SMPTE STANDARD

Data Encoding Protocol Using Key-Length-Value

Table Of Contents Page

Approved
July 21, 2017

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 30MR - Metadata and Registers.

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.

1 Scope

This standard defines a byte-level data encoding protocol for representing data items and data groups. This protocol defines a data structure which is independent of the application or transportation method used.

The standard defines a Key-Length-Value (KLV) triplet as a data interchange protocol for data items or data groups where the Key identifies the data, the Length specifies the length of the data and the Value is the data itself. The KLV protocol provides a common interchange point for all compliant applications irrespective of the method of implementation or transport.

The standard also provides methods for combining associated KLV triplets in data sets where the set of KLV triplets is itself coded with KLV data coding protocol. Such sets can be coded in either full form (Universal Sets) or in one of four increasingly bit-efficient forms (Global Sets, Local Sets, Variable Length Packs and Defined Length Packs). The standard provides a definition of each of these data constructs.

The standard also describes implications of KLV coding including the use of a SMPTE Universal Label as a value within a KLV coding triplet or whose meaning is entirely conveyed by the SMPTE UL itself. The two kinds of usage for such standalone SMPTE Universal Labels are a) as a value in a K L V construct and b) as a Key that has no Length and no Value. This standard defines where SMPTE ULs can be used for each kind of construct.

The standard also defines the use of KLV coding to provide a means to carry information that is registered with a non-SMPTE external agency.

The encoding byte range (length of the payload) specified in this document can accommodate unusually large volumes of data. Consequently, a specific application of KLV encoding might require only a limited operating data range and those details be defined in a relevant application document.

This revision is intended to yield identical encodings to previous revisions of ST 336 given identical dictionaries.

Section 4 provides a glossary of terms used in this standard.

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.

2.1 Conforming Implementations

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.

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 298:2009 Universal Labels for Unique Identification of Digital Data.

ISO/IEC 8825-1:2008 |ITU-T X.690:2008, Information Technology – ASN.1 Encoding Rules – Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER): BER Paragraphs 8.1.3.4 and 8.1.3.5 for Length, 8.19 for Object Identifier Coding.

4 Terms and Definitions

ASN
Abstract Syntax Notation (see ISO/IEC 8825-1 (ITU-T X.690)).

Basic Encoding Rules (BER)
ISO standard encoding for various constructs in ASN.1. Includes the encoding of Object Identifiers, and also of Length fields. The length bytes of the KLV packet conform to the Basic Encoding Rules (BER) for either the short form or long form encoding specified in ISO/IEC 8825-1, Pars. 8.1.3.4 and 8.1.3.5.

Big-Endian
Multi-octet (multi-byte) data entity that has the most significant octet (byte) first in time or leftmost in diagrams.

Byte
Widely used alternative for the term ‘octet’ (see Octet, below).

CER
Canonical Encoding Rules (see ISO/IEC 8825-1 (ITU-T X.690)).

Container
Generic name for a data object which provides a framework to ‘contain’ different kinds of information. The term is commonly applied to multimedia where audio, video, data essence and metadata are formed into a single data object.

Control Data
Item of data that is used to provide a control function for essence data or metadata.

Data Group
Collection of data items.

Data Item
Data entity in this standard. The term ‘item’ is widely used in other documents and can have a different meaning. A data item is not a group in this standard.

Data Type
(see definition of Type below).

DER
Distinguished Encoding Rules (see ISO/IEC 8825-1 (ITU-T X.690)).

Dictionary
Register that provides for the semantic interpretation of the data items within the register.

Essence
Abstract term that describes any data or signal necessary to represent any single type of visual, aural or other sensory experience independent of the method of coding. Also identified by the SMPTE/EBU “Task Force for Harmonized Standards for the Exchange of Program Material as Bitstreams” (TFHS) as Video, Audio, and/or Data information. Essence can also be Graphics, Telemetry, Photographs, Haptic / Tactile or other information.

ISAN
International Standard Audiovisual Number.

Key
16-byte SMPTE administered Universal Label used for KLV coding of data.

KLV
Key-Length-Value; data format defined by this standard.

Metadata
Generally referred to as “data about data” or “data describing other data”. Metadata is information that is considered ancillary to or otherwise directly complementary to the essence. Also any information considered useful or of value when associated with the essence.

Metadata Dictionary
Standard database of approved Metadata Items including definitions and allowed formats.

Metadata Item
Broad term for a unit of metadata.

Object Identifier (OID)
First byte in the UL that identifies it as a UL — abbreviated OID. Always “06” in hexadecimal (hex) notation (0x06).

Octet
Data word comprising 8 binary digits.

Primitive Encoding
In ASN.1 notation, a definite-length encoding method that applies to simple encoding types and types derived from simple types by implicit tagging. It requires that the length of sub-identifiers be known in advance.

Register
Information store or database that is maintained by a registry.

Registry
Information system for registering data.

SMPTE Administered UL (abbreviated to UL in this standard)
UL that is administered by SMPTE in accordance with SMPTE ST 298. All SMPTE administered ULs are 16 bytes in length.

SMPTE Label
SMPTE UL that is self identifying. (See Section 9).

SMPTE Registration Authority
Registration organization which keeps record of the use of SMPTE ST 298 UL Keys and other reference data.

ST / RP
Abbreviation for “SMPTE Standard or Recommended Practice”. In this document, it is used as a generic indication that other SMPTE documents (maybe not yet written) will be needed to fully define a particular usage of KLV. Those documents will be identified against each item in the appropriate register.

Type or Data Type
Information that defines how data is represented.

SMPTE UL
Abbreviation of the term ***SMPTE Administered UL.***

Tag
Special form of identification that is local to the coding format. In its fully expanded form, a Tag can be identical to the Item Designator.

UL
Universal Label; object identifier according to ISO/IEC 8824-1 (see also SMPTE ST 298). In this standard, this term is used to mean a SMPTE administered UL.

