Amendment 2:2012 to
SMPTE ST 377-1:2011

SMPTE STANDARD

Material Exchange Format (MXF) — File Format Specification — Amendment 2

Table of Contents

Page

Approved
October 7, 2012

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices, Engineering Guidelines and Registered Disclosure Documents, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in Part XIII of its Operations Manual.

SMPTE Amendment 2 to SMPTE 377-1:2011 was prepared by Technology Committee 31FS.

1 Scope

The purpose of this amendment is to address various issues identified since the publication of SMPTE ST 377-1:2011.

2 Descriptive vs Application Plugin Metadata

Change Section 9 (Header Metadata) from:

The Header Metadata contains Metadata Sets that define the contents of the file as a whole, including any Essence Containers. It may also describe Essence that is stored external of the file.

The Header Metadata is broadly split into two categories: Structural Metadata (see Sections 9.5 and 9.6) and Descriptive Metadata (see Section 9.8).

Application-Specific Metadata (see Section 9.7) can be used to extend either of the two categories.

All three categories shall be encoded as a single sequence of KLV coded packets.

This specification defines Structural Metadata as a single extensible scheme. There shall be no other Structural Metadata schemes in MXF; i.e., there shall be no other scheme of similar or identical functionality that replaces the scheme that is defined in Section 9.5.

To: (redline shows difference)

The Header Metadata contains Metadata Sets which defines the contents of the file as a whole, including any Essence Containers. It may also describe Essence that is stored external of the file.

The Header Metadata is broadly split into two categories: Structural Metadata (see Sections 9.5 and 9.6) and Descriptive Metadata (see Section 9.8).

Note: Application-Specific Metadata (see Section 9.7) can be used to extend either of the two categories. However, extending Descriptive Metadata is not recommended unless the application that defines the Application Scheme is the same as the application that defines the Descriptive Metadata Scheme.

All three categories shall be encoded as a single sequence of KLV coded packets.

This specification defines Structural Metadata as a single extensible scheme. There shall be no other Structural Metadata schemes in MXF, i.e. there shall be no other scheme of similar or identical functionality that replaces the scheme that is defined in Section 9.5.

3 Application Plugin Schema Publication Requirements

Change Section 9.7.2 (Application Metadata Scheme Specification) from:

For each Application Metadata Scheme, there shall be an Application Plug-in Scheme specification. For MXF Application Metadata Schemes, the Application Plug-in Scheme specification should be an SMPTE Engineering Document.

In addition to the Application Metadata Plug-In payload elements and semantics, Application Metadata Scheme specifications shall define the Application Scheme Label.

For further requirements on Application Metadata Scheme specifications, see Annex C.

To: (redline shows change)

For each Application Metadata Scheme, there shall be an Application Plug-in Scheme specification. ~~For~~ An MXF ~~Application Metadata Schemes, the~~ Application Plug-in Scheme specification should be a ~~an SMPTE Engineering Documen~~public standard from an internationally accredited standards body. This is to ensure that all Application Plug-in Schemes can be used for interchange.

In addition to the Application Metadata Plug-In payload elements and semantics, Application Metadata Scheme specifications shall define the Application Scheme Label.

For further requirements on Application Metadata Scheme specifications, see Annex C.

4 AFD

Replace Annex G.2.5 with the following:

Active Format Description, as defined by SMPTE ST 2016-1, describes a video picture in terms of the aspect ratio and other characteristics of the active image within the coded frame. In general, Active Format Description can vary within a video stream, reflecting changes to the formatting of the video picture within the coded frame over time. In particular cases, Active Format Description might be constant for an individual video clip or stream.

Note 1: The SMPTE RP 210 name for the element used here in the Picture Descriptor is Active Format Descriptor, in contrast to the SMPTE ST 2016-1 term Active Format Description.

