# SMPTE STANDARD

# MXF KLV-Encoded Extension Syntax (KXS)



## Table of Contents




445 Hamilton Avenue., White Plains, NY 10601
+1 (914) 761-1100

Approved July 19, 2019

## 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 its Standards Operations Manual. This SMPTE Engineering Document was prepared by Technology Committee 31FS.

## Intellectual Property

At the time of publication no notice had been received by SMPTE claiming patent rights essential to the implementation of this Engineering Document. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

## Introduction (Informative)

This section is entirely informative and does not form an integral part of this Engineering Document.

MXF permits encoders to include KLV packets in files other than those that are defined and documented by SMPTE ST 377-1 and companion documents. In the text of SMPTE ST 377-1 these are known as KLV-coded dark components. A particular method of incorporating dark components into MXF files is described by SMPTE ST 377-1 section 9.7 Application Metadata Plugins.

SMPTE ST 377-1 does not define a format for describing the syntax or the contents of the dark metadata. Therefore, decoders that wish to interpret the packets require extra information from external sources that might not be available at the time of decoding. Without this extra information, decoders are limited to treating the contents as dark metadata, removing them, or passing them through without explicit knowledge of their contents (extensions conforming to Application Metadata Plugins can be removed from MXF files without the necessity of the extra information).

This specification defines a format for describing the syntax and the contents of KLV packets that can be used by KXS decoders to interpret the KLV-coded dark components. This enables KXS encoders to include both the desired additional KLV data and the definition of this data within one MXF file, so that it can be interpreted at its intended destination by KXS decoders that can parse the extra information. The extra information is called KLV-Encoded Extension Syntax (KXS).

KXS Encoders might include such KXS packets to describe any of a number of extensions, including (but not limited to) the following:

- Packets defined by new parts of the MXF specification – for example, the Sub-descriptors defined by SMPTE ST 422
- Packets defined by SMPTE specifications that are not part of the MXF specification – for example, the Camera Positioning Information defined by SMPTE ST 315
- Packets that encode user information in addition to the standard MXF structural and descriptive metadata, intended to be interchanged to cognizant decoders – for example, user annotation of mark points in a sequence
- Packets that encode parametric information in addition to the standard MXF metadata, intended to be available as optional hints to cognizant decoders – for example, MPEG Recoding Data Set information as described by SMPTE ST 327
- Packets defined in general terms by SMPTE specifications where the detailed KLV encoding is contained in registers defined by SMPTE – for example, as described by SMPTE ST 380 section 5.8

- Packets defined by revisions of the MXF specification, for example Application Metadata Plug-Ins as defined by SMPTE ST 377-1 section 9.7.

Examples of these extensions and of the KXS that describes them are contained in section 9 below.

KXS describes KLV-coded dark components using Class, Property and Type Definitions that specify how they are derived from the standard KLV packets defined by SMPTE ST 377-1. Thus, KXS decoders can identify the standard component with which the extension is associated, and if desired, can remove the extensions.

KXS defines not only extensions that are intended to be used identically by all KXS encoders, but also extensions that can be used repeatedly by the same encoder or several encoders. This permits several applications to add extensions independently of each other.

KXS can describe Application Metadata Plugins, thus allowing KXS encoders to create files that can be used both by decoders with only SMPTE ST 377-1 capability and by KXS decoders with KXS capability.

# 1 Scope

This standard specifies a method and data structures known as MXF – KLV-Encoded Extension Syntax (KXS), which can be used within files conforming to SMPTE ST 377-1 – Material Exchange Format.

The extension definitions can be omitted from MXF files.

# 2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; Tables shall be next; followed by formal languages; then figures; and then any other language forms.

# 3 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 ST 335:2012 Metadata Element Dictionary Structure
SMPTE ST 395:2014 Metadata Groups Register
SMPTE ST 336:2017 Data Encoding Protocol Using Key-Length-Value
SMPTE ST 377-1:2011 Material Exchange Format (MXF) — File Format Specification
SMPTE ST 377-1:2011 Am1:2012 Material Exchange Format (MXF) — File Format Specification — Amendment 1
SMPTE ST 377-1:2011 Am2:2012 Material Exchange Format (MXF) — File Format Specification — Amendment 2
SMPTE ST 400:2012 SMPTE Labels Structure
SMPTE ST 2001-1:2013 XML Representation of SMPTE Registered Data (Reg-XML) - Mapping Rules
SMPTE ST 2003:2012 Types Dictionary Structure
OMG Unified Modeling Language™ (OMG UML), Superstructure, Version 2.2 (January 2009), http://www.omg.org

# 4 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

MXF-related acronyms, terms, and data types are defined in SMPTE ST 377-1, and those definitions are not repeated here to avoid any divergence of meaning.

## 4.1 Terms

This specification introduces the following new terms:

**Attribute**: A generic term meaning either a property or a behavior of a class. In this specification, the sections labeled “Attributes” describe the data model. The representation of attributes as properties of sets is described in the sections labeled “KLV Encoding”

**Definition Object**: Describes extensions. The purpose of Definition Objects is to provide sufficient information for interchange of Dictionary Definitions between applications.

**Dictionary Definition:** Describes the meaning of specific ULs and UUIDs when they appear as values of properties within the MXF file.

**Hosting Property**: A property of type strong reference to a Mixin Class.

**Host Class**: A class that contains a hosting property.

**KXS**: KLV-Encoded Extension Syntax as defined by sections 6, 7, and 8 of this specification.

**Meta Reference:** An Association by reference relationship from source Objects to target Objects, implemented in MXF using AUIDs. Meta References are a many-to-one relationship (one Object may be the target of many Meta References in source Objects). Meta References are typed. The targets of Meta References need not be contained within the file; however, all targets of a Meta Reference of a given type that are contained in the file shall be aggregated by Strong Reference in a single Property of a single Class within the file.

**Mixin Class**: A set of properties to be used by another class; a Mixin Class is not meant to stand alone. A Mixin Class is not a form of specialization, but rather a means to collect functionality.

**Mixin Object**: An instance of a Mixin Class that provides values of a set of properties to be used as part of another object, but is not meant to stand alone.

## 4.2 Notation

This document uses Unified Modeling Language (UML) class diagrams to depict the class model.

# 5 General

## 5.1 Extension Mechanisms

KLV-Encoded Extension Syntax (KXS) builds upon the KLV structure of SMPTE ST 377-1, and provides additional extension definitions by KLV encoding of a dictionary and metadictionary.

The extension definitions are transparent to MXF decoders, being encoded fully within the provisions of SMPTE ST 377-1 “KLV-coded dark components”. The extension definitions augment and do not supersede or modify the information that is carried in the Primer Pack.

Applications may add Dictionary Definitions to MXF Files, and they may extend MXF files using any of the following mechanisms, which are described in detail in subsequent sections of this specification:

- Addition of Optional Properties to MXF Sets
- Addition of Mandatory Properties to MXF Sets
- Derivation of new Classes and Addition of Objects to MXF Files
- Addition of Mixin Objects to MXF Sets

Dictionary Definitions are described by Definition Objects. In an MXF file, these are placed within the Dictionary in the MXF Preface. Definition Objects are described in detail in section 6.

All other extensions are described by MetaDefinition Objects. In an MXF file, these are placed within the Extensions Header in the MXF Header Metadata. The Extensions Header is described in detail in section 8, which also defines the scope over which they apply. MetaDefinition Objects are further described in subsections of section 8.

The data types used to represent the values of Properties are described by Type Definitions. Applications may define new data types that are derived from a few fundamental data types. In an MXF file, Type Definitions are placed within the Extensions Header in the MXF Header Metadata. They are described in detail in section 8.8 and following sections.

## 5.2 Addition of Dictionary Definitions to MXF Files

Dictionary Definitions describe the meaning of specific ULs and UUIDs that are the values of properties within the MXF file.

Note: For example, MXF Sets derived from the Structural Component class have properties called DataDefinition that identify a kind of essence that is used – Picture, Sound, or other Data. The meaning of the values of these properties can be described by Dictionary Definitions.

Dictionary Definitions may be added to MXF files using Definition Objects as described in section 6.

## 5.3 Wrapping of Properties

Optional and mandatory properties may be used by the class that is specified in their PropertyDefinition or any class derived from that class.

Properties may also be used in other classes, by providing a PropertyAliasDefinition as described in section 8.7. The PropertyAliasDefinition references the original PropertyDefinition, and specifies the class in which the reuse may occur, and provides a globally unique UL or UUID to identify the specific reuse of the original property.

## 5.4 Derivation of New Classes and Addition of Objects to MXF Files

New classes may be defined as subclasses of an existing class as illustrated in Figure 1.

Note: New classes are typically defined in order to delineate specialized behavior of the base class, which might require the presence of mandatory properties or provide a vehicle to which to add optional properties.

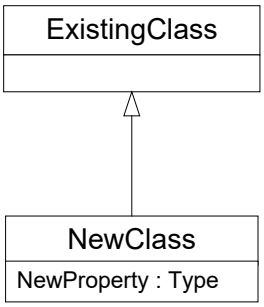


**Figure 1 – Defining a New Class**

A Class Definition object shall be provided for every new class, and a Property Definition object shall be provided for every new property, as described below.

Class Definition are described in section 8.5 below, and specify the base class from which the new class is derived.

Property Definitions are described in section 8.6 below, and specify the class that may contain instances of the Property or whose subclasses may contain instances of the Property. Property Definitions also specify the Identification and Type of instances of the Property.

Instances of the specified class shall include instances of all mandatory properties of the class. Instances of the specified class shall include instances of all mandatory properties of the class from which they are derived. Instances of the specified class may include instances of optional properties.

Instances of the specified class shall identify the class using the Identification property of the Class Definition object. The value of the Identification property may be encoded in the ObjectClass property of the base class, or it may be used as the KLV Key of the instance.

Encoders should assume that some decoders will ignore instances of classes other than classes whose KLV Key is defined by SMPTE ST 377-1.

## 5.5 Addition of Mixin Objects to MXF Sets

One or more Mixin Objects may be added to MXF files as illustrated in Figure 2.

Note 1: A Mixin Object is an instance of a Mixin Class that provides values of a set of properties to be used as part of another object, but is not meant to stand alone. An example of a Mixin Class is the Essence Descriptor class that is added to MXF Source Packages via the Descriptor property, as shown by SMPTE ST 377-1.

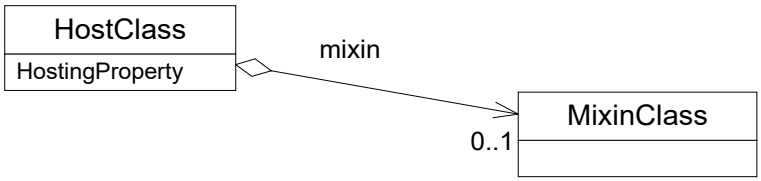


**Figure 2 – Adding a single Mixin Object**

The Mixin Class may be specialized by subclassing or by the addition of optional properties.

Individual instances of a Mixin Class may be added to a host class by adding a property to the host class, of type Strong Reference to the base Mixin Class.

The hosting property may be an optional property on an existing class, in which case each host object in the file may have a Mixin Object. The hosting property may be a mandatory property defined in the course of deriving a new subclass, in which case each host object in the file shall have a Mixin Object.

A host class may have any number of different Mixin Classes added to it, by adding a different hosting property for each Mixin Class. A host class may have any number of instances of the same Mixin Class added to it, by adding a different hosting property for each Mixin Class. A Mixin Class may be added to several host classes, by adding a different hosting property to each host class.

Several instances of the same Mixin Class may be added to a single host class by adding a property to the host class, of type Array of Strong Reference to the base Mixin Class or Batch of Strong Reference to the base Mixin Class as illustrated in Figure 3.

Note 2: Multiple instances of a Mixin Class might describe multiple data (for example, the SubDescriptors within a MultipleDescriptor in SMPTE ST 377-1), or perhaps describe multiple facets of a single piece of data (for example, the SubDescriptors defined on the MXF Generic Descriptor by SMPTE ST 422, of which one specialization is the JPEG2000 Picture SubDescriptor).

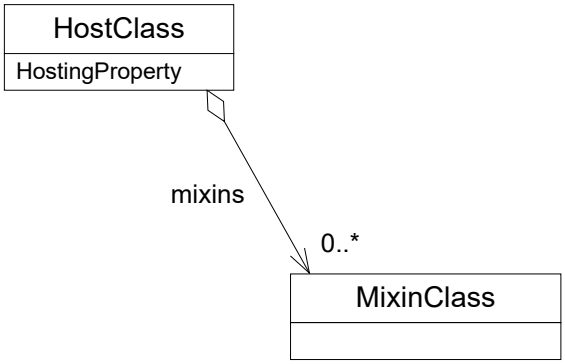


**Figure 3 – Adding multiple Mixin Objects**

The hosting property may be an optional property on an existing class, in which case each host object in the file may have an array or batch of Mixin Objects. The hosting property may be a mandatory property defined in the course of deriving a new subclass, in which case each host object in the file shall have an array or batch of Mixin Objects.

## 5.6 Addition of Shared Mixin Objects to MXF Files

One or more shared Mixin Objects may be added to MXF files as illustrated in Figure 4.

A shared Mixin Object is an instance of a Mixin Class that is expected to be referenced from several objects in a file. Shared Mixin Objects are also known as Plug-Ins.

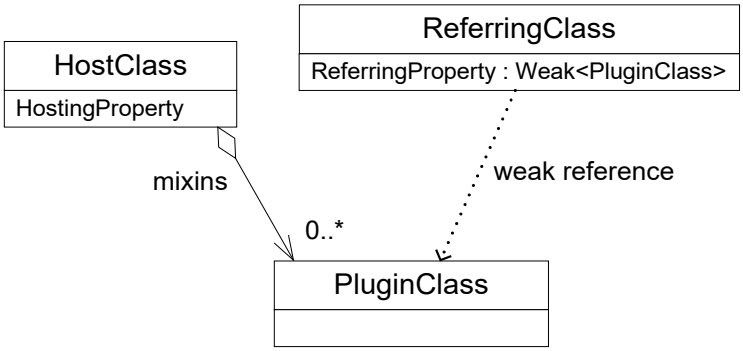


**Figure 4 – Adding Shared Mixin Objects**

Shared Mixin Classes may be added to MXF files by providing hosting properties that strongly reference the Mixin Class, as properties of the Dictionary or Content Storage.

Other classes in the file may weakly reference shared Mixin Classes via properties of type weak reference to the Mixin Class, or array or batch of weak reference to the class.

# 6 Addition of Definition Objects to MXF Files

## 6.1 Dictionary Class

### 6.1.1 Dictionary Class description

Applications that add Dictionary Definitions to MXF files shall also provide Definition Objects that describe those extensions. The purpose of Definition Objects is to provide additional information for interchange of Dictionary Definitions between applications.

Applications may encode Definition Objects as KLV Packets contained within the MXF File in a set of Definitions property in the optional Dictionary Mixin Class of the Preface, or they may publish Definition Objects separately from the MXF File in a publicly accessible Registry maintained by the SMPTE Registration Authority.

The Dictionary Mixin Class contains sets of Definition Objects that describe the meaning of specific ULs and UUIDs that are the values of properties within the MXF file.

The Dictionary Mixin Class shall be strongly referenced by the optional Dictionary hosting property of the Preface as illustrated in Figure 5.

Note: The Dictionary Hosting property has the UL: urn:smpte:ul:060e2b34.01010102.06010104.02020000.

An MXF file may contain zero or one instance of the Dictionary Mixin Class.

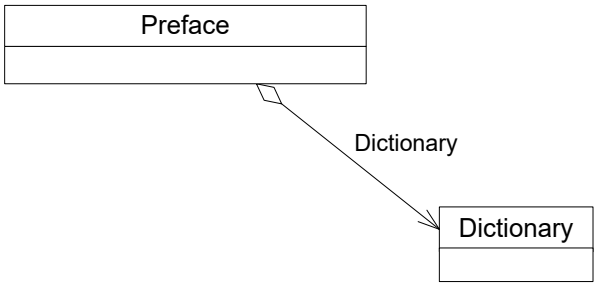


**Figure 5 – Adding Dictionary to Preface**

### 6.1.2 Attributes of the Dictionary class

The Dictionary class is a subclass of the InterchangeObject class as illustrated in Figure 6.

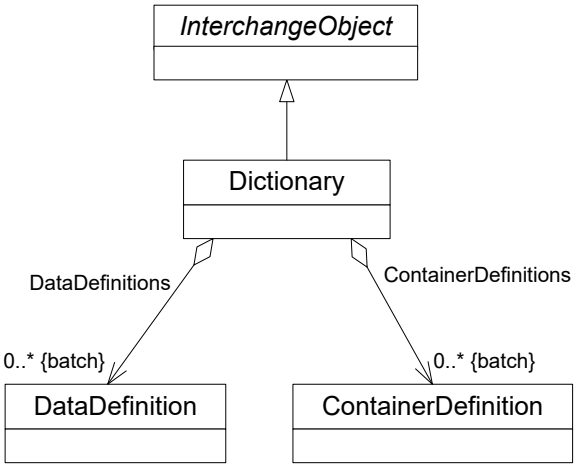


**Figure 6 – Dictionary class**

A Dictionary object shall contain the required attributes of its parent class.

A Dictionary object may contain the optional attributes of its parent class and may contain the following attributes:

- DataDefinitions; specifies the Data Definitions that are used in the file.
- ContainerDefinitions; specifies Container Definitions that are used in the file.

Note: Since the Dictionary is a subclass of the InterchangeObject class, it is an MXF Set as defined by ST 377-1 and can be extended or subclassed as described by ST 377-1 section 9.

### 6.1.3 KLV Encoding of the Dictionary class

When encoded as KLV Packets, the Dictionary shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths with the label in Table 1.

**Table 1 – Universal Label for MXF Dictionary Set**

| Byte No. | Description | Value (hex) | Meaning |
|---|---|---|---|
| 1-13 | Defined in the Structural Header Metadata Implementation section of SMPTE ST 377-1 | | |
| 14 | Set Kind (1) | 01h | Structural Metadata Set |
| 15 | Set Kind (2) | 22h | Dictionary Set |
| 16 | Reserved | 00h | Reserved |

The attributes shall be encoded as shown in Table 2:

**Table 2 - KLV Encoding of MXF Dictionary Set**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| Dictionary | Set Key | 16 | | See Table 1 | Req | Defines the Dictionary set |
| Length | BER Length | var | | | Req | Set length |
| All items from the Interchange Object defined in SMPTE ST 377-1. | | | | | | |
| Data Definitions | DataDefinitionStrongReferenceSet | 8+16n | 26.05 | 06.0E.2B.34 01.01.01.02 06.01.01.04 05.05.00.00 | Opt | Specifies the Data Definitions that are used in the file |
| Container Definitions | ContainerDefinitionStrongReferenceSet | 8+16n | 26.08 | 06.0E.2B.34 01.01.01.02 06.01.01.04 05.08.00.00 | Opt | Specifies Container Definitions that are used in the file |

## 6.2 Definition Object Class (Abstract)

### 6.2.1 Definition Object Description

The DefinitionObject class defines the meaning of Unique Identifiers used as the values of properties in an MXF file. These may include but not limited to Data Definition Property of Structural Component, Essence Container Property of File Descriptor and Codec Property, also of File Descriptor.