Wrapper
Identified by the SMPTE/EBU “Task Force for Harmonized Standards for the Exchange of Program Material as Bitstreams” (TFHS) as a means of wrapping video, audio, data essence and metadata information into a common framework. In this definition, it is identical to the definition of a container, but Wrappers can further be used to ‘wrap’ further metadata around an already defined container. In this sense, a container is a multi-purpose box which has audio-visual information and a wrapper is the packaging around the box including labeling and other supporting metadata.

5 KLV Protocol

5.1 Overview

Table 1 and Figure 1 present an overview of the KLV Protocol for encoding data. The data encoded can be a single data item, or a data group. The coding of data items is described in Section 6 while the coding of data groups is described in Section 7 of this standard.

The KLV Coding Protocol is composed of a Universal Label (UL) identification “Key”, followed by a numeric “Length” (Value Length), followed by the data “Value”.

The composition of the Key is described in Section 5.2 of this standard. The length of the full Key is 16 bytes. The Length field is described in Section 5.3 of this standard. The byte-length of the Length field varies as defined in Section 5.3 of this standard. The Value is described in Section 5.4 of this standard. The Value is a sequence of bytes of the data type as specified in a relevant standard and is not further encoded by the KLV Protocol. The length of the Value field is variable and any limitations are defined in a relevant defining standard.

Table 1 - KLV Fields for Encoding of Data

Field	Description	Length	Contents/Format
Key	UL for identification of the Value	16 bytes	Section 5.2
Length	Length of the Value field	Variable, with constraints defined by an application document	Section 5.3
Value	Value associated with the Key	Variable	Section 5.4

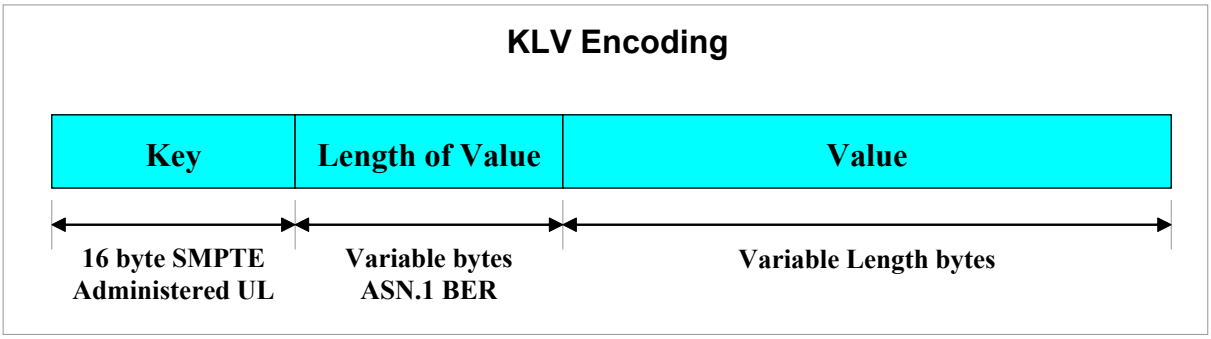


Figure 1 – KLV Encoding

5.2 Encoding of the KLV Key Field

5.2.1 Key Structure

The KLV Coding Protocol shall use a fixed 16-byte SMPTE-administered Universal Label, according to SMPTE ST 298, as the Key to identify the data in the Value field. The term UL is used throughout this standard to refer to a SMPTE-administered Universal Label.

The UL of the Key shall be as specified in Table 2.

The sub-identifiers in the Key shall have left-to-right significance with the first sub-identifier as the most significant. The leftmost sub-identifier of value 0x00 in the Key shall define the termination of the label and all sub-identifiers of lower significance shall also be set to 0x00. Sub-identifiers of value 0x00 shall have no significance to the meaning of the Key.

NOTE 1 - Each word in the UL is coded using ASN.1 Basic Encoding Rules (BER) for Object Identifier coding specified in ISO/IEC 8825-1, paragraph 8.19 as required by SMPTE ST 298 (For convenience, this is reproduced in Annex J). The BER Object Identifier coding constrains the value of all single-byte UL sub-identifiers to the range 0x01 to 0x7F.

NOTE 2 - SMPTE ST 298 defines only the first four bytes of a UL: the Object ID, UL Size, UL Code and SMPTE Designator. This standard specifies the application of SMPTE ST 298 ULs for the purpose of Key-Length-Value coding and defines the semantics of sub-identifier bytes 5 to 8. The semantics of the Item Designator (sub-identifier bytes 9-16) are defined by a number of separate documents, which are identified in the associated register and together cover all the defined values of the Key.

Table 2 (Normative) - Encoding of the KLV Key Field

No.	Field	Description	Length	Content/Format
UL bytes defined by ST 298				
1	**OID**	Object Identifier	1 byte	Always 0x06
2	**UL Size**	16-byte size of the UL	1 byte	Always 0x0E
3	**UL Code**	Concatenated sub-identifiers ISO, ORG	1 byte	Always 0x2B
4	**SMPTE Designator**	SMPTE sub-identifier	1 byte	Always 0x34
UL sub-identifiers defined by ST 336				
5	**Category Designator**	Category designator identifying the category of registry described (e.g. Dictionaries)	1 byte	Defined in Table 3
6	**Registry Designator**	Registry Designator identifying the specific register in a category (e.g. Metadata Dictionaries)	1 byte	Defined in Table 3
7	**Structure Designator**	Designator of the structure variant within the given registry designator	1 byte	Defined in Table 3
8	**Version Number**	Version of the given register which first defines the item specified by the Item Designator	1 byte	Incrementing number
UL sub-identifiers defined by application standards listed in given register				
9 – 16	**Item Designator**	Unique identification of the particular item	8 bytes	Defined in relevant standard and version

