect logical way to create the association, which brings a benefit of much higher flexibility than the URL-based one, though some additional tools are required for the UMID-based association to work in practice, as discussed in the next subsection.

UMID-Based Material Search and UMID Resolution Protocol

One of the primary roles of metadata is its use for material search. Because of the huge difference in data sizes between material and its associated metadata, it is reasonable and in common practice for the metadata to be stored separately from the material.

Based on the association between material and its metadata via UMID, an application scenario, called UMID-based material search, is schematically demonstrated in **Fig. 7**, where materials are stored in various kinds of material servers ("Ingest Server," "Near-Line Material Server," "Archive," and "Playout Server") connected to the network in a media production system, and all the metadata associated with materials via respective UMID are collected and separately stored together in a dedicated metadata database ("Metadata Database") for their uniform management.

In this scenario, when an external application ("Application"), such as video editing, desires to obtain a material item that captures an "Ichiro Homerun" scene, for example, it will submit a query to the metadata database accordingly. The metadata database will then reply to the application with the desired material by its UMID.

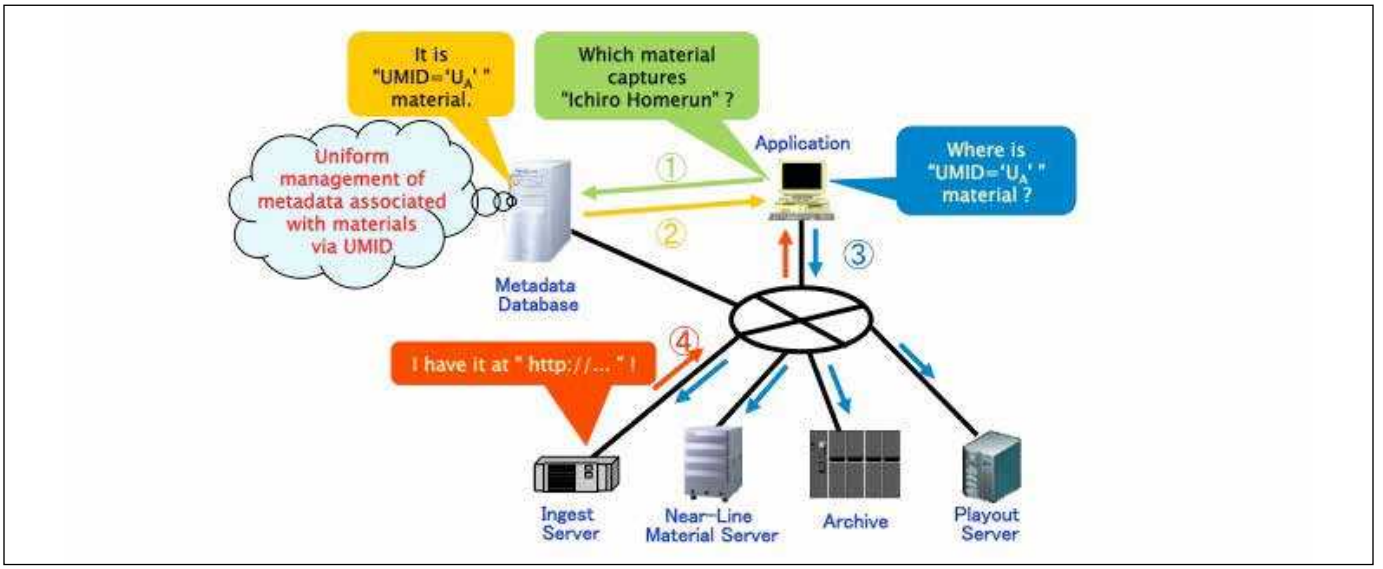

Figure 7. UMID based material search.

Because the UMID itself cannot tell anything about where to access the desired material, the application needs to resolve the UMID—to convert the UMID into its corresponding URL. In this scenario, the application distributes a query asking "Where is 'UMID = U_A' material?" to the material servers, and the ingest server ("Ingest Server") storing the desired material responds with the URL for the material, which is to be used by the application to actually access it.

Note that, in this application scenario, because each material server would come from a different vendor, the UMID Resolution Protocol, or the conversation method between the application and the material servers for the UMID resolution, needs to be industry standardized, which is the issue on which the UMID Application Project is working intensively at the time of this writing.

UMID Application Principles (UAPs)

In addition, the application scenario shown in **Fig. 7** implicitly assumes that all the materials stored in any material server are appropriately managed by using their UMIDs as globally unique material identifiers. However, what does it mean by "materials are appropriately managed by using their UMIDs"?

To answer this question, it is essential to clearly define the UAPs, the fundamental rules for the UMID to be treated in a reliable and consistent way over the media products from multiple vendors, which constitute the basis of any UMID applications.

Based on the analysis of several existing and future-envisaged UMID applications, the UMID Application Project successfully distilled the UAPs, which were then industry standardized in the latest SMPTE RP 205.[2]

In the following, UAPs composed of seven principles are briefly introduced so that they can serve as a basis for discussion on UMID applications in MXF and streaming media.

- *Principle 1: Definitions*

  This principle defines the terms used in the statements that follow, including a strict definition of "material" (the original material with UMID of newly created Mat.# and zero Inst.# values) and "instance" (the derived material with UMID of inherited Mat.# and nonzero Inst.# values).

- *Principle 2: UMID Creation*

  This principle specifies when new material is created, a UMID with a newly created Mat.# and a zero Inst.# values must be created and attached to the material **(Fig. 5 (a))**.

- *Principle 3: UMID Integrity*

  This principle specifies that different materials must be globally uniquely identified by different UMIDs.

- *Principle 4: UMID Identification*

  This principle specifies if more than one material is uniquely identified by a single UMID, their representations at playout must be identical bit by bit on the time line. In other words, what a UMID uniquely identifies is not a media file, but a baseband bit stream at its playout in a strict sense according to this principle.

- *Principle 5: UMID Inheritance*

  This principle denotes that an instance derived from original material can be attached with the UMID of the Mat.# being inherited from that of the original material and an Inst.# set to a nonzero value **(Fig. 5 (b))**.

- *Principle 6: Extended UMID*

  This principle recommends that the extended UMIDs attached to the material units within the material share the same basic UMID that uniquely identifies the material as a whole (**Fig. 4**).

- *Principle 7: Source Pack*

  This principle recommends that the source pack in the extended UMID, once created, not be replaced with a new one, so that the source pack attached at an initial material creation will be preserved throughout the processes of the media production workflow chain.

MXF OVERVIEW

What is the MXF?

Structure of an MXF File

An overview of MXF technology is briefly introduced here, starting with the structure of an MXF file. While MXF has elaborate file format specifications, only those needed in the following discussions are described. Hence, readers desiring a full knowledge of the MXF specifications are directed to the relevant SMPTE standards[4] and/or literature.[9]

MXF is a container format in the sense that an MXF file may contain any kind of audiovisual essence(s) as well as metadata such as time code and titles.