The DefinitionObject class is an abstract class.

### 6.2.2 Attributes of the Definition Object class

The DefinitionObject class is a subclass of the InterchangeObject class as illustrated in Figure 7.

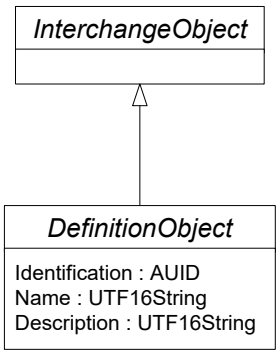


**Figure 7 – DefinitionObject Class**

DefinitionObject objects shall contain the required attributes of their parent class and shall contain the following attributes:

- Identification; the Unique Identifier being defined.
- Name; specifies the display name of the Unique Identifier being defined.

DefinitionObject objects may contain the optional attributes of their parent class and may contain the following attributes:

- Description; provides an explanation of the use of the Unique Identifier being defined.

Note: Since the DefinitionObject is a subclass of the InterchangeObject class, it is an MXF Set as defined by ST 377-1 and can be extended or subclassed as described by ST 377-1 section 9.

When a Definition Object is constructed from an entry in the Labels Registry (SMPTE ST 400), the values of these attributes shall be set according to Table 3.

**Table 3 - Definition Object Attribute Values**

| Attribute | Mapping | ST 400 Field | ST 400 Section |
|---|---|---|---|
| Identification | value of | URN representation of the universal label | 4.2.6 |
| Name | value of | Name | 4.2.9 |
| Description | value of | Definition | 4.2.10 |

## 6.2.3 KLV Encoding of the Definition Object class

When encoded as KLV Packets, Definition Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths with the Universal Label in Table 4

**Table 4 - Universal Label for MXF Definition Object Set**

| Byte No. | Description | Value (hex) | Meaning |
|---|---|---|---|
| 1-13 | Defined in the Structural Header Metadata Implementation section of SMPTE ST 377-1 | | |
| 14 | Set Kind (1) | 01h | Structural Metadata Set |
| 15 | Set Kind (2) | 1Ah | Definition Object Set |
| 16 | Reserved | 00h | Reserved |

The attributes shall be encoded according to Table 5.

**Table 5 - MXF Definition Object Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| Definition Object | Set Key | 16 | | See Table 4 | Req | Defines the Definition Object set |
| Length | BER Length | var | | | Req | Set length |
| All items from the Interchange Object defined in SMPTE ST 377-1 | | | | | | |
| Definition Object Identification | AUID | 16 | 1B.01 | 06.0e.2B.34 01.01.01.02 01.01.15.03 00.00.00.00 | Req | The Unique Identifier being defined |
| Definition Object Name | UTF16String | var | 1B.02 | 06.0E.2B.34 01.01.01.02 01.07.01.02 03.01.00.00 | Req | The display name of the Unique Identifier being defined |
| Definition Object Description | UTF16String | var | 1B.03 | 06.0E.2B.34 01.01.01.02 03.02.03.01 02.01.00.00 | Opt | Provides an explanation of the use of the Unique Identifier being defined |

## 6.3 DataDefinition class

### 6.3.1 DataDefinition Description

The DataDefinition class specifies the kind of data that can be stored in an MXF Structural Component.

Note: The UL identifying a DataDefinition is used as the value of properties that specify data kinds. These include, but not limited to, the Data Definition property of MXF Structural Component.

An MXF file may contain any number of instances of subclasses of DataDefinition objects; all instances shall be contained within the DataDefinitions property of the Dictionary object.

### 6.3.2 Attributes of the DataDefinition class

The DataDefinition class is a subclass of the DefinitionObject class as illustrated in Figure 8.

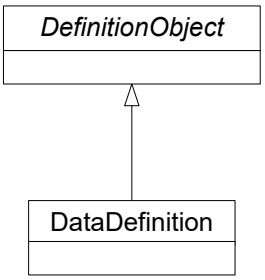


**Figure 8 – DataDefinition Class**

DataDefinition objects shall contain the required attributes of their parent class.

DataDefinition objects may contain the optional attributes of their parent class.

### 6.3.3 KLV Encoding of the DataDefinition class

When encoded as KLV Packets, DataDefinition objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths with the Universal Label inTable 6.

**Table 6 : Universal Label for MXF DataDefinition Set**

| Byte No. | Description | Value (hex) | Meaning |
|---|---|---|---|
| 1-13 | Defined in the Structural Header Metadata Implementation section of SMPTE ST 377-1 | | |
| 14 | Set Kind (1) | 01h | Structural Metadata Set |
| 15 | Set Kind (2) | 1Bh | Data Definition Set |
| 16 | Reserved | 00h | Reserved |

## 6.4 ContainerDefinition class

### 6.4.1 ContainerDefinition Description

The ContainerDefinition class specifies the identifier of the Essence Container defined in ST 377-1.

Note 1: A container can be either a kind of file, such as an MXF file or it can be another mechanism for storing essence data.

Note 2: The UL identifying a ContainerDefinition is used as the value of properties that specify containers. These include, but not limited to, the Essence Container property of MXF File Descriptor.

An MXF file may contain any number of instances of subclasses of ContainerDefinition objects; all instances shall be contained within the ContainerDefinitions property of the Dictionary object.

### 6.4.2 Attributes of the ContainerDefinition class

The ContainerDefinition class is a subclass of the DefinitionObject class as illustrated in Figure 9.

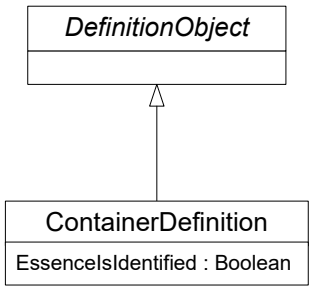


**Figure 9 – ContainerDefinition Class**

ContainerDefinition objects shall contain the required attributes of their parent class.

ContainerDefinition objects may contain the optional attributes of their parent class and may contain the following attributes:

- EssenceIsIdentified; specifies whether the Container includes unique identifiers of the instance of the essence.

### 6.4.3 KLV Encoding of the ContainerDefinition class

When encoded as KLV Packets, ContainerDefinition objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths with the Universal Label in Table 7.

**Table 7 - Universal Label for MXF ContainerDefinition Set**

| Byte No. | Description | Value (hex) | Meaning |
|---|---|---|---|
| 1-13 | Defined in the Structural Header Metadata Implementation section of SMPTE ST 377-1 | | |
| 14 | Set Kind (1) | 01h | Structural Metadata Set |
| 15 | Set Kind (2) | 20h | Container Definition Set |
| 16 | Reserved | 00h | Reserved |

The attributes shall be encoded according to Table 8.

**Table 8 - MXF ContainerDefinition Encoding**

| | Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|---|
| | Container Definition | Set Key | 16 | | See Table 10 | Req | Defines the Container Definition set |
| ↔ | Length | BER Length | var | | | Req | Set length |
| | All items from the DefinitionObject class | | | | | | |
| | Essence IsIdentified | Boolean | 1 | 24.01 | 06.0E.2B.34 01.01.01.01 03.01.02.01 03.00.00.00 | Opt | TRUE if the Container includes unique identifiers of the instance of the essence |

# 7 Addition of MetaDefinition Objects to MXF Files

## 7.1 MetaDefinition Object

Applications that extend MXF files shall also provide MetaDefinition Objects that describe those extensions. The purpose of MetaDefinition Objects is to provide precise information for interchange of extensions between applications.

Applications may encode the MetaDefinition Objects as KLV Packets contained within the MXF File in an Extension Set in the Extensions Header, or they may publish the MetaDefinition Objects separate from the MXF File in a publicly accessible Registry maintained by the SMPTE Registration Authority.

Every instance of a MetaDefinition Object with a given value of the Identification property shall contain the same values of all mandatory properties.

This means that MetaDefinition Objects are immutable, and decoders can assume that the meaning of one instance is the same as the meaning of every other instance. Thus, decoders can obtain MetaDefinition objects from within an MXF file, or can use MetaDefinitions contained in one MXF file to decode the contents of other MXF files, or can obtain MetaDefinitions from an external source such as a Registry. All these methods produce equivalent results.

## 7.2 Data Model of MetaDefinition Objects

MetaDefinition Objects shall include at least the attributes listed in the “Attributes” subsections of section 8 below.

Note: Since the MetaDefinition object is not a subclass of the InterchangeObject class, it is not an MXF Set as defined by ST 377-1 and cannot be extended or subclassed as described by ST 377-1 section 9.

Associations between MetaDefinition Objects are described as attributes whose Type is Strong Reference or Meta Reference.

Figure 42 illustrates the MetaDefinition Object class hierarchy.

## 7.3 KLV Encoding of MetaDefinition Objects

When encoded as KLV Packets, MetaDefinition Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths.

Local tags shall be statically allocated from the range defined in SMPTE ST 377-1:
00.01h to 00.FFh Reserved for compatibility with AAF

The KLV encoding of strong references shall be implemented as an aggregation by reference between KLV Sets using UUIDs as the identifier of the target.

Targets of KLV encoded strong references shall be contained in the same file as the referencing object.

The KLV encoding of Meta References shall be implemented as an aggregation by reference between KLV Sets using the value of the Identification attribute as the identifier of the target.

Targets of KLV encoded Meta References need not be contained in the same file as the referencing object.

Attributes of Meta Definition Objects shall be encoded as described in the “KLV Encoding” subsections of section 8 below.

The ULs for MXF MetaDefinition sets shall be as defined in Table 9 and Table 10.

**Table 9 - Universal Label for MXF MetaDefinition Sets**

| Byte No. | Description | Value (hex) | Meaning |
|---|---|---|---|
| 1 | Object Identifier | 06h | |
| 2 | Label size | 0Eh | |
| 3 | Designator | 2Bh | ISO, ORG |
| 4 | Designator | 34h | SMPTE |
| 5 | Registry Category Designator | 02h | Groups (Sets and Packs) |
| 6 | Registry Designator: | xxh | Local Sets: 2-byte Local Tags with either 2-byte length (default) or BER encoded length |
| 7 | Structure Designator | 01h | Set/Pack Registry |
| 8 | Version Number | 01h | Registry Version |
| 9 | Item Designator | 0Dh | Organizationally Registered |
| 10 | Organization | 01h | AAF Association |
| 11 | Application | 01h | MXF / AAF Association Structural Metadata Sets |
| 12 | Structure version | 01h | Structure Version 1 |
| 13 | Structure Kind | xxh | MetaDefinitions (02h) or Root (03h) |
| 14 | Set Kind (1) | yyh | MetaDefinition Set |
| 15 | Set Kind (2) | zzh | MetaDefinition Set Variant |
| 16 | Reserved | 00h | Reserved |

**Table 10 – Byte Number 14 and 15 for MXF MetaDefinition Sets**

| Set Name | Value of Byte 14 (hex) | Value of Byte 15 (hex) | Comments |
|---|---|---|---|
| Root | 00h | 00h | |
| ExtensionScheme | 26h | 00h | New |
| MetaDefinition | 24h | 00h | |
| ClassDefinition | 01h | 00h | |
| PropertyDefinition | 02h | 00h | |
| PropertyAliasDefinition | 27h | 00h | New |
| TypeDefinition | 03h | 00h | |
| TypeDefinitionInteger | 04h | 00h | |
| TypeDefinitionStrongObjectReference | 05h | 00h | |
| TypeDefinitionWeakObjectReference | 06h | 00h | |
| TypeDefinitionEnumeration | 07h | 00h | |
| TypeDefinitionFixedArray | 08h | 00h | |
| TypeDefinitionVariableArray | 09h | 00h | |
| TypeDefinitionSet | 0Ah | 00h | |
| TypeDefinitionString | 0Bh | 00h | |
| TypeDefinitionStream | 0Ch | 00h | |
| TypeDefinitionRecord | 0Dh | 00h | |
| TypeDefinitionRename | 0Eh | 00h | |
| TypeDefinitionExtendibleEnumeration | 20h | 00h | |
| TypeDefinitionIndirect | 21h | 00h | |
| TypeDefinitionOpaque | 22h | 00h | |
| TypeDefinitionCharacter | 23h | 00h | |
| MetaDictionary | 25h | 00h | Deprecated |
| ExtendibleEnumerationElement | 28h | 00h | New |

# 8 Extensions Header

## 8.1 Extensions Header usage

The MetaDefinitions that are contained in the MXF file shall be contained in the Extensions Header as a series of KLV packets. The Extensions Header shall be placed immediately after the Primer Pack in the Header Partition of an MXF File.

The Extensions Header shall be placed before the Preface and the remainder of the Structural and Descriptive Metadata.

Decoders shall not be required to insert an Extensions Header in Partitions that have a Header Byte Count equal to zero.

Decoders may ignore an Extensions Header if the Partition Pack is marked Open or Incomplete.

Note: this allows decoders to avoid look-ahead processing to interpret any extensions that are encountered. All MetaDefinitions are read and can be decoded before starting to process the Preface. The MetaDefinitions apply to all metadata that follows. The entire Preface and all structural and descriptive Metadata Sets are thus within the scope of the MetaDefinitons.

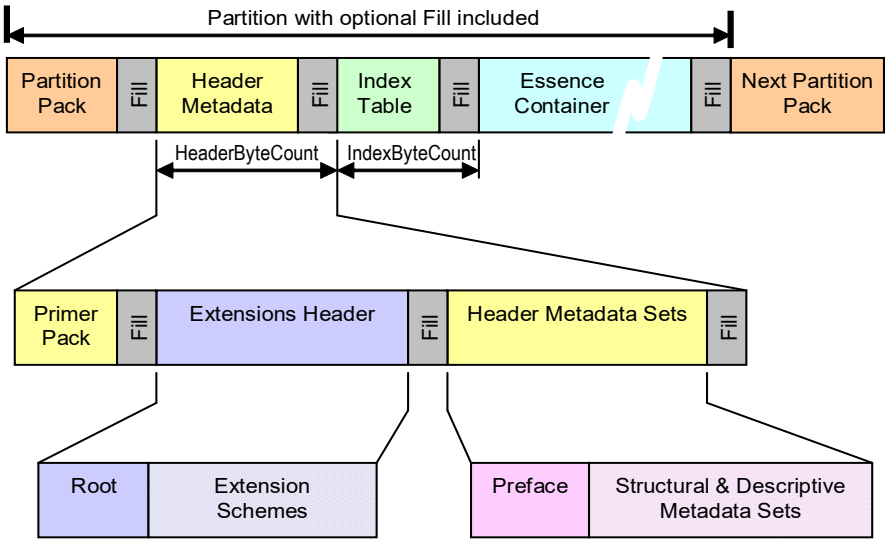


**Figure 10 - Placement of Extensions Header in a partition**

Applications should collect MetaDefinitions into Extension Schemes. Each Extension Scheme shall be assigned a globally unique identifier. The identifier shall be recorded in the ExtensionSchemeID attribute of the Extension Scheme.

The Extensions Header shall start with the Root object. The Root object contains strong references to all Extension Schemes that are used in the file.

## 8.2 Root Class

### 8.2.1 Root Class Description

The role of the Root class is to logically associate the definitions of the extensions used in a file with the structural and descriptive metadata in the file.

An MXF file shall contain zero or one Root object.

For backwards compatibility, the Root object is optional. When no Root object is present in a file, the structural and descriptive metadata in the file shall be interpreted in accordance with SMPTE ST 377-1.

### 8.2.2 Attributes of the Root class

The Root class shall strongly reference the Preface and all the Extensions Schemes that are contained in the file, and shall contain a format version number to provide for future revisions to the meta-model as illustrated in Figure 11.

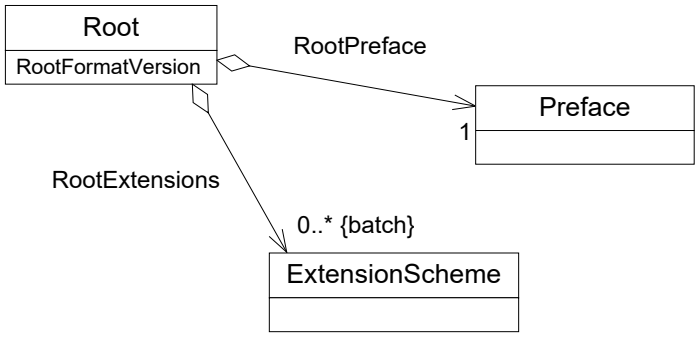


**Figure 11 – Root**

Root objects shall contain the following attributes:

- RootPreface
- RootExtensions

Root objects may contain the following attributes:

- RootFormatVersion

### 8.2.3 KLV Encoding of the Root class

When encoded as KLV Packets, Root Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 11.

**Table 11 - Root Class Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| Root | Set Key | 16 | | See Table 10 | Req | Defines the Root set |
| Length | BER Length | var | | | Req | Set length |
| InstanceUID | UUID | 16 | 3C.0A | 06.0e.2b.34 01.01.01.01 01.01.15.02 00.00.00.00 | Req | Unique ID of this instance |
| RootPreface | PrefaceStrongReference | 16 | 00.02 | 06.0e.2b.34 01.01.01.0a 06.01.01.07 17.00.00.00 | Req | References the Preface of the file |
| RootExtensions | ExtensionSchemeStrongReferenceSet | 8+16n | 00.23 | 06.0e.2b.34 01.01.01.0d 06.01.01.07 1a.00.00.00 | Req | References the Extension Schemes that are used in the file |
| RootFormatVersion | UInt32 | 4 | 00.22 | 06.0e.2b.34 01.01.01.0a 06.01.01.07 19.00.00.00 | Opt | Simple integer version number of MetaModel. The value, if present, shall be 12h or greater |

## 8.3 ExtensionScheme class

### 8.3.1 ExtensionScheme Class Description

The role of the Extension Scheme class is to group together a collection of MetaDefinitions for an extension to an MXF file and to provide a globally unique identifier for the collection.

An MXF file may contain any number of Extension Scheme objects.

### 8.3.2 Attributes of the ExtensionScheme class

The Extension Scheme class references all the MetaDefinition objects that are included in the collection, and contains a globally unique identifiers and additional descriptive information for the collection as illustrated in Figure 12.

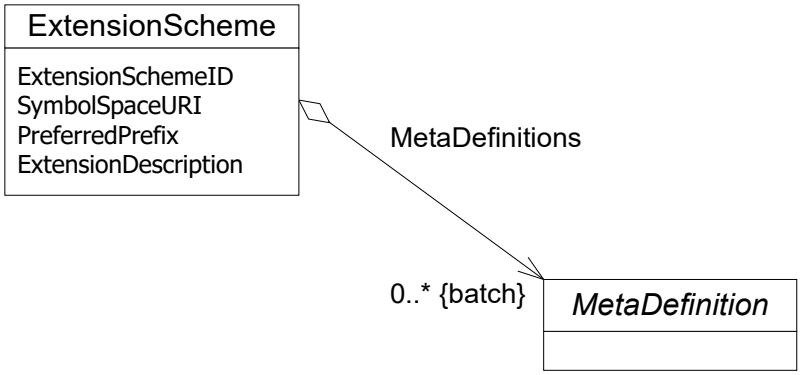


**Figure 12 – ExtensionScheme**

Extension Scheme objects shall contain the following attributes:

- ExtensionSchemeID
- SymbolSpaceURI

Extension Scheme objects may contain the following attributes:

- PreferredPrefix
- ExtensionDescription
- MetaDefinitions

### 8.3.3 KLV Encoding of the ExtensionScheme class

When encoded as KLV Packets, Extension Scheme Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 12.