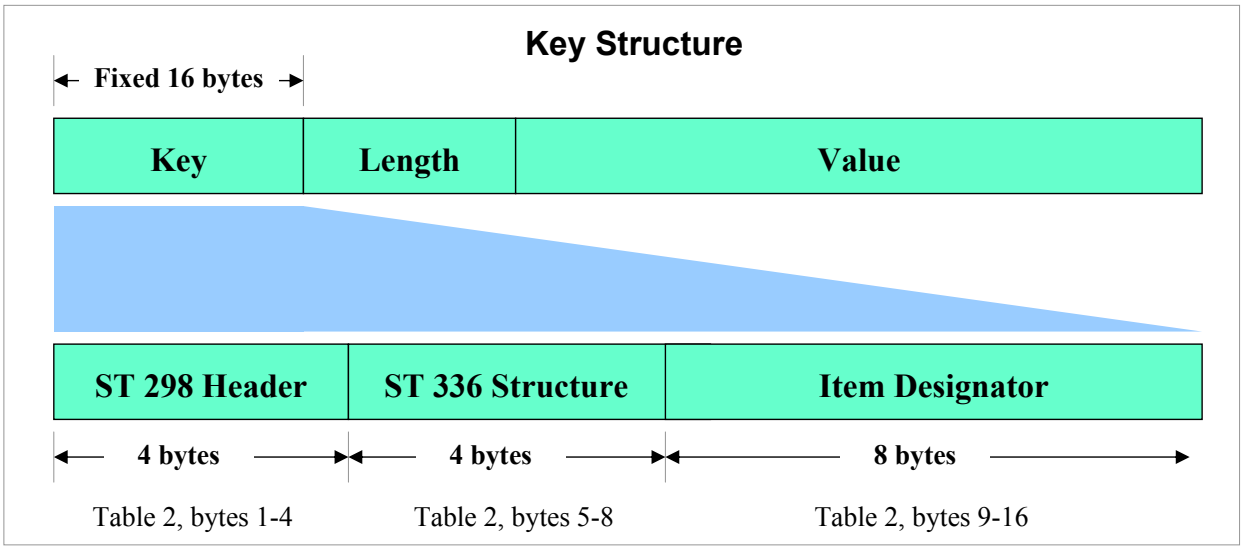
Annex A shows an example of a Metadata Key from the SMPTE Metadata Dictionary in tabular and figure formats.

Decoders that recognize the Key but do not want to, or cannot, decode the associated Value may ignore the item and continue the decoding process of subsequent items using the Length value to ‘skip’ the Value of the un-decoded item. If decoders store and forward the item, they shall forward the item unaltered.

Bytes 5 and 6 of the Key shall identify the contents of, and define the interpretation of the Value for all values of the Item Designator within a given category and registry designator. Table 3 defines the use of bytes 5 and 6. When bytes 5 and 6 do not match any of the values in Table 3, the decoder shall not interpret the contents of the "V" bytes, shall make the K L and V available to application processing, and shall continue parsing with the byte immediately following the end of the "V".

NOTE - Public and private registers of SMPTE UL number spaces exist and these registers will contain valid KLV keys which can be unknown to the decoder. Provision of application level interpretation of unrecognized KLV keys is important for interoperability.

Figure 2 – Key Structure



5.2.2 Key Bytes 5 – 7

5.2.2.1 Definitions for Key Bytes 5-7

The values for bytes 5 to 7 of Keys shall conform to Table 3.
SMPTE Standards (STs) and Recommended Practices (RPs) that define a Key with the value of byte 5 (Registry Category Designator) in the range 0x01 to 0x04 shall register the Key or Keys used with the SMPTE Registration Authority in the register identified by bytes 6 and 7 (Registry Designator and Structure Designator).
Structure Designator values are allocated to distinguish between incompatible versions of the same register. They can be thought of as a major version number.

Table 3 (Normative) - Definitions for Key Bytes 5 - 7

Category Designator	Registry Designator	Defined In:	Structure Designator	Defining Documents (informative)
Byte 5	**Byte 6**		**Byte 7**	
0x01 – Dictionaries		**Section 6**		
	0x01 – Metadata Dictionaries	Section 6.2.2	0x01~0x7F	Byte 7 = 0x01: SMPTE ST 335
	0x02 – Essence Dictionaries	Section 6.2.3	0x01~0x7F	Byte 7 =0x01: SMPTE ST 2088
	0x03 – Control Dictionaries	Section 6.2.4	0x01~0x7F	
	0x04 – Types Dictionaries	Section 6.2.5	0x01~0x7F	Byte 7 = 0x01: SMPTE ST 2003
0x02 – Groups (Sets and Packs)		**Section 7**		**SMPTE ST 395**
	0x01 – Universal Sets	Section 7.2	0x01~0x7F	
	0x02 (default) – Global Sets	Section 7.3, Table 4	Section 7.3	
	0x03 (default) – Local Sets	Section 7.4, Table 5	0x01~0x7F	
	0x04 (default) – Variable Length Packs	Section 7.5, Table 6	0x01~0x7F	
	0x05 – Defined-Length Packs	Section 7.6	0x01~0x7F	
	0x06 – Reserved		0x01~0x7F	
0x03 – Wrappers and Containers		**Section 8**		
	0x01 – Simple Wrappers and Containers		0x01~0x7F	
	0x02 – Complex Wrappers and Containers		0x01~0x7F	
0x04 – Labels		**Section 9**		
	Labels Registers	Specific Labels Register - Section 9	0x01~0x7F	Byte 7 = 0x01: SMPTE ST 400
0x05 – Registered Private Information		**Section 10**		**SMPTE RP 225**
0x06 – 0x7E - Reserved				

5.2.2.2 Dictionaries

SMPTE STs and RPs that define Keys with the value of byte 5 set to 0x01 shall define Dictionary Keys that shall define single data items.

The coding of individual data items is defined in Section 6.

5.2.2.3 Groups (Sets and Packs)

SMPTE STs and RPs that define Keys with the value of byte 5 set to 0x02 shall define Set Keys or Pack Keys that shall be used to define groups of KLV coded data items.

The coding of data groups is defined in Section 7. Global Sets, Local Sets and Variable Length Packs use more than one value to identify the lengths of the Length field and, in the case of Local Sets, the Local Tag field.

5.2.2.4 Wrappers and Containers

SMPTE STs and RPs that define Keys with the value of byte 5 set to 0x03 shall define Wrapper Keys or Container Keys that shall identify the Wrapper or Container and its contents. The terms 'Wrapper' and 'Container' are defined in Section 4 Terms and Definitions.
The use of Wrappers and Containers is defined in Section 8.