An MXF file, in its physical representation, is composed of a sequence of SMPTE key-length-value (KLV) coded chunks of data

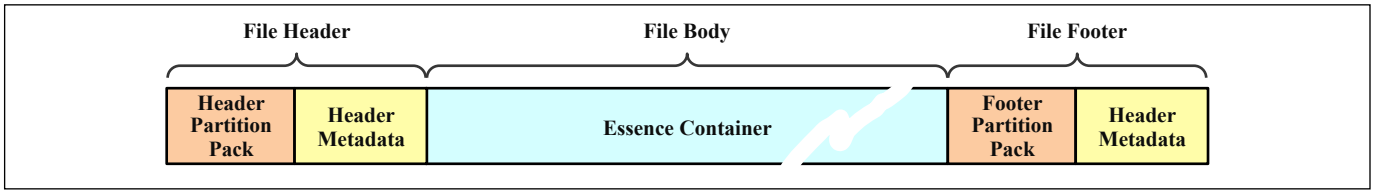

Figure 8. Structure of an MXF file.

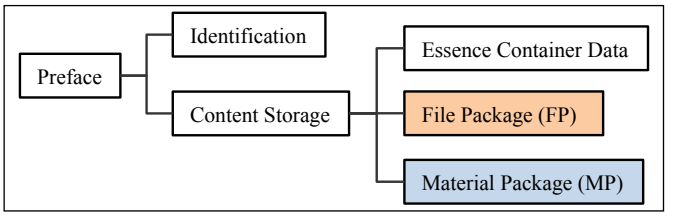

Figure 9. Logical structure of MXF Header Metadata.

(hereafter called KLV packet),[10] which logically form a file header, starting with a header partition pack, and a file footer, starting with footer partition pack, and the file body placed between them, as shown in **Fig. 8**.

In a typical MXF file, the essence is individually KLV wrapped to form an essence container, and it is stored in the file body. The granularity of the essence data to be KLV wrapped varies depending on the type of MXF file, ranging from per frame (frame wrapped) to an entire clip (clip wrapped). In the case of the frame-wrapped essence, metadata associated with a frame such as time code are also attached to the KLV packet containing the frame.

Structural and Descriptive Metadata

Metadata associated with an MXF file are called MXF header metadata and contained in the file header and additionally in the file footer if desired. Two kinds of header metadata are defined in MXF: structural metadata and descriptive metadata.

The structural metadata describes one or more essence types and their relationship along a time line at their playout. The structural metadata is mandatory because it describes how to playout the essence(s) by specifying their synchronization and technical parameters such as the frame rate. Consequently, modification of structural metadata has a strong impact on the playout of an MXF file.

The descriptive metadata provides information mainly for human use, such as a title and content synopsis. While it is important for an efficient material search, it is not mandatory in the sense that an MXF file can be played out without the descriptive metadata. Hence, we will not go into more details about it in this paper.

Logical Structure of MXF Header Metadata

The MXF header metadata content by itself is also a sequence of the KLV packets, but they logically form a tree structure by connecting them via a so-called "strong reference." **Figure 9** shows a simplified logical structure of the MXF header metadata (the structural metadata).

In **Fig. 9**, the preface as a root of the MXF header metadata signals the kind of MXF file, such as a type of MXF operational pattern (OP), and the types of essence container, including essence kinds contained in the MXF file.

Below the preface are the identification and the content storage. The identification describes the media product used to produce the MXF file and when, and the content storage contains three fundamental metadata items, the essence container data, the material package (MP), and the file package (FP) (also called, top-level source package).

The essence container data describes the individual essence container in the file body. Theoretically, an MXF file is permitted to contain more than one type of essence, such as the high-resolution picture essence and its proxy, each of which is stored in a specific essence container, which is then described by the essence container data, including the FP associated with the essence container.

According to SMPTE ST 377-1,[4] the MP and FP are specified to describe the essence on a time line at the playout of an MXF file (called "the output time line") and a time line for the essence stored in an essence container (called "the input time line"), respectively. In fact, this is one of the most characteristic parts of the MXF technology, and it is also vital for the UMID applications in MXF as to be discussed later.

MXF Internal Behavior Model

MXF with a Dual-Layered Structure

Among other valuable characteristics of the MXF technology, its dual-layered structure is one of the most distinguishing characteristics of MXF. In short, what is to be produced to output at the playout of an MXF file is not always the same as what is contained in the MXF file, as shown in **Fig. 10**, which schematically demonstrates that only the middle part of the essence specified by the in/out points in the file body (the essence container) of an MXF file is produced to output at its playout.

The FP and MP describe the input and output time lines, respectively. More specifically, the FP attaches a time line to a sequence of chunks of essence data in the essence container, while the MP specifies how they are to be placed on the time line at the playout. As a result, though the input time line can contain discontinuity (e.g., for the composite essence container), the output time line must be always contiguous, while both are represented using time code.

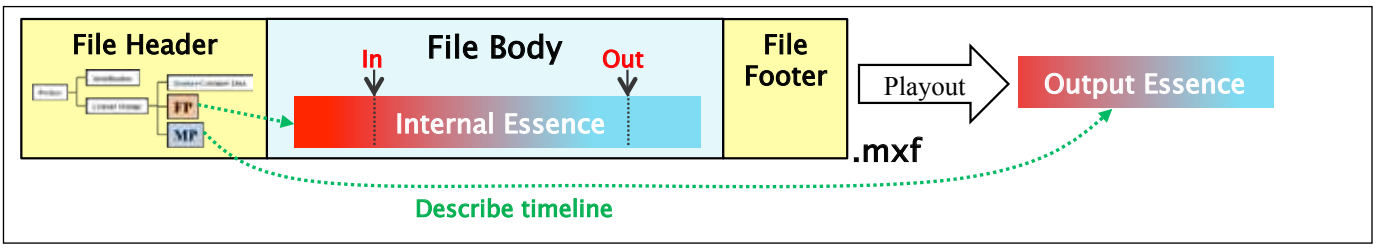

Figure 10. An MXF file at playout.

