2024-03-04

Withdrawal of SMPTE ST 434

A document should be Withdrawn only if there is a significant possibility of its use causing harm. A Withdrawn document shall still be made available and offered for sale by the Society, but it shall be prefaced by a cover page explaining its current status including a statement that some or all of the content is no longer endorsed by the Society.

SMPTE ST 434 Material Exchange Format – XML Encoding for Metadata and File Structure Information is withdrawn. The document specifies an encoding of the header metadata, partition multiplex structure and index tables of MXF files as XML documents. It has not been revised since 2014 and no longer covers the range of Header Metadata values that might appear in the SMPTE registry and, as a consequence, MXF files.

A withdrawn document can still be obtained from SMPTE. Implementers are advised that implementations of SMPTE ST 434 are deprecated. XML interchange of MXF and IMF data can be achieved by using SMPTE ST 2001-1 XML Representation of SMPTE Registered Data (Reg-XML) — Mapping Rules.

SMPTE 434-2006

SMPTE STANDARD

Material Exchange Format – XML Encoding for Metadata and File Structure Information

Table of Contents

Approved
November 9, 2006

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 and Engineering Guidelines, 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 Administrative Practices.

SMPTE 434 was prepared by Technology Committee W25.

1 Scope

This standard specifies an encoding of the header metadata, partition multiplex structure and index tables of MXF files as XML documents. It specifies methods to support descriptive metadata plug-ins, dark metadata and private and future public extensions to the MXF specification.

This standard applies to the lossless representation of the metadata contained in MXF files as XML documents.

This standard applies to the interchange of the information of the header metadata, partition multiplex structure and index tables of MXF files as XML documents where the receiving application supports all XML namespaces from which there are elements in the XML document.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

SMPTE 335M-2001, Television — Metadata Dictionary Structure

SMPTE 336M-2001, Television — Data Encoding Protocol using Key-Length-Value

SMPTE 377M-2004, Television — Material Exchange Format (MXF) – File Format Specification

SMPTE 380M-2004, Television — Material Exchange Format (MXF) – Descriptive Metadata Scheme-1

SMPTE 381M-2005, Television — Material Exchange Format (MXF) – Mapping MPEG Streams into the MXF Generic Container

SMPTE 385M-2004, Television — Material Exchange Format (MXF) — Mapping SDTI-CP Essence and Metadata into the MXF Generic Container

SMPTE 422M-2006, Material Exchange Format — Mapping JPEG 2000 Codestreams into the MXF Generic Container

SMPTE 429-6-2006, D-Cinema Packaging — MXF Track File Essence Encryption

XML, Extensible Markup Language (XML) 1.0, W3C Recommendation

Namespaces in XML, W3C Recommendation

XML Schema Part 1: Structures, W3C Recommendation

XML Schema Part 2: Datatypes, W3C Recommendation

3 Introduction

This standard defines an XML coding for MXF structural and descriptive metadata. It does not provide for the XML coding of essence. Other standards may exist which offer potential advantage in some applications. See Section 3.3 for further information on application scenarios for this specification.

3.1 Definition of Terms

Table 1 –- Definition of Terms

Term	Definition
MXF-File	A file compliant with the MXF-Specification-Suite.
MXF-Specification-Suite	SMPTE 377M, its amendments and corrigenda and the set of SMPTE Engineering Documents that define metadata sets which are intended for interchange within the framework of SMPTE 377M header metadata.
MXF-to-XML-Encoder	A process that receives an MXF-File at its input and produces an MXF-XML-Document at its output.
MXF-XML-Document	An XML document that satisfies the provisions of this specification. MXF-XML-Documents can either represent the MXF-File's header metadata or the MXF-File's partition multiplex structure including header metadata and index tables, but not essence container data.
MXF-XML-Schema	The set of XML Schema files incorporated in this specification that defines the format and syntax of MXF-XML-Documents or a version of this set of XML Schema files that has been modified in accordance with the provisions of this specification.
XML-to-MXF-Encoder	A process that receives an MXF-XML-Document at its input and produces an MXF-File at its output.

3.2 General Considerations (informative)

An XML encoding can be specified by defining the format and syntax of XML instance documents, and by defining further constraints on their content. This specification uses a MXF-XML-Schema written in XML Schema to define the format and syntax of MXF-XML-Documents. Additional constraints on the contents of the MXF-XML-Documents result from the provisions of the documents of the MXF-Specification-Suite.

This specification also contains provisions how to extend the MXF-XML-Schema to support additions to the MXF-Specification-Suite, user-defined extensions of metadata sets defined in documents of the MXF-Specification-Suite and user-defined metadata schemes.

The MXF-XML-Schema does not need to be an element of an MXF-XML implementation. While the MXF-XML-Schema can be used to validate the format and syntax of MXF-XML-Document, it cannot be used to validate the contents of MXF-XML-Documents. In particular, the MXF-XML-Schema does not represent all of the rules and constraints of the MXF-Specification-Suite and, for example, does not validate if the MXF operational pattern properties of the MXF Partition packs and of the MXF Preface set do contain the same value. According to SMPTE 377M this is required.

3.3 Application Scenario (informative)

The MXF-XML application scenario is represented in Figure 1.

The lower section of the figure shows the MXF domain. In this domain MXF files are interchanged between MXF aware applications. The content of the MXF files is multiplexed into partitions according to the rules of SMPTE 377M. They contain MXF header metadata and may contain essence. The data structures and

properties of the files conform to the provisions of the relevant documents of the MXF-Specification-Suite. The MXF aware applications may use different metadata schemes. Examples for such schemes are MXF structural metadata, MXF structural metadata in combination with DMS-1 or another descriptive scheme or MXF structural header metadata with proprietary extensions. The independent extensibility supported by the design of MXF ensures that each of the MXF aware applications can successfully decode the information that is supported by its scheme, while being able to ignore and pass on the information that lies outside its supported scheme.

The structure of an MXF file and the MXF header metadata contained within it may be represented in XML through the use of an MXF-to-XML-Encoder that produces MXF-XML-Documents. MXF-XML-Documents need to be compliant with provisions of the relevant documents of the MXF-Specification-Suite as well as with the provisions of this specification. MXF-XML-Documents do not contain essence container data.