5.2.2.5 Labels

The coding of Labels is defined in Section 9.

5.2.2.6 Registered Private Information

The coding of Registered Private Information is defined in Section 10.

5.2.2.7 Reserved

Reserved Registry Categories are reserved for future extension to this standard or for provisions defined by other SMPTE standards. No other specifications shall use these reserved values.

5.2.3 Version Number

For items registered by SMPTE, byte 8 shall define the version number of the given register, which first defines the item specified by the Item Designator (defined in Section 5.2.4).
For items not registered by SMPTE, the entity responsible for registration of items shall define its own version numbering policy.

NOTE - In any register, SMPTE can assign parts of the register to another organization whether as a public or private organization, or for other reasons make a part of the register an open space for undefined use. The policy of assigning these parts of the registry is not specified by this standard but by the individual registry standards. Typically, SMPTE will assign the first node that identifies the non-SMPTE organization and this node will be given a SMPTE version number. The values of the version number that lie under the SMPTE assigned node will then be assigned by the appropriate non-SMPTE organization.

New items will be added to registers after initial approval of the controlling Standard or RP. Each time a set of item definitions is added, the current Version Number of the particular register is incremented. Each entry in a register includes the version number in which the item was first defined. This Version Number is carried in byte 8.

Decoders may ignore the version number or use the version number as an additional guide and consistency check in the process of parsing a Key.

5.2.4 Item Designator

Bytes 9 through 16 of the Key comprise the Item Designator.

The Item Designator field shall be a fixed 8 bytes in length. Item Designator values are from 1 to 8 bytes long, padded on the right with zero bytes to fill the 8 byte field, and coded using ASN.1 BER Object Identifier coding as described in Section 5.2.

The precise meaning and construction of the Item Designator depends upon the specific register and Structure Designator value, and is described further in the sections below.

5.3 Encoding of the KLV Length Field

In the KLV coding protocol, the value of the Length field shall be encoded using the Basic Encoding Rules (BER) for either the short form or long form encoding of length bytes specified in ISO/IEC 8825-1, paragraph 8.1.3, 8.1.3.3 through 8.1.3.5 (a copy of which is included in Annex I). This method of encoding the Length field is self-contained and allows for efficient parsing of KLV encoded data. When the KLV coding protocol is applied to groups of KLV coded units, the Length field for individual units may adopt a different method as defined by the standard for the coding of that group (see Section 7).

Where appropriate, individual application ST / RPs may define the maximum byte-length of the Length field or may place limitations on the value range of the Length field in order to simplify decoder requirements.

NOTE 1 - While there are no restrictions in this standard on the maximum number of bytes in the Length field, the presence of large Length fields can be determined from the first byte of the ASN.1 BER Long Form Length Encoding.

NOTE 2 - It is suggested that the short form of ASN.1 BER be used for all Value fields smaller than or equal to 127 (0x7f).

Implementations should make every effort to apply a valid value to the Length field. However, in certain operations it can prove impractical to establish the length of the Value field. Such a case is an incoming data stream which is assigned a Key and a Length field at the start point. In this case, the value of the Length field cannot be established until the termination of the stream and at that point, it can prove impossible to return to the Length field to enter the value. In such cases, the Length field shall be set to [0x80] which shall indicate a non-deterministic length of the Value field. Any application document which allows the length of the Value field to be undefined shall define an alternative method of locating the end of the Value field.

NOTE 3 - The length value [0x80] is used because it is normally meaningless as an ASN.1 BER long-form value as it indicates zero subsequent bytes.

5.4 Encoding of Data Values

Data values may be either individual data items or data groups. In either case, the data is a byte string whose length is specified by the Length field value. The last byte of the Value field shall be the terminating byte of the data sequence.

5.5 Empty Data Items

Specifications for contiguity of KLV packets including any gaps between KLV packets are outside the scope of this standard and are addressed in the appropriate transport-layer documents.

However, if applications so require, breaks in the data sequence may be inserted by the use of a specific “Empty” data item. Use of “Empty” data items is not mandatory.

The “Empty” data item is a KLV coded packet which shall define a length value followed by an empty Value field. No attempt should be made to interpret the data in the Value field.

"Empty" data items may be coded as individual items, or within sets when allowed by the specific set definition.

Applications may delete or skip any or all "Empty" data items upon receipt. Applications may insert "Empty" data items, but shall not require other applications to preserve such items.

The "Empty" data item shall be defined in the metadata dictionary and may be defined in other dictionaries.
In the particular case of the empty data item, implementations have been deployed with different version values so it is recommended that decoders ignore the version number value.

NOTE 1 - Section 5.2.3 provides guidance on the use of the version number byte.
NOTE 2 - The empty data item is widely known as the Fill Item.

6 KLV Coding of Individual Data Items

6.1 General

The KLV coding of individual data items is a simple application of the Key, Length and Value as defined in Section 5.

The Key of individual data items is defined in a register together with the ranges of Length and the specification of the Value itself. For individual data items the value of byte 5 of the Key shall be 0x01.

6.2 Registers of Individual Data Items

6.2.1 Dictionaries

Individual data items that are KLV coded are collected into registers for the purpose of data management.

This standard defines four different registers as identified in Table 3. These registers are 'dictionaries' and they are identified by the value of byte 6 of the Key as follows:

6.2.2 Metadata Dictionaries

Metadata dictionaries shall be identified by byte 6 having the value of 0x01. Metadata dictionaries shall be registers of metadata items.

Metadata is information other than Essence that has no inherent stand-alone value, but is related to Essence (i.e. it is contextual and has no meaning outside its relationship to the associated Essence). Examples of Metadata include: URL, URI, timecode, MPEG-2 PCR, filename, program labels, copyright information, version control, watermarking, conditional-access keys, etc.

6.2.3 Essence Dictionaries

Essence dictionaries shall be identified by byte 6 having the value of 0x02. Essence dictionaries shall be registers of essence items.