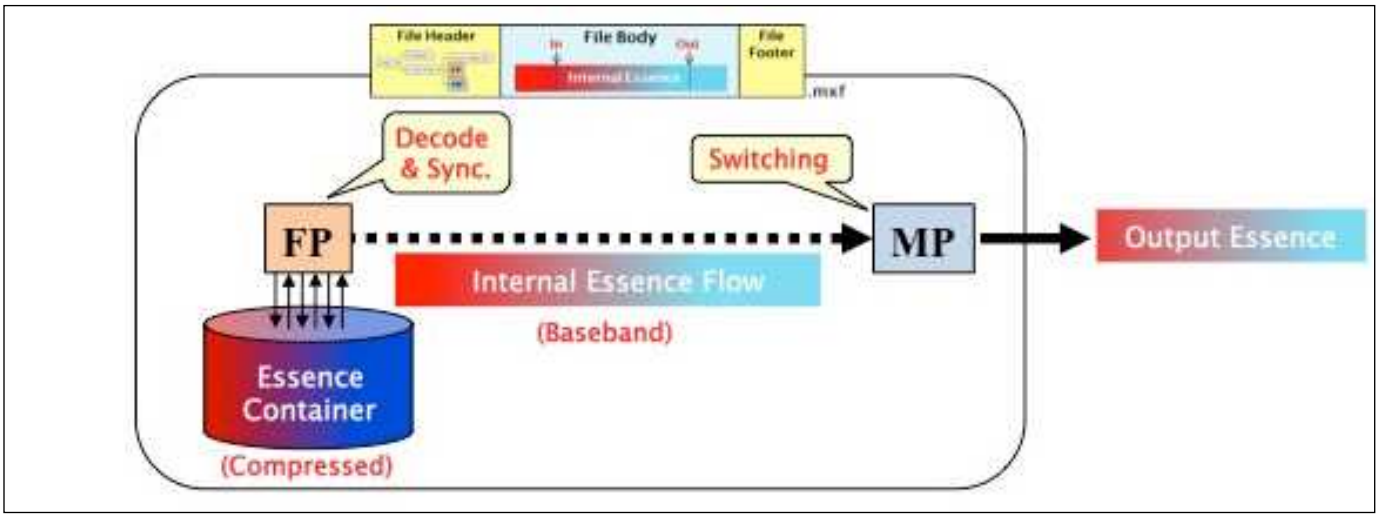

Figure 11. MXF internal behavior model at playout.

MXF Internal Behavior at Playout

Another interesting point to be noted is that although the internal essence in the file body can take various forms, from those based on a variety of compression schemes to an uncompressed one, the output essence resulting from the playout is always the baseband signal we can directly *perceive* (via devices such as a video monitor or the sound speakers, of course).

Taking the roles of the MP and the FP into consideration, the MXF internal behavior that would occur at the playout of an MXF file is schematically demonstrated in **Fig. 11**.

Based on **Fig. 11**, the MXF internal behavior model is described as follows: the essence data, encoded by a certain compression scheme and stored in the essence container, are fetched and decoded by the FP. After the synchronization, if more than one type of essence data is decoded, the FP supplies the baseband essence flow to the MP, which is then switched by the MP so that only the part specified by the in/out points is produced to output.

Because of such roles of the MP and FP, the following features are observed:

- The FP describes not only the temporal information of an essence container, but also technical properties of the essence data stored in it such as the codec, the sampling rate (frame rate), a frame size, and so on, by using so-called file descriptor, so that it can provide the internal essence flow as a baseband signal.
- The MP describes not only the temporal information of an essence at playout, but also the switching information for it so that it can control the essence flow to be actually produced to output. No file descriptor is attached to the MP because it receives the (already decoded) baseband signal from the FP as input.

Note that the baseband essence flow supplied by the FP to the MP is *imperceptive* and thus cannot be accessed from an external application because it is just a conceptual essence flow derived, based on the MXF internal behavior model.

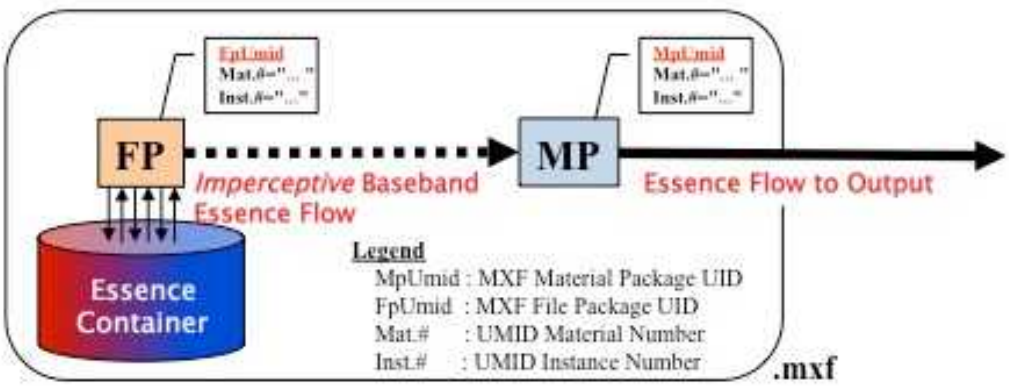

Figure 12. MXF Package UID's (MpUmid an FpUmid).

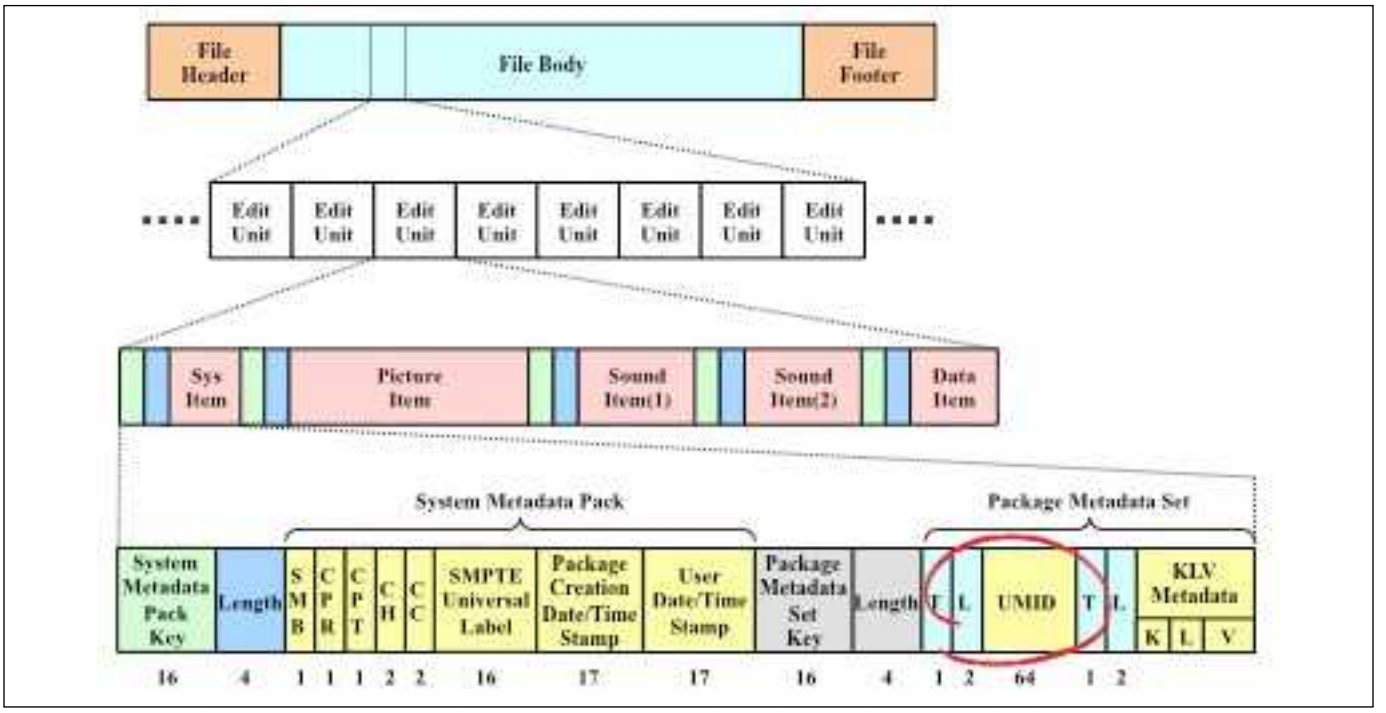

Figure 13. MXF Generic Container in file body.