**Table 12 - Extension Scheme Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| ExtensionScheme | Set Key | 16 | | See Table 10 | Req | Defines the ExtensionScheme set |
| Length | BER Length | var | | | Req | Set length |
| InstanceUID | UUID | 16 | 3C.0A | 06.0e.2b.34 01.01.01.01 01.01.15.02 00.00.00.00 | Req | Unique ID of this instance |
| ExtensionSchemeID | AUID | 16 | 00.24 | 06.0e.2b.34 01.01.01.0d 06.01.01.07 1b.00.00.00 | Req | Globally unique identification of the ExtensionScheme |
| SymbolSpaceURI | UTF16String | Var | 00.25 | 06.0e.2b.34 01.01.01.0d 06.01.01.07 1c.00.00.00 | Req | Namespace URI for the Extension Scheme |
| PreferredPrefix | UTF16String | Var | 00.26 | 06.0e.2b.34 01.01.01.0d 06.01.01.07 1d.00.00.00 | Opt | Preferred namespace tag when SMPTE ST 2001-1 Reg-XML encoding is used |
| Extension Description | UTF16String | Var | 00.27 | 06.0e.2b.34 01.01.01.0d 06.01.01.07 1e.00.00.00 | Opt | Description of the Extension Scheme |
| MetaDefinitions | MetaDefinitionStrongReferenceSet | 8+16n | 00.28 | 06.0e.2b.34 01.01.01.0d 06.01.01.07 1f.00.00.00 | Opt | References all the MetaDefinitions in this Extension Scheme when they are contained in the MXF file |

## 8.4 MetaDefinition class (Abstract)

### 8.4.1 MetaDefinition Class Description

The MetaDefinition class is an abstract class that defines a class, type, or property in a file.

The MetaDefinition class is an abstract class. Only subclasses of MetaDefinition shall be instantiated in MXF files.

An MXF file may contain any number of subclasses of MetaDefinition objects.

### 8.4.2 Attributes of the MetaDefinition class

The MetaDefinition class shall contain a globally unique Identification and a Description for the item being defined, and a Name that is unique within a SymbolSpace as illustrated in Figure 13.

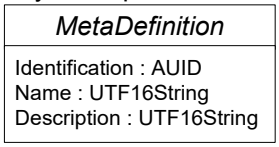


**Figure 13 – MetaDefinition**

MetaDefinition objects shall contain the following attributes:

- Identification
- Name

MetaDefinition objects may contain the following attributes:

- Description

When a MetaDefinition Object is constructed from an entry in the Groups Registry (SMPTE ST 395), the values of these attributes shall be set to the UL, Name, and Definition field of the entry according to Table 13.

**Table 13 - Metadefinition Object Attribute Values**

| Attribute | Mapping | ST 395 Field | ST 395 Section |
|---|---|---|---|
| Identification | value of | Universal label | 4.5.7 |
| Name | value of | Name | 4.5.10 |
| Description | value of | Definition | 4.5.11 |

Note: The value of the Identification is of type AUID. If these are ULs, they can be found in Registries, whereas UUIDs can be found by lookup within ExtensionSchemes but are not necessarily registered. For easiest possible lookup, applications are advised to use registered ULs whenever possible.

### 8.4.3 KLV Encoding of the MetaDefinition class

When encoded as KLV Packets, MetaDefinition Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 14.

**Table 14 - Metadefinition Class Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| MetaDefinition | Set Key | 16 | | See Table 10 | Req | Defines the MetaDefinition set |
| Length | BER Length | var | | | Req | Set length |
| InstanceUID | UUID | 16 | 3C.0A | 06.0e.2b.34 01.01.01.01 01.01.15.02 00.00.00.00 | Req | Unique ID of this instance |
| Identification | AUID | 16 | 00.05 | 06.0E.2B.34 01.01.01.02 06.01.01.07 13.00.00.00 | Req | Globally unique identification of the MetaDefinition |
| Name | UTF16String | var | 00.06 | 06.0E.2B.34 01.01.01.02 03.02.04.01 02.01.00.00 | Req | Specifies the display name of the item being defined |
| Description | UTF16String | Var | 00.07 | 06.0E.2B.34 01.01.01.02 06.01.01.07 14.01.00.00 | Opt | Provides an explanation of the use of the item being defined |

## 8.5 ClassDefinition class

### 8.5.1 ClassDefinition Class Description

The ClassDefinition class extends the existing class hierarchy by specifying a new class or by defining additional optional properties for an existing class.

All ClassDefinition objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of ClassDefinition objects.

### 8.5.2 Attributes of the ClassDefinition class

The ClassDefinition class is a sub-class of the MetaDefinition class as illustrated in Figure 14.

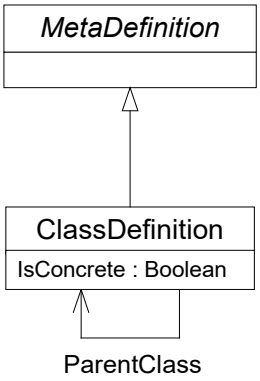


**Figure 14 – ClassDefinition**

ClassDefinition objects shall contain the required attributes of their parent class and shall contain the following attributes:

- ParentClass; specifies the parent of the class being defined.
- IsConcrete; specifies if the class is concrete.

ClassDefinition objects may contain the optional attributes of their parent class.

When a MetaDefinition Object is constructed from an entry in the Groups Registry (SMPTE ST 395), the values of these attributes shall be set according to Table 15.

**Table 15 - Metadefinition Object Attribute Values**

| Attribute | Mapping | ST 395 Field | ST 395 Section |
|---|---|---|---|
| ParentClass | value of | Parent Group | 4.5.12 |
| IsConcrete | value of | Abstract | 4.5.13 |

Note: MXF files created by legacy applications may contain the following attributes:

- Properties

Additional description is contained in section 10.2 below.

If the class is concrete, it may be used to create concrete instances. If the class is not concrete, then it is abstract. Any object in a file that belongs to an abstract class shall also belong to a concrete subclass of the abstract class.

### 8.5.3 KLV Encoding of the ClassDefinition class

When encoded as KLV Packets, ClassDefinition Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 16.

**Table 16 - Class Definition Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| ClassDefinition | Set Key | 16 | | See Table 10 | Req | Defines the ClassDefinition set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| ParentClass | ClassDefinitionWeakReference | 16 | 00.08 | 06.0E.2B.34 01.01.01.02 06.01.01.07 01.00.00.00 | Req | Specifies the parent of the class being defined |
| IsConcrete | Boolean | 1 | 00.0A | 06.0E.2B.34 01.01.01.02 06.01.01.07 03.00.00.00 | Req | Specifies if the class is concrete. If the class is not concrete, then it is abstract. Any object in a file that belongs to an abstract class shall also belong to a concrete subclass of the abstract class |

Any class extension shall be descended from the InterchangeObject class. A ClassDefinition object specifying the InterchangeObject class shall have a ParentClass property with a weak reference to itself.

## 8.6 PropertyDefinition class

### 8.6.1 PropertyDefinition Class Description

The PropertyDefinition class describes properties allowed for a class.

All PropertyDefinition objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

### 8.6.2 Attributes of the PropertyDefinition class

The PropertyDefinition class is a sub-class of the MetaDefinition class as illustrated in Figure 15.

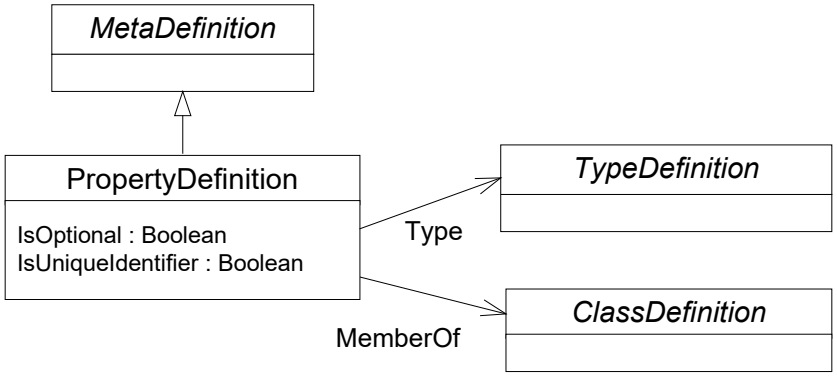


**Figure 15 – PropertyDefinition**

PropertyDefinition objects shall contain the required attributes of their parent class and shall contain the following attributes:

- Type; specifies the Property type.
- IsOptional; specifies whether the Property is optional or required.

PropertyDefinition objects may contain the optional attributes of their parent class and may contain the following attributes:

- MemberOf; specifies the Class in which this Property may be present.

- IsUniqueIdentifier; specifies whether this property provides a unique identification for instances within a collection of unique instances of this class.

When a PropertyDefinition Object is constructed from an entry in the Groups Registry (SMPTE ST 395), the values of these attributes shall be set according to Table 17.

**Table 17 - PropertyDefinition Attribute Values**

| Attribute | Mapping | ST 395 Field | ST 395 Section |
|---|---|---|---|
| Type | value of Type UL of | Item | 4.5.16 |
| IsOptional | value of | Contents Optional | 4.5.20 |
| MemberOf | value of | Definition | 4.5.11 |
| IsUniqueIdentifier | value of | IsUniqueIdentifier | 4.5.19 |

The values of the attributes of the MetaDefinition superclass shall be set from the fields of each ST 395 Contents Item according to Table 18.

**Table 18 - Metadefinition Superclass Attribute Values**

| Attribute | Mapping | ST 395 Field | ST 395 Section |
|---|---|---|---|
| Identification | value of | Item | 4.5.16 |
| Name | value of Name of | Item | 4.5.16 |
| Description | value of Definition of | Item | 4.5.16 |

### 8.6.3 KLV Encoding of the Property Definition class

When encoded as KLV Packets, PropertyDefinition Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 19.

**Table 19 - PropertyDefinition Class Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| PropertyDefinition | Set Key | 16 | | See Table 10 | Req | Defines the PropertyDefinition set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| PropertyType | TypeDefinitionWeakReference | 16 | 00.0B | 06.0E.2B.34 01.01.01.02 06.01.01.07 04.00.00.00 | Req | Specifies the property type |
| MemberOf | ClassDefinitionWeakReference | 16 | 00.2B | 06.0E.2B.34 01.01.01.0d 06.01.01.07 22.00.00.00 | Opt | Specified the class in which this property may be present |
| IsOptional | Boolean | 1 | 00.0C | 06.0E.2B.34 01.01.01.02 03.01.02.02 01.00.00.00 | Req | Specifies whether instances of this class may omit a value for the property |
| IsUniqueIdentifier | Boolean | 1 | 00.0E | 06.0E.2B.34 01.01.01.02 06.01.01.07 06.00.00.00 | Opt | Specifies that this property provides a unique identification for instances within a collection of unique instances of this class |

A PropertyDefinition object specifies that a property can be used in instances of the class identified by the MemberOf property.

## 8.7 PropertyAliasDefinition class

### 8.7.1 PropertyAliasDefinition Class Description

The PropertyAliasDefinition class specifies a globally unique alias for a property that was originally defined to be used in another class. This allows the property to be used unambiguously in more than one class.

Note: If the alias that is provided is a UL, the property can also be used unambiguously when the class is KLV encoded as a Universal Set outside of MXF files.

### 8.7.2 Attributes of the PropertyAliasDefinition class

The PropertyAliasDefinition class is a sub-class of the MetaDefinition class as illustrated in Figure 16.

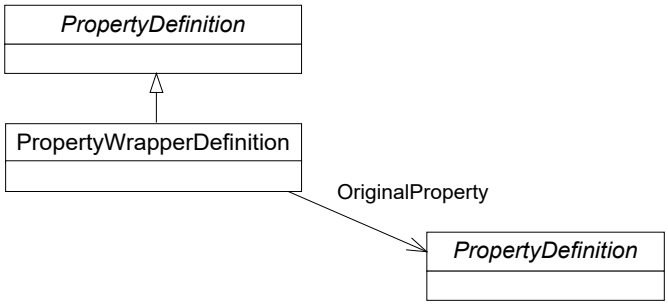


**Figure 16 – PropertyAliasDefinition**

PropertyAliasDefinition objects shall contain the required attributes of their parent class and shall contain the following attributes:

- OriginalProperty; specifies the original definition of the reused Property.

PropertyAliasDefinition objects may contain the optional attributes of their parent class.

### 8.7.3 KLV Encoding of the PropertyAliasDefinition class

When encoded as KLV Packets, PropertyAliasDefinition Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 20.

**Table 20 - PropertyAliasDefinition Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| PropertyAliasDefinition | Set Key | 16 | | See Table 10 | Req | Defines the PropertyAliasDefinition set |
| Length | BER Length | var | | | Req | Set length |
| All items from the PropertyDefinition class defined in 8.6 above | | | | | | |
| OriginalProperty | PropertyDefinitionWeakReference | 16 | 00.29 | 06.0E.2B.34 01.01.01.0d 06.01.01.07 20.00.00.00 | Req | Specifies the original definition of the reused Property |

Note: a PropertyAliasDefinition object specifies that a property is a redefinition of the property identified by OriginalProperty, that can be used in instances of the class identified by the MemberOf property of the parent class.

## 8.8 TypeDefinition class (Abstract)

### 8.8.1 TypeDefinition Class Description

The TypeDefinition class defines a property type.

The TypeDefinition class is an abstract class.

An MXF file may contain any number of TypeDefinition objects.

### 8.8.2 Attributes of the TypeDefinition class

The TypeDefinition class is a sub-class of the MetaDefinition class as illustrated in Figure 17.

The TypeDefinition class does not define any additional properties.

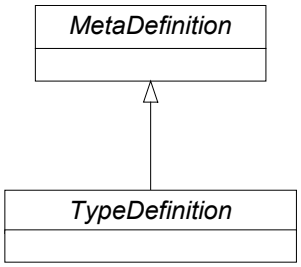


**Figure 17 – TypeDefinition**

TypeDefinition objects shall contain the required attributes of their parent class.

TypeDefinition objects may contain the optional attributes of their parent class.

### 8.8.3 KLV Encoding of the TypeDefinition class

When encoded as KLV Packets, TypeDefinition Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 21.

**Table 21 - TypeDefinition Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinition | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinition set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |

## 8.9 TypeDefinitionInteger class

### 8.9.1 TypeDefinitionInteger Class Description

The TypeDefinitionInteger class describes a property type that is an integer with the specified number of bytes.
All TypeDefinitionInteger objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionInteger objects.

## 8.9.2 Attributes of the TypeDefinitionInteger class

The TypeDefinitionInteger class is a sub-class of the TypeDefinition class as illustrated in Figure 18.

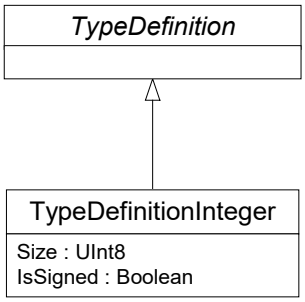


**Figure 18 – TypeDefinitionInteger**

TypeDefinitionInteger objects shall contain the required attributes of their parent class and shall contain the following attributes:

- Size; specifies the number of bytes to store the integer.
- IsSigned; specifies if the integer is signed or unsigned.

TypeDefinitionInteger objects may contain the optional attributes of their parent class.

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the values of these attributes shall be set according to Table 22.

**Table 22 - TypeDefinitionInteger Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| Size | value of | Type Size | 4.4.12 |
| IsSigned | value of IsSigned qualifier of | IsSigned | 4.4.15 |

## 8.9.3 KLV Encoding of the TypeDefinitionInteger class

When encoded as KLV Packets, TypeDefinitionInteger Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according toTable 23.

**Table 23 - TypeDefinitionInteger Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinitionInteger | Set Key | 16 | | See Table 10 | Req | Defines the Meta TypeDefinitionInteger Definition set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| Size | UInt8 | 1 | 00.0F | 06.0E.2B.34 01.01.01.02 03.01.02.03 01.00.00.00 | Req | Specifies the number of bytes to store the integer. Legal values are 1, 2, 4, and 8 |
| IsSigned | Boolean | 1 | 00.10 | 06.0E.2B.34 01.01.01.02 03.01.02.03 02.00.00.00 | Req | Specifies if the integer is signed (True) or unsigned (False) |

## 8.10 TypeDefinitionCharacter class

### 8.10.1 TypeDefinitionCharacter Class Description

The TypeDefinitionCharacter class defines a property type that has a value of a single 2-byte character.

All TypeDefinitionCharacter objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionCharacter objects.

### 8.10.2 Attributes of the TypeDefinitionCharacter class

The TypeDefinitionCharacter class is a sub-class of the TypeDefinition class as illustrated in Figure 19.

The TypeDefinitionCharacter class does not define any additional properties.

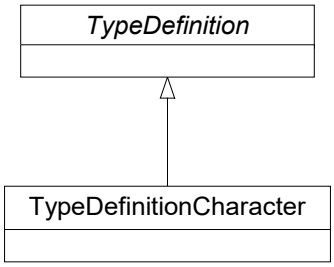


**Figure 19 – TypeDefinitionCharacter**

TypeDefinitionCharacter objects shall contain the required attributes of their parent class.

TypeDefinitionCharacter objects may contain the optional attributes of their parent class.

### 8.10.3 KLV Encoding of the TypeDefinitionCharacter class

When encoded as KLV Packets, TypeDefinitionCharacter Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 24.

**Table 24 - TypeDefinitionCharacter Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinitionCharacter | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionCharacter set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |

## 8.11 TypeDefinitionString class

### 8.11.1 TypeDefinitionString Class Definition

The TypeDefinitionString class defines a property type that consists of a zero-terminated array of the underlying character or integer type.

All TypeDefinitionString in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionString objects.

### 8.11.2 Attributes of the TypeDefinitionString class

The TypeDefinitionString class is a sub-class of the TypeDefinition class as illustrated in Figure 20. The TypeDefinitionString class references a Type Definition.

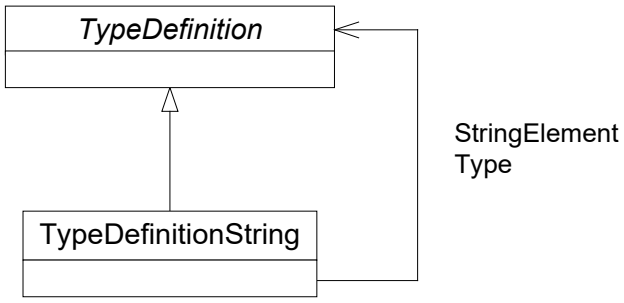


**Figure 20 – TypeDefinitionString**

TypeDefinitionString objects shall contain the required attributes of their parent class and shall contain the following attributes:

- StringElementType; specifies the string element.