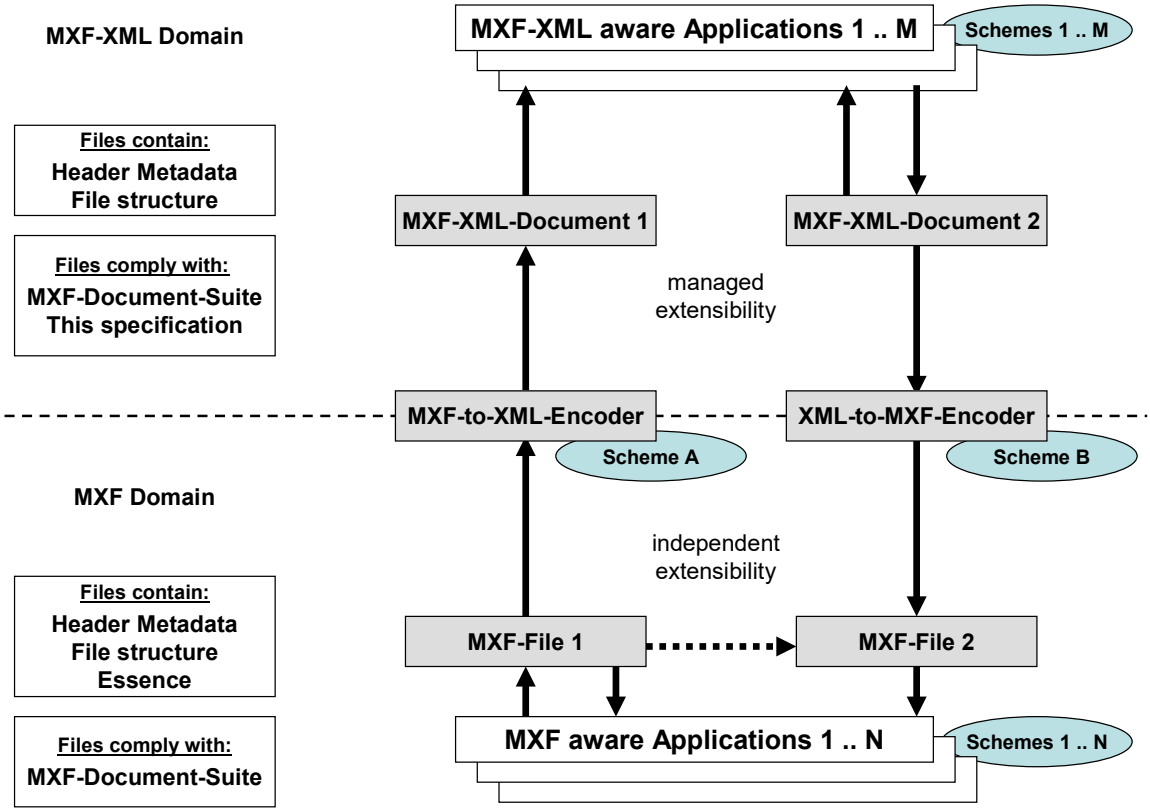

Figure 1 – Application Scenario

MXF aware applications can decode the KLV stream, process the MXF partition multiplex information and are able to decode MXF header metadata contained in the MXF file that is supported by their scheme. They may also be able to generate and/or update MXF files.

The MXF-to-XML-Encoder decodes the partition multiplex information as well as MXF header metadata and index table information contained in the MXF file that is supported by its scheme, and encodes all or a subset if it into MXF-XML. All MXF header metadata or index table information that is not supported by the scheme of the MXF-to-XML-Encoder can be encoded lossless into XML using the mechanisms of the MXF-XML specification to represent dark metadata.

MXF-XML aware applications read existing MXF-XML-Documents, update existing one or produce entirely new MXF-XML-Documents.

XML-to-MXF-Encoders may take MXF-XML-Documents as input and produce and/or update MXF files.

The MXF-XML-Schema in Annex B and its extensions that satisfy the rules defined in Section 5 do not represent a complete validation tool for MXF-XML-Documents. However, the validation of an MXF-XML-Document by the MXF-XML-Schema used by an MXF-XML aware application constitutes a necessary condition that an application that supports all namespaces from which there are elements in the MXF-XML-Document can successfully read the MXF-XML-Document and that an XML-to-MXF-Encoder can produce an MXF file that contains a lossless representation of the MXF header metadata contained in the MXF-XML-Document.

MXF-XML does not aim to be an interchange format that supports independent extensibility. This problem has already been solved by the encoding mechanisms of SMPTE 377M. In environments that require independent extensible interchange, it is therefore recommended to rely on the provisions of the MXF-Specification-Suite.

Within one MXF-XML installation domain, extensibility can be supported by managing the schemes used by the MXF-XML aware applications. Information that is supported or needed only by one of the MXF-XML aware applications can be encoded into MXF-XML-Documents using the dark metadata mechanisms of this specification. All MXF-XML aware applications will be able to ignore this information (that is irrelevant to them) and can forward it losslessly. MXF-XML aware applications may try to parse the KLV data carried in *ParsedDarkSet* XML elements, *UnparsedDarkGroup* XML elements and *DarkProperty* XML elements to detect and decode usable metadata.

Information that is required by more than one application should be represented at the XML level in decoded form (as opposed to dark). This information needs also to be supported by the schemes of the relevant applications of the installation domain. Figure 1 emphasizes this management requirement by introducing the term “managed extensibility” in the MXF-XML part of the application scenario.

Wherever possible and desirable, systems should be designed such that all known MXF extensions in use within the system are contained in schemes available to each MXF-to-XML Encoder and to each XML-to-MXF Encoder.

In Figure 1, MXF-to-XML-Encoders and XML-to-MXF-Encoders are drawn to lie in the MXF domain as well as in the MXF-XML domain. They are both XML aware and MXF-XML aware.

While MXF-XML-Documents do not contain essence container data, they can be used by MXF-XML aware applications to access essence container data stored in an MXF file. In particular, if the MXF-XML document contains the partition multiplex information and index table information, a MXF-XML aware application may not need to decode any of the header metadata or multiplex information from the MXF file in order to access specific essence container data elements.

MXF-XML-Documents can also be used as element of a representation that contains information of an equivalent MXF file, including the essence. If the equivalent MXF contains index table information, the MXF-XML-Document needs to contain the partition multiplex information and index table information. The essence container data for all essence containers is extracted into external files and network locators are added to the descriptors of the File Packages to link each of the files containing the extracted essence container data to the package that was associated with the essence container. If the original file does not contain index table information (or if it is not required to preserve index table information that may be present in the original file), the partition multiplex information may be discarded and it is sufficient if the MXF-XML-Document contains only the header metadata.

In combination with an MXF-to-XML encoder and an XML-to-MXF encoder, MXF-XML aware applications can also be used to update the header metadata of an MXF file while preserving the essence containers and index tables of the original file.

In Figure 1, the functionality of the previous two application examples is indicated by the dotted arrow pointing from MXF File 1 to MXF File 2.

3.4 Relation to Architecture of the MXF Documentation

The specification of the XML encoding follows the modularity of the architecture of the MXF-Specification-Suite. Namely, it has a core and supports plug-ins.

The core defines the XML representation for the MXF structural metadata sets defined in SMPTE 377M, partition multiplex structure and index tables. The plug-in mechanism allows the extension of the XML encoding by adding new properties to existing metadata sets, new metadata sets to existing metadata schemes and new metadata schemes. Some of these plug-ins are already included in this specification.

In the MXF-Specification-Suite, a new scheme is defined by adding one or more new documents to the suite. Examples are descriptive metadata schemes and essence mapping documents. In the MXF-XML-Schema there shall be one file and one unique namespace that contain the definitions of the metadata sets for each such document.

Annex B consists of a description of the directory structure of the MXF-XML-Schema documents and of the MXF-XML-Schema documents which constitute a part of this specification.

4 Conformance to MXF Specifications

The property values and property hierarchy of the data encoded in MXF-XML-Documents shall conform to the provisions of SMPTE 377M and the provisions of the applicable documents of the SMPTE-Document-Suite.

5 XML Encoding Rules

This standard complies to XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.

5.1 Namespaces

All groups defined in a particular SMPTE Engineering Document and the MXF partition multiplex structure and index tables information shall lie in namespaces that are identified by an URI of the form

http://www.smpte-ra.org/schemas/434/<ref-revision>/<shortname>/<ref-number>/<ref-revision>

All properties and data types defined a particular SMPTE Engineering Document shall lie in namespaces that are identified by an URI of the form

http://www.smpte-ra.org/schemas/434/<ref-revision>/<shortname>/<ref-number>

The <shortname> shall be one of:

- “types” – if the namespace identified by this URI holds data types, appropriately defined in an SMPTE Engineering Document, and expressed as defined in section 5.3;
- “property” – if the namespace identified by this URI holds metadata dictionary elements defined in an SMPTE Engineering Document, and expressed as defined in section 5.4;
- “groups” – if the namespace identified by this URI holds groups, appropriately defined in an SMPTE Engineering Document, and expressed as defined in section 5.5;
- “storage” - if the namespace identified by this URI holds information expressed as defined in section 5.8.