UMID APPLICATIONS IN MXF

UMIDs in an MXF File

Material Package UID (MpUmid) and File Package UID (FpUmid)

According to SMPTE ST 377-1,[4] the basic UMID is used as a unique identifier (UID) of the package that describes the essence on a time line. While the package UID is originally introduced to uniquely identify a package instance within an MXF file by definition, it is also considered to uniquely identify the essence flow the package supplies based on the MXF internal behavior model. **Figure 12** reproduces the MXF internal behavior model with the package UIDs being attached to the MP and the FP.

In the following, let MpUmid and FpUmid denote the package UIDs attached to the MP and FP, respectively. As a result, the MpUmid uniquely identifies the essence flow to output, which we can *perceive* when an MXF file is played out, while the FpUmid uniquely identifies the *imperceptive* baseband essence flow that the FP with it supplies to the MP, as shown in **Fig. 12**.

Body UMID (BodyUmid)

In addition to the package UIDs, there is another kind of UMID in an MXF file called body UMID. Specifically, it is either the basic or the extended UMID associated with a frame (or a group of frames if desired) and inserted into the MXF file body via the so-called MXF generic container[11] as an essence container. **Figure 13** schematically illustrates an example of the detailed structure of the MXF generic container containing the body UMIDs.

As shown in **Fig. 13**, the file body is composed of a sequence of so-called edit units, which are on per frame basis in this case. An edit unit for a frame is then composed of the system item, the picture item containing picture essence data per frame, the sound items containing sound data synchronized with the frame, and finally the data item for the frame, and it is the system item that stores the body UMID as indicated by the red circle in **Fig. 13**,[12] which is denoted as BodyUmid hereafter.

Applications of MpUmid (Material Package UID)

What Does the MpUmid Identify?

While the originally expected role of the MpUmid is a unique identifier of an instance of the MP within an MXF file, it is also considered to uniquely identify the baseband essence flow produced to output at the playout of an MXF file (i.e., what we actually observe when the MXF file is to be played out).

According to the UAP 4 (UMID Identification), it is a representation of the material at its playout, which a UMID uniquely identifies in a strict sense. Therefore, it is the MpUmid, which uniquely identifies the material that is stored in the form of an MXF file.

It is worthwhile to note that in many cases, the MpUmid is also regarded as a unique identifier of an MXF file, but this is not always true.

In fact, there exists a complicated MXF file that can produce more than one type of baseband essence flow at its playout by containing multiple MPs, each of which corresponds to an individual baseband essence flow. Because each MP, and thus the baseband essence flow it produces, has its own MpUmid, a single MXF file can be uniquely identified by more than one MpUmid in this case.

Furthermore, there is a case where multiple MXF files are uniquely identified by a single MpUmid. Because multiple media files can share a single UMID when they produce an identical result at their playout according to the UAP 4 (UMID Identification), a single MpUmid uniquely identifies multiple clones of an MXF file all at once, though they can be distinguished by different MpUmid if desired.

MpUmid as a Globally Unique Material Identifier

Because of the MpUmid uniquely identifying a material as an MXF file in general, the most important application of the MpUmid is its use as a globally unique material identifier, which corresponds to "U_A" and "U_B" in **Fig. 6**.

Since the UMID as a globally unique material identifier is its primary use, a comprehensive discussion is given in the latest SMPTE RP 205,[2] in which the most important concept is the UMID managed domain that MXF files with the MpUmid need to constitute.

The UMID managed domain is a conceptual domain composed of materials with valid UMIDs in the sense of UAPs 2 to 4 (UMID Creation, Integrity, and Identification), which are applied to the UMID as a globally unique material identifier.

In order for the MpUmid in the UMID managed domain to be always maintained valid in the sense of those UAPs, certain MpUmid treatments are always required at every manipulation of an MXF file in the domain. These are executed by the material manager.

The material manager is not only a tool responsible for management of MXF files, but it also maintains the integrity of MpUmids assigned to them in the UMID managed domain. In addition, because an MpUmid by itself tells nothing about where to access a desired MXF file identified by the MpUmid, the material manager is also expected to maintain the correspondence between the MpUmid and a URL of the MXF file by using a mapping list.

While the detailed MpUmid treatments for various manipulations of an MXF file are described in the Standard Committee Report,[7] some treatments are worth mentioning here.

For New MXF File Creation in the UMID Managed Domain

When a new MXF file is created from scratch by, for example, the camera acquisition, in the domain, the material manager assigns a newly created MpUmid (composed of newly created Mat.# and zero Inst.# values) to the MXF file, and it registers the pair of the MpUmid and the file's URL to the mapping list.

For a New MXF File Import into the UMID Managed Domain

When an MXF file existing elsewhere is imported into the domain, the material manager usually replaces the MpUmid of the incoming MXF file with a newly created UMID value (and registers the pair of the new MpUmid and the file's URL to the mapping list) because it cannot trust the validity of the MpUmid of an incoming MXF file in general.

An exception occurs when the location from which an MXF file is to be imported is also known as a UMID managed domain in advance. Because the MpUmid in this case is valid by definition, it can be reused when the MXF file is imported into the domain exactly as is according to UAP 4 (UMID Identification).

For an Existing MXF File Modification at its Essence in the UMID Managed Domain

When an MXF file existing in the domain is modified at its essence by, for example, insert editing, the material manager usually finds the MpUmid of the MXF file, replaces it with a newly created UMID value, and reflects the change in the mapping list.

This MpUmid replacement is required because the existence of identical MXF files sharing the same MpUmid elsewhere is unknown, which could otherwise lead to a breach of UAP 3 (UMID Integrity).

An exception is given in a special case when the modification causes no change to the playout result of an MXF file at all, such as a mathematically lossless picture encoding, because of the UAP 4 (UMID Identification).

Applications of FpUmid (File Package UID)

What Does the FpUmid Identify?

While the originally expected role of the FpUmid was as a unique identifier of an instance of the FP within an MXF file, it is also considered to uniquely identify the internal baseband essence flow supplied to the MP by the FP with the FpUmid.

Unlike the MpUmid, the global usefulness of FpUmid is insignificant because the internal baseband essence flow supplied by the FP is just a conceptual entity that no external application can access.

On the other hand, the uniqueness of FpUmid within an MXF file is crucial because the MP needs to unambiguously specify the FP(s) from which the internal baseband essence flow(s) is supplied by using its FpUmid (via so-called "source clip").