TypeDefinitionString objects may contain the optional attributes of their parent class.

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the values of these attributes shall be set according to Table 25.

**Table 25 - TypeDefinitionString Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| StringElementType | value of | Base Type | 4.4.13 |

### 8.11.3 KLV Encoding of the TypeDefinitionString class

When encoded as KLV Packets, TypeDefinitionString Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 26.

**Table 26 - TypeDefinitionString Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinitionString | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionString set |
| Length | BER Length | Var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| StringElementType | TypeDefinitionWeakReference | 16 | 00.1B | 06.0E.2B.34 01.01.01.02 06.01.01.07 0F.00.00.00 | Req | Specifies the string element, which may be a character (TypeDefinitionCharacter) or integer (TypeDefinitionInteger) |

## 8.12 TypeDefinitionStream class

### 8.12.1 TypeDefinitionStream Class Definition

The TypeDefinitionStream class defines a property type that is stored in a stream and has a value that consists of a varying number of the bytes. The order of the bytes is meaningful.

All TypeDefinitionStream objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionStream objects.

### 8.12.2 Attributes of the TypeDefinitionStream class

The TypeDefinitionStream class is a sub-class of the TypeDefinition class as illustrated in Figure 21.

The TypeDefinitionStream class does not define any additional properties.

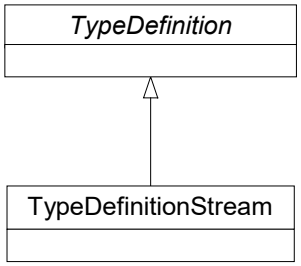


**Figure 21 – TypeDefinitionStream**

TypeDefinitionStream objects shall contain the required attributes of their parent class.

TypeDefinitionStream objects may contain the optional attributes of their parent class:

### 8.12.3 KLV Encoding of the TypeDefinitionStream class

When encoded as KLV Packets, TypeDefinitionStream Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths. according to Table 27.

**Table 27 - TypeDefinitionStream Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinitionStream | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionStream set |
| Length | BER Length | Var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |

## 8.13 TypeDefinitionRecord class

### 8.13.1 TypeDefinitionRecord Class Description

The TypeDefinitionRecord class defines a property type that consists of an ordered set of fields, where each field has a name and type.

All TypeDefinitionRecord objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionRecord objects.

## 8.13.2 Attributes of the TypeDefinitionRecord class

The TypeDefinitionRecord class is a sub-class of the TypeDefinition class as illustrated in Figure 22.

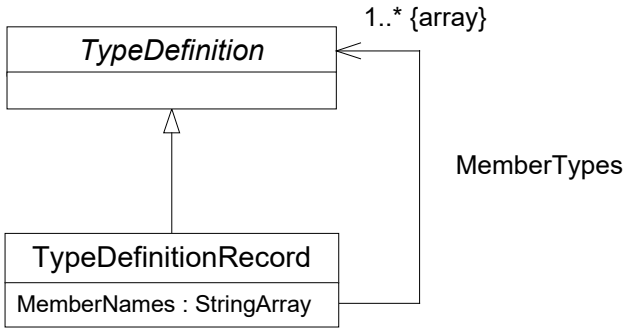


**Figure 22 – TypeDefinitionRecord**

TypeDefinitionRecord objects shall contain the required attributes of their parent class and shall contain the following attributes:

- MemberTypes; specifies the type of each element of the record.
- MemberNames; specifies the name of each element of the record.

TypeDefinitionRecord objects may contain the optional attributes of their parent class.

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the values of these attributes shall be set according to Table 28.

**Table 28 - TypeDefinitionRecord Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| MemberTypes | list of values of | Facet Type | 4.4.16 |
| MemberNames | list of values of | Facet Symbol | 4.4.16 |

## 8.13.3 KLV Encoding of the TypeDefinitionRecord class

When encoded as KLV Packets, TypeDefinitionRecord Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 29.

**Table 29 - TypeDefinitionRecord Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinition Record | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionRecord set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| MemberTypes | TypeDefinitionWeakReferenceVector | 8+16n | 00.1C | 06.0E.2B.34 01.01.01.02 06.01.01.07 11.00.00.00 | Req | Specifies the type of each element of the record |
| MemberNames | UTF16StringArray | var | 00.1D | 06.0E.2B.34 01.01.01.02 03.01.02.03 06.00.00.00 | Req | Specifies the name of each element of the record |

## 8.14 TypeDefinitionEnumeration class

### 8.14.1 TypeDefinitionEnumeration Class Description

The TypeDefinitionEnumeration class defines a property type that can have one of a set of integer values.

All TypeDefinitionEnumeration objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionEnumeration objects.

### 8.14.2 Attributes of the TypeDefinitionEnumeration class

The TypeDefinitionEnumeration class is a sub-class of the TypeDefinition class as illustrated in Figure 23.

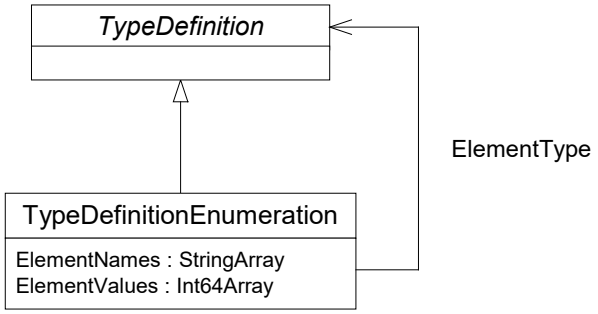


**Figure 23 – TypeDefinitionEnumeration**

TypeDefinitionEnumeration objects shall contain the required attributes of their parent class and shall contain the following attributes:

- ElementType; specifies the TypeDefinition that defines the underlying integer type.
- ElementNames; specifies the names associated with each enumerated value.
- ElementValues; specifies the valid enumerated values.

TypeDefinitionEnumeration objects may contain the optional attributes of their parent class.

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the values of these attributes shall be set according to Table 30.

**Table 30 - TypeDefinitionEnumeration Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| ElementType | value of | Base Type | 4.4.13 |
| ElementNames | list of values of | Facet Symbol | 4.4.16 |
| ElementValues | list of values of | Facet Value | 4.4.16 |

### 8.14.3 KLV Encoding of the TypeDefinitionEnumeration class

When encoded as KLV Packets, TypeDefinitionEnumeration Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 31.

**Table 31 - TypeDefinitionEnumeration Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinitionEnumeration | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionEnumeration set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| ElementType | TypeDefinitionWeakReference | 16 | 00.14 | 06.0E.2B.34 01.01.01.02 06.01.01.07 0B.00.00.00 | Req | Specifies the TypeDefinition that defines the underlying integer type |
| ElementNames | UTF16StringArray | var | 00.15 | 06.0E.2B.34 01.01.01.02 03.01.02.03 04.00.00.00 | Req | Specifies the names associated with each enumerated value |
| ElementValues | Int64Array | 8+8n | 00.16 | 06.0E.2B.34 01.01.01.02 03.01.02.03 05.00.00.00 | Req | Specifies the valid enumerated values. Each value is a positive integer value. Each name and value is unique within the enumeration typedefinition |

## 8.15 TypeDefinitionExtendibleEnumeration class

### 8.15.1 TypeDefinitionExtendibleEnumeration Class Description

The TypeDefinitionExtendibleEnumeration class defines a property type that can have one of an extendible set of AUID values.

All TypeDefinitionExtendibleEnumeration objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionExtendibleEnumeration objects.

### 8.15.2 Attributes of the TypeDefinitionExtendibleEnumeration class

The TypeDefinitionExtendibleEnumeration class is a sub-class of the TypeDefinition class as illustrated in Figure 24.

The TypeDefinitionExtendibleEnumeration class does not define any additional properties.

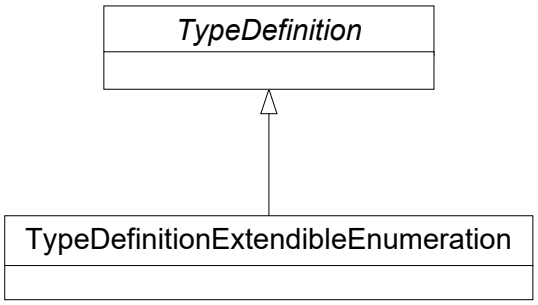


**Figure 24 – TypeDefinitionExtendibleEnumeration**

TypeDefinitionExtendibleEnumeration objects shall contain the required attributes of their parent class.

TypeDefinitionExtendibleEnumeration objects may contain the optional attributes of their parent class.

Note: MXF files created by legacy applications may contain the following attributes:

- ElementNames
- ElementValues

Additional description is contained in section 10.3 below.

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the values of these attributes shall be set according to Table 32.

**Table 32 - TypeDefinitionExtendibleEnumeration Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| ElementNames | list of values of | Facet Symbol | 4.4.16 |
| ElementValues | list of values of | Facet Value | 4.4.16 |

### 8.15.3 KLV Encoding of the TypeDefinitionExtendibleEnumeration class

When encoded as KLV Packets, TypeDefinitionExtendibleEnumeration Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths. according to Table 33

**Table 33 - TypeDefinitionExtendibleEnumeration Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinition Extendible Enumeration | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionExtendibleEnumeration set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |

## 8.16 ExtendibleEnumerationElement class

### 8.16.1 Attributes of the ExtendibleEnumerationElement class

The ExtendibleEnumerationElement class is a concrete subclass of the Definition Object class described in section 6.2 above, as illustrated in Figure 25.

An ExtendibleEnumerationElement references the Extendible Enumerations in which it is known to be used. An ExtendibleEnumerationElement may be used elsewhere in MXF files or other KLV structures.

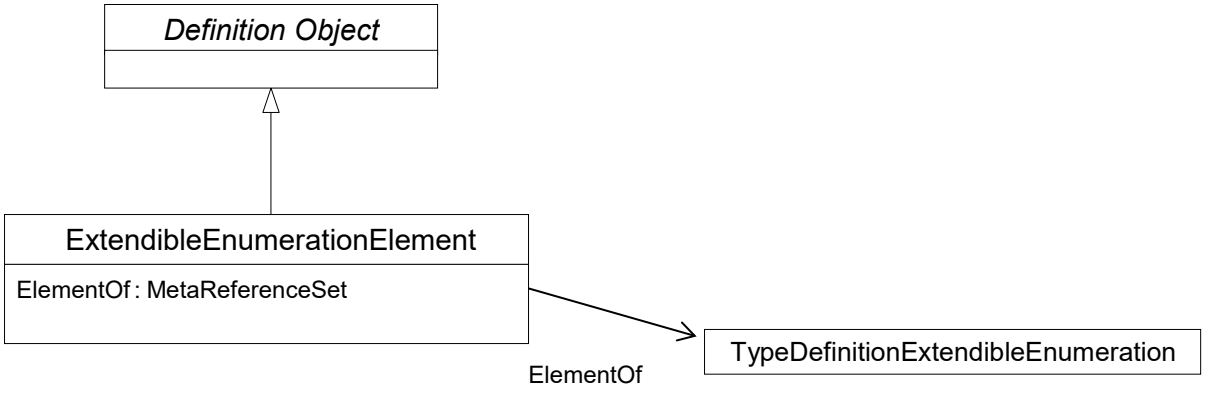


**Figure 25 – ExtendibleEnumerationElement**

ExtendibleEnumerationElement objects shall contain the required properties of their parent class.

ExtendibleEnumerationElement objects may contain the optional attributes of their parent class and may contain the following attributes:

- ElementOf

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the value of this attribute shall be set according to Table 34.

**Table 34 - ExtendibleEnumerationElement Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| ElementOf | value of | Base Type | 4.4.13 |

## 8.16.2 KLV Encoding of the ExtendibleEnumerationElement class

When encoded as KLV Packets, ExtendibleEnumerationElement Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 35.

**Table 35 - ExtendibleEnumerationElement Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| Extendible Enumeration Element | Set Key | 16 | | See Table 10 | Req | Defines the Extendible Enumeration Element set |
| Length | BER Length | var | | | Req | Set length |
| All items from the Definition Object class defined in 6.2 above | | | | | | |
| ElementOf | TypeDefinitionExtendibleEnumerationWeakReferenceSet | 8+16n | 00.2a | 06.0E.2B.34 01.01.01.0d 06.01.01.07 21.00.00.00 | Opt | References the ExtendibleEnumerations in which this element is known to be used |

## 8.17 TypeDefinitionRename class

## 8.17.1 TypeDefinitionRename Class Description

The TypeDefinitionRename class defines a property type that has the same structure and representation as its underlying type but has a different meaning.

All TypeDefinitionRename objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionRename objects.

## 8.17.2 Attributes of the TypeDefinitionRename class

The TypeDefinitionRename class is a sub-class of the TypeDefinition class as illustrated in Figure 26.

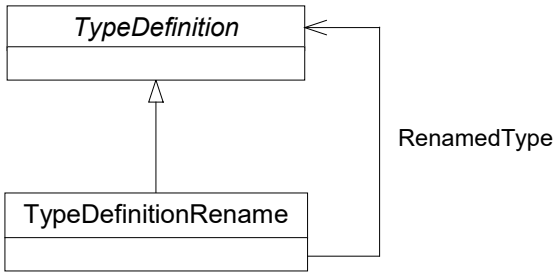


**Figure 26 – TypeDefinitionRename**

TypeDefinitionRename objects shall contain the required attributes of their parent class and shall contain the following attributes:

- RenamedType; specifies the underlying type.

objects may contain the optional attributes of their parent class.

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the value of this attribute shall be set according to Table 36.

**Table 36 - TypeDefinitionRename Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| RenamedType | value of | Base Type | 4.4.13 |

## 8.17.3 KLV Encoding of the TypeDefinitionRename class

When encoded as KLV Packets, TypeDefinitionRename Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 37.

**Table 37 - TypeDefinitionRename Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinitionRename | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionRename set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| RenamedType | TypeDefinitionWeakReference | 16 | 00.1E | 06.0E.2B.34 01.01.01.02 06.01.01.07 12.00.00.00 | Req | Specifies the underlying type |

## 8.18 TypeDefinitionIndirect class

## 8.18.1 TypeDefinitionIndirect Class Description

The TypeDefinitionIndirect class defines a property type that has a value whose type is specified in each instance.

All TypeDefinitionIndirect objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionIndirect objects.

### 8.18.2 Attributes of the TypeDefinitionIndirect class

The TypeDefinitionIndirect class is a sub-class of the TypeDefinition class as illustrated in Figure 27.

The TypeDefinitionIndirect class does not define any additional properties.

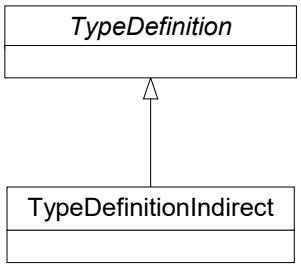


**Figure 27 – TypeDefinitionIndirect**

TypeDefinitionIndirect objects shall contain the required attributes of their parent class.

TypeDefinitionIndirect objects may contain the optional attributes of their parent class.

### 8.18.3 KLV Encoding of the TypeDefinitionIndirect class

When encoded as KLV Packets, TypeDefinitionIndirect Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 38.

**Table 38 - TypeDefinitionIndirect Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinition Indirect | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionIndirect set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |

## 8.19 TypeDefinitionOpaque class

### 8.19.1 TypeDefinitionOpaque class Description

The TypeDefinitionOpaque Class defines an unknown property type. TypeDefinitionOpaque indicates that the MXF encoder does not know the actual type of the data.

MXF encoders that do not know the actual type of a data value shall create a TypeDefinitionOpaque object to provide an identification of the origin of the data to downstream MXF decoders.

MXF decoders may interpret the data value if they recognize the Identification of a TypeDefinitionOpaque object. MXF decoders may attempt to interpret the data value by looking up the Identification of a TypeDefinitionOpaque object in an external system or in a SMPTE Register. MXF decoders that do not recognize the Identification of a TypeDefinitionOpaque object and cannot discover the Identification shall treat the data value as Dark Metadata i.e. they shall make no assumptions about the actual data type.

All TypeDefinitionOpaque objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionOpaque objects.

### 8.19.2 Attributes of the TypeDefinitionOpaque class

The TypeDefinitionOpaque Class is a concrete Subclass of the TypeDefinitionIndirect Class as illustrated in Figure 28.

The TypeDefinitionOpaque class does not define any additional properties.

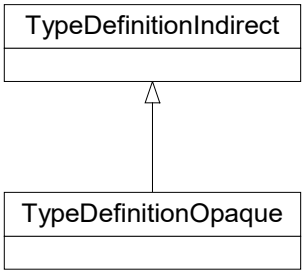


**Figure 28 – TypeDefinitionOpaque**

TypeDefinitionOpaque objects shall contain the required attributes of their parent class.

TypeDefinitionOpaque objects may contain the optional attributes of their parent class.

### 8.19.3 KLV Encoding of the TypeDefinitionOpaque class

When encoded as KLV Packets, TypeDefinitionOpaque Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 39.

**Table 39 - TypeDefinitionOpaque Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinition Opaque | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionOpaque set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |

## 8.20 TypeDefinitionStrongObjectReference class

### 8.20.1 TypeDefinitionStrongObjectReference Class Description

The TypeDefinitionStrongObjectReference class defines a property type that defines an object relationship where the target of the strong reference is owned by the object with the property with the TypeDefinitionStrongObjectReference type. An object can be the target of only one strong reference.

All TypeDefinitionStrongObjectReference objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionStrongObjectReference objects.

### 8.20.2 Attributes of the TypeDefinitionStrongObjectReference class

The TypeDefinitionStrongObjectReference class is a sub-class of the TypeDefinition class as illustrated in Figure 29.

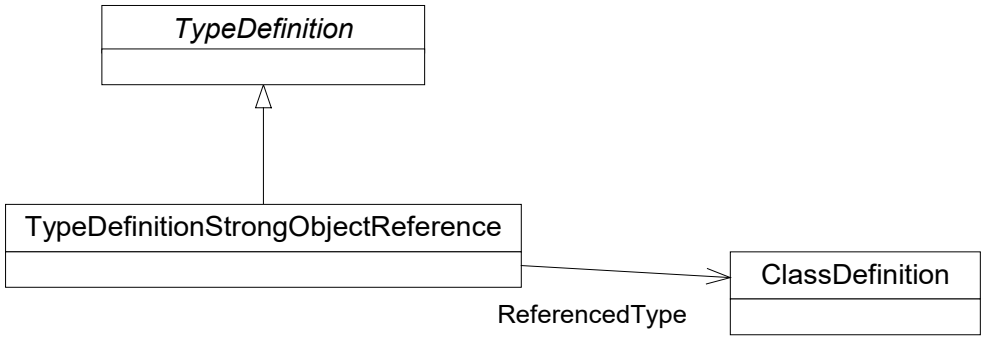


**Figure 29 – TypeDefinitionStrongObjectReference**

TypeDefinitionStrongObjectReference objects shall contain the required attributes of their parent class and shall contain the following attributes:

- ReferencedType; specifies the class that the referenced object shall belong to.