The <ref-number> shall be a string representing the official unique number assigned by SMPTE to the SMPTE Engineering Document where the data types, properties or groups are defined.

The <ref-revision> shall be a string of digits identifying the year and version of the SMPTE Engineering Document immediately preceding this value. Its format shall be of the form yyyy or yyyy.n. The characters yyyy shall be replaced by the four digit year during which the SMPTE Engineering Document was published or amended. The term ".n" shall be absent for the first publication or amendment within year yyyy. For all subsequent revisions or amendments of the SMPTE Engineering Document within the same year the term ".n" shall be present and n shall equal the count of publication events (publication, revision and amendment) of the Specification in year yyyy minus 1.

NOTE – XML Namespaces are used to assure globally unique tag names and to link MXF-XML-Documents to their defining MXF-XML-Schema. Amendments or new revisions of documents of the MXF-Specification-Suite may change provisions of the original versions such as adding new groups or adding new properties to already existing groups. For this reason, a new XML Namespace is assigned that identifies that an MXF-XML-Document conforms to the revised or amended document of the MXF-Specification-Suite. According to SMPTE 377M, all MXF files that are of the same major version are backwards compatible. As long as the major version number of the MXF file does not change, the XML Namespace associated with an older version of the documents of the MXF-Specification-Suite may be safely used to represent the information that conforms to that version (sets or properties that have been added with the amendment or revision will be represented as dark metadata). This can be used to manage extensibility.

The namespace URI, defined according to the above stated rules, and recommended default namespace prefixes used for the MXF-XML-Schema in Annex B are as shown in Table 2.

The definition of private properties and groups (such as class 14 entries in the registry controlled by SMPTE 335M) shall be accompanied with the definition of appropriate namespaces. The definition of the URI for these namespaces lies outside the scope of this specification.

The private addition of properties to groups, the private addition of groups to metadata schemes and the private addition of properties or groups to the partition multiplex structure and index table information defined in SMPTE engineering documents shall be accompanied with the definition of appropriate namespaces. The definition of the URI for these namespaces lies outside the scope of this specification.

The URI of the namespaces of all groups that are intended to be part of MXF header metadata shall be added as *<xs:any namespace="<namespace URI>" processContents="strict"/>* declarations to the contents of the second xs:choice model group in the definition of the *MetadataSets* XML element and to contents of the xs:choice model group in the definition of the *FloatingMetadataGroups* XML element.

NOTES

1 The *MetadataSets* and the *FloatingMetadataGroups* XML elements are defined in the MXF-XML-Schema file Annex B/file/metadata.xsd.

2 Annex D.2 provides examples for extensions of the MXF-XML encoding and their impacts on namespace assignment.

Table 2 – XML Namespace URI Definitions

Specification and Identified Elements	Namespace URI	Namespace Prefix used in the remainder of this specification (informative)
Groups (sets and packs)		
All groups defined in SMPTE 377M. Dark metadata sets and their properties (XML attributes and XML elements) as defined in this specification.	http:://www.smpte-ra.org/schemas/434/2006/types/S377M/2004	s377mGroups
All groups defined in SMPTE 380M	http:://www.smpte-ra.org/schemas/434/2006/groups/S380M/2004	s380mGroups
All groups defined in SMPTE 381M	http:://www.smpte-ra.org/schemas/434/2006/goups/S381M/2005	s381mGroups
All groups defined in SMPTE 385M	http:://www.smpte-ra.org/schemas/434/2006/groups/S385M/2004	s385mGroups
All groups defined in SMPTE 422M	http:://www.smpte-ra.org/schemas/434/2006/groups/S422M/2006	s422mGroups
All groups defined in SMPTE 429-2	http:://www.smpte-ra.org/schemas/434/2006/groups/S429-6/2006	s429-6Groups
MXF Storage Layer (partitions and partition multiplex)		
Storage layer related information defined in SMPTE 377M	http:://www.smpte-ra.org/schemas/434/2006/multiplex/S377M/2004	s377mMux
Properties of Groups		
All dictionary elements defined in SMPTE 335M classes 1 through 13	http:://www.smpte-ra.org/schemas/434/2006/properties/S335M	s335mElements
Data Types		
All data types defined in SMPTE 377M	http:://www.smpte-ra.org/schemas/434/2006/types/S377M	s377mTypes
All data types defined in SMPTE 422M	http:://www.smpte-ra.org/schemas/434/2006/types/S422M	s422mTypes

5.2 General Encoding Rules

MXF-XML-Documents shall comply with the XML document syntax and XML document structure defined in Annex B of this specification.

Extensions of MXF structural metadata or MXF descriptive metadata shall be expressed according to the extension rules defined in this document. Definitions of descriptive metadata schemes (as defined by SMPTE 377M) shall be expressed according to the extension rules defined in the remainder of this document.

NOTE – The support for SMPTE 380M may be used as an example for defining support for DM schemes.

MXF-XML-Documents shall declare only the namespaces from which elements are present in the MXF-XML-Document.

MXF-XML-Encoders shall not use the encoding mechanisms for dark metadata defined in Section 5.7 to encode any group that is defined in a namespace that is declared in the MXF-XML-Document.

MXF-XML-Encoders shall use the encoding mechanisms for dark metadata defined in Section 5.7 to encode all properties of a group that, according to the namespace of that group declared in the MXF-XML-Document are not defined to be part of that group.

Applications that create MXF-XML-Documents may discard parts of the header metadata (such as descriptive metadata frameworks or dark metadata. The resulting MXF-XML-Document shall conform to the provisions of Section 4.

NOTES

1 Applications that discard parts of the header metadata may need to update or may need to discard other metadata sets in order to guarantee compliance with Section 4. In the case of discarding descriptive metadata frameworks this may involve discarding the DM Segment set as well as changes to properties of DM Sequence, DM Track and DM Source Clip sets, changes to Packages and changes to the DM Schemes property of the Preface set.

2 Great care should be taken when eliminating dark metadata. Since references to and between dark metadata sets cannot be resolved, elimination of dark metadata may only be possible under certain, well-defined circumstances.

3 SMPTE 377M, section 8.2.3 states that “MXF Decoders are not required to handle dark metadata, but they should preserve any dark metadata in the file in case later processes require it.

5.3 Data Types

NOTE – The XML schema files that contain the type definitions are located in the directory Annex-B/dataTypes.

5.3.1 XML Encoding

The encoding in XML of the data conveyed as metadata properties is specified through the data types formally declared in the MXF-XML-Schema documents [Annex B], by restricting or extending XML Schema native data types.

The MXF-XML-Schema specifies the rules of XML encoding for each data type by deriving, aggregating and/or otherwise restricting native XML Schema data types and/or data types already included in the MXF-XML-Schema.

NOTE – XML Schema type names are subject to XML restrictions on symbol names. Therefore, type names have been derived from the data type name defined in the MXF-Specification-Suite by eliminating all characters that are not permitted according to the XML restrictions on symbol names. These characters include spaces and dashes.

5.3.2 MXF-XML-Schema Extension

Any addition of a data type to the MXF-XML-Schema shall specify the XML encoding for this data type, as defined in 5.3.1. Whenever possible, the definition of data types should rely on extending or restricting data types already defined in the MXF-XML-Schema.

The new MXF-XML-Schema entry shall additionally specify the XML name of such data type for further reference in MXF-XML-Schema constructs.

NOTE – If additions to the MXF-Specification-Suite use new data types as properties of groups, it is advised that an XML encoding for the values of these properties is defined as part of the document defining the new data type.

The new MXF-XML-Schema entry may optionally include the following information in a xs:annotation XML Schema element:

1. further documentation and any other relevant notes on the data type included in the xs:documentation element;

2. the name for the data type defined in the MXF-Specification-Suite, which may differ from the XML data type name in the MXF-XML-Schema (which is subject to the XML restrictions for data type names). This information may be included using an xs:appinfo element. The source attribute of the element shall have the value http://www.smpte-ra.org/schemas/434/Types/TypeName.