Essence is the data that represents pictures, sound and text. Types of Essence include Video, Audio and Data of various kinds, including captions, graphics, still images, text, enhancements and other data as needed by each application.

6.2.4 Control Dictionaries

Control dictionaries shall be identified by byte 6 having the value of 0x03. Control dictionaries shall be registers of control data items.

6.2.5 Types Dictionaries

Types dictionaries shall be identified by byte 6 having the value of 0x04. Types dictionaries shall be registers of data types.

Many register values share a common set of definitions for multiple data representations. To simplify the register definitions, the 'Types' Dictionary shall be used to define these data representations. The Types Dictionary shall be used as a shared resource for all other dictionaries.

6.3 Identification of Value Data Representations

The Value of many data items can be represented in more than one way. For example, a start time in the metadata dictionary can be represented as a character string of the timecode or as an efficient bit-packed form. The first offers direct mapping to a display whereas the second offers high transmission efficiency for use in narrow-band data channels. There are many such metadata dictionary items that have multiple data representations for the same descriptor.

Where a data item has more than one data representation for the value, one representation shall be designated as the default representation and shall be assigned a Key with at least one trailing zero byte. Alternate representations shall be assigned Keys by replacing the leftmost trailing zero byte with non-zero values, which shall be assigned sequentially. Each representation shall be documented as a separate entry in the register.

Example:

01.02.03.04.00.00.00.00 is 'Name' (default data representation in 16-bit Unicode characters)

01.02.03.04.01.00.00.00 is 'Name' (different data representation in ISO 7-bit characters)

01.02.03.04.02.00.00.00 is 'Name' (another data representation in UTF-8 Unicode characters)

The parser treats all representations as the same data item, i.e. it recognizes 01.02.03.04.00, then looks for xx in place of the '00' to identify different encodings. Since the default representation is defined, the extra non-zero term in the 5th position is known to be a new data representation of the default register item. Different representations may constrain the values that a data item may use.

Annex B shows an example of KLV coding for a single metadata item.

7 KLV Group Coding

7.1 Introduction to Groups

Group coding of data items can be used to reduce the overhead of repeating redundant information that appears in the Key of each item in the Group. Group coding also allows logical groups of individual data items, or groups of items, to be encoded together and provides options for increased bit-efficiency. In order of increasing coding efficiency, KLV Coding Protocol can be used to support the following structures:

Universal Sets are used to construct a logical grouping of data items and other KLV encoded items. Universal Sets use the full KLV Coding Construct throughout.

Global Sets are defined as per Universal Sets, but offer coding efficiency by sharing a common Key header. This coding gain is lossless and every Key can be fully recovered from the data in the Global Set alone.

Local Sets are defined as per Universal Sets, but offer coding efficiency through the use of short Local Tags whose meaning is defined within the context of the Local Set. Local Sets retain the KLV data construct but require a separate ST / RP to define the meaning of the Local Tags and to provide a map from the Local Tag value to the Key value.

Variable-Length Packs are defined as a further grouping of data items that eliminates the use of Keys and Local Tags for all individual items within the group. Variable-length Packs therefore rely on a ST / RP to define the order of data items within the pack and the UL of each item in the pack.

Defined-Length Packs are the most efficient (and least flexible) grouping of data items that eliminates the use of both Keys and Local Tags and removes the length for all individual items within the group. Thus Defined-Length packs rely on a ST / RP which defines both the order of data items, the length of each data item within the pack and the UL of each item in the pack.

Group coding shall only be used for the encoding of sets and packs described in this standard.

Sets and Packs shall consist of a number of individual data items that are coded as a group by the KLV Set or Pack data construct. The Set or Pack shall be defined by a full Key whose value shall be registered with the SMPTE Registration Authority.

Sets and Packs may encode data that are themselves Sets or Packs as well as individual register items. This is called KLV recursive coding and this standard provides no limit on the number of levels of recursion which may be used by any particular application.

The presence of Sets or Packs shall be indicated by 0x02 in the Registry Category Designator field (byte 5) of the Set or Pack Key. The Registry Designator field (byte 6) shall be used to identify the type of Set or Pack. The Set or Pack register shall be identified by the Structure Designator field (byte 7) and the version of the register shall be identified by the Version Number field (byte 8).

The Set or Pack Value shall be comprised of a number of individual data items with coding as defined by the Set or Pack type. In a Pack, the order and presence of items is defined. By default, the order and presence of items in a Set are not defined. Specific ST / RPs may define constraints for the order and presence of data items within any specific Set or group of Sets.

The following sections define how the data items are encoded for Universal Sets, Global Sets, Local Sets, Variable-Length Packs and Defined-Length Packs.

7.2 Universal Sets

A Universal Set is defined as a number of data items that are grouped for application or management reasons. Data items may be in any order within the Universal Set and may be present or absent.

The use of Keys for Universal Set coding shall be defined by an accompanying ST / RP including a Structure Designator and an accompanying Register including a version number.

The Key of a Universal Set is 16 bytes in length.

The Length of a Universal Set is coded as per ASN.1 notation; long or short form as needed.

The Value of a Universal Set shall be a sequence of KLV-encoded items whose total length is given by the Length field. Each and every data item in a Universal Set shall apply KLV Data Coding protocol including the full Key value.

Relevant Application ST / RPs may specify constraints upon the Value of a Universal Set such as the number and size of items, the allowed sequence of items and whether any items are mandatory or optional.

The Universal Set Designator is defined within the last 8 bytes in the Universal Set Key. Universal Set Keys shall be defined in an associated ST / RP and the Key value shall be registered with the SMPTE Registration Authority in accordance with the provisions of the accompanying ST / RP to guarantee a unique Key value.

Figure 3 illustrates the data structure for the encoding of Universal Sets.