FpUmid as a Linking Tool

Although the FpUmid is also used as a globally unique material identifier, its usefulness is very limited, as discussed in the Standard Committee Report.[7] Taking account of the scope of uniqueness of the FpUmid within an MXF file, the use of FpUmid as a linking tool further enhances its usefulness.

Figure 14 schematically demonstrates such a use of the FpUmid, where an originally created MXF file called "Original Material.mxf" is partially retrieved, with its in/out points being specified to create a derived MXF file called "Derived Material.mxf."

When "Original Material.mxf" is created, its MpUmid ("U_1") is newly created with its newly created Mat.# of "FE_h DC_h BA_h ... 32_h 10_h" and zero Inst.# values, so that the MXF file is independently managed by using its MpUmid as a globally unique material identifier.

Because "Derived Material.mxf" is also a newly created MXF file to be managed independently, its MpUmid as a globally unique material identifier ("U_3") is also newly created with a different Mat.# value ("98_h 76_h 54_h ... AB_h CD_h").

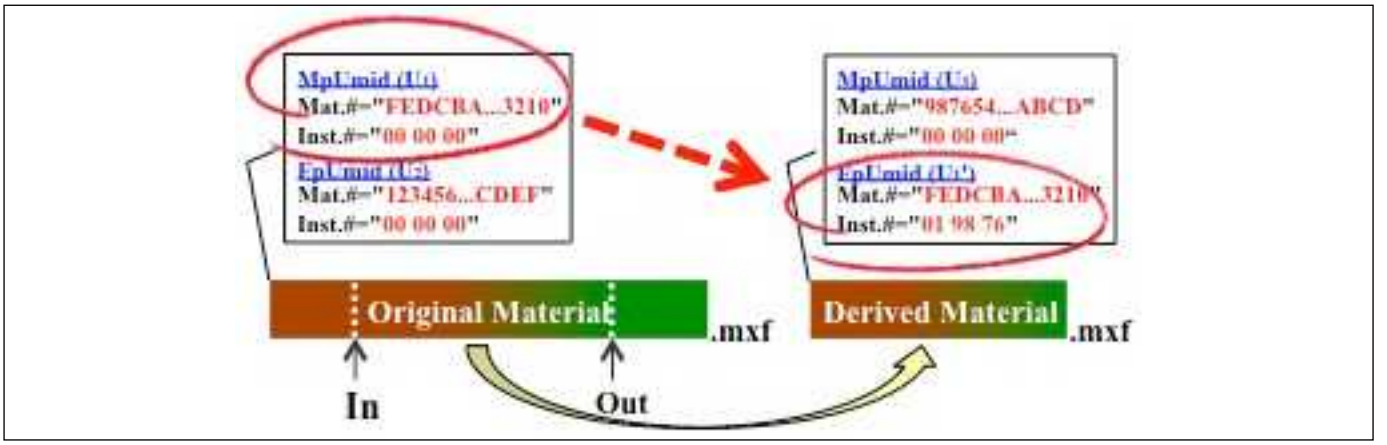

Figure 14. FpUmid application for partial retrieval of an MXF OP1a file.

When the FpUmid is employed as a linking tool, there is a way for the FpUmid of "Derived Material.mxf" ("U1′") to be created based on the MpUmid of "Original Material.mxf"; the Mat.# of the FpUmid inherits the value "FE_h DC_h BA_h ... 32_h 10_h" from the MpUmid of "Original Material.mxf," and its Inst.# is set to a non-zero value ("01_h 98_h 76_h") in the case of **Fig. 14**.

This kind of FpUmid treatment brings additional usefulness for the FpUmid. As it is obvious in the figure, the resolution of FpUmid of "Derived Material.mxf" with its Inst.# masked to zero will lead to the URL of "Original Material.mxf." In other words, thanks to the FpUmid as a linking tool, a source MXF file from which a derived MXF file is created is easily obtained by using the UMID Resolution Protocol.

Note that while the partial retrieval is taken for example in this case, this use of FpUmid is applicable to any kind of media processing that treats an MXF file as input and output, such as transcoding, a text overlay, and so on.

Applications of BodyUmid (Body UMID)

What Does the BodyUmid Identify?

As shown in **Fig. 4**, the BodyUmid as an extended UMID globally uniquely identifies a material unit by definition. This is valid when its basic part is a globally unique material identifier with a newly created Mat.# and zero Inst.# values. However, what happens to the BodyUmid when its basic part is a linking tool (the inherited Mat.# and nonzero Inst.# values)?

According to UAP 6 (Extended UMID), the basic part of BodyUmid in an originally created MXF file is equivalent with the MpUmid globally uniquely identifying the MXF file as a whole. It is therefore logically expected that when the basic part of the BodyUmid attached to a material unit as a linking tool is resolved with its Inst.# masked to zero by using the UMID Resolution Protocol, the URL of the original MXF file from which this material unit derives in any way is obtained.

More specifically, there should exist a material unit for which the BodyUmid shares the Mat.# but has zero Inst.# values in the original MXF file, from which the material unit in question derives, though the source pack is required to determine the exact position of the material unit in the original MXF file.

BodyUmid Applications for an Original MXF File Creation from a Live Feed

While the material unit is defined as a quantum duration of material such as a frame, it is also regarded as the quantum duration, the sequencing of which forms a media stream. In both cases, each quantum duration is uniquely identified by the extended UMID.

Figure 15 schematically demonstrates an original MXF file creation by capturing a live feed from a camera as a media stream. The media stream generated by the camera is composed of a sequence of frames as the material units, and the extended UMID is attached to each frame for its own unique identification in the stream.

Thanks to the source pack in the extended UMID, the information on "when," "where," and "who" creates each frame, which is also automatically generated by the camera typically using an internal clock, a global positioning system (GPS) unit, and the system preset values, respectively, is attached to the frame.

An MXF recorder that receives the media stream creates an original MXF file. More specifically, after the file header creation, the MXF recorder receives incoming frames as the material units and stores them as the edit units with newly created BodyUmid being assigned to each edit unit during the file body creation.