For cases where the Active Format Description is constant for the duration of the essence container (whether encoded as MXF essence elements or embedded in the essence), then the Active Format Descriptor element may be encoded in the Picture Descriptor. For cases where the Active Format Description varies with time, the Active Format Descriptor element shall not be encoded in the Picture Descriptor. If the Active Format Description is encoded in the Picture Descriptor, then it shall be encoded if and only if it is constant in the essence container (whether encoded as MXF essence elements or embedded in the essence).

If the Active Format Description is time-varying, then it should be encoded as VANC data according to SMPTE ST 2016-3 and embedded in an MXF essence container according to SMPTE ST 436.

For decoders of MXF files: if the Active Format Descriptor value is encoded in the Picture Descriptor, the essence track shall be assumed to have a constant Active Format Description value in the essence container. If the Active Format Description value is not encoded in the picture descriptor, no assumptions can be made about whether a per-frame Active Format Description value is encoded in the essence container, or whether it is constant or not.

The Active Format Descriptor element in the Picture Descriptor is expressed as an 8-bit unsigned integer (b7-b0) with characteristics as follows:

Encoders

Encoders compliant with SMPTE 377-1 shall encode b7-b0 as defined in SMPTE ST 2016-1 Table 4, where b7, b1, b0 are each '0' (reserved), b6-b3 are the AFD code data bits a3-a0, and b2 is an aspect ratio bit.

Note: AFD code data bits a3-a0 are defined in SMPTE ST 2016-1 Table 1. The aspect ratio (AR) bit uses value '1' to indicate 16:9 and '0' to indicate 4:3.

Decoders

For decoders compliant with SMPTE 377-1:

If the MXF file being read has the Version property of the Preface Set equal to 1.3 or higher:
decoders shall ignore b7,
decoders shall interpret b6-b3 as AFD a3a2a1a0,
decoders shall interpret b2 as AR,
and
decoders shall ignore b1-b0.

If the MXF file being read has the Version property of the Preface Set lower than 1.3 and both b6 and b5 equal 0:
decoders should ignore b7,
decoders should interpret b3-b0 as AFD a3a2a1a0,
and
decoders may interpret b4 as AR.

If the MXF file being read has the Version property of the Preface Set lower than 1.3 and either b6 equals 1 or b5 equals 1:
decoders should ignore b7,
decoders should interpret b6-b3 as AFD a3a2a1a0,
decoders should interpret b2 as AR,
and
decoders should ignore b1-b0.

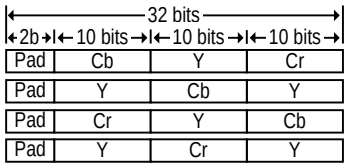
AFD code data bits a3-a0 are defined in SMPTE ST 2016-1 Table 1. The aspect ratio (AR) bit uses value '1' to indicate 16:9 and '0' to indicate 4:3.

In all cases for decoders, if the Aspect Ratio property of the GenericPictureEssenceDescriptor Set is specified in the file being read, it shall take precedence over the AR bit.

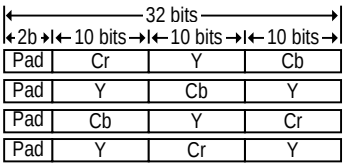
Note: It is possible that decoders compliant with SMPTE 377-1, when reading MXF files with the Version property of the Preface Set lower than 1.3, will not correctly decode AFD codes 0000, 0010, and 0011 carried in the byte defined in SMPTE ST 2016-1 Table 4.

5 Picture Essence Coding Word Ordering for 'v210'

For compatibility with industry practices, change the 5th row of the second table in Section G.2.25 (Picture Essence Coding) from:

06.0E.2B.34 04.01.01.0A 04.01.02.01 01.02.02.01		Interleaved, 4:2:2, 10-bit component. A pattern of 10-bit components Cb, Y, Cr, Y with a 2-bit pad inserted at the start of every 32 bits. This results in a rotating pattern of three components for every 32-bits. A full rotation occurs every 6 Pixels, so the full pattern is repeated ((W/2)·H)/3 times. The value of the Pad bits is unspecified. This layout is sometimes referred to as 'v210'. No equivalency, although ComponentDepth=10, HorizontalSubsampling=2, VerticalSubsampling=1 may be specified. Stored Width shall be a multiple of 6.