TypeDefinitionStrongObjectReference objects may contain the optional attributes of their parent class.

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the value of this attribute shall be set according to Table 40.

**Table 40 - TypeDefinitionStrongObjectReference Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| ReferencedType | value of | Base Type | 4.4.13 |

## 8.20.3 KLV Encoding of the TypeDefinitionStrongObjectReference class

When encoded as KLV Packets, TypeDefinitionStrongObjectReference Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 41.

**Table 41 - TypeDefinitionStrongObjectReference Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinition StrongObject Reference | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionStrongObjectReference set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| ReferencedType | ClassDefinitionWeakReference | 16 | 00.11 | 06.0E.2B.34 01.01.01.02 06.01.01.07 09.00.00.00 | Req | Specifies the class that the referenced object shall belong to (the referenced object may also belong to a subclass of the referenced class) |

## 8.21 TypeDefinitionWeakObjectReference class

## 8.21.1 TypeDefinitionWeakObjectReference Class Description

The TypeDefinitionWeakObjectReference class defines a property type that defines an object relationship where the target of the weak reference is referenced by the object with the property with the TypeDefinitionWeakObjectReference type. Only objects that define a unique identification can be the targets of weak object references. An object can be the target of one or more than one weak references.

All TypeDefinitionWeakObjectReference objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionWeakObjectReference objects.

### 8.21.2 Attributes of the TypeDefinitionWeakObjectReference class

The TypeDefinitionWeakObjectReference class is a sub-class of the TypeDefinition class as illustrated in Figure 30.

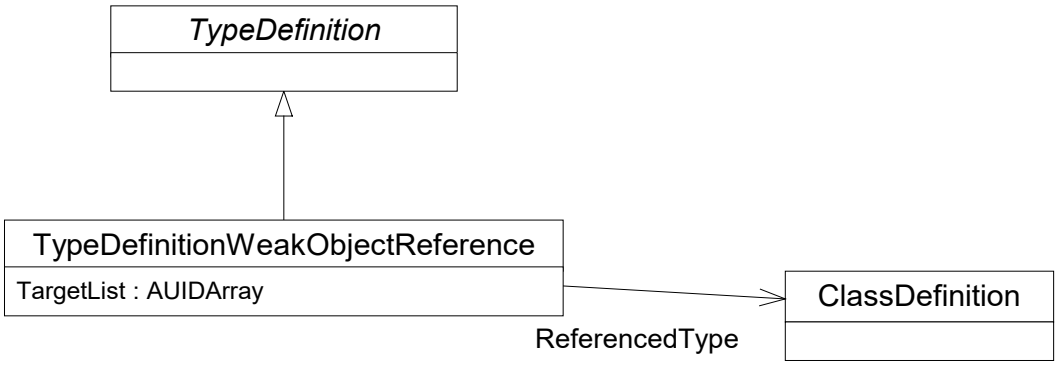


**Figure 30 – TypeDefinitionWeakObjectReference**

TypeDefinitionWeakObjectReference objects shall contain the required attributes of their parent class and shall contain the following attributes:

- WeakReferencedType; specifies the class that the referenced object shall belong to.
- TargetSet; specifies the AUIDs that specify the properties from the root of the file to the property that has the StrongReferenceSet containing the uniquely identified objects that may be the target of the weak reference.

TypeDefinitionWeakObjectReference objects may contain the optional attributes of their parent class.

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the value of this attribute shall be set according to Table 42.

**Table 42 - TypeDefinitionWeakObjectReference Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| WeakReferencedType | value of | Base Type | 4.4.13 |
| TargetSet | list of values of | Facet Symbol | 4.4.16 |

### 8.21.3 KLV Encoding of the TypeDefinitionWeakObjectReference class

When encoded as KLV Packets, TypeDefinitionWeakObjectReference Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 43.

**Table 43 - TypeDefinitionWeakObjectReference Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinition WeakObject Reference | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionWeakObjectReference set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| WeakReferenced Type | ClassDefinitionWeakReference | 16 | 00.12 | 06.0E.2B.34 01.01.01.02 06.01.01.07 0A.00.00.00 | Req | Specifies the class that the referenced object shall belong to (the referenced object may also belong to a subclass of the referenced class) |
| TargetSet | AUIDArray | 8+16n | 00.13 | 06.0E.2B.34 01.01.01.02 03.01.02.03 0B.00.00.00 | Req | Specifies the AUIDs that specify the properties from the root of the file to the property that has the StrongReferenceSet containing the uniquely identified objects that may be the target of the weak reference. The first AUID in the array identifies the object in the file's root storage. The last AUID in the array identifies the property containing the set of uniquely identified objects. The AUIDs between the first and the last identify properties that have a TypeDefinitionStrongObjectReference and define the containing hierarchy from the object in the root storage to the object containing the StrongReferenceSet. |

## 8.22 TypeDefinitionFixedArray class

### 8.22.1 TypeDefinitionFixedArray Class Description

The TypeDefinitionFixedArray class defines a property type that has a fixed number of values of the underlying type. The order of the values is meaningful.

All TypeDefinitionFixedArray objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionFixedArray objects.

### 8.22.2 Attributes of the TypeDefinitionFixedArray class

The TypeDefinitionFixedArray class is a sub-class of the TypeDefinition class as illustrated in Figure 31.

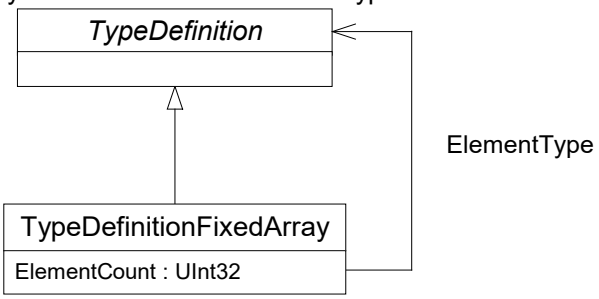


**Figure 31 – TypeDefinitionFixedArray**

TypeDefinitionFixedArray objects shall contain the required attributes of their parent class and shall contain the following attributes:

- FixedArrayElementType; specifies the TypeDefinition that defines the type of each element of the array
- ElementCount; specifies the number of elements in the array.

TypeDefinitionFixedArray objects may contain the optional attributes of their parent class.

When a TypeDefinitionFixedArray Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the values of these attributes shall be set according to Table 44.

**Table 44 - TypeDefinitionFixedArray Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| FixedArrayElementType | value of | Base Type | 4.4.13 |
| ElementCount | value of | Type Size | 4.4.12 |

## 8.22.3 KLV Encoding of the TypeDefinitionFixedArray class

When encoded as KLV Packets, TypeDefinitionFixedArray Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths.

**Table 45 - TypeDefinitionFixedArray Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinitionFixedArray | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionFixedArray set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| FixedArrayElementType | TypeDefinitionWeakReference | 16 | 00.17 | 06.0E.2B.34 01.01.01.02 06.01.01.07 0C.00.00.00 | Req | Specifies the TypeDefinition that defines the type of each element of the array |
| ElementCount | UInt32 | 4 | 00.18 | 06.0E.2B.34 01.01.01.02 03.01.02.03 03.00.00.00 | Req | Specifies the number of elements in the array |

# 8.23 TypeDefinitionVariableArray class

## 8.23.1 TypeDefinitionVariableArray Class Description

The TypeDefinitionVariableArray class defines a property type that has a varying number of values of the underlying type. The order of the values is meaningful.

All TypeDefinitionVariableArray objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionVariableArray objects.

## 8.23.2 Attributes of the TypeDefinitionVariableArray class

The TypeDefinitionVariableArray class is a sub-class of the TypeDefinition class as illustrated in Figure 32.

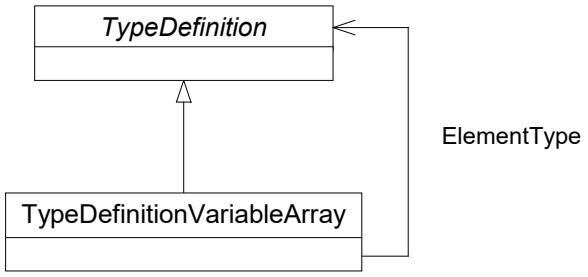


**Figure 32 – TypeDefinitionVariableArray**

TypeDefinitionVariableArray objects shall contain the required attributes of their parent class and shall contain the following attributes:

- VariableArrayElementType; specifies the type of the element of the array.

TypeDefinitionVariableArray objects may contain the optional attributes of their parent class.

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the value of this attribute shall be set as follows

**Table 46 - TypeDefinitionVariableArray Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| VariableArrayElement Type | value of | Base Type | 4.4.13 |

### 8.23.3 KLV Encoding of the TypeDefinitionVariableArray class

When encoded as KLV Packets, TypeDefinitionVariableArray Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths.

**Table 47 - TypeDefinitionVariableArray Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinitionVariableArray | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionVariableArray set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| VariableArray ElementType | TypeDefinitionWeakReference | 16 | 00.19 | 06.0E.2B.34 01.01.01.02 06.01.01.07 0D.00.00.00 | Req | Specifies the type of the element of the array |

## 8.24 TypeDefinitionSet class

### 8.24.1 TypeDefinitionSet Class Description

The TypeDefinitionSet class defines a property type that has a collection of object references to uniquely identified objects. The order of the objects has no meaning.

All TypeDefinitionSet objects in an MXF file shall be strongly referenced from within the ExtensionScheme object.

An MXF file may contain any number of TypeDefinitionSet objects.

### 8.24.2 Attributes of the TypeDefinitionSet class

The TypeDefinitionSet class is a sub-class of the TypeDefinition class as illustrated in Figure 33.

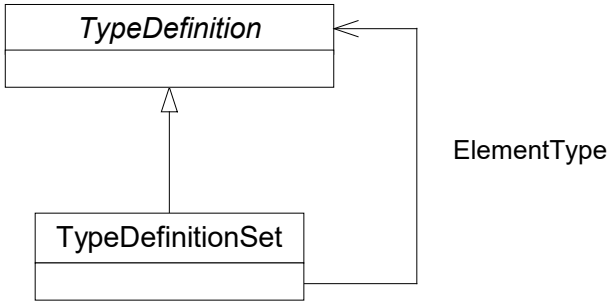


**Figure 33 – TypeDefinitionSet**

TypeDefinitionSet objects shall contain the required attributes of their parent class and shall contain the following attributes:

- ElementType; specifies the TypeDefinition that identifies the kind of object reference.

TypeDefinitionSet objects may contain the optional attributes of their parent class.

When a MetaDefinition Object is constructed from an entry in the Types Registry (SMPTE ST 2003), the value of this attribute shall be set as follows

**Table 48 - TypeDefinitionSet Attribute Values**

| Attribute | Mapping | ST 2003 Field | ST 2003 Section |
|---|---|---|---|
| ElementType | value of | Base Type | 4.4.13 |

### 8.24.3 KLV Encoding of the TypeDefinitionSet class

When encoded as KLV Packets, TypeDefinitionSet Objects shall be encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths.

**Table 49 - TypeDefinitionSet Encoding**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinitionSet | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionSet set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| SetElementType | TypeDefinitionWeakReference | 16 | 00.1A | 06.0E.2B.34 01.01.01.02 06.01.01.07 0E.00.00.00 | Req | Specifies the TypeDefinition that identifies the kind of object reference. This TypeDefinition shall belong to either the TypeDefinitionStrongObjectReference or TypeDefinitionWeakObjectReference |

# 9 Examples of Describing MXF Extensions using the KLV-Encoded Extension Syntax (Informative)

## 9.1 Adding a Single Optional Property

Additional optional properties can be defined for MXF according to the provisions of SMPTE ST 377-1 section 9.3.

For example, SMPTE ST 422 section 7.2 defines an additional optional property for the MXF Generic Descriptor as shown in Table 50.

**Table 50 - Copy of Table 4 from ST422**

| Element Name | Type | Len | Local Tag | UL Designator | Req ? | Element Description | Default |
|---|---|---|---|---|---|---|---|
| All items from the Generic Descriptor defined in SMPTE ST 377-1 | | | | | | | |
| Sub Descriptors | StrongRefArray (Sub Descriptors) | 8+16n | Dyn | 06.01.01.04.06.10.00.00 | Opt | Ordered array of strong references to sub descriptor sets | |

Table 4 (of ST422) – Additional Optional Property for the MXF Generic Descriptor

The KXS Meta Definition describing this is shown in Table 51.

**Table 51 - Example Metadefinition for ST422 subdescriptors**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Sub Descriptors property | 06.0E.2B.34 01.01.01.09 06.01.01.04 06.10.00.00 |
| MetaDefinition Name | UTF16String | The property name | Sub Descriptors |
| MetaDefinition Description | UTF16String | The property description | Ordered array of strong references to sub descriptor sets |
| PropertyType | TypeDefinitionWeakReference | UL of the StrongRefArray (Sub Descriptors) type | 06.0E.2B.34 01.04.01.01 05.06.0E.00 00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Generic Descriptor class | 06.0E.2B.34 02.7F.01.01 0D.01.01.01 01.01.24.00 |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

NOTE: from the perspective of a SMPTE ST 377-1 decoder, the MetaDefinition will be ignored, and the Header Metadata of the resultant file remains fully compliant with SMPTE ST 377-1.

## 9.2 Reusing an Existing Property

Properties defined for MXF according to the provisions of SMPTE ST 377-1 are used only within the classes that define them or their subclasses.

It is sometimes desired to add an additional use of a previously defined property to another class within an MXF file. For example, a particular application might wish to add an Active Format Description property to a DMS-1 Picture Format set.

The original definition of Active Format Description is as a Property of the MXF Generic Picture Descriptor in SMPTE ST 377-1. The KXS Meta Definition that describes this is shown in Table 52.

**Table 52 - Example metadefinition of Active Format Description**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Active Format Descriptor property | 06.0E.2B.34<br>01.01.01.05<br>04.01.03.02<br>09.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | Active Format Descriptor |
| MetaDefinition Description | UTF16String | The property description | Specifies the intended framing of the content within the displayed image (4:3 in 16:9 etc.) |
| PropertyType | TypeDefinitionWeakReference | UL of the UInt8 type | 06.0E.2B.34<br>01.04.01.01<br>01.01.01.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Generic Picture Descriptor class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.24.00 |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

The new use of Active Format Description as a Property of the DMS-1 Picture Format set is defined using a KXS PropertyAliasDefinition as shown in Table 53.

**Table 53 - Example of KXS PropertyAlias in DMS-1 Picture Set**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyAlias Definition** | **Set Key** | **Defines the PropertyAliasDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | Globally unique identification of the MetaDefinition<br>Note: the UL used here was generated for the purposes of this example and is not valid outside its context. | 06.0E.2B.34<br>01.01.01.01<br>0F.01.01.00<br>00.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | Active Format Descriptor |
| MetaDefinition Description | UTF16String | The property description | Specifies the intended framing of the content within the displayed image (4:3 in 16:9 etc.) |
| PropertyType | TypeDefinitionWeakReference | UL of the UInt8 type | 06.0E.2B.34<br>01.04.01.01<br>01.01.01.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | Specifies the class in which this property may be present | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.04.01<br>01.1D.01.00<br>(DMS-1 Picture Format set) |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |
| OriginalProperty | PropertyDefinitionWeakReference | References the original definition of the reused Property | 06.0E.2B.34<br>01.01.01.05<br>04.01.03.02<br>09.00.00.00 |

Note: from the perspective of a SMPTE ST 377-1 decoder, the MetaDefinitions will be ignored, and the Header Metadata of the resultant file remains fully compliant with SMPTE ST 377-1. Also, the new property instance will be ignored.

## 9.3 Describing a New Class

### 9.3.1 Overview

When it is desired to add extend a class by adding a new mandatory property, this can be achieved by defining a new subclass and assigning a new UL as the class ID. It may also be desired to provide an explicit subclass as a container for optional properties. Another motivation for defining new subclasses is to explicitly specify different class IDs to indicate different use cases.

### 9.3.2 Example Scenario

The example here shows the Definition Objects that are required to describe a PackageMarkerObject as illustrated in Figure 34. PackageMarkerObject is a good example, since that class is fully described by SMPTE ST 377-1:2011 and is an extension of SMPTE ST 377M:2004.

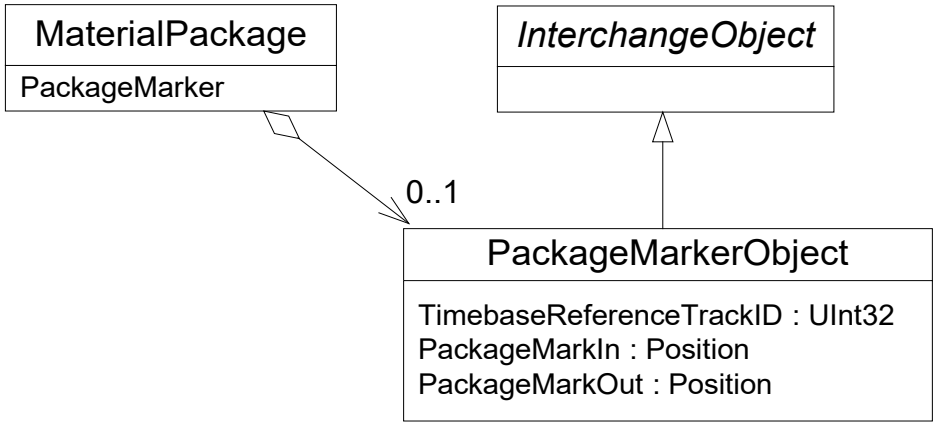


**Figure 34 – Package Marker Object**

### 9.3.3 Extensions to Be Described

PackageMarkerObject is specified by SMPTE ST 377-1 Annex B.34 and defines the mandatory TimebaseReferenceTrackID property and two optional properties PackageMarkInPosition and PackageMarkOutPosition.

A PackageMarkerObject may be present in any MaterialPackage in an MXF file. It is strongly referenced by the optional PackageMarker property of the MaterialPackage class. This property is an extension of MXF specified by SMPTE ST 377-1 Annex E.1.

Since this is an optional property, it is possible for it to be present in a SMPTE ST 377:2004 file, and thus for these files to also include PackageMarkerObjects.

### 9.3.4 Definition Objects

The KXS Meta Definition that describes the PackageMarker property is shown in Table 54.

**Table 54 - Example Metadefinition of PackageMarker**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Package Marker property | 06.0E.2B.34<br>01.01.01.0C<br>06.01.01.04<br>02.0F.00.00 |
| MetaDefinition Name | UTF16String | The property name | PackageMarker |
| MetaDefinition Description | UTF16String | The property description | A strong reference to a Package Marker Object. |
| PropertyType | TypeDefinitionWeakReference | UL of the StrongRef to (PackageMarkerObject) type. | 06.0E.2B.34<br>01.04.01.01<br>05.02.2E.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Material Package class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.36.00 |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

The KXS Meta Definitions that describe the PackageMarkerObject class are shown in Table 55 through Table 59.