Table 3 – Example – MXF-XML-Schema Definition of the Unsigned 32 Bit Integer Data Type

```
<xs:simpleType name="UInt32">
    <xs:annotation>
        <xs:documentation>Unsigned 32 bit integer</xs:documentation>
        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Types/TypeName">UInt32</xs:appinfo>
    </xs:annotation>
    <xs:restriction base="xs:unsignedInt"/>
</xs:simpleType>
```

5.4 Dictionary Elements

NOTE – The XML schema files that contain the type dictionary element definitions are located in the dictionary subdirectory of Annex B.

5.4.1 XML Encoding

Dictionary elements included in MXF-XML-Documents defined either in the SMPTE Metadata Dictionary (SMPTE 335M) or in private metadata dictionaries, shall be encoded in XML as formally specified in the MXF-XML-Schema documents [Annex B]. The MXF-XML-Schema normatively specifies:

1. the XML tag names to be used for each dictionary element in MXF-XML-Documents;
2. the encoding rules for the data conveyed in the element, by referencing the appropriate data type definition in the MXF-XML-Schema.

For each dictionary element included in MXF-XML-Documents, the MXF-XML-Schema shall contain the associated SMPTE Universal Label as the value of an appinfo element with source http://www.smpte-ra.org/schemas/434/Dictionary/ElementUL". The representation for the Universal Label values is the one defined for the UniversalLabel type in Annex-B/types/smpte377m_types.xsd.

5.4.2 MXF-XML-Schema Extensions

All dictionary elements added to the MXF-XML-Schema shall have a valid SMPTE Label assigned to them.

The addition of a dictionary element to the MXF-XML-Schema shall include the XML encoding information for the element as defined in 5.4.1.

The data type of the dictionary element shall include the reference to the appropriate data type definition. The way this reference is implemented depends on the exact data type of the dictionary element.

1. If the data type to be referenced, declared according to 5.3, is not abstract, the type attribute in the xs:element declaration shall be used (e.g., Table 4).

2. If the data type of the dictionary element is an Array or a Batch of data items which are not References, the dictionary element declaration shall include an anonymous complexType definition, extending the abstract Array or Batch data types, respectively (e.g., Table 5). In such case, an xs:element declaration is added specified in the content model of the complexType (which shall be a xs:sequence) with the following settings:

    a. attribute name set to the string 'Element'

    b. attribute type referencing to the non-abstract data type for such data items

    c. attribute minOccurs set to the minimum number of data items allowed in the Batch or Array

    d. attribute maxOccurs set to the maximum number of data items allowed in the Batch or Array

3. If the data type of the dictionary element is a Strong Reference, a Weak Reference, a Strong Reference Array, Weak Reference Array, a Strong Reference Batch or a Weak Reference Batch, the dictionary element declaration shall include an anonymous complexType definition, extending the abstract *StrongRefHolder*, *WeakRefHolder*, *StrongRefArray*, *WeakRefArray*, *StrongRefBatch* or WeakRefBatch, respectively.

    NOTE – These abstract data types are defined in the file Annex-B/dataTypes/s377m_types.xsd.

    For each Metadata Set allowed to be referenced from an instance of a dictionary element that is a Strong Reference, Strong Reference Array or Strong Reference Batch, two xs:element declarations shall be added in the content model of the complexType, which shall be a xs:choice (e.g., Table 6):

    a. one declaration shall use the ref mechanism to reference the tag name of the Metadata Set (e.g.: *ContentStorag*e), as defined in 5.5;

    b. the other declaration shall use the ref mechanism to reference the tag name of the reference to a Metadata Set (e.g.: *ContentStorage_REF*) , as defined in 5.6.

    For each Metadata Set allowed to be referenced from an instance of a dictionary element that is a Weak Reference, Weak Reference Array or Weak Reference Batch, only xs:element declarations from point 3.b from above shall be added in the xs:choice content model of the complexType.

The new MXF-XML-Schema entry can include the following information under a xs:annotation element:

1. further documentation and any other relevant notes on the metadata element included in the xs:documentation element;

2. the name used in SMPTE S377M for the dictionary element, which may differ from the XML tag name since the later is subject to the XML restrictions for data type names. This information shall be encoded in a xs:appinfo element identified with a source attribute holding the value “http://www.smpte-ra.org/schemas/434/Dictionary/ElementName”;

3. the Universal Label that uniquely identifies this dictionary element, encoded in a xs:appinfo element identified with a source attribute holding the value “http://www.smpte-ra.org/schemas/434/Dictionary/ElementUL”;

4. the length of the data as encoded in a MXF file. This information shall be encoded in a xs:appinfo element identified with a source attribute holding the value “http://www.smpte-ra.org/schemas/434/Dictionary/Length”;

5. the documentation on this dictionary element included in the SMPTE Metadata dictionary. This information shall be encoded in a xs:appinfo element identified with a source attribute holding the value “http://www.smpte-ra.org/schemas/434/Dictionary/DataElementDefinition”.

   NOTE – This corresponds to the value of column “Data Element Definition” in the metadata dictionary.

Item number 3 of the above list shall be present. All other items are optional.

Table 4 – Example – MXF-XML-Schema Definition of the Last Modified Date Metadata Element

```
<xs:element name="FileLastModified" type=" s377mTypes:TimestampType">
    <xs:annotation>
        <xs:documentation>Date &amp;amp; time of the last modification of the file</xs:documentation>
        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Dictionary/ElementName">Last Modified Date</xs:appinfo>
        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Dictionary/ElementUL">urn:oid:1.3.52.1.1.1.2.7.2.1.16.2.4</xs:appinfo>
        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Dictionary/Length">8</xs:appinfo>
        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Dictionary/DataElementDefinition">Identifies date and time at the point
            of most recent modification of any item in the container.</xs:appinfo>
    </xs:annotation>
</xs:element>
```

Table 5 – Example – MXF-XML-Schema definition of the Video Line Map

```
<xs:element name="VideoLineMap">
    <xs:annotation>
        <xs:documentation>First active line in each field e.g. {16,278} (see E.2.15). Distinguished Value = 0</xs:documentation>
        <xs:appinfo source=" http://www.smpte-ra.org/schemas/434/Dictionary/ElementName">Video Line Map</xs:appinfo>
        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Dictionary/ElementUL">urn:oid:1.3.52.1.1.1.2.4.1.3.2.5</xs:appinfo>
        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Dictionary/Length">8+8</xs:appinfo>
        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Dictionary/DataElementDefinition"/>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="s377mTypes:Array">
                <xs:sequence>
                    <xs:element name="Element" type="s377mTypes:Int32" minOccurs="2" maxOccurs="2"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
```

Table 6 – Example – MXF-XML-Schema Definition of the ContentStorageObject

```
<xs:element name="ContentStorageObject">
    <xs:annotation>
        <xs:documentation>Strong reference to Content Storage object</xs:documentation>
        <xs:appinfo source=" http://www.smpte-ra.org/schemas/434/Dictionary/ElementName">Content Storage</xs:appinfo>
        <xs:appinfo source=" http://www.smpte-ra.org/schemas/434/Dictionary/ElementUL">urn:oid:1.3.52.1.1.1.2.6.1.1.4.2.1</xs:appinfo>
        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Dictionary/Length">16 bytes</xs:appinfo>
        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Dictionary/DataElementDefinition">[RP210 Specifies a reference to
            the packages and essence in a file]</xs:appinfo>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="s377mTypes:StrongRefHolder">
                <xs:choice>
                    <xs:element ref="s377mGroups:ContentStorage"/>
                    <xs:element ref="s377mGroups:ContentStorage_REF"/>
                </xs:choice>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
```

5.5 Sets and Packs

NOTE – The XML schema files that contain the definitions of sets and packs are located in the subdirectory Annex-B/groups.