To: (graphic not redlined)

06.0E.2B.34 04.01.01.0A 04.01.02.01 01.02.02.01		Interleaved, 4:2:2, 10-bit component. A pattern of 10-bit components Cr, Y, Cb, Y with a 2-bit pad inserted at the start of every 32 bits. This results in a rotating pattern of three components for every 32-bits. A full rotation occurs every 6 pixels, so the full pattern is repeated (W·H)/6 times. The value of the Pad bits is unspecified. This layout is sometimes referred to as 'v210'. No equivalency, although ComponentDepth=10, HorizontalSubsampling=2, VerticalSubsampling=1 may be specified. Stored Width shall be a multiple of 48.

6 VBEByteCount Semantics

Change Section 11.2.3, Table 26, VBEByteCount row, right column from:

The number of bytes of Essence data in an external Essence file.

To:

The count of bytes of the last essence element in the last Edit Unit indexed by the Index Table Segment.

Change Section 11.2.3 from:

12. In a VBE Essence Container, the value of VBEByteCount shall equal the count of bytes in the last Edit Unit indexed by the Index Table Segment starting at the first byte after the last indexed byte of Essence data until the last byte in the Edit Unit of stored Essence. The same definition shall apply when indexing an external Essence file containing VBE Essence.

To:

12. If present, the value of VBEByteCount shall equal the count of bytes of the last essence element in the last Edit Unit indexed by the Index Table Segment. This definition shall apply when indexing an internal Essence Stream or an external Essence file containing VBE Essence.

Replace Section 11.5.2.2 with:

The format of the Essence byte stream in the external file shall be such that it could be the value of a clip-wrapped MXF Essence Container.

The MXF encoder shall index the Essence byte stream as if indexing the equivalent single Essence Element clip wrapped MXF Essence Container that holds the identical byte stream.

If the Edit Unit Byte Count Property of an Index Table Segment is non-zero and if the Essence byte stream in the external file begins at an offset greater than zero, the ExtStartOffset shall be present and its value shall equal the offset of the beginning of the Essence byte stream in the external file. The absolute byte offset of an Edit Unit of stored Essence within the external file shall be computed as the sum of the value of ExtStartOffset and the product of the Edit Unit Byte Count and the edit unit number.

If the Edit Unit Byte Count Property of an Index Table Segment is absent or zero and if the Essence byte stream in the external file begins at an offset greater than zero, the value of ExtStartOffset shall be set to equal the offset of the beginning of the Essence byte stream in the external file. The absolute byte offset of an

Edit Unit of stored Essence within the external file shall be computed as the sum of the value of ExtStartOffset and the value of Stream Offset of the corresponding Index Array entry.

7 Cleanup of references to SMPTE RP 210

7.1 General

Except as listed below, replace all occurrences of:

"SMPTE RP 210 …"

With:

"Note: SMPTE RP 210 definition …".

7.2 Specific instances that do *not* change

1. P10, "Dictionary: A list of values with defined meanings (e.g. SMPTE RP 210 metadata dictionary)".

2. P14, "[RP 210 text]: The definition of a term copied from the SMPTE Dictionary RP 210 at the time of balloting this specification. This text is Informative in this document. SMPTE RP 210 remains the defining document. This text is provided where the specific text in the MXF document constrains the RP 210 definition such that the wording might appear different between the two documents."

3. P19, "Item UL: The UL Designator and Item Designator portions of the UL as defined in SMPTE 336M. It defines the SMPTE Metadata Dictionary entry. The normative definition of all 16 bytes of the ULs is given by SMPTE 335M (SMPTE RP 210)."

4. P33: "The KLV Fill item is defined in the SMPTE Metadata Dictionary (SMPTE RP 210) as 06.0e.2b.34.01.01.01.02.03.01.02.10.01.00.00.00."