**Table 55 - Example Metadefinition of Reference to PackageMarkerObject**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **TypeDefinition StrongObjectReference** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the StrongRef to (PackageMarkerObject) type. | 06.0E.2B.34<br>01.04.01.01<br>05.02.2E.00<br>00.00.00.00 |
| MetaDefinition Name | UTF16String | The type name | StrongRefPackageMarkerObject |
| ReferencedType | ClassDefinitionWeakReference | UL of the Package Marker Object class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.60.00 |

**Table 56 - Example Metadefinition of PackageMarkerObject**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **ClassDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Package Marker Object class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.60.00 |
| MetaDefinition Name | UTF16String | The class name | PackageMarkerObject |
| MetaDefinition Description | UTF16String | The class description | Defines the Package Marker Object |
| ParentClass | ClassDefinitionWeakReference | UL of the Interchange Object class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.01.00 |
| IsConcrete | Boolean | The class is concrete | True |

**Table 57 - Example Metadefinition of TimebaseReferenceTrackID**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Timebase Reference Track ID property | 06.0E.2B.34 01.01.01.0C 06.01.01.03 0E.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | TimebaseReferenceTrackID |
| MetaDefinition Description | UTF16String | The property description | Specifies the value of the Track ID of the target Track in the Material Package that provides the Edit Rate |
| PropertyType | TypeDefinitionWeakReference | UL of the UInt32 type | 06.0E.2B.34 01.04.01.01 01.01.03.00 00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Package Marker Object class | 06.0E.2B.34 02.7F.01.01 0D.01.01.01 01.01.60.00 |
| IsOptional | Boolean | The property is optional | False |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

**Table 58 - Example Metadefinition of PackageMarkInPosition**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Package Mark In Position property | 06.0E.2B.34 01.01.01.0A 07.02.01.03 01.0E.00.00 |
| MetaDefinition Name | UTF16String | The property name | PackageMarkInPosition |
| MetaDefinition Description | UTF16String | The property description | Start of the optional sub-section on the Material Package Timebase Reference Track timeline. |
| PropertyType | TypeDefinitionWeakReference | UL of the PositionType type | 06.0E.2B.34 01.04.01.01 01.01.20.01 00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Package Marker Object class | 06.0E.2B.34 02.7F.01.01 0D.01.01.01 01.01.60.00 |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

**Table 59- Example Metadefinition of PackageMarkOutPosition**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Package Mark Out Position property | 06.0E.2B.34<br>01.01.01.0A<br>07.02.01.03<br>02.04.00.00 |
| MetaDefinition Name | UTF16String | The property name | PackageMarkOutPosition |
| MetaDefinition Description | UTF16String | The property description | Stop of the optional sub-section on the Material Package Timebase Reference Track timeline. |
| PropertyType | TypeDefinitionWeakReference | UL of the PositionType type | 06.0E.2B.34<br>01.04.01.01<br>01.01.20.01<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Package Marker Object class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.60.00 |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

Note: from the perspective of a SMPTE ST 377-1 decoder, the MetaDefinitions will be ignored, and the Header Metadata of the resultant file remains fully compliant with SMPTE ST 377-1. Also, the new class instance will be ignored.

## 9.4 Example of an Extension Scheme object

Sets of MetaDefinitions are grouped into an instance of the ExtensionScheme class. This following example shows an ExtensionScheme object to contain the Extensions to SMPTE ST 377:2004 defined by SMPTE ST 377-1, such as those in section 9.3 above. An example is shown in Table 60.

**Table 60 - Example Metadefinition of Extension Scheme**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| ExtensionScheme | **Set Key** | **Defines the** ExtensionScheme **set** | **See Table 10** |
| ExtensionSchemeID | AUID | Globally unique identification of the ExtensionScheme.<br>Note: the UL used here was generated for the purposes of this example and is not valid outside its context. | 06.0E.2B.34<br>01.01.01.01<br>0F.02.01.00<br>00.00.00.00 |
| SymbolSpaceURI | URI (UTF16String) | Namespace URI for the Extension Scheme | http://example.org/kxs-examples/ |
| PreferredPrefix | UTF16String | Preferred namespace tag when SMPTE ST 2001-1Reg-XML encoding is used | kxs-example |
| Extension Description | UTF16String | Description of the Extension Scheme | Extensions to SMPTE ST 377:2004 defined by SMPTE ST 377-1 |
| MetaDefinitions | MetaDefinitionStrongReferenceSet | References all the MetaDefinitions in this Extension Scheme when they are contained in the MXF file | InstanceUIDs of all the MetaDefinitions in section 9.3 |

Note: from the perspective of a SMPTE ST 377-1 decoder, the MetaDefinitions will be ignored, and the Header Metadata of the resultant file remains fully compliant with SMPTE ST 377-1.

## 9.5 Describing MXF Application Metadata Plug-Ins

### 9.5.1 Example Scenario

Application Metadata Plug-Ins are defined according to the provisions of SMPTE ST 377-1.

The example scenario shows what Definition Objects are required to describe the Application Metadata Plug-In mechanism defined by SMPTE ST 377-1, and then provides two simple Application Metadata Plug-Ins that are used simultaneously to extend an underlying MXF File.

### 9.5.2 Extensions to Be Described

SMPTE ST 377-1 Application Metadata Plug-Ins extend the data model of SMPTE ST 377:2004 by defining optional properties of the abstract superclass InterchangeObject, and by defining new subclasses of InterchangeObject, as illustrated in Figure 35.

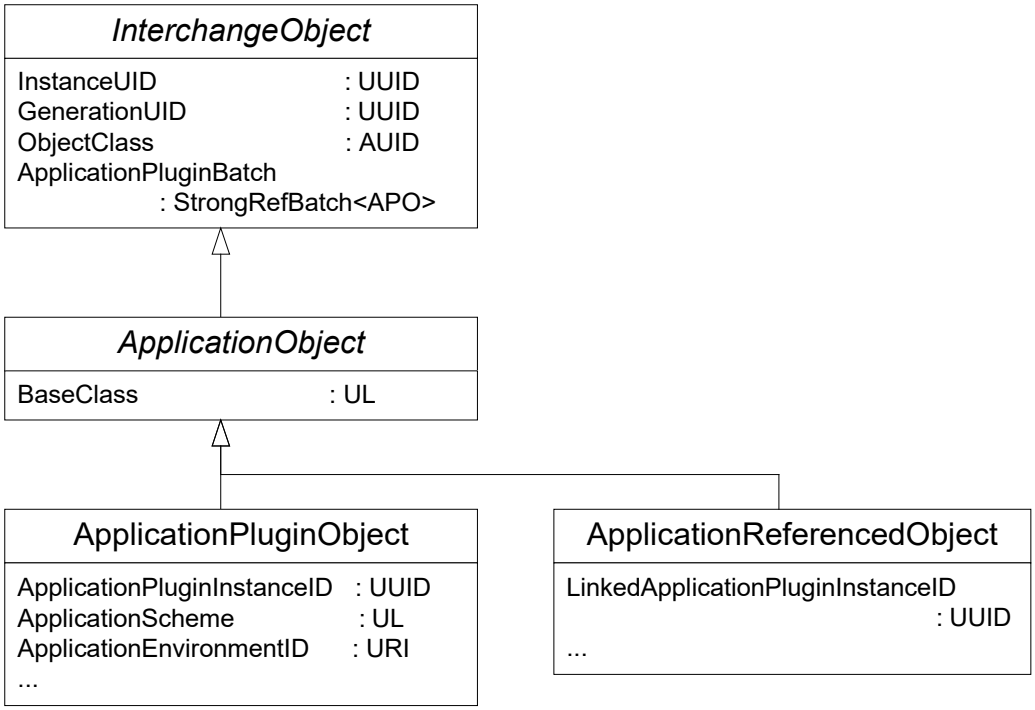


**Figure 35 – Application-Specific Metadata Plug-In Objects**

### 9.5.3 Definition Objects

The KXS MetaDefinitions to describe the extensions to InterchangeObject are described in Table 60 through Table 72.

**Table 61 - Example Metadefinition of ObjectClass**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the ObjectClass property | 06.0E.2B.34 01.01.01.02 06.01.01.04 01.01.00.00 |
| MetaDefinition Name | UTF16String | The property name | ObjectClass |
| MetaDefinition Description | UTF16String | The property description | Class identifier of this Object. |
| PropertyType | TypeDefinitionWeakReference | UL of MetaRefClassDefinition | 06.0E.2B.34 01.04.01.01 05.01.01.00 00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the InterchangeObject class | 06.0E.2B.34 02.7F.01.01 0D.01.01.01 01.01.01.00 |
| IsOptional | Boolean | The property is optional | true |
| IsUniqueIdentifier | Boolean | The property is a unique identifier | false |

**Table 62 - Example Metadefinition of ApplicationPluginBatch**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the ApplicationPluginBatch property | 06.0E.2B.34 01.01.01.0C 06.01.01.04 02.0E.00.00 |
| MetaDefinition Name | UTF16String | The property name | ApplicationPluginBatch |
| MetaDefinition Description | UTF16String | The property description | A batch of strong references to ApplicationPluginObject. |
| PropertyType | TypeDefinitionWeakReference | UL of the StrongRefBatch <ApplicationPluginObject> type. | 06.0E.2B.34 01.04.01.01 05.05.11.00 00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the InterchangeObject class | 06.0E.2B.34 02.7F.01.01 0D.01.01.01 01.01.01.00 |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

**Table 63 - Example Metadefinition of References to ApplicationPluginObjects**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **TypeDefinitionSet** | **Set Key** | **Defines the TypeDefinitionSet set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the StrongRefBatch <ApplicationPluginObject> type. | 06.0E.2B.34<br>01.04.01.01<br>05.05.11.00<br>00.00.00.00 |
| MetaDefinition Name | UTF16String | The type name | StrongRefBatch ApplicationPluginObject |
| SetElementType | TypeDefinitionWeakReference | UL of the StrongRef ApplicationPluginObject type definition | 06.0E.2B.34<br>01.04.01.01<br>05.02.2F.00<br>00.00.00.00 |

**Table 64 - Example Metadefinition of Reference to ApplicationPluginObject**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **TypeDefinition Strong ObjectReference** | **Set Key** | **Defines the particular TypeDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the StrongRef <ApplicationPluginObject> type. | 06.0E.2B.34<br>01.04.01.01<br>05.02.2F.00<br>00.00.00.00 |
| MetaDefinition Name | UTF16String | The type name | StrongRef ApplicationPluginObject |
| ReferencedType | ClassDefinitionWeakReference | UL of the ApplicationPluginObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.61.00 |

The KXS Meta Definitions that describe the ApplicationObject class are as follows:

**Table 65 - Example Metadefinition of ApplicationObject**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **ClassDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the ApplicationObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.66.00 |
| MetaDefinition Name | UTF16String | The class name | ApplicationObject |
| MetaDefinition Description | UTF16String | The class description | Defines the Abstract Superclass of the Application Plug-in Objects and Application Referenced Objects |
| ParentClass | ClassDefinitionWeakReference | UL of the Interchange Object class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.01.00 |
| IsConcrete | Boolean | The class is concrete | False |

**Table 66 - Example Metadefinition of BaseClass**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the BaseClass property | 06.0E.2B.34<br>01.01.01.0C<br>06.01.01.04<br>01.0B.00.00 |
| MetaDefinition Name | UTF16String | The property name | BaseClass |
| MetaDefinition Description | UTF16String | The property description | Class Identifier of the immediate Superclass defined in an MXF specification that this Object extends. |
| PropertyType | TypeDefinitionWeakReference | UL of the AUID type | 06.0E.2B.34<br>01.04.01.01<br>01.03.01.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the ApplicationObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.66.00 |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

The KXS Meta Definitions that describe the ApplicationPluginObject class are as follows:

**Table 67 - Example Metadefinition of ApplicationPluginObject**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **ClassDefinition** | **Set Key** | **Defines the ClassDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the ApplicationPluginObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.61.00 |
| MetaDefinition Name | UTF16String | The class name | ApplicationPluginObject |
| MetaDefinition Description | UTF16String | The class description | Defines the Application Plug-in Object Set |
| ParentClass | ClassDefinitionWeakReference | UL of the ApplicationObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.66.00 |
| IsConcrete | Boolean | The class is concrete | True |

**Table 68 - Example Metadefinition of ApplicationPluginInstanceID**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Application Plugin Instance ID property | 06.0E.2B.34<br>01.01.01.0C<br>05.20.07.01<br>0D.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | ApplicationPluginInstanceID |
| MetaDefinition Description | UTF16String | The property description | An immutable ID of this application metadata plug-in instance |
| PropertyType | TypeDefinitionWeakReference | UL of the UUID type | 06.0E.2B.34<br>01.04.01.01<br>01.03.03.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the ApplicationPluginObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.61.00 |
| IsOptional | Boolean | The property is optional | False |
| IsUniqueIdentifier | Boolean | The property is a unique identifier | True |

**Table 69 - Example Metadefinition of ApplicationScheme**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Application Scheme property | 06.0E.2B.34<br>01.01.01.0C<br>04.06.08.03<br>00.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | ApplicationScheme |
| MetaDefinition Description | UTF16String | The property description | Universal Label of the Application Metadata Scheme contained in this Plug-In Object |
| PropertyType | TypeDefinitionWeakReference | UL of the AUID type | 06.0E.2B.34<br>01.04.01.01<br>01.03.01.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the ApplicationPluginObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.61.00 |
| IsOptional | Boolean | The property is optional | False |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

**Table 70 - Example Metadefinition of ApplicationEnvironmentID**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Application Environment ID property | 06.0E.2B.34<br>01.01.01.0C<br>05.20.07.01<br>0F.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | ApplicationEnvironmentID |
| MetaDefinition Description | UTF16String | The property description | RFC 3986 Uniform Resource Identifier that identifies the application to which the information in this Plug-In Object applies |
| PropertyType | TypeDefinitionWeakReference | UL of the UTF16String type | 06.0E.2B.34<br>01.04.01.01<br>01.10.02.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the ApplicationPluginObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.61.00 |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

The KXS Meta Definitions that describe the ApplicationReferencedObject class are as follows:

**Table 71 - Example Metadefinition of ApplicationReferencedObject**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **ClassDefinition** | **Set Key** | **Defines the ClassDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the ApplicationReferencedObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.62.00 |
| MetaDefinition Name | UTF16String | The class name | ApplicationReferencedObject |
| MetaDefinition Description | UTF16String | The class description | Defines the Application Referenced Object Set |
| ParentClass | ClassDefinitionWeakReference | UL of the ApplicationObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.66.00 |
| IsConcrete | Boolean | The class is concrete | True |

**Table 72 - Example Metadefinition of LinkedApplicationPluginInstanceID**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Linked Application Plugin Instance ID property | 06.0E.2B.34 01.01.01.0C 05.20.07.01 0B.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | LinkedApplicationPlugin InstanceID |
| MetaDefinition Description | UTF16String | The property description | Global Weak Reference to the Application Plug-In Object that (directly or indirectly) strongly references this Application Metadata Referenced Object Set |
| PropertyType | TypeDefinitionWeakReference | UL of the Global WeakRef (Application Plug-In Object) type | 06.0E.2B.34 01.04.01.01 05.09.04.00 00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the ApplicationReferencedObject class | 06.0E.2B.34 02.7F.01.01 0D.01.01.01 01.01.62.00 |
| IsOptional | Boolean | The property is optional | False |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

If present in an MXF file, these MetaDefinitions would be contained within an ExtensionScheme object that identifies them as forming part of SMPTE ST 377-1.

Note: from the perspective of a SMPTE ST 377-1 decoder, the MetaDefinitions will be ignored, and the Header Metadata of the resultant file remains fully compliant with SMPTE ST 377-1.

### 9.5.4 First Plugin: Addition of TargetAFD to a Material Package

The first example shows the use of an Application Metadata Plugin to add a new Property called "TargetAFD" (which is a reuse of the property GenericPictureEssenceDescriptior::ActiveFormatDescription) to a MaterialPackage.

The instances of the Material Package and the ApplicationPluginObject together with their superclasses are illustrated in Figure 36.

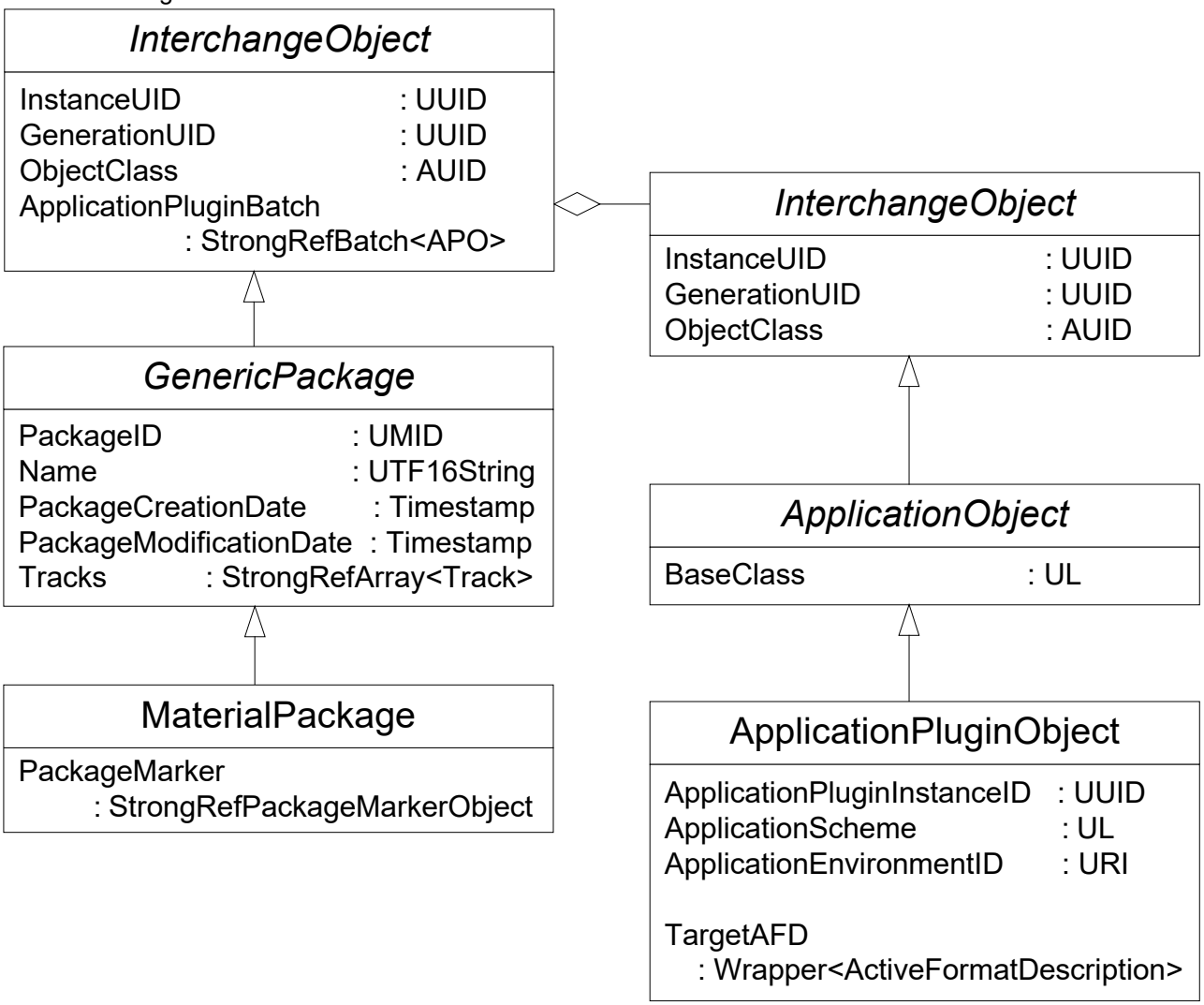


**Figure 36 – Addition of TargetAFD to a Material Package**

The KXS MetaDefinition to achieve this is shown in Table 73.