5.5.1 XML Encoding

Instances of groups included that are defined in SMPTE standards shall be encoded in MXF-XML-Documents as formally specified in the MXF-XML-Schema documents [Annex B].

The MXF-XML-Schema normatively specifies:

1. the XML tag name to be used for the XML element representing the group in MXF-XML-Documents (e.g., Table 7);

2. the content model for this XML element which shall be defined by referencing the type definition for the group (e.g., Table 8):

    a. the type name;

    b. the base type from which this group derives;

    c. the list of XML encoded dictionary elements which may appear under the metadata set XML tag, by referencing the appropriate dictionary element definitions;

    d. the order in which such XML encoded properties shall appear;

        NOTE – In KLV encoding of the equivalent information, the items KLV sets can appear in arbitrary order; the items of KLV Packs are ordered. MXF-to-XML-Encoders need enforce the ordering defined in this specification when emitting the properties of KLV Set-encoded groups in MXF-XML-Documents. The ordering information is lost (and irrelevant) when an XML-to-MXF-Encoder emits a KLV Set-encoded groups.

    e. the cardinality for each property using the minOccurs and maxOccurs attributes. All properties with status “Required” or “Encoder Required” as defined in SMPTE 377M shall have minOccurs set to “1”. All other properties shall have the minOccurs attribute set to “0”.

3. For each set or pack included in MXF-XML-Documents, the MXF-XML-Schema shall contain the associated SMPTE Universal Label as the value of an appinfo element with source http://www.smpte-ra.org/schemas/434/Groups/Key”.

Metadata set definitions can include the following information in a xs:annotation XML Schema element:

4. further documentation and any other relevant notes on the metadata set, included in the xs:documentation element;

5. the name of the metadata set as defined in the specification of the metadata scheme which includes it, encoded as a xs:appinfo element. This may differ from the XML name since the later is subject to the XML restrictions for XML tag names; the xs:appinfo shall be identified by a source attribute holding the value “http://www.smpte-ra.org/schemas/434/Groups/Name”;

6. further comments on the metadata set, encoded as a xs:appinfo element identified by a source attribute holding the value “http://www.smpte-ra.org/schemas/434/Groups/Comments”;

7. for each property in the metadata set:

    a. the required status (as defined in SMPTE 377M), added under the xs:annotation associated with the property, and encoded in a xs:appinfo element identified with a source attribute holding the value “http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus”;

the following character sequences should be used for each of the required status possibilities as defined in SMPTE S377M:

i. Required: “Req”

ii. Optional: “Opt”

iii. Encoder Required: “E/req”

iv. Decoder Required: “D/req”

v. Best Effort: “B.Effort”

b. the need for further semantic validation rules associated with the required status is flagged with an “*” following the required status information in SMPTE S377M. The semantics validation rules shall be encoded as natural language text in a xs:appinfo element identified with a source attribute holding the value http://www.smpte-ra.org/schemas/434/Groups/RequiresSemanticValidation. The value of this xs:appinfo element is the definition of the validation requirement in text form (i.e. the text copied from the respective SMPTE engineering document);

Example: The ComponentLength property of the DM Segment metadata set specified in SMPTE S377M has additional semantic validation rules based on the type of the Track that includes it. In particular, the ComponentLength property is “B.Effort” in a Timeline Track, “Opt” in an Event Track and shall be omitted in a Static Track.

c. the Distinguished Value for the dictionary element, if applicable, encoded xs:appinfo element identified with a source attribute holding the value “http://www.smpte-ra.org/schemas/434/Groups/DistinguishedValue”;

d. the Local Tag statically assigned to the property, if any, encoded in a xs:appinfo element identified with a source attribute holding the value http://www.smpte-ra.org/schemas/434/Groups/LocalTag. The representation for the local tag value is the one defined for the LocalTag type definition in Annex-B/types/s377m_types.xsd;

In the list of items 3 through 7 above, item number 3 shall be present. All other items are optional.

5.5.2 MXF-XML-Schema Extensions

All metadata groups that are added as extensions to the MXF-XML-Schema shall be associated with a valid SMPTE Key.

The addition of a group to the MXF-XML-Schema shall follow the rules defined section 5.5.1 items 1 through 3. It may contain the information defined in 5.5.1 items 4 through 8.

Groups that are specified to be part of an inheritance hierarchy shall be defined as extension of the type of the group from which they directly derive. All group types that have an Instance UID property shall directly or indirectly derive from *AbstractSetType.* All metadata group types that have a Generation UID property shall directly or indirectly derive from *InterchangeObjectType.*

NOTES

1 *AbstractSetType* and *InterchangeObjectType* are defined in Annex-B/groups/s377m_metadata.xsd.

2 No semantic meaning should be inferred from the use of the term "Set" in *AbstractSetType* and the term "Object" in *InterchangeObjectType*. Both are nothing but symbol names for XML schema type definitions.

The definition of all groups that derive indirectly or directly from *AbstractSetType* shall include the definition of a reference XML element as defined in section 5.6.1. If such an XML reference element is defined, the contents of the xs:choice model groups of all dictionary elements that contain the XML reference element of the base group shall be updated to also include XML reference element of the derived group. The contents of the xs:choice model groups of all dictionary elements that contain the XML element of the base group shall be updated to also include XML element of the derived group.

The MXF-XML-Schema may also be extended by adding metadata properties to a existing group. In this case, the rules outlined in 5.5.1 clause 7 shall apply for each added metadata property.

The MXF-XML-Schema may be extended by adding new descriptive metadata schemes. In this case, the rules for adding metadata sets (as defined above) shall apply. Such metadata sets shall be declared in a separate namespace that is identified by an appropriate namespace URI.

NOTE – Refer to section 5.1 for the specification of namespace URI.

Table 7 – Informative Example – Metadata Set Element

```
<xs:element name="Preface" type="s377mGroups:PrefaceType">
        <xs:annotation>
            <xs:documentation>Defines the Preface set</xs:documentation>
            <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/Name">Preface</xs:appinfo>
            <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/Key">urn:oid:1.3.52.2.83.1.1.13.1.1.1.1.1.47</xs:appinfo>
        </xs:annotation>
</xs:element>
```