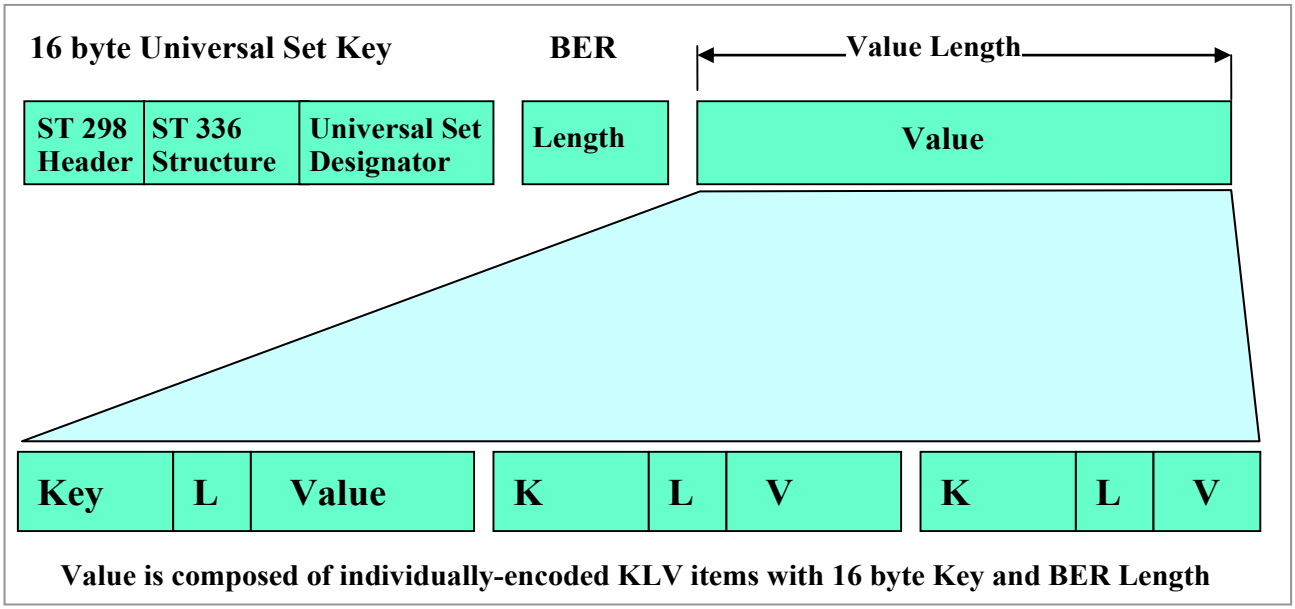


Figure 3 – KLV Coded Universal Set Data Structure

An example of the operation of Universal Sets is given in Annex C

7.3 Global Sets

A Global Set is defined as a number of data items that are grouped to losslessly reduce the length of the Keys for each item within the Set. Data items may be in any order within the Global Set and may be present or absent.

The use of Keys for Global Set coding shall be defined by an accompanying ST / RP including a Structure Designator and an accompanying Register including a version number.

The Key of a Global Set is 16 bytes in length.

The Length of a Global Set is coded by default as per ASN.1 notation; long or short form as required.

The Value of a Global Set shall be a sequence of KLV-encoded items whose total length is given by the Length field. Each and every data item of a Global Set shall apply KLV Data Coding protocol but with a shortened Global Tag value replacing the Key as described next.

The Global Set UL shall be defined in two parts:

- The first group of 8 bytes (UL Header and UL Designator) shall be registered with the SMPTE Registration Authority and shall be used to identify the Global Set Standard or RP including the structure designator. Each entry in the Global Set Register shall record the version number in which it was first defined.
- The second group of 8 bytes is called the Global Set Designator and shall be used to define the common UL Header and UL Designator for all the Keys within the Global Set. This second group of 8 bytes shall include the UL Header fields together with as much of the UL Designator as is common to all items in the Global Set and indicated by the Structure Designator (byte 7). The Global Set Designator may be terminated by a zero-value byte to indicate termination of the common UL Designator root. The significant length of the second group to the zero value terminator shall be 2 to 8 bytes. If the length of the second group is 8 bytes, then the zero-value terminator byte is not required.

Each Global Tag is from 2 to 12 words in length. Global Tags of length less than 12 words shall be terminated by a single zero value thus removing redundant UL data.

The full 16-byte Key of each data item in the Global Set can be losslessly re-created by concatenating the non-zero bytes of the Global Set Designator and the Global Tag of the item. If the resulting concatenation is less than 16 bytes in length, the remaining bytes in the 16-byte space shall be zero filled.

Application ST / RPs may specify constraints upon the Value of a Global Set such as the number and size of items, the allowed sequence of items and whether any items are mandatory or optional.

The Global Set Designator is defined within the last 8 bytes in the Global Set Key. It defines the common portion of the Key shared by all Global Tags. The active number defines the bytes needed to establish the common root for all Global Tags (2-8 bytes). Global Set Keys shall be defined in an associated Standard or RP and the Key value shall be registered with the SMPTE Registration Authority in accordance with the provisions of the accompanying ST / RP to guarantee a unique Key value.

The value of the Structure designator shall be equal to 1 plus the number of initial bytes of the Key which are copied from the Key before the start of the common portion. Values shall be in the range 1 (0 bytes copied) through 9 (8 bytes copied). A Value of 5 (4 bytes copied) is most reasonable.

Figure 4 illustrates the structure for the encoding of Global Data Sets.