5. P134, Table at top of page, col 5: "Appropriate value from SMPTE 335M (SMPTE RP 210) or UUID defined according to the Application Metadata Scheme"

6. P135, C.3 Row 4 Meaning column: "Appropriate value from SMPTE 335M (SMPTE RP 210) or UUID defined according to the Application Metadata Scheme"

7. P136, ditto for Meaning column entries in Tables D1 and D2

8. P137, ditto for Tables E1 and E2.

9. P139 onwards..., ditto for all uses in Tables F2, F3, F4.1, F4.2, F4.3, F5 and F6.

10. P181: in the Bibliography

8 Number of Picture/Sound/Data Track Sets

In Annex B.18 Picture Track Sets, change the note from:

Note: The number of Picture Track sets is determined by the number of editable picture tracks in the Essence Container.

To (redline shows change):

Note: The number of Picture Track sets is determined by the number of ~~editable picture tracks~~ Picture Elements in the Essence Container. See Section 9.5.5 Top-Level File Packages.

In Section B. 21 Sound Track Sets, change the note from:

Note: The number of Sound Track sets is determined by the number of sound tracks present in the Essence Container.

To (redline shows change):

Note: The number of Sound Track sets is determined by the number of ~~sound tracks present~~ Sound Elements in the Essence Container. See Section 9.5.5 Top-Level File Packages.

In Section B.24 Data Track sets, change the note from:

Note: The number of Data Track sets is determined by the number of data tracks present in the Essence Container.

To (redline shows change):

Note: The number of Data Track sets is determined by the number of ~~data tracks present~~ Data Elements in the Essence Container. See Section 9.5.5 Top-Level File Packages.

9 Color Siting for IEC 61834 4:2:0 Essence

Change the text description for vertical midpoint (value 06h) within the table of Annex G.2.29 from:

“The color sample is sited at the point vertically midway between the luma sample on each column, as in MPEG-2 4:2:0.”

To (redline shows change):

"The color sample is sited ~~at the point~~ vertically midway between the luma sample on each column, as in MPEG-2 4:2:0 and H.264/AVC 4:2:0 (default)."

10 Index Table Layout

In Table 26 of Section 11.2.3 (Index Table Segment), append to the end of the table the rows labeled "Single Index Location", "Single Essence Location" and "Forward Index Direction" as defined in the table below.

Single Index Location	Boolean	1	3F.11	06.0E.2B.34 01.01.01.0E 04.04.05.01 00.00.00.00	Opt	If all Index Table Segments that compose one Complete Index Table are in one Partition, the Boolean value shall be TRUE. Else, (Index Table Segments that compose one Complete Index Table are distributed into multiple Partitions), the Boolean value shall be False. [**RP210** Specifies whether the Index Table Segments are in one partition or multiple partitions.]	
Single Essence Location	Boolean	1	3F.12	06.0E.2B.34 01.01.01.0E 04.06.02.06 00.00.00.00	Opt	If all Essence Containers are in one Partition, the Boolean value shall be TRUE. Else, (the Essence Container Segments are placed into multiple Partitions), the Boolean value shall be FALSE. [**RP210** Specifies whether the Essence Containers are in one partition or multiple partitions.]	
Forward Index Direction	Boolean	1	3F.13	06.0E.2B.34 01.01.01.0E 04.04.05.02 00.00.00.00	Opt	If all Index Table Segments that compose one Complete Index Table precede Essence Container Segments that they index, then the Boolean value shall be TRUE. Else (all Index Table Segments that compose one Complete Index Table follow Essence Container Segments that they index), the Boolean value shall be FALSE. [**RP210** Specifies whether the Index Table Segments are pointing forward or backward.	

In Section 11.2.3 (Index Table Segment), add the following two new conditions to the list following the table:

16. Single Index Location and Forward Index Direction shall be omitted when the Index Table is either sparse or partial.

17. If the encoder cannot guarantee the validity of the Single Index Location, Single Essence Location and/or Forward Index Direction, the properties shall be omitted.