**Table 73 - Example Metadefinition of TargetAFD**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyAlias Definition** | **Set Key** | **Defines the PropertyAliasDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | Globally unique identification of the MetaDefinition<br>Note: the UL used here was generated for the purposes of this example and is not valid outside its context. | 06.0E.2B.34<br>01.01.01.01<br>0F.01.01.00<br>00.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | TargetAFD |
| MetaDefinition Description | UTF16String | The property description | Specifies the intended framing of the content within the displayed image (4:3 in 16:9 etc.) when the content is played out |
| PropertyType | TypeDefinitionWeakReference | UL of the UInt8 type | 06.0E.2B.34<br>01.04.01.01<br>01.01.01.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the ApplicationPluginObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.61.00 |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |
| OriginalProperty | PropertyDefinitionWeakReference | References the original definition of the reused Property | 06.0E.2B.34<br>01.01.01.05<br>04.01.03.02<br>09.00.00.00 |

### 9.5.5 Second Plugin: Addition of TargetDialNorm to a Material Package

The second example shows the use of an Application Metadata Plugin to add a new class called “TargetSoundParameters” containing a property called “ContentType” and a property called “TargetDialNorm” (which is a reuse of the property GenericSoundEssenceDescriptior::DialNorm) to a MaterialPackage.

The instances of the Material Package and the TargetSoundParameters together with their superclasses are illustrated in Figure 37.

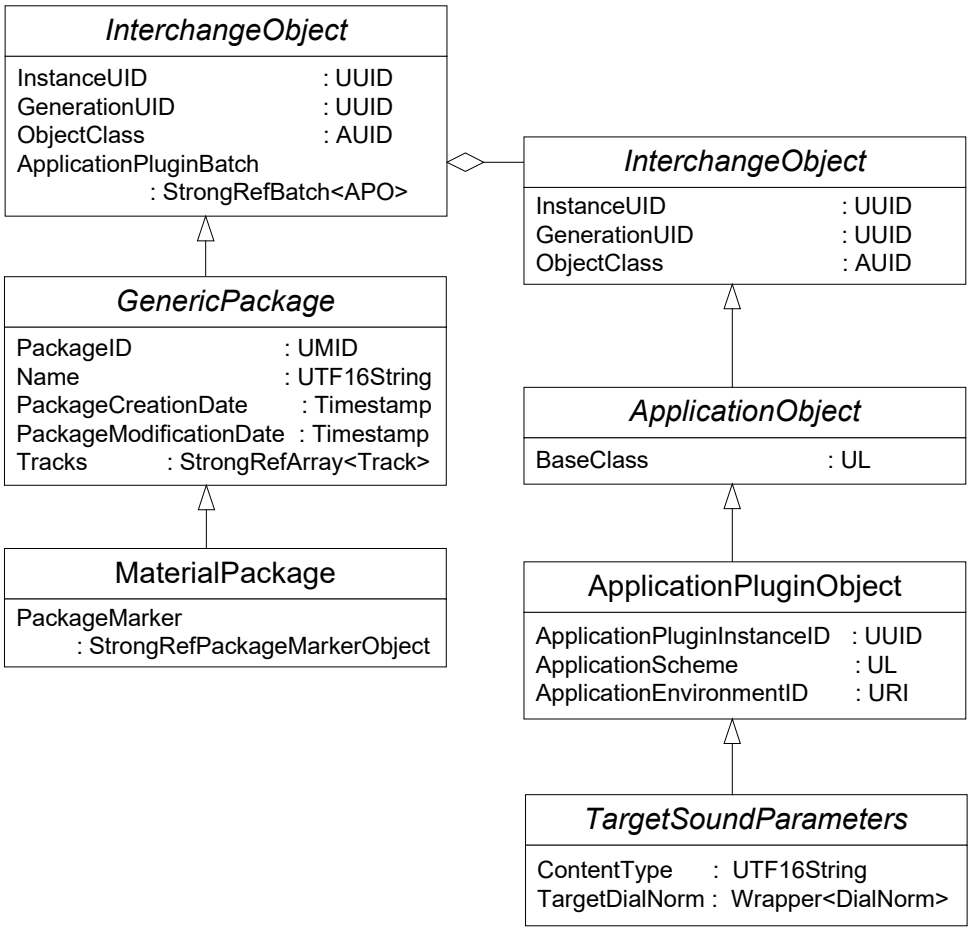


**Figure 37 – Addition of TargetDialNorm to a Material Package**

The KXS MetaDefinitions to achieve this is shown in Table 74, Table 75 and Table 76.

**Table 74 - Example Metadefinition of TargetSoundParameters**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **ClassDefinition** | **Set Key** | **Defines the ClassDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the TargetSoundParameters class<br>Note: the UL used here was generated for the purposes of this example and is not valid outside its context. | 06.0E.2B.34<br>02.7F.01.01<br>0F.01.01.00<br>00.00.00.00 |
| MetaDefinition Name | UTF16String | The class name | TargetSoundParameters |
| MetaDefinition Description | UTF16String | The class description | An application plug-in to define the user-specified content type and the target DialNorm when played out |
| ParentClass | ClassDefinitionWeak Reference | UL of the ApplicationPluginObject class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.61.00 |
| IsConcrete | Boolean | The class is concrete | True |

**Table 75 - Example Metadefinition of ContentType**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the ContentType property<br>Note: the UL used here was generated for the purposes of this example and is not valid outside its context. | 06.0E.2B.34<br>01.01.01.01<br>0F.01.01.00<br>00.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | ContentType |
| MetaDefinition Description | UTF16String | The property description | Specifies the type of the content in the associated package. This is a user-specified string |
| PropertyType | TypeDefinitionWeakReference | UL of the UTF16String type | 06.0E.2B.34<br>01.04.01.01<br>01.10.02.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the TargetSoundParameters class<br>Note: the UL used here was generated for the purposes of this example and is not valid outside its context. | 06.0E.2B.34<br>02.7F.01.01<br>0F.01.01.00<br>00.00.00.00 |
| IsOptional | Boolean | The property is optional | True |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |

**Table 76- Example Metadefinition of TargetDialNorm**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyAlias Definition** | **Set Key** | **Defines the PropertyAliasDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the TargetDialNorm property<br>Note: the UL used here was generated for the purposes of this example and is not valid outside its context. | 06.0E.2B.34<br>01.01.01.01<br>0F.01.02.00<br>00.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | TargetDialNorm |
| MetaDefinition Description | UTF16String | The property description | Specifies the target DialNorm when the content is played out |
| PropertyType | TypeDefinitionWeakReference | UL of the Int8 type | 06.0E.2B.34<br>01.04.01.01<br>01.01.05.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the TargetSoundParameters class<br>Note: the UL used here was generated for the purposes of this example and is not valid outside its context. | 06.0E.2B.34<br>02.7F.01.01<br>0F.01.01.00<br>00.00.00.00 |
| IsOptional | Boolean | The property is optional | False |
| IsUniqueIdentifier | Boolean | The property is not a unique identifier | False |
| OriginalProperty | PropertyDefinitionWeakReference | References the original definition of the reused Dial Norm Property | 06.0E.2B.34<br>01.01.01.05<br>04.02.07.01<br>00.00.00.00 |

## 9.6 Describing an MXF Descriptive Metadata Plugin

### 9.6.1 Examples

Descriptive Metadata Plug-Ins are defined according to the provisions of SMPTE ST 377-1.

Descriptive Metadata Plug-Ins are instances of Descriptive Metadata Schemes (DM Scheme) that make use of one or more Framework classes, each of which contain a number of Descriptive Set Mixin Objects.

Two examples are provided. The first example shows what Definition Objects are required to describe a simple DM Scheme consisting of a single Framework containing a single Descriptive Set.

This example can be readily (if long-windedly) expanded to describe a complex DM Scheme such as SMPTE ST-380 DMS-1.

The second example shows what Definition Objects are required to describe a user extension of the simple example DM Scheme. This example can be applied also to describe a user extension of DMS-1.

Note: from the perspective of a SMPTE ST 377-1 decoder, the MetaDefinitions will be ignored, and the Header Metadata of the resultant file remains fully compliant with SMPTE ST 377-1.

### 9.6.2 Extensions to Be Described

The DMS-1 Project class is specified by SMPTE ST 380:2004 Annex A.2 and defines the optional Project Name property.

A Project is strongly referenced by the optional Project Set property of the DMS-1 Clip Framework class (which it inherits from its abstract parent class Production-Clip Framework). The Clip Framework is specified by SMPTE ST 380:2004 Annex A.1.

A Clip Framework can be present in any DM Segment in an MXF file. It is strongly referenced by the optional DM Framework property of the DM Segment class.

### 9.6.3 Definition Objects

The KXS Meta Definitions that partially describe the Production-Clip Framework class are as follows:

**Table 77- Example Metadefinition of ClipFramework**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **ClassDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Production-Clip Framework class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.04.01<br>01.7F.02.00 |
| MetaDefinition Name | UTF16String | The class name | ProductionClipFramework |
| MetaDefinition Description | UTF16String | The class description | Defines the DMS-1 Production-Clip Framework |
| ParentClass | ClassDefinitionWeakReference | UL of the DMS-1 Framework class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.04.01<br>01.7F.01.00 |
| IsConcrete | Boolean | The class is concrete | False |

**Table 78 - Example Metadefinition of ProjectSet**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Project Set property | 06.0E.2B.34<br>01.01.01.05<br>06.01.01.04<br>02.40.21.00 |
| MetaDefinition Name | UTF16String | The property name | ProjectSet |
| MetaDefinition Description | UTF16String | The property description | Specifies strong reference to the Project set |
| PropertyType | TypeDefinitionWeakReference | UL of the StrongReference (DescriptiveObject) type | 06.0E.2B.34<br>01.04.01.01<br>05.02.22.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Production-Clip Framework class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.04.01<br>01.7F.02.00 |
| IsOptional | Boolean | The property is optional | True |

The KXS Meta Definitions that partially describe the DMS-1 Project class are as follows:

**Table 79 - Example Metadefinition of Project**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **ClassDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Project class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.04.01<br>01.20.02.00 |
| MetaDefinition Name | UTF16String | The class name | Project |
| MetaDefinition Description | UTF16String | The class description | Defines the DMS-1 Project set |
| ParentClass | ClassDefinitionWeakReference | UL of the DMS-1 Object class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.04.01<br>01.7F.10.00 |
| IsConcrete | Boolean | The class is concrete | True |

**Table 80 - Example Metadefinition of ProjectName**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | Globally unique identification of the MetaDefinition | 06.0E.2B.34 01.01.01.05 01.03.01.08 01.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | ProjectName |
| MetaDefinition Description | UTF16String | The property description | Specifies the name for a particular project or mission |
| PropertyType | TypeDefinitionWeakReference | UL of the UTF16String type | 06.0E.2B.34 01.04.01.01 01.10.02.00 00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Project class | 06.0E.2B.34 02.7F.01.01 0D.01.04.01 01.20.02.00 |
| IsOptional | Boolean | The property is optional | True |

### 9.6.4 Extensions to Be Described

The extended DMS-1 Project is specified by SMPTE ST 380:2004 Annex A.2, and defines the optional Project Name property and the optional user extension property.

An extended Project is strongly referenced by the optional Project Set property of the DMS-1 Clip Framework class. The Clip Framework is specified by SMPTE ST 380:2004 Annex A.1.

A Clip Framework can be present in any DM Segment in an MXF file. It is strongly referenced by the optional DM Framework property of the DM Segment class.

### 9.6.5 Definition Objects

The KXS Meta Definitions that describe the extended DMS-1 Clip Framework are identical to those describing the standard Clip Framework (see the previous example).

The KXS Meta Definitions that describe the extended DMS-1 Project class are identical to those in the standard Project (see the previous example), with an additional KXS PropertyDefinition for a user extension property of the Project class as shown in Table 81.

**Table 81 - Example Metadefinition of UserDefinedProjectVersion**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the User Defined Project Version property. Note: the UL used here was generated for the purposes of this example and is not valid outside its context. | 06.0E.2B.34 01.01.01.01 0F.01.02.00 00.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | UserDefinedProjectVersion |
| MetaDefinition Description | UTF16String | The property description | Specifies the user defined version number for a particular project |
| PropertyType | TypeDefinitionWeakReference | UL of the UInt32 type | 06.0E.2B.34 01.04.01.01 01.01.03.00 00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Project class | 06.0E.2B.34 02.7F.01.01 0D.01.04.01 01.20.02.00 |
| IsOptional | Boolean | The property is optional | True |

## 9.7 Describing an MXF Structural Metadata Class

### 9.7.1 Extensions to Be Described

MXF Specifications other than SMPTE ST 377-1 can also define new structural metadata classes. One such class is the WaveAudioPhysicalDescriptor defined by SMPTE ST 382 Annex A.3.

The example shows what MetaDefinition Objects are required to describe the WaveAudioPhysicalDescriptor class and 2 of its properties: BextCodingHistory of type UTF-16 String and QltyFileSecurityReport of type UInt32.

### 9.7.2 Definition Objects

The KXS Meta Definitions that describe the WaveAudioPhysicalDescriptor class, the BextCodingHistory property, and the QltyFileSecurityReport property are shown in Table 82, Table 83 and Table 84.

**Table 82 - Example Metadefinition of WaveAudioPhysicalDescriptor**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **ClassDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Wave Audio Physical Descriptor class | 06.0E.2B.34 02.7F.01.01 0D.01.01.01 01.01.50.00 |
| MetaDefinition Name | UTF16String | The class name | WaveAudioPhysicalDescriptor |
| MetaDefinition Description | UTF16String | The class description | Defines the SMPTE ST 382 Wave Audio Physical Descriptor |
| ParentClass | ClassDefinitionWeakReference | UL of Generic Descriptor | 06.0E.2B.34 02.7F.01.01 0D.01.01.01 01.01.24.00 |
| IsConcrete | Boolean | The class is concrete | True |

**Table 83 - Example Metadefinition of BextCodingHistory**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Bext Coding History property | 06.0E.2B.34<br>01.01.01.05<br>04.02.05.02<br>01.01.00.00 |
| MetaDefinition Name | UTF16String | The property name | BextCodingHistory |
| MetaDefinition Description | UTF16String | The property description | Coding History from BWF <bext> chunk |
| PropertyType | TypeDefinitionWeakReference | UL of the UTF16String type | 06.0E.2B.34<br>01.04.01.01<br>01.10.02.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Wave Audio Physical Descriptor class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.50.00 |
| IsOptional | Boolean | The property is optional | True |

**Table 84 - Example Metadefinition of QltyFileSecurityReport**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Qlty File Security Report property | 06.0E.2B.34<br>01.01.01.05<br>04.02.03.02<br>05.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | QltyFileSecurityReport |
| MetaDefinition Description | UTF16String | The property description | FileSecurityCode (checksum) of quality report |
| PropertyType | TypeDefinitionWeakReference | UL of the UInt32 type | 06.0E.2B.34<br>01.04.01.01<br>01.01.03.00<br>00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of the Wave Audio Physical Descriptor class | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.50.00 |
| IsOptional | Boolean | The property is optional | True |

Note: from the perspective of a SMPTE ST 377-1 decoder, the MetaDefinitions will be ignored, and the Header Metadata of the resultant file remains fully compliant with SMPTE ST 377-1.

## 9.8 Example - Describing a New Essence Mapping (Informative)

### 9.8.1 Example of the implementation of ST422 – Mapping of JPEG2000 Codestreams

New Essence Mappings can be defined for MXF according to the provisions of SMPTE ST 377-1.

Typically, such new mappings define some new labels and a new subclass of an Essence Descriptor. In some cases, new mappings define new optional properties on existing classes.

The example shows what Definition Objects and MetaDefinition Objects are required to describe a new essence mapping that did not exist at the time of publication of SMPTE ST 377:2004. The example that is used is SMPTE ST 422 – Mapping of JPEG 2000 Codestreams.

### 9.8.2 Extensions to Be Described

JPEG2000PictureSubDescriptor is specified in SMPTE ST 422 and defines a number of mandatory and optional properties. For simplicity, this example only describes the Rsiz property of the descriptor. JPEG2000PictureSubDescriptor is a supplementary essence descriptor that can be strongly referenced by any File Descriptor. In order that the strong reference can be made, the specification defines the SubDescriptors optional property for Generic Descriptor, which is described by this example. Also provided is a ContainerDefinition describing Essence Element mapping to Essence Container specified by SMPTE ST 422.

### 9.8.3 Definition Objects

Example of a Metadata Object that describes the optional SubDescriptors property of Generic Descriptor is provided in section 9.1.
The KXS MetaDefinition Objects that describe the JPEG2000PictureSubDescriptor class and Rsiz property, and the KXS Definition Object that describes Frame-wrapped JPEG-2000 Picture MXF Generic Container are shown in Table 85, Table 86 and Table 87.