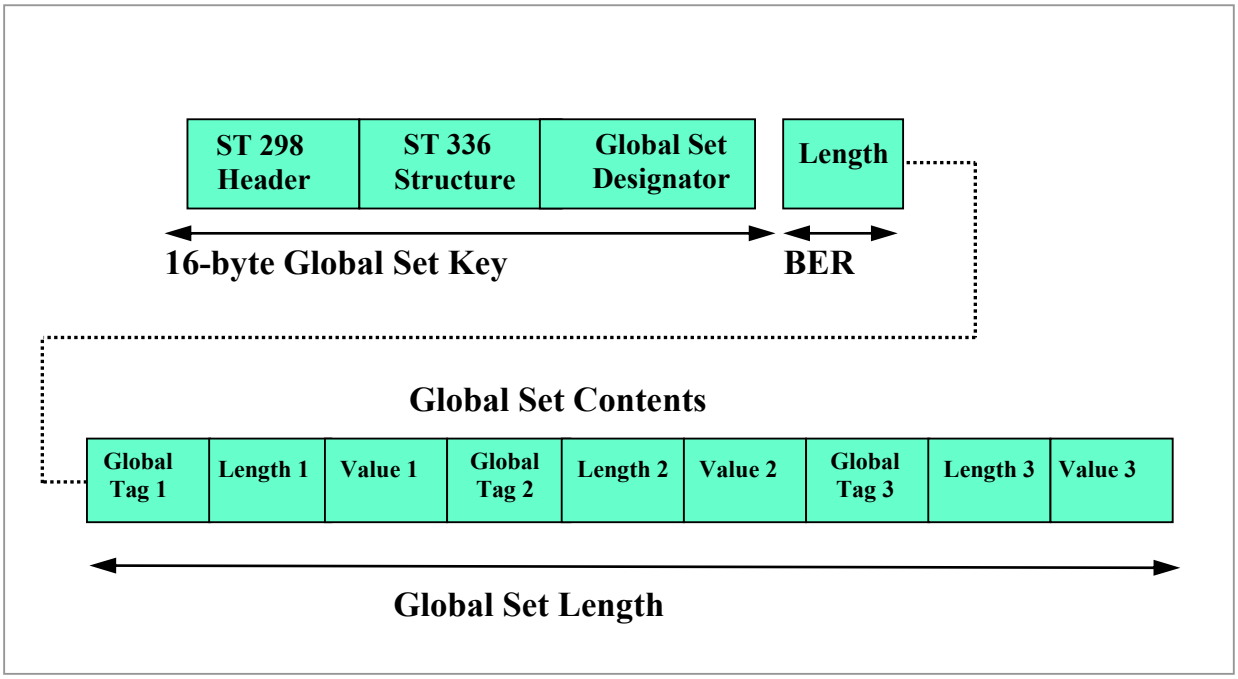


Figure 4 – KLV Coded Global Set Data Structure

The 16-byte Global Set Key is followed by the Global Set Length (encoded using ASN.1 BER length encoding) which is followed by a number of data items which shall each be triplets of Global Tag, Length and Value.

When non-ASN.1 BER encoding is used it shall be big-endian byte ordered. The full range of allowed Length field lengths shall be defined by the Registry Designator according to Table 4. All Length fields in the Global Set shall follow the same syntax.

Table 4 (Normative) - Coding of Registry Designator (Byte 6) for Global Set Syntax

Byte 6 Value	Length field	Description
0x02	ASN.1 BER short or long	Any length (default)
0x22	1 byte	Length up to 255
0x42	2 bytes	Length up to 65535
0x62	4 bytes	Length up to 2^{32}-1

Global Sets can accommodate recursion, so that the Key linked to a Global Tag may identify either a single data item from a register or a data set or pack from a Set or Pack Standard or RP and the corresponding Register.

NOTE – Global Sets are useful for containing a group of data items that share a common hierarchy of individual key values, for example, metadata items that are listed under a common node in a register.

An example of the operation of Global Sets is given in Annex D

7.4 Local Sets

A Local Set is defined as a number of data items that are grouped to reduce the length of the Keys for each item within the Set. Data items may be in any order within the Local Set and may be present or absent.

The use of Keys for Local Set coding shall be defined by an accompanying Standard or RP including a Structure Designator and an accompanying Register including a version number.

The Key of a Local Set is 16 bytes in length.

The Length of a Local Set is coded by default as per ASN.1 BER length notation; long or short form as required.

The Value of a Local Set shall be a sequence of KLV-encoded items whose total length is given by the Length field.

The Local Set Designator is defined within the last 8 bytes in the Local Set Key. It defines the Local set placement in a hierarchical structure. Local Set Keys shall be defined in an associated ST / RP and the Key value shall be registered with the SMPTE Registration Authority in accordance with the provisions of the accompanying ST / RP to guarantee a unique Key value.

The data structure for the encoding of Local Sets is illustrated by Figure 5.

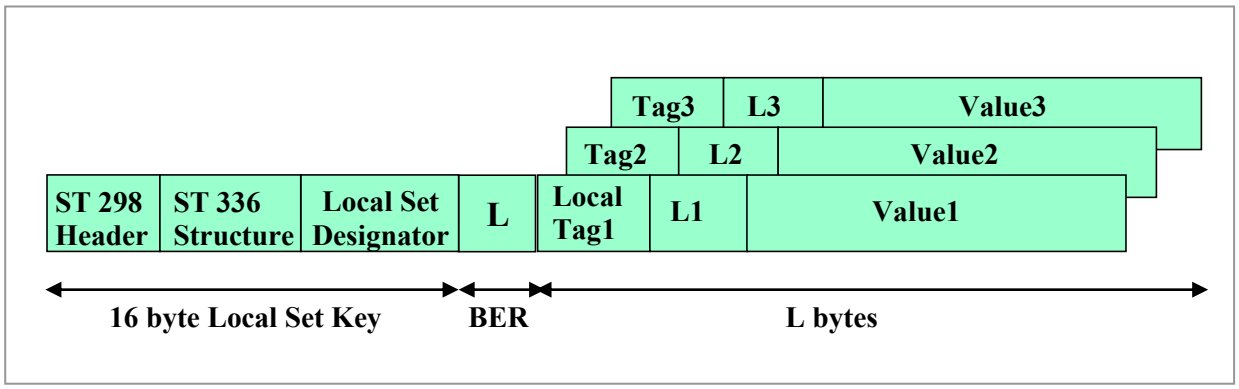


Figure 5 – KLV Coded Local Set Stucture

The 16-byte Local Set Key is followed by the Set Length, which is followed by a number of data items, which shall each be triplets of Local Tag, Length and Value.

The preferred size of Local Tag fields is 1 byte. When non-ASN.1 BER encoding is used it shall be big-endian byte ordered. The full range of allowed combinations of Local Tag and Length field lengths shall be defined by the Registry Designator according to Table 5.