Table 8 – Informative Example – Metadata Set Type

```
<xs:complexType name="PrefaceType">
    <xs:complexContent>
        <xs:extension base="s377mGroups:InterchangeObject">
            <xs:sequence>
                <xs:element ref="s335mElements:FileLastModified">
                    <xs:annotation>
                        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">Req
                        </xs:appinfo>
                        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/LocalTag">0x3B02</xs:appinfo>
                    </xs:annotation>
                </xs:element>
                <xs:element ref="s335mElements:FormatVersion">
                    <xs:annotation>
                        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">Req
                        </xs:appinfo>
                        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/LocalTag">0x3B05</xs:appinfo>
                    </xs:annotation>
                </xs:element>
                <xs:element ref="s335mElements:ObjectModelVersion" minOccurs="0">
                    <xs:annotation>
                        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">Opt
                        </xs:appinfo>
                        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/LocalTag">0x3B07</xs:appinfo>
                    </xs:annotation>
                </xs:element>
```

```
<xs:element ref="s335mElements:PrimaryPackage" minOccurs="0">
        <xs:annotation>
                <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">Opt
                </xs:appinfo>
                <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/LocalTag">0x3B08</xs:appinfo>
        </xs:annotation>
</xs:element>
<xs:element ref="s335mElements:IdentificationList">
        <xs:annotation>
                <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">E/req
                </xs:appinfo>
                <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/LocalTag">0x3B06</xs:appinfo>
        </xs:annotation>
</xs:element>
<xs:element ref="s335mElements:ContentStorageObject">
        <xs:annotation>
                <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">Req
                </xs:appinfo>
                <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/LocalTag">0x3B03</xs:appinfo>
        </xs:annotation>
</xs:element>
<xs:element ref="s335mElements:OperationalPattern">
        <xs:annotation>
                <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">Req
                </xs:appinfo>
                <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/LocalTag">0x3B09</xs:appinfo>
```

```
                    </xs:annotation>
                </xs:element>
                <xs:element ref="s335mElements:EssenceContainers" minOccurs="0">
                    <xs:annotation>
                        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">Req
                        </xs:appinfo>
                        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/LocalTag">0x3B0A</xs:appinfo>
                    </xs:annotation>
                </xs:element>
                <xs:element ref="s335mElements:DescriptiveSchemes">
                    <xs:annotation>
                        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">Req
                        </xs:appinfo>
                        <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/LocalTag">0x3B0B</xs:appinfo>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

5.6 References

5.6.1 XML Encoding

Strong and weak references shall be represented in MXF-XML-Documents as formally specified in the MXF-XML-Schema documents in Annex B.

The MXF-XML-Schema normatively defines:

1. for each group that contains an Instance UID property, the XML tag name of an XML element representing a MXF reference to that group;
2. that the reference XML element may appear in an MXF-XML-Document as a replacement of the group XML element;

   NOTE – In the MXF-XML-Schema this is accomplished by adding to the declaration of dictionary entries from which a specific metadata set is referenced, the option to use a reference XML element instead. e.g.: The *ContentStorageObject* dictionary entry (member of the relational dictionary class), is declared in MXF-XML-Schema using a xs:choice content model, which enables the *ContentStorageObject* XML tag in an instance to hold as a child element either a *ContentStorage* XML tag (the actual metadata set representation) or a *ContentStorage_REF* (a reference to the metadata set representation).
3. that a dark reference XML element may appear in an MXF-XML-Document as a replacement of the group XML element;

   NOTES

   1 In the MXF-XML-Schema this is accomplished by adding to the xs:choice mentioned above a *DarkMetadataGroup_REF* XML element.

   2 This provides a mechanism that enables representation of references that cannot be resolved because they point to dark metadata sets. The mechanism also enables the representation of references that point to sets that are missing in the header metadata. The latter can be useful for validating applications or for applications that implement error recovering mechanisms.
4. that the reference XML element or the dark reference XML element shall hold a text representation of the value of the Instance UID property of the target group;
5. that a group in the MXF-XML-Document which has an *InstanceID* element with a value that matches the one represented in each reference XML element in the document, and that the group instance is of the appropriate type (typed references) according to the reference XML element definition.

The fact that a reference XML element tag may be used in place of the metadata set XML element leads to two possible representations of the same data in XML documents:

- a logical model, where each strong reference to a metadata set is resolved such that the target metadata set is represented in its place. This results in a nesting of metadata sets where, in line with SMPTE 377M, all metadata set XML tags shall be descendants of the XML tag representing the Preface metadata set. Weak references are never resolved.
- a physical model, where none of the strong references to metadata sets are resolved, but the reference element are represented instead. This results in metadata sets being represented as a list of sibling XML tag names, resembling the way they are encoded in the MXF stream, hence the name physical model. Weak references are never resolved.

NOTE – Annex D 1.1 contains an example for physical model. Annex D 1.2 contains an example for logical model. Please refer to the representation of the *ContantStorage* property of the Preface set for illustration: In physical model, the *ContentStorageObject* contains a *ContentStorage_REF* element. The *TargetInstance* attribute of this element is the value of the *ContentStorage* element in the MXF file. In logical model, the strong reference has been resolved such that the *ContentStorage_REF* element has been replaced by the *ContentStorage* set element.

An MXF-XML-Document shall follow either the logical model or the physical model.

5.6.2 MXF-XML-Schema Extensions

The addition of a group that directly or indirectly derives from *AbstractSetType* to the MXF-XML-Schema shall include the definition of a reference XML element definition which shall enforce the XML Encoding rules for the reference XML element (as defined in 5.6.1).

The definitions of all dictionary elements that can hold a reference to the new header metadata set shall be updated to include the new reference XML element.

Table 9 – Informative Example – Metadata Set Reference Element

```
<!—Metadata Refrence element -->

<xs:element name="ContentStorage_REF" type=“s377mTypes:Ref"/>

<!—Metadata Set element -->

<xs:element name="ContentStorage" type=“s377mGroups:ContentStorageType">

        <xs:annotation>

                <xs:documentation>Defines the Content Storage set</xs:documentation>

                <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/Name">Content Storage</xs:appinfo>

                <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/Key">urn:oid:1.3.52.2.83.1.1.13.1.1.1.1.1.24</xs:appinfo>

        </xs:annotation>

</xs:element>

<!—Metadata Set Type -->

<xs:complexType name="ContentStorageType">

        <xs:complexContent>

                <xs:extension base=“s377mGroups:InterchangeObject">

                        <xs:sequence>

                                <xs:element ref=“s335mElements:Packages">

                                        <xs:annotation>

                                            <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">Req

                                            </xs:appinfo>

                                        </xs:annotation>

                                </xs:element>

                                <xs:element ref=“s335mElements:EssenceDataObjects" minOccurs="0">

                                        <xs:annotation>

                                            <xs:appinfo source="http://www.smpte-ra.org/schemas/434/Groups/PropertyRequiredStatus">Opt

                                            </xs:appinfo>

                                        </xs:annotation>
```

```
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

To guarantee the concept of typed references defined above, the XML Schema *key* and *keyref* mechanisms defined in the file Annex-B/mxf/metadata.xsd shall be used as follows:

1. for each metadata set, a *key* element shall be defined specifying the value of the *InstanceID* element as the unique identifier in the MXF-XML-Document for the metadata set.

2. for each metadata set, a *keyref* element shall ensure that a metadata set reference element references a metadata set of the correct type.

Table 10 – Informative Example – Metadata Set *key* and *keyref* Elements

```
<xs:key name="mxf_Preface_PK">
        <xs:selector xpath=".//groups:Preface"/>
        <xs:field xpath="@s335m:InstanceID"/>
</xs:key>
<xs:keyref name="mxf_Preface_FK" refer="s377m:mxf_Preface_PK">
        <xs:selector xpath=".//groups:Preface_REF"/>
        <xs:field xpath="@types:TargetInstance"/>
</xs:keyref>
```

5.7 Dark Metadata

Dark metadata, as considered in this text, may appear in an MXF file as:

– unknown groups or

– unknown extension of known groups.

In the second case, the extensions may comprise:

– dark properties, i.e. properties with a Universal Label which is unknown to the application or

– identified properties, i.e. properties with a Universal Label that is known to the application, but which are not defined to appear inside the group.

5.7.1 XML Encoding

a. the set key as an attribute with name *Key*;

b. the number of octets used for BER coding (1 for short form, 2 or more for long form) as an attribute with name *BEROctetCount*;

c. all properties encoded as dark properties as defined in item number 4 below.

2. that all other unknown groups (such as metadata sets with 1-byte local tags or with 4-byte local tags) shall be encoded in XML using the tag name *UnparsedDarkGroup* and shall include the following information:

   a. the group key an attribute with name *Key* ;

   b. the number of octets used for BER coding (1 for short form, 2 or more for long form) as an attribute with name *BEROctetCount*;

   c. the value of the KLV packet encoded as *xs:hexBinary*.