**Table 85 - Example Metadefinition of JPEG2000PictureSubDescriptor**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **ClassDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of JPEG 2000 Picture Sub-descriptor | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.5A.00 |
| MetaDefinition Name | UTF16String | The class name | JPEG2000PictureSubDescriptor |
| MetaDefinition Description | UTF16String | The class description | Defines the SMPTE ST 422 JPEG 2000 Picture Sub-descriptor |
| ParentClass | ClassDefinitionWeak Reference | UL of SubDescriptor | 06.0E.2B.34<br>02.7F.01.01<br>0D.01.01.01<br>01.01.59.00 |
| IsConcrete | Boolean | The class is concrete | True |

**Table 86 - Example Metadefinition of Rsiz**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **PropertyDefinition** | **Set Key** | **Defines the PropertyDefinition set** | **See Table 10** |
| MetaDefinition Identification | AUID | UL of the Rsiz property | 06.0E.2B.34 01.01.01.0A 04.01.06.03 01.00.00.00 |
| MetaDefinition Name | UTF16String | The property name | Rsiz |
| MetaDefinition Description | UTF16String | The property description | An enumerated value that defines the decoder capabilities |
| PropertyType | TypeDefinitionWeakReference | UL of UInt16 | 06.0E.2B.34 01.04.01.01 01.01.02.00 00.00.00.00 |
| MemberOf | ClassDefinitionWeakReference | UL of JPEG 2000 Picture Subdescriptor | 06.0E.2B.34 02.7F.01.01 0D.01.01.01 01.01.5A.00 |
| IsOptional | Boolean | The property is required | False |

**Table 87 - Example Metadefinition of MXF-GC Frame-wrapped JPEG-2000Pictures**

| Item Name | Type | Meaning | Value |
|---|---|---|---|
| **ContainerDefinition** | **Set Key** | **Defines the ContainerDefinition set** | **See Table 10** |
| InstanceUID | UUID | Unique ID of this ContainerDefinition instance. | A5.BE.A0.FB 2C.41.0F.43 00.02.53.F9 C9.6D.4C.2A |
| Definition Object Identification | AUID | UL of Frame-wrapped JPEG-2000 Picture MXF Generic Container as specified in SMPTE ST RP224 | 06.0E.2B.34 04.01.01.07 0D.01.03.01 02.0C.01.00 |
| Definition Object Name | UTF16String | The Essence Container name | MXF-GC Frame-wrapped JPEG-2000 Pictures |
| Definition Object Description | UTF16String | The Essence Container description | Identifier for MXF-GC JPEG 2000 frame wrapped pictures |

Note: From the perspective of a SMPTE ST 377-1 decoder, the MetaDefinitions will be ignored, and the Header Metadata of the resultant file remains fully compliant with SMPTE ST 377-1. Also, the new class and property instance will be ignored by a decoder that does not already have knowledge of the new Essence Mapping.

# 10 Legacy Information (Informative)

## 10.1 MetaDictionary class

### 10.1.1 MetaDictionary Class Description

The MetaDictionary class is the deprecated earlier version of the ExtensionScheme class.
The MetaDictionary class contains MetaDefinition objects.

In MXF files created by some legacy applications, the MXF file contains one and only one MetaDictionary object.

### 10.1.2 Attributes of the MetaDictionary class

The MetaDictionary class references ClassDefinition and TypeDefinition objects as illustrated in Figure 38.

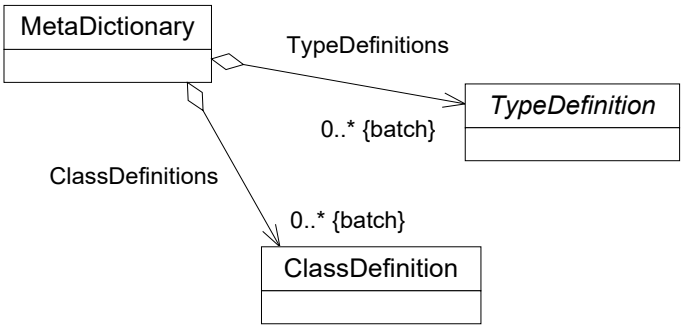


**Figure 38 – MetaDictionary**

### 10.1.3 KLV Encoding of the MetaDictionary class

When encoded as KLV Packets, MetaDictionary Objects are encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths as shown in Table 88.

**Table 88 – KLV Set encoding of MetaDictionary**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| MetaDictionary | Set Key | 16 | | See Table 10 | Req | Defines the MetaDictionary set |
| Length | BER Length | var | | | Req | Set length |
| ClassDefinitions | ClassDefinitionStrongReferenceSet | 8+16n | 00 03 | 06.0E.2B.34 01.01.01.02 06.01.01.07 07.00.00.00 | Opt | Specifies the ClassDefinitions that are used in the file |
| TypeDefinitions | TypeDefinitionStrongReferenceSet | 8+16n | 00 04 | 06.0E.2B.34 01.01.01.02 06.01.01.07 08.00.00.00 | Opt | Specifies the TypeDefinitions that are used in the file |

## 10.2 PropertyDefinitions contained within ClassDefinitions

### 10.2.1 PropertyDefinitions and ClassDefinition Structure

In MXF files created by some legacy applications, Property Definitions are contained within the ClassDefinition of which the property is a member, using the optional Properties property, as illustrated in Figure 39.

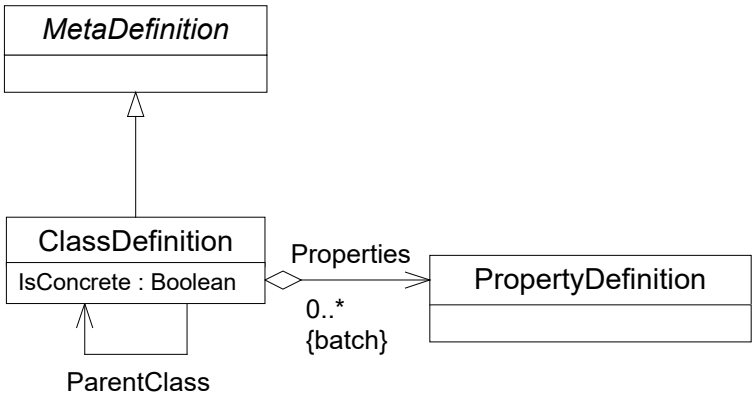


**Figure 39 – ClassDefinition contains PropertyDefinitions**

ClassDefinition objects contain the following attributes:

- ParentClass
- IsConcrete

ClassDefinition objects can contain the following attributes:

- Properties

### 10.2.2 KLV Encoding of the ClassDefinition class

When encoded as KLV Packets, ClassDefinition Objects are encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 89.

**Table 89 - KLV Set Encoding of ClassDefinition**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| ClassDefinition | Set Key | 16 | | See Table 10 | Req | Defines the ClassDefinition set |
| Length | BER Length | var | | | Req | Set length |
| All items from the ClassDefinition class defined in 8.4 above | | | | | | |
| Properties | PropertyDefinition StrongReferenceSet | 8+16n | 00 09 | 06.0E.2B.34 01.01.01.02 06.01.01.07 02.00.00.00 | Opt | Specifies the set of PropertyDefinition objects that define the properties for a class |

The ClassDefinition object specifying the InterchangeObject class has a ParentClass property with a weak reference to itself.

## 10.3 TypeDefinitionExtendibleEnumeration class

### 10.3.1 TypeDefinitionExtendibleEnumeration Class Description

In MXF files created by some legacy applications, known ExtendibleEnumerationElements are contained within the TypeDefinitionExtendibleEnumeration of which the property is a member, using the optional ElementNames and ElementValues properties.

The TypeDefinitionExtendibleEnumeration class defines a property type that can have one of an extendible set of AUID values.
The TypeDefinitionExtendibleEnumeration class is a sub-class of the TypeDefinition class.
All TypeDefinitionExtendibleEnumeration objects is owned by the MetaDictionary object.

An MXF file can contain any number of TypeDefinitionExtendibleEnumeration objects.

### 10.3.2 Attributes of the TypeDefinitionExtendibleEnumeration class

The TypeDefinitionExtendibleEnumeration class contains ElementNames and ElementValues as illustrated in Figure 40.

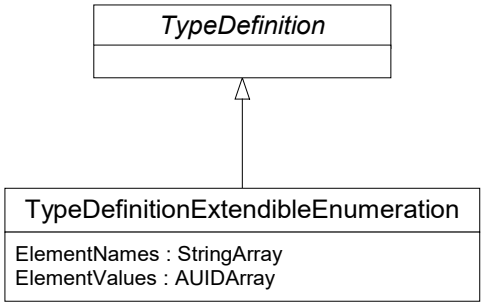


**Figure 40 – TypeDefinitionExtendibleEnumeration**

TypeDefinitionExtendibleEnumeration objects contain the following attributes:

- ElementNames
- ElementValues

### 10.3.3 KLV Encoding of the TypeDefinitionExtendibleEnumeration class

When encoded as KLV Packets, TypeDefinitionExtendibleEnumeration Objects are encoded as SMPTE ST 336 KLV Sets with 2-byte local tags and 2-byte or BER length encoded lengths according to Table 90.

**Table 90 - KLV Set encoding of TypeDefinitionExtendibleEnumeration**

| Item Name | Type | Len | Local Tag | Item UL | Req ? | Meaning |
|---|---|---|---|---|---|---|
| TypeDefinition Extendible Enumeration | Set Key | 16 | | See Table 10 | Req | Defines the TypeDefinitionExtendibleEnumeration set |
| Length | BER Length | var | | | Req | Set length |
| All items from the MetaDefinition class defined in 8.4 above | | | | | | |
| ElementNames | UTF16StringArray | var | 00.1F | 06.0E.2B.34 01.01.01.02 03.01.02.03 07.00.00.00 | Req | Specifies the names associated with each enumerated value |
| ElementValues | AUIDArray | 8+16n | 00.20 | 06.0E.2B.34 01.01.01.02 03.01.02.03 08.00.00.00 | Req | Specifies the known AUID values that can be used in this type |

# Annex A. Static Local Tags assigned by this specification (Informative)

Table 91 below contains the complete list of static Local Tags defined by this specification. For more information on static Local Tag usage see SMPTE ST 377-1.

**Table 91 – Static Local tags in this specification**

| Local Tag | Universal Label |
|---|---|
| 00.01 | 06.0E.2B.34.01.01.01.0A.06.01.01.07.16.00.00.00 |
| 00.02 | 06.0E.2B.34.01.01.01.0A.06.01.01.07.17.00.00.00 |
| 00.03 | 06.0E.2B.34.01.01.01.02.06.01.01.07.07.00.00.00 |
| 00.04 | 06.0E.2B.34.01.01.01.02.06.01.01.07.08.00.00.00 |
| 00.05 | 06.0E.2B.34.01.01.01.02.06.01.01.07.13.00.00.00 |
| 00.06 | 06.0E.2B.34.01.01.01.02.03.02.04.01.02.01.00.00 |
| 00.07 | 06.0E.2B.34.01.01.01.02.06.01.01.07.14.01.00.00 |
| 00.08 | 06.0E.2B.34.01.01.01.02.06.01.01.07.01.00.00.00 |
| 00.09 | 06.0E.2B.34.01.01.01.02.06.01.01.07.02.00.00.00 |
| 00.0A | 06.0E.2B.34.01.01.01.02.06.01.01.07.03.00.00.00 |
| 00.0B | 06.0E.2B.34.01.01.01.02.06.01.01.07.04.00.00.00 |
| 00.0C | 06.0E.2B.34.01.01.01.02.03.01.02.02.01.00.00.00 |
| 00.0D | 06.0E.2B.34.01.01.01.02.06.01.01.07.05.00.00.00 |
| 00.0E | 06.0E.2B.34.01.01.01.02.06.01.01.07.06.00.00.00 |
| 00.0F | 06.0E.2B.34.01.01.01.02.03.01.02.03.01.00.00.00 |
| 00.10 | 06.0E.2B.34.01.01.01.02.03.01.02.03.02.00.00.00 |
| 00.11 | 06.0E.2B.34.01.01.01.02.06.01.01.07.09.00.00.00 |
| 00.12 | 06.0E.2B.34.01.01.01.02.06.01.01.07.0A.00.00.00 |
| 00.13 | 06.0E.2B.34.01.01.01.02.03.01.02.03.0B.00.00.00 |
| 00.14 | 06.0E.2B.34.01.01.01.02.06.01.01.07.0B.00.00.00 |
| 00.15 | 06.0E.2B.34.01.01.01.02.03.01.02.03.04.00.00.00 |
| 00.16 | 06.0E.2B.34.01.01.01.02.03.01.02.03.05.00.00.00 |
| 00.17 | 06.0E.2B.34.01.01.01.02.06.01.01.07.0C.00.00.00 |
| 00.18 | 06.0E.2B.34.01.01.01.02.03.01.02.03.03.00.00.00 |
| 00.19 | 06.0E.2B.34.01.01.01.02.06.01.01.07.0D.00.00.00 |
| 00.1A | 06.0E.2B.34.01.01.01.02.06.01.01.07.0E.00.00.00 |
| 00.1B | 06.0E.2B.34.01.01.01.02.06.01.01.07.0F.00.00.00 |
| 00.1C | 06.0E.2B.34.01.01.01.02.06.01.01.07.11.00.00.00 |
| 00.1D | 06.0E.2B.34.01.01.01.02.03.01.02.03.06.00.00.00 |
| 00.1E | 06.0E.2B.34.01.01.01.02.06.01.01.07.12.00.00.00 |
| 00.1F | 06.0E.2B.34.01.01.01.02.03.01.02.03.07.00.00.00 |
| 00.20 | 06.0E.2B.34.01.01.01.02.03.01.02.03.08.00.00.00 |
| 00.21 | 06.0E.2B.34.01.01.01.0A.06.01.01.07.18.00.00.00 |
| 00.22 | 06.0E.2B.34.01.01.01.0A.06.01.01.07.19.00.00.00 |
| 00.23 | 06.0E.2B.34.01.01.01.0D.06.01.01.07.1A.00.00.00 |
| 00.24 | 06.0E.2B.34.01.01.01.0D.06.01.01.07.1B.00.00.00 |
| 00.25 | 06.0E.2B.34.01.01.01.0D.06.01.01.07.1C.00.00.00 |
| 00.26 | 06.0E.2B.34.01.01.01.0D.06.01.01.07.1D.00.00.00 |
| 00.27 | 06.0E.2B.34.01.01.01.0D.06.01.01.07.1E.00.00.00 |
| 00.28 | 06.0E.2B.34.01.01.01.0D.06.01.01.07.1F.00.00.00 |
| 00.29 | 06.0E.2B.34.01.01.01.0D.06.01.01.07.20.00.00.00 |
| 00.2A | 06.0E.2B.34.01.01.01.0D.06.01.01.07.21.00.00.00 |
| 00.2B | 06.0E.2B.34.01.01.01.0D.06.01.01.07.22.00.00.00 |

## Annex B. Class Hierarchy (Normative)

Figure 41, Figure 42 and Figure 43 illustrate the hierarchy of classes defined in this standard.

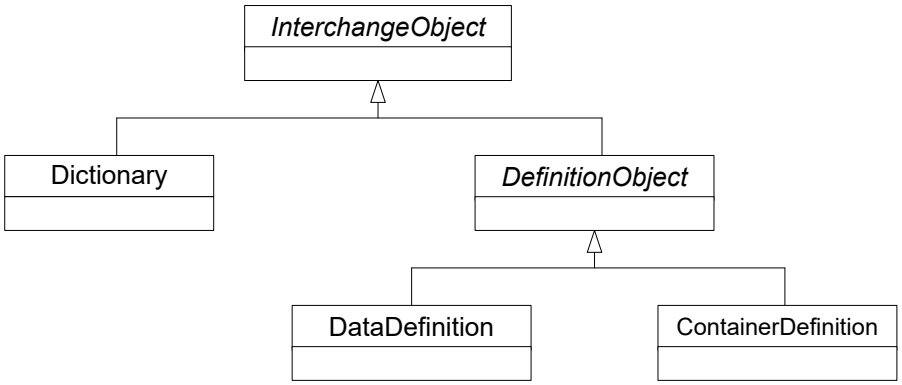


**Figure 41 – Dictionary and Definition Object classes**

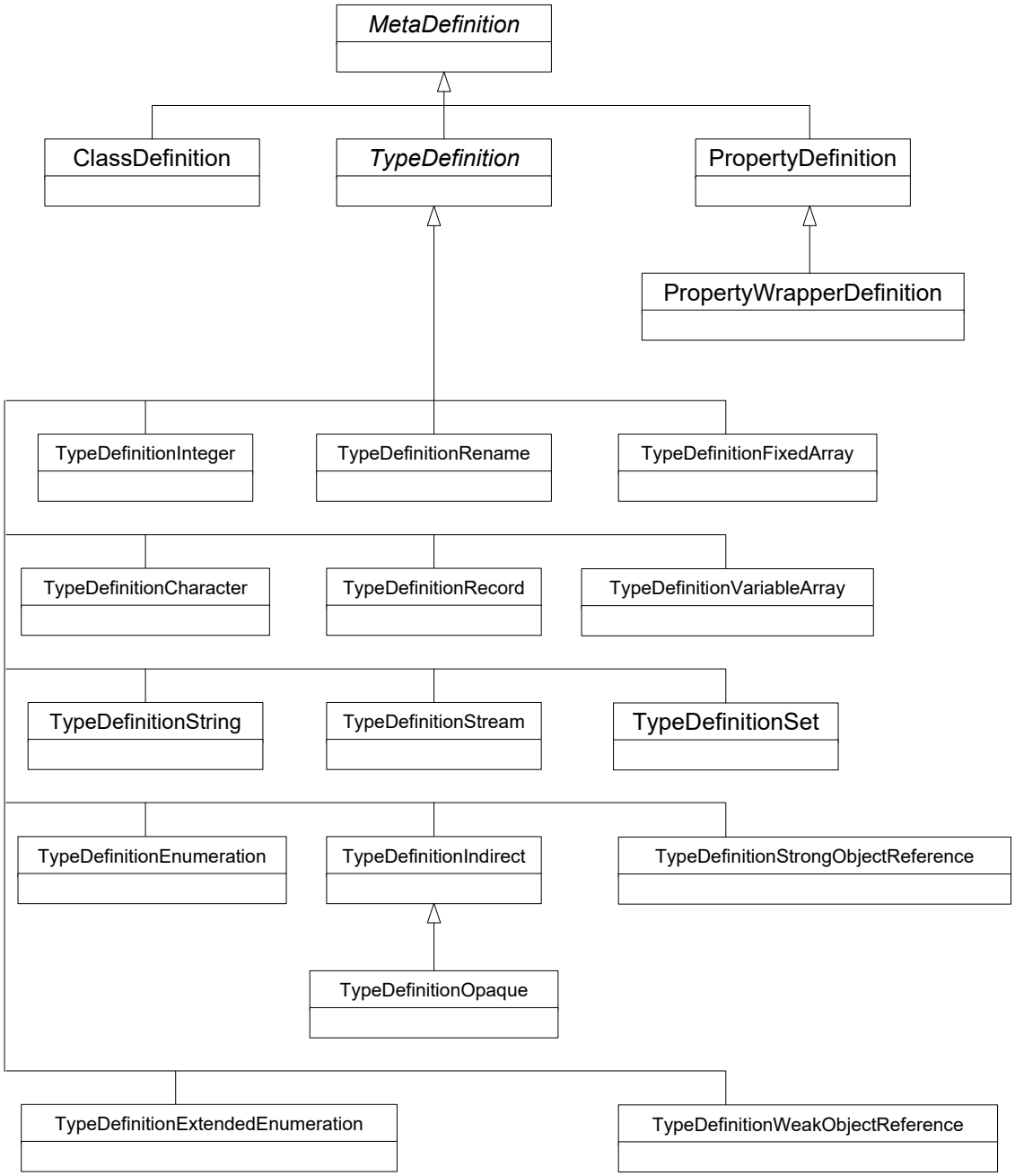


**Figure 42 – MetaDefinition classes**


**Figure 43 – Extension Header classes**

## Bibliography (Informative)

SMPTE ST 380:2004: Material Exchange Format (MXF) Descriptive Metadata Scheme-1

SMPTE ST 422:2014: Material Exchange Format —Mapping JPEG 2000 Codestreams into the MXF Generic Container

SMPTE ST 377:2004: Material Exchange Format