In Annex A.2 (Preface), append to the table the item labeled "is RIP present" below:

Is RIP Present	Boolean	1	dyn	06.0E.2B.34 01.01.01.0E 04.04.05.03 00.00.00.00	Opt	If a RIP exists in the file, the Boolean value shall be TRUE. Else (a RIP doesn't exist in the file), the Boolean value shall be FALSE. [**RP210** Specifies whether the file includes a Random Index Pack.]	

In Annex A.2 (Preface), change from:

The contents of EssenceContainers should be complete where possible.

If the Partition in which this Set is located is Closed, the values shall be complete and correct.

To (redline shows change):

The contents of EssenceContainers should be complete where possible.

If the encoder cannot guarantee the validity of "is RIP present", the property shall be omitted.

If the Partition in which this Set is located is Closed, the values shall be complete and correct.

In Annex A.5 (Essence Container Data), append to the end of the items labeled: "Header Partition Usage", "Singular Partition Usage", "Footer Partition Usage" and "is Sparse" defined below:

Preceding Index Table	Boolean	1	dyn	06.0E.2B.34 01.01.01.0E 04.04.05.04 00.00.00.00	Opt	If a Complete Index Table for this Essence Container precedes all Essence that it indexes , the Boolean value shall be TRUE.Else, FALSE. [**RP210** Specifies whether a Complete Index Table is in a partition preceding all Essence that it indexes.]	
Singular Partition Usage	Boolean	1	dyn	06.0E.2B.34 01.01.01.0E 04.06.02.07 00.00.00.00	Opt	See Section A.5.1 below. [**RP210** Specifies whether the Partition is Singular.]	
Following Index Table	Boolean	1	dyn	06.0E.2B.34 01.01.01.0E 04.04.05.05 00.00.00.00	Opt	If a Complete Index Table for this Essence Container follows all Essence that it indexes, then, the Boolean value shall be TRUE. Else, FALSE. [**RP210** Specifies whether a Complete Index Table is in a partition following all Essence that it indexes.]	
Is Sparse	Boolean	1	dyn	06.0E.2B.34 01.01.01.0E 04.04.05.06 00.00.00.00	Opt	If a Sparse Index Table is in the essence container, the Boolean value shall be TRUE. Else (a sparse Index Table isn't in the essence container) FALSE.	

						[**RP210** Specifies whether a sparse Index Table is in the essence container.]	

In Annex A.5 (Essence Container Data), add the following text directly under the table:

A.5.1 Singular Partition Usage Semantics

If every Partition contains only an Essence Container Segment, or contains only one or more Index Table Segments, or neither of these segments, then the Singular Partition Usage property may be encoded, in which case its value shall be true.

If every Partition that contains an Essence Container Segment also contains one or more Index Table Segments, and the Footer does not contain any Index Table Segments, then the Singular Partition Usage property may be encoded, in which case its value shall be false.

If neither of the two conditions above are true or they are unknown, then the Singular Partition Usage property shall not be encoded.

Note: When Index Table Segments and Essence Container Segments are distributed among partitions, neither Preceding Index Table nor Following Index Table properties can express whether distributed Index Table Segments precede or follow the Essence they index. Preceding Index Table, Singular Partition Usage and Following Index Table properties can be used for example as follows.

Each Partition in an MXF file can only contain Index Table Segments for a single Essence Container. Where a file contains more than one Essence Container it is still useful to be able to locate a Complete Index Table for each Essence Container at the start or end of the file. Encoders can write a sequence of partitions at the start of a file, each containing a Complete Index Table for an Essence Container. The first will be the Header Partition and must also contain Header Metadata (but no essence). The following partitions will be Body Partitions with no Header Metadata and no essence, just a Complete Index Table for the next Essence Container. Similarly the last partitions in a file can contain a complete set of Index Tables for all Essence Containers in the file, the last will be the Footer Partition; the other Index Table Segments will be in Body Partitions (again with no essence and no Header Metadata)

In Annex H (Static Local Tags Assigned by MXF Specifications), extend the table with the following three Local Tags and the corresponding Universal Labels (redline shows change and context):