In **Fig. 15**, when a new frame is created by a camera, a newly created extended UMID ("U_pS_i") is attached to the frame, in which its basic part ("U_p") is a constant value specific to the camera, and its source pack ("S_i") is newly created specifically for the frame. Note that the bottom nibble of byte 12 in the extended UMID is set to "F_h" to signal the "live stream," or it indicates that a frame with the UMID has never been recorded to form a persistent material so far, while its top nibble of "1_h" indicates the "SMPTE method" used to create the Mat.# value.[1]

Because an original MXF file created by the MXF recorder is managed independently, its MpUmid ("U_1") is newly created with a new Mat.# ("1xx") and zero Inst.# values. Its FpUmid ("U_p'") is created by inheriting the Mat.# value in the basic part of the ex-

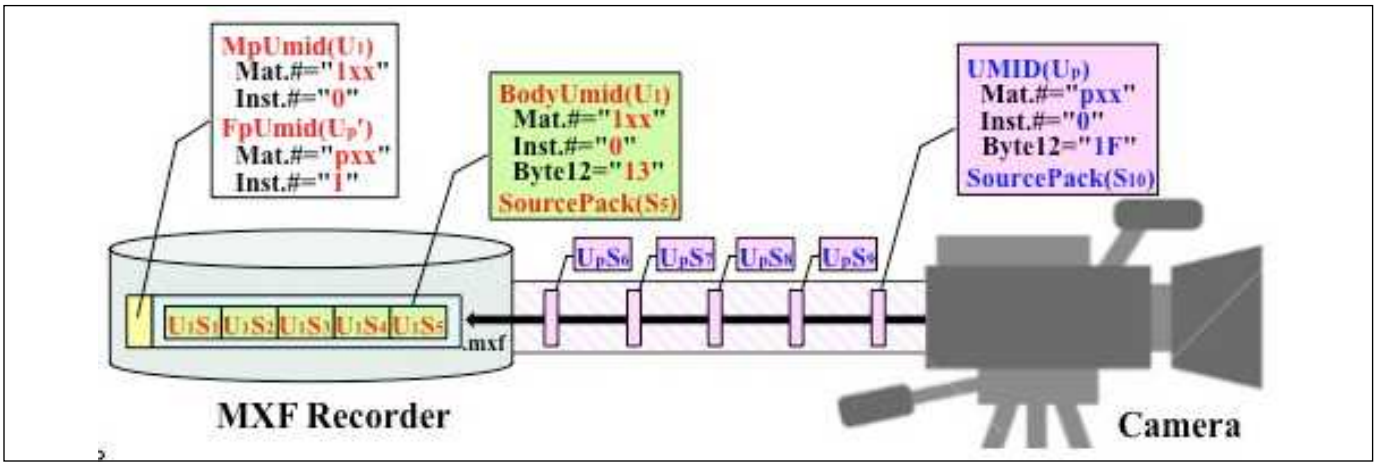

Figure 15. An original MXF file creation from a live feed.

tended UMID attached to every frame ("pxx") in the incoming media stream and using nonzero Inst.# values ("1").

As for the BodyUmid assigned to an edit unit in the original MXF file, based on the UAPs 6 (Extended UMID) and 7 (Sources Pack), the basic part of the BodyUmid is aligned with the MpUmid value ("U_1"), while the source pack of BodyUmid ("S_i") is created by inheriting the source pack of extended UMID attached to the incoming frame exactly as is.

Note that the bottom nibble of byte 12 in the BodyUmid is reset from "F_h" to "3_h" to indicate the "copy number and 16 bit pseudo-random sequence (PRS) generator" method for the Inst.# generation1 (insignificant for a newly created UMID with zero Inst.# value, though) because the edit unit was recorded as a part of persistent material as the MXF file.

BodyUmid Applications for an Existing MXF File Playout and Another MXF File Creation

Figure 16 schematically demonstrates another MXF file creation by capturing a media stream over the serial digital interface (SDI), which is generated by the playout of the original MXF file shown in **Fig. 15**.

For the MXF file playout, each edit unit in the original MXF file, usually stored in a compressed fashion, is decoded to the baseband signal and emitted to the SDI as a frame. The BodyUmid stored beside the edit unit is also read and embedded exactly as is into the vertical ancillary (VANC) data space of the frame (as the extended UMID attached to it) in the media stream over the SDI.

Another MXF recorder that receives the media stream over the SDI creates a new MXF file. More specifically, after the file header creation, the MXF recorder receives incoming frames of the baseband signal as the material units, makes them lossy compressed, and stores them as the edit units with newly created BodyUmids during the file body creation.

Because the new MXF file created by another MXF recorder is also managed independently, its MpUmid ("U_3") is newly created with another new Mat.# ("3xx") and zero Inst.# values. Following the discussion in **Fig. 14**, its FpUmid ("U_1'") is created by inheriting the Mat.# value in the MpUmid of the original MXF file ("1xx") and using a newly created nonzero Inst.# value ("1").

The BodyUmid in the new MXF file is created in a different way from that of the original MXF file shown in **Fig. 15**. Instead of aligning with its MpUmid value, the basic part of BodyUmid is created by inheriting the Mat.# value of the basic part in the extended UMID attached to an incoming frame ("1xx") and using a newly created nonzero Inst.# value ("1"), while the source pack of the BodyUmid ("S_i") is created in the same way as that in **Fig. 15**.

This BodyUmid creation strategy is based on UAPs 5 (UMID Inheritance) and 7 (Source Pack), which are also valid because of the lossy compression, which makes the edit unit stored in the new MXF file a derived material from a frame of the baseband signal in the incoming media stream.

Note that this BodyUmid creation strategy is more useful, especially for derived MXF file creation. If this strategy is applied not only to an MXF file derived directly from an original MXF file (the first generation), but also to any other derived MXF files from another derived MXF file (the Nth generation), their BodyUmids always inherit the Mat.# of the MpUmid of the original MXF file from which the edit unit derives directly or indirectly. As a result, when an edit unit with the BodyUmid in any MXF file in the media production workflow chain is obtained, the URL of the original MXF file from which the edit unit derives can be accessed via resolving the basic part of BodyUmid with its Inst.# masked to zero by using the UMID Resolution Protocol.

In addition, this function is further enhanced when the nonzero Inst.# value is created by using the aforementioned "copy number and 16-bit PRS generator" method,[1] which is indicated by the bottom nibble of byte 12 in the BodyUmid as "3_h." According to this Inst.# generation method, the 3 byte Inst.# value is composed of a 1 byte copy number, which actually indicates the generation number since the material is original, and a 2 byte random value.

Therefore, if this Inst.# generation method is adopted by all media products over the media production workflow chain, and the

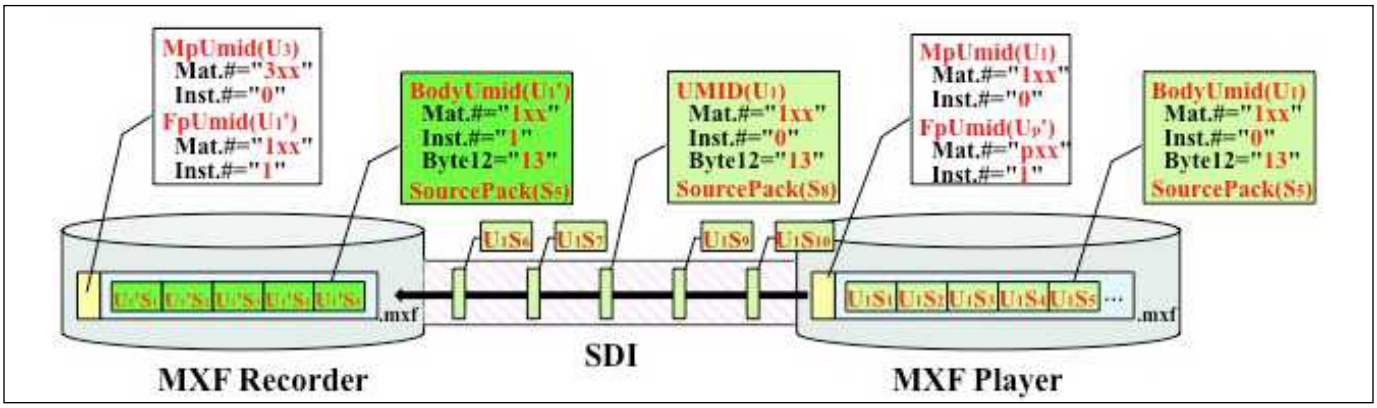

Figure 16. A new MXF file creation from a playout of an existing MXF file.

copy number is employed by them appropriately, it is expected that the quality of the finished program can be evaluated on a frame basis, and, when the quality of certain parts of the program does not satisfy the quality requirements, the original materials used to produce those parts can be easily obtained via the BodyUmid resolution in order to reproduce the parts in a higher quality.