3. that known sets extended with dark properties or with identified properties shall include each of them as *DarkProperty* inside the container *ExtensionProperties* (defined in *AbstractSetType*), which shall be the first child element of the group node;

4. that dark properties of sets shall be encoded in XML using the element name *DarkProperty* and shall include the following information:

   a. the Universal Label of the property unless the MXF-XML-Document is being generated from an MXF file and the Universal Label is missing from the Primer Pack;

   b. the Local Tag unless the local tag is a dynamic local tag or the MXF-XML document is being generated from an MXF file and it cannot be determined that the local tag has been registered in the Primer Pack;

      NOTE – SMPTE 377M-2004 requires only local tags of MXF sets to be registered in the Primer Pack.

   c. the data encoded as *xs:hexBinary*.

5. that identified properties of known sets shall be encoded according to the XML encoding rules defined in section 5.5.1.

5.7.2 Reference Resolution

Strong references to dark metadata sets cannot be resolved by MXF-to-XML-Encoders. The same is true for strong references to known metadata sets, if these references are dark properties. All such metadata sets shall be encoded as defined in section 5.5.1 or 5.7.1, respectively, and:

1. placed under the *MetadataSets* element, as siblings following the Preface set, in the physical model;

2. placed under the *FloatingMetadataGroups* element (child element of the *MetadataSets* element) in the logical model.

5.8 Partiton Multiplex and Index Table Information

For MXF-XML-Documents including only structural and descriptive metadata, the document element shall be *MetadataSets (defined in Annex-B/mxf/metadata.xsd)*.

MXF-XML-Documents may also include information on partition multiplex and index table data structures within the MXF file. In this case the document element shall be *MXFFile (defined in Annex-B/mxf/file.xsd)* and *MetadataSets* shall be a descendant of this element. In this case, child elements of *MXFFile* shall encode in XML information on index tables and packs within the file, as set out formally in the MXF-XML-Schema [Annex B].

NOTE – The XML schema files that contain the definitions for partition multiplex structure and index table information are Annex-B/mxf/file.xsd and Annex-B/mxf/partitions.xsd.

Annex A (informative)
User Requirements

1. Implementation efficiency for processing by encoders/decoders and mappable to database storage.
2. Efficient mapping for lossless (in terms of content) bidirectional (reversible) translation between MXF and MXF-XML.
3. Schema understandable for MXF specialists and for XML specialists.
4. Schema should be able to reflect the logical object model. Some applications need representations of the storage layer (the partition multiplex of MXF). MXF header metadata and storage layer information should be separable.
5. Platform-neutral and programming-language-neutral.
6. Support means of encoding dictionaries (SMPTE).
7. Must be amenable to processing and validation using commonly available XML Schema tools.

   NOTE – This may not allow supporting complete validation of MXF files based on the schema.
8. Must conform to XML 1.0 and XML Schema.
9. Select either element-normal form or attribute-normal form.
10. Schema must be able to lossless represent and transport dark metadata.
11. The schema must be maintainable, extensible and modular.
12. ML instance files do not contain essence container data.

Annex B
MXF-XML-Schema Definition

The XML Schema documents that constitute the MXF-XML-Schema are organized according to the following directory structure.

Table B.1 – Directory and File Structure of the MXF-XML-Schema

```
Annex-B
¦
+---mxf
¦   +---file.xsd (Core)
¦   +---partitions.xsd (Core)
¦   +---metadata.xsd (Core)
¦
¦
+---groups
¦   +---s377m_packs.xsd (Core)
¦   +---dark_metadata.xsd (Core)
¦   +---s377m_index_tables.xsd (Core)
¦   +---s377m_metadata.xsd (Core)
¦   +---s380m_metadata.xsd (Plug-in)
¦   +---s381m_metadata.xsd (Plug-in)
¦   +---s385m_metadata.xsd (Plug-in)
|   +---s422m_metadata.xsd (Plug-in)
|   +---s429-6_metadata.xsd (Plug-in)
¦
¦
+---dictionary
¦   +---rp210.xsd (Core)
¦
¦
+---dataTypes
    +---s377m_types.xsd (Core)
    +---s422m_types.xsd (Plug-in)
```

The files qualified by “(Core)” constitute required parts of the MXF-XML-Schema. They shall always be present.

The files qualified by “(Plug-in)” constitute optional parts of the MXF-XML-Schema.

NOTE – In order to eliminate an optional part from the MXF-XML-Schema, all elements and attributes associated with the namespace defined in this document need to be eliminated from the MXF-XML-Schema file Annex-B/mxf/metadata.xsd and Annex-B/dictionary/rp210.xsd

Annex C (informative)
Steps to Extend the Encoding

There are three distinct cases when extending the MXF-XML encoding: Addition of properties to sets, addition of sets or packs to structural metadata or to existing descriptive metadata schemes, and addition of a new descriptive metadata schemes. This section outlines the actions necessary for each of the cases, and the relation of these cases to each other.

C.1 Adding Properties to Sets

In order to add properties to existing sets, the following steps need to be taken:

1. If the property is not defined as a dictionary element in the registry controlled by SMPTE 335M (i.e., it does not have a SMPTE Universal Label) register it.

2. If the property is defined as a class 1 through 13 dictionary element in the registry controlled by SMPTE 335M, check if the dictionary element is already included in the file dictionary/rp210.xsd.

    a. If not, add the declaration of the dictionary element to file dictionary/rp210.xsd as defined in Section 5.2.

    NOTE – In case the data type of this dictionary element has not yet been defined in dataTypes/smpte377m_types.xsd, the type definition needs to be added. Please refer to Section 5.3.

3. If the property is defined as a class 14 dictionary element in the registry controlled by SMPTE 335M, add it to an XML Schema file for that class, with an appropriate XML namespace URI as illustrated in the file Annex-D.2.1/dictionary/extension_dictionary.

4. Edit the XSD file in schema/groups that contains the declaration of the set and add the property to the declaration of the set.

    a. If the property is defined in a document other than dictionary/rp210.xsd (as in point 2 above), this document must be imported in the schema/groups XSD file.

NOTE – Please refer Annex D Section 2 for an example on how to add properties to sets.

C.2 Adding Sets and Packs

In order to add sets or packs to MXF structural metadata or to existing descriptive metadata schemes, the following steps need to be taken:

1. When adding a set or pack, one of the following two options applies:

    a. If the group is defined in a new essence mapping document or if it is a group for a new (not yet implemented) scheme, the group is added in a new file created for that scheme and identified with a namespace URI, according to 5.1.

    b. If the group is defined in an amendment or revision of SMPTE 377M, an essence mapping document or if it is an extension to an already implemented descriptive scheme, the group is added to the file that already contains all other groups associated with the original specification of that scheme and a new namespace URI is assigned according to 5.1.

    If the group is part of a scheme that is indented to be part of the MXF header metadata, the xs:choice model groups in the definition of the *MetadataSets* and *FloatingMetadataGroups* elements need to be updated according to 5.1.

2. When adding a group that has an Instance UID property, the base type of the new group must be determined. Examples for base types of metadata groups are the *AbstractSetType*, the *InterchangeObjectType* or the *PictureDescriptorType*, defined in Annex-B/groups/s377m_metadata.xsd.

3. Add the declaration of the set or pack as defined in section 5.5 in a new XML Schema file with an appropriate XML namespace URI, as illustrated in groups/extension_metadata.xsd.

   NOTE – Please refer to section C.1 for a description how to add properties to sets and packs.

4. When adding a group that has an Instance UID property and that is intended to be part of the MXF header metadata, add the declarations for the reference mechanism as defined in section 5.6.2.

NOTE – Please refer to file groups/extension_metadata.xsd of Annex D 2.1 for an example on how to add sets.