3F.10 06.0E.2B.34.01.01.01.0A.04.06.02.05.00.00.00.00
3F.11 06.0E.2B.34.01.01.01.0E.04.04.05.01.00.00.00.00
3F.12 06.0E.2B.34.01.01.01.0E.04.06.02.06.00.00.00.00
3F.13 06.0E.2B.34.01.01.01.0E.04.04.05.02.00.00.00.00
40.01 06.0E.2B.34.01.01.01.01.01.02.01.01.01.00.00.00

11 Header and Body Partition Pack Values

Change Section 7.2.2 Header Partition Pack Values from:

The value of the FooterPartition Property shall be as defined in Section 7.1 for Closed Partitions. In Open Partitions, the value shall be as defined in Section 7.1 or zero (0). If the Footer Partition is not present in the file then the value of this Property shall be zero (0).

To (redline shows change):

The value of the FooterPartition Property shall be as defined in Section 7.1 ~~for Closed Partitions. In Open Partitions, the value shall be as defined in Section 7.1~~ or zero (0). If the Footer Partition is not present in the file then the value of this Property shall be zero (0).

Change section 7.3.2 Body Partition Pack Value from:

The value of the FooterPartition Property shall be as defined in Section 7.1 for Closed Partitions. In Open Partitions, the value shall be as defined in Section 7.1 or zero (0). If the Footer Partition is not present in the file then this Property shall be zero.

To (redline shows change):

The value of the FooterPartition Property shall be as defined in Section 7.1 ~~for Closed Partitions~~ or zero (0). ~~In Open Partitions, the value shall be as defined in Section 7.1 or zero (0).~~ If the Footer Partition is not present in the file then this Property shall be zero.

12 Timecode Values in Source and Material Package

Add a new Section 9.9:

9.9 Start Timecode Values

This section provides rules for assigning proper values for the Start Timecode item in the Timecode Component in both Material Packages and Source Packages. Picking the correct value for Start Timecode can be influenced by various metadata values as well as the intended use of the final MXF file. These factors include the presence or absence of timecode values in the System Element in the underlying Essence Container, timecode discontinuities in the Essence Container, the value of the Origin item in the File Package, and whether the Material Package is used to redefine the timecode. The following rules are intended to provide clear, normative constraints based on these factors.

1. For a Source Package Timecode Component in the primary timecode track at offset N, the value of the Start Timecode should be equal to the timecode in ContentPackage[N]. However, if the Essence Container does not include a System Element with a valid Timecode, the Start Timecode value shall be unconstrained.

2. For a Material Package Timecode Component at position N, and given a corresponding Source Package Origin value of M, the value of the Start Timecode should be equal to the timecode in ContentPackage[N+M]. However, since the Material Package can be used to redefine the timecode for playout, the Material Package Start Timecode may have any value.

13 Node Leaf Conformance

In Section F.4.1 for the Picture Essence Coding row, change the 7th column as follows (redline shows addition):

UL identifying the Picture Compression Scheme (see G.2.25) Individual UL values are listed in the Registry defined by SMPTE ST 400 (RP 224) The value for the UL should not contain an RP 224 node. [**RP 210** Specifies the Compression scheme used]

14 Section 5 Cleanup

Change the first sentence in 5.2 from:

"SMPTE 377M was originally published in 2004. This specification is the first revision of the original document."

To:

The following is an overview of in-file version numbers for SMPTE 377-1.

Delete Section 5.2.1.

Change the Section number for 5.2.2 to 5.2.1.

Move Section 5.2.3 to a new Section 6.5 (so it is after 6.4.3 and before 6.5).

Change the Section numbers for 6.5, 6.6 and 6.7 to 6.6, 6.7 and 6.8, respectively.

Delete Section 5.3

Change the Section number for 5.4 to 5.3.

15 Correct Orphaned References to New Versions of 379

Delete SMPTE ST 379-1 and SMPTE ST 379-2 from the Bibliography.

16 AMP Scheme vs Generic Stream ID

Change Table 20 byte 11 (”Application”) from 05h to 06h