Source Pack Applications

The source pack in the extended UMID, and so the BodyUmid in an MXF file, is a compact placeholder to accommodate the information on "when," "where," and "who" originally created each frame. Because the source packs synchronize with frames, they are also *played back* together with the frames. **Figure 17** shows an example of such a playback result where a media stream generated by the MXF player in **Fig. 16** (or a live feed generated by the camera in **Fig. 15**) is displayed on a video monitor.

In **Fig. 17**, the media stream resulting from the playout of material that captures a seaside scene during a typhoon is displayed, with the date/time ("when") and place ("where") information for its acquisition being superimposed over the image. Because the source pack varies with frames, the superimposed information is also continuously updated during the playout of the material if it is also played back.

Figure 17. Material playback with the source pack.

Another source pack application is its use for material classification and/or search, as shown in **Fig. 7**. Unlike the metadata shown in **Fig. 6**, the source pack is created automatically without any human intervention. This characteristic of the source pack is helpful, particularly when a large number of materials are created almost simultaneously, such as in a large-scale disaster, where even consumer electronics (CE) devices such as smartphones play an important role for material acquisition.

In fact, the source pack was introduced in the initial stage of UMID development in 2000, and a trial was made for its use even in a traditional VTR/SDI-based media production environment.[13] With material as a physical entity (video tape) in the traditional environment, the usability of the source pack was very much limited, though.

Now, the file-based environment where any material is available online has become a reality, and, with the provision of standard methods for source pack transfer (in the extended UMID on the wire) and its recording (in the BodyUmid), as well as the world-wide map information available today, it is just a matter of time for the source pack (especially the GPS information in it) to be proactively utilized for professional use in a similar way to the CE applications on mobile devices.

UMID APPLICATIONS IN MXF AND STREAMING MEDIA

UMID Applications in Streaming Media

While the extended UMID can be seamlessly used over an MXF file and a media stream resulting from the playout of an MXF file as shown in **Fig. 16**, it can also be effectively utilized even only within the world of streaming media, especially when the copy number in the Inst.# field is employed based on the "copy number and 16-bit PRS generator" method[1] to generate a value for the field.

The copy number was originally introduced for the number of repetitions of lossy compression and decompression caused by tape dubbing in the traditional VTR/SDI environment. In the file-based environment, it is similarly used to indicate the file's generation number from the original, as **Fig. 16** depicts.

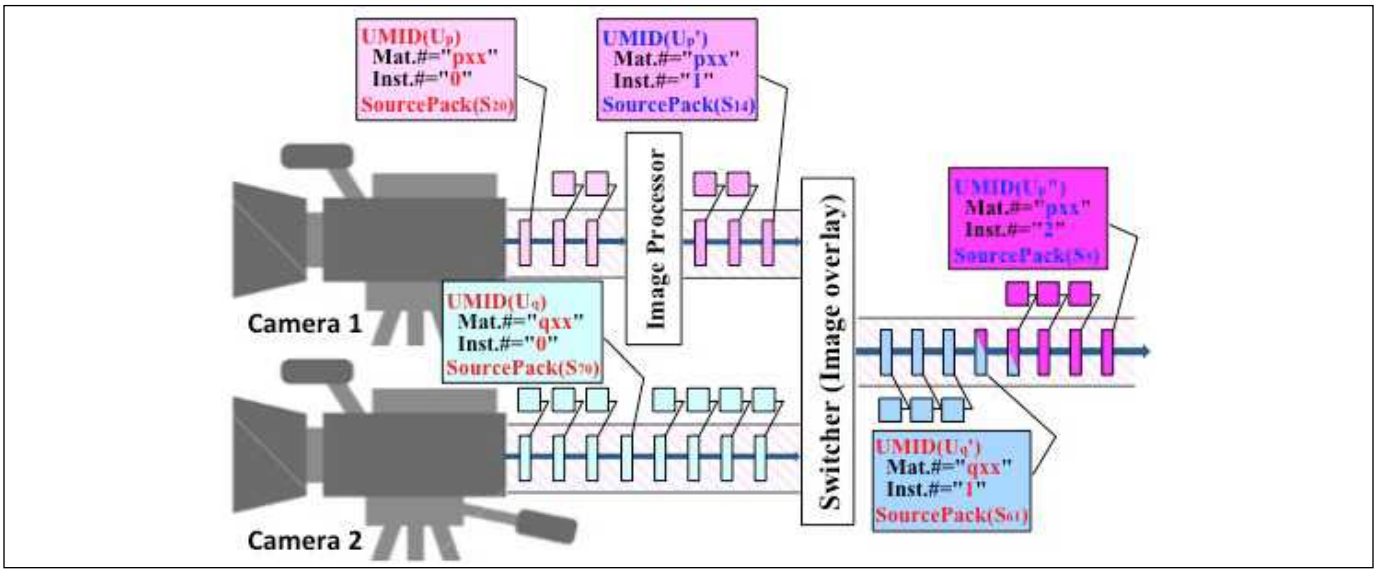

Figure 18. Extended UMID applications for streaming media.

For streaming media, the copy number is further generalized to indicate the number of media processing steps applied to an edit unit since its origin. **Figure 18** schematically demonstrates such an example where two cameras ("Camera 1" and "Camera 2") generate their respective live feeds as media streams, which are then switched over by a production switcher ("Switcher") to form a single media stream.

In this example, an image processor ("Image Processor") applies image processing such as color grading to the media stream from "Camera 1" before switching, and the "Switcher" applies a transition effect such as a wipe to the switching point of the incoming two streams over two frames as well as applying an image overlay such as a corporate logo to all frames to output.

As shown in **Fig. 18**, the extended UMID is attached to every frame as the material unit, the sequencing of which forms a media stream. A frame newly created by a camera is attached with a newly created extended UMID having a newly created Mat.# ("pxx" and "qxx" by "Camera 1" and "Camera 2," respectively) and zero Inst.# values (ditto "U_p" and "U_q") together with a newly created source pack ("S_i").

Assuming the "copy number and 16-bit PRS generator" method for the generation of nonzero Inst.# values, the copy number in the Inst.# field of the extended UMID attached to a frame is employed so that it is incremented by one when certain media processing is applied to the frame.