C.3 Adding Metadata Schemes

In order to add a new metadata scheme (such as a new descriptive metadata scheme), the following steps need to be taken:

1. Add the metadata sets of the descriptive metadata scheme as described in C.2.

2. Add reference(s) to the descriptive metadata framework set(s) of the new scheme to the content model of the *DescriptiveFrameworkObject* element declared in dictionary/rp210.xsd.

Annex D (informative)
Examples

D.1 MXF-XML Documents conformant with Annex B

D.1.1 Metadata Only – Physical Model

The MXF-XML-Document Annex-D.1/MetadataSets-Physical+Descriptive.xml demonstrates XML encoding of an MXF file's metadata following the physical model.

D.1.2 Metadata Only – Logical Model

The MXF-XML-Document Annex-D.1/MetadataSets-Logical+Descriptive.xml demonstrates XML encoding of an MXF file's metadata following the logical model.

D.1.3 Partition Multiplex, Index Table and Metadata – Physical Model

The MXF-XML-Document Annex-D.1/File-Multiplex-Physical+Descriptive.xml demonstrates XML encoding of the partition multiplex of an MXF file including index tables and physical model MXF header metadata.

D.1.4 Metadata Only – Logical Model – DarkProperty

The MXF-XML-Document Annex-D.1/MetadataSets-Logical+Descriptive+DarkProperty.xml demonstrates XML encoding of dark properties of known sets.

The dark property is associated with the *ProductionFramework* set. In the MXF-XML-Document this dark property is encoded as child element of the XML element *ExtensionProperties*, which is the first child element of the *ProductionFramework* element.

D.1.5 Metadata Only – Logical Model – UnparsedDarkGroup

The MXF-XML-Document Annex-D.1/MetadataSets-Logical+Descriptive+UnparsedDarkGroup.xml demonstrates the XML encoding of a dark metadata set that uses a local tag sizes other than two bytes.

The *UnparsedDarkGroup* element is a child element of the *FloatingMetadataGroups* element, which is the second child of the *MetadataSets* element.

D.1.6 Metadata Only – Logical Model – ParsedDarkSet

The MXF-XML-Document Annex-D.1/MetadataSets-Logical+Descriptive+ParsedDarkSet.xml demonstrates the XML encoding of a dark metadata set that uses a local tag sizes of two bytes.

The *ParsedDarkSet* element is a child element of the *FloatingMetadataGroups* element, which is the second child of the *MetadataSets* element.

D.2 MXF-XML Encoding Extension Examples

D.2.1 Schema Extension

An extended version of the MXF-XML-Schema incorporated into this document as an example on how to apply the extension rules defined in this specification.

The following is the structure of directories that contains the XML schema documents for this example.

Table D.1 – Directory and File Structure of the Extended Schema

```
Annex-D.2.1
¦
+---mxf
¦  +---file.xsd (Core)
¦  +---partitions.xsd (Core)
¦  +---metadata.xsd (Core, edited)
¦
+---groups
¦  +---s377m_packs.xsd (Core)
¦  +---dark_metadata.xsd (Core)
¦  +---s377m_index_tables.xsd (Core)
¦  +---s377m_metadata.xsd (Core)
¦  +---s380m_metadata.xsd (Plug-in, edited)
¦  +---s381m_metadata.xsd (Plug-in)
¦  +---s385m_metadata.xsd (Plug-in)
¦  +---s422m_metadata.xsd (Plug-in)
¦  +---s429-6_metadata.xsd (Plug-in)
¦  +---extension_metadata.xsd xsd (Plug-in)
¦
+---dictionary
¦  +---rp210.xsd (Core)
¦  +---extension_dictionary.xsd (Plug-in)
¦
+---dataTypes
   +---s377m_types.xsd (Core)
   +---s422m_types.xsd (Plug-in)
   +---extension_types.xsd (Plug-in)
```

The names of new files that have been added to the original MXF-XML-Schema of Annex-B are written in bold fonts. Files that are present in Appendix-A, but that have been edited in order to implement the extensions are identified by the word “edited” that has been added in bold font in the parentheses. All modifications inside files that where already present in Annex-B start with the comment <!-- Schema Extension Start --> and end with the comment <!-- Schema Extension End -->.

The file Annex-D.2.1/dataTypes/extension_types.xsd defines the data type *InternalIDType*. The example uses urn:OrganisationName:Types as the namespace URI for this data type.

The file Annex-D.2.1/dictionary/dictionary_extensions.xsd defines a number of dictionary elements. In the example, all of the elements have SMPTE Labels associated with them that fall into category 15 (Experimental) of the SMPTE metadata dictionary. The example uses urn:OrganisationName:Dictionary as the namespace URI for these dictionary elements. The values of the elements *InternalProcessID* and *EmployeeID* use the *InternalIDType* data type.

The file Annex-D.2.1/groups/extension_metadata.xsd contains all definitions required for a metadata set named *PostIt*. The set extends *InterchangeObjectType*. In addition of the properties defined in this file it therefore has an *InstanceID* property, an optional Linked*GenerationID* property and an optional

ExtensionProperties XML element than can hold *DarkProperties*. The example uses urn:OrganisationName:Groups as the namespace URI for this metadata set.

NOTE – The definition of this single set could also have been directly added to the file Annex-D.2.1/groups/s380m_metadata.xsd. It has been added in a separate file to also illustrate the addition of a new scheme (consisting of one set) to already defined header metadata.

The file Annex-D.2.1/groups/s380m_metadata.xsd has been edited to import the files Annex-D.2.1/dictionary/dictionary_extensions.xsd and Annex-D.2.1/groups/extension_metadata.xsd. In addition to this, the definitions of the *ProductionFramework* and the *SceneFramework* sets have been updated to hold additional properties. One of these properties (*PostItLis*') is an array of strong references to *PostIt* metadata sets, defined in groups/extension_metadata.xsd. This demonstrates how to add a new metadata set to hierarchy of MXF metadata sets. According to the provisions of section 0, the namespace for this file has been changed. The example uses urn:OrganisationName:S380M:2005 as the new namespace URI for this scheme.

The file Annex-D.2.1/mxf/metadata.xsd has been edited to import the files Annex-D.2.1/dictionary/dictionary_extensions.xsd and Annex-D.2.1/groups/extension_metadata.xsd. The declaration of the choice model groups inside the *MetadataSets* and *FloatingMetadataGroups* elements declaration has been updated to include <xs:any namespace="urn:OrganisationName:Groups" processContents="strict" minOccurs="0" maxOccurs="unbounded"/>. The namespace URI http::www.smpte-ra.org/schemas/434/2006/groups/S380M/2004 has been changed to urn:OrganisationName:S380M:2005 and the reference mechanism of *PostItList* has been defined according to 5.6.2.

NOTE – The above extensions are examples that illustrate the use of the mechanisms defined in this specification. The Universal Labels used for the new dictionary entries and the KLV key for new metadata set lie within the Experimental class of the SMPTE registries. They should not be used for implementation.

D.2.2 MXF-XML Document conformant with Extended Schema

The MXF-XML-Document Annex-D.2.2/MetadataSets-Logical+Descriptive+Extensions.xml incorporated into this annex demonstrates the extension mechanisms.

The extensions included in the MXF-XML-Document are compliant with the extended schema described in section D.2.1.

The extensions can be found in the MXF-XML-Document as XML elements with prefix *extElements*, for dictionary extensions, and *extGroups*, for the extension metadata set *PostIt*.

Annex E (informative)
Bibliography

SMPTE RP 210, Metadata Dictionary Registry of Metadata Element Descriptions

SMPTE EG 41-2004, Material Exchange Format (MXF) – Engineering Guideline

SMPTE EG 42-2004, Material Exchange Format (MXF) – MXF Descriptive Metadata