As a result, the extended UMID attached to a frame input to the "Image Processor" is replaced with that for which the Inst.# value is set to "01_h xx_h yy_h" (the copy number "01_h" immediately followed by the PRS value "xx_h yy_h"), as is simply represented by "Inst.# = "1" " in **Fig. 18**. In addition, because of the image overlay applied by the "Switcher," all the frames the switcher outputs are attached with the extended UMID for which the copy number is "2" and "1" for the frame input from "Camera 1" and "Camera 2," respectively.

As for the composite frames in the transition effect region, a frame in the region is considered as a derived one from the input frame that contributes to the composition most. Therefore, the right and left frames in the region in **Fig. 18** are considered as ones derived from a frame by "Camera 1" and "Camera 2," respectively, though the frame composition and the image overlay applied to those frames are regarded as a single media processing step.

Note that because none of the frame is recorded to form a persistent material in **Fig. 18**, all the extended UMIDs attached to the frames have the UMID "live stream" flag, or the bottom nibble of their byte 12 is set to "F_h." In fact, this nibble is also used to indicate the Inst.# generation method, which should have been set to "3_h" for this case. However, the UMID "live stream" flag prevails because the recorded material can coexist in a system without any problem when a device in the system has a recording capability, which should be distinguished by a higher priority than the Inst.# generation method.

Consequently, the copy number in the Inst.# field is also employed effectively even only within the world of streaming media when the "copy number and 16-bit PRS generator" method is adopted for the Inst.# value to be generated by all media products, which is useful to optimize the system configuration for the quality of streaming media.

Proposed Guidelines for Interoperability

Because of the wide range and high flexibility of MXF technology, some guidelines to restrict its use are often required to maximize the interoperability, and the UMID applications in MXF and streaming media are no exception.

In the following, the assumptions for the UMID applications discussed so far are itemized, which will be the starting point for the guideline study to obtain better interoperability for them.

- A newly created MpUmid is assigned to a newly created MXF file (UAP 2).

- The MpUmid is appropriately managed to be a globally unique material identifier (UAP 3).
- Clone MXF files share a single MpUmid (UAP 4).
- The FpUmid is used as a linking tool back to the source MXF file (UAP 5).
- The basic part of BodyUmid aligns with the MpUmid for an original MXF file creation (UAP 6).
- The source pack is always inherited regardless of the creation method of the basic part of an extended UMID and regardless of whether it is used for an MXF file or a media stream (UAP 7).
- The edit unit assigned with the BodyUmid in an MXF file corresponds to the material unit attached with the extended UMID on the wire one by one, which is a frame.
- The "copy number and 16-bit PRS generator" method, indicated by the bottom nibble of byte 12 as "3_h," is always applied for the nonzero Inst.# generation regardless of whether it is for an MXF file or a media stream.
- The UMID "live stream" flag ("F_h") prevails against the Inst.# generation method ("3_h") for the bottom nibble of byte 12 in the case of a media stream before recording.

CONCLUSION

In this paper, plausible UMID applications in an MXF file and a media stream are discussed, based on the UMID Application Principles recently standardized by SMPTE.

It was found that neither new invention nor extension is required for the MXF technology to enable those UMID applications. With the wide range and high flexibility of the technology, however, some guidelines to restrict its use need to be established to maximize the interoperability.

This paper constitutes a digest version of the SMPTE Standard Committee Report on this topic available online,[7] which is a deliverable of the UMID Application Project (SMPTE TC-30MR Study Group UMID Applications). Because that report contains more comprehensive discussions on this topic, those interested in this paper are kindly requested also to review the report and to provide the project members with any feedback in order for the dream envisioned by the EBU/SMPTE Task Force to be fully realized.

REFERENCES

1. SMPTE ST 330:2011, "Television—Unique Material Identifier (UMID)," https://www.smpte.org/standards.
2. SMPTE RP 205:2014, "Application of Unique Material Identifiers in Production and Broadcast Environments," https://www.smpte.org/standards.
3. EBU Technical Review Special Supplement 1998, "EBU/SMPTE Task Force for Harmonized Standards for the Exchange of Programme Material as Bitstreams, Final Report: Analyses and Results," http://tech.ebu.ch/docs/techreview/ebu-smpte-tf-bitstreams.pdf.
4. SMPTE ST 377-1:2011, "Material Exchange Format (MXF)—File Format Specification," https://www.smpte.org/standards.
5. Y. Shibata and J. Wilkinson, "UMID Applications in Practices," *SMPTE Mot. Imag. J.*, 121(2):58-67, Mar. 2012.
6. SMPTE TC-30MR SG UMID Applications, https://kws.smpte.org/kws/public/projects/project/details?project_id=90.
7. SMPTE TC-30MR Study Group Report, "Study of UMID Applications in MXF and Streaming Media," available at https://www.smpte.org/standards/reports.
8. SMPTE RP 210v13:2012, "Metadata Element Dictionary," https://www.smpte.org/standards.
9. N. Wells, O. Morgan, J. Wilkinson, and B. Devlin, *The MXF Book*, Focal Press: Burlington, MA, May 2006.
10. SMPTE ST 336:2007, "Data Encoding Protocol Using Key-Length-Value," https://www.smpte.org/standards.
11. SMPTE ST 379:2009, "Material Exchange Format (MXF)—MXF Generic Container," https://www.smpte.org/standards.
12. SMPTE ST 385:2012, "Material Exchange Format (MXF)—Mapping SDTI-CP Essence and Metadata into the MXF Generic Container," https://www.smpte.org/standards.
13. For example, the optional GPS unit (HKDW-704) for the Sony HDCAM™ Camcorder (HDW-730).

A contribution received from the author, October 2015.

Yoshiaki Shibata is a chair of SMPTE TC-30MR Study Group (SG) UMID Applications. In 2001 Shibata met the UMID for the first time at Sony Corp. Since then, he was heavily involved and played a crucial role in the development and implementation of the UMID applications, initially for the traditional professional VTRs (HDCAM™) and then for the modern file-based media products (XDCAM™), during which he was fascinated by the potential key roles of UMID in the future file-based media production environment. At the end of 2010, Shibata left Sony, and later, founded metaFrontier.jp, LLC, Japan's first independent consulting firm specifically for the media and metadata technology. With his well-received presentation at the SMPTE 2011 Annual Technical Conference, regarding the challenges and solutions to realize an ever-envisioned UMID application in practice as a trigger, Shibata joined the SMPTE Standard Community and established the SG under TC-30MR in 2012. So far, the SG has already standardized the UMID Application Principles as the latest RP 205 for which Shibata is awarded by the ITE (The Institute of Image Information and Television Engineers, Japan), has successfully developed the UMID Resolution Protocol, which is now under SMPTE standardization, and has intensively explored the promising UMID applications specifically for the MXF technology whose deliverable is now made available online. Shibata is a member of SMPTE, ITE, and MPTE (Motion Picture and Television Engineering Society of Japan).