Table 5 (Normative) - Coding of Registry Designator (Byte 6) for Local Set Syntax

Byte 6 Value	Length field	Local Tag field	Description
0x03	ASN.1 BER short or long	1 byte	Any length
0x0B	ASN.1 BER short or long	ASN.1 OID BER	
0x13	ASN.1 BER short or long	2 bytes	
0x1B	ASN.1 BER short or long	4 bytes	
0x23	1 byte	1 byte	Length up to 255
0x2B	1 byte	ASN.1 OID BER	
0x33	1 byte	2 bytes	
0x3B	1 byte	4 bytes	
0x43	2 bytes	1 byte	Length up to 65535
0x4B	2 bytes	ASN.1 OID BER	
0x53	2 bytes	2 bytes	
0x5B	2 bytes	4 bytes	
0x63	4 bytes	1 byte	Length up to 2^{32}-1
0x6B	4 bytes	ASN.1 OID BER	
0x73	4 bytes	2 bytes	
0x7B	4 bytes	4 bytes	

NOTE – ASN.1 OID BER coding is also known as “primitive BER” coding and differs from the ASN.1 BER coding used for the length field. A description of the use of both forms of BER coding is provided in EG 377-3 Section 4.3

A relevant Local Set ST / RP shall define the link between the Local Tag of each data item and the corresponding Key value. This link shall be defined in a relevant Local Set ST / RP that provides, for each Local Tag, the Key of the defining item. This linking definition is a mechanism that gives users of this Standard the flexibility to define their own aliases for highly efficient coding. The relevant Local Set ST / RP shall also define the intended scope of applicability of the local tag or alias within the specification. Developers of Local Sets shall provide the mapping between each Tag in a Local Set and the defining Key. Unlike Universal Sets and Global Sets where the Key of each data item in the set can be losslessly recreated, the Key of each Local Set Tag cannot be reconstructed without recourse to the defining ST / RP and the corresponding Register.

Local Sets can accommodate recursion, so that the Key linked to a Local Tag may identify either a single data item from a register or a data set or pack from a Set or Pack ST / RP and the corresponding Register.

Figure 6 is an informative illustration of the linking between a Local Tag and a full Key.
An example of the operation of Local Sets is given in Annex E

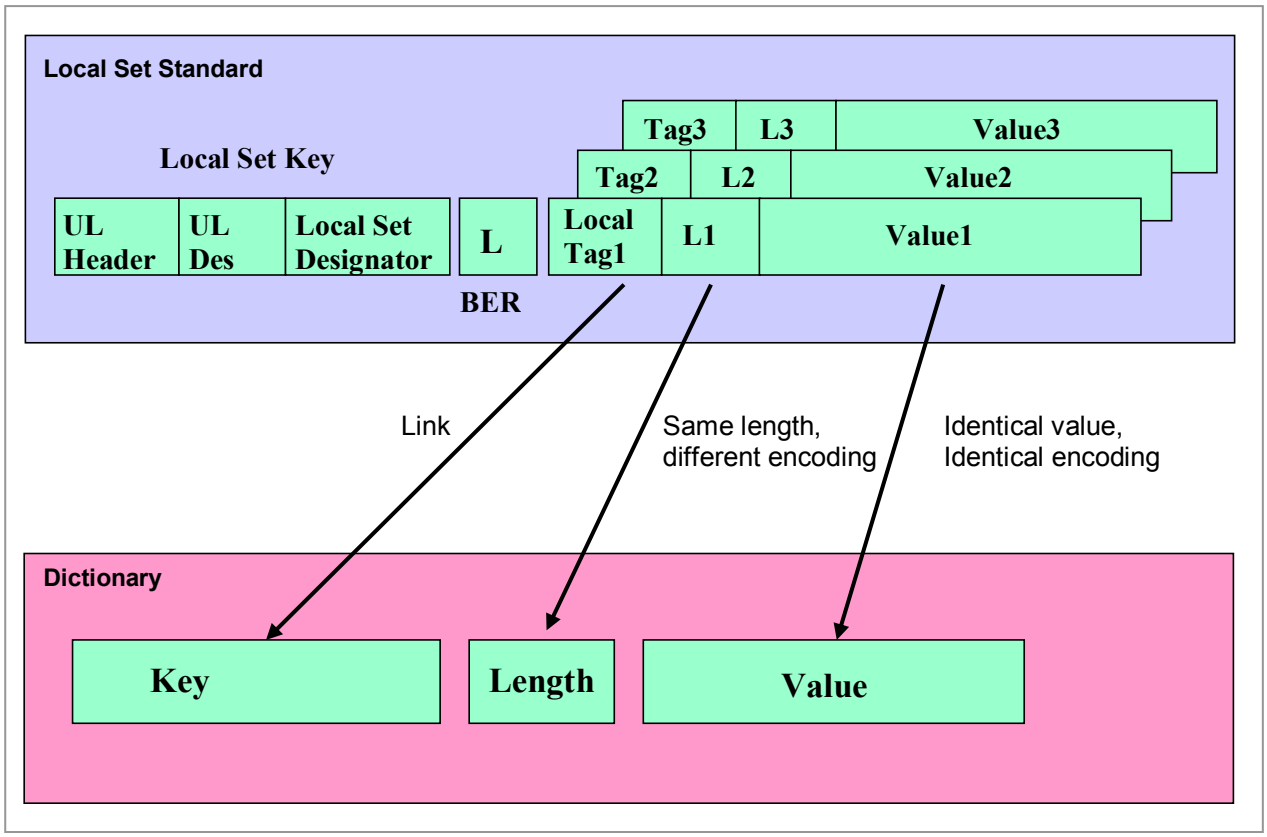


Figure 6 – Informative Illustration of Local Set Label-to-Global Key Linking

7.5 Variable-Length Packs

A Variable-Length Pack is similar to a Local Set but does not have Local Tags. Thus each item of a Variable-length Pack comprises only a Length field and a Value field. Items in a Variable-Length Pack must appear in the defined order.

The use of Keys for Variable-Length Pack coding shall be defined by an accompanying ST / RP including a Structure Designator and an accompanying Register including a version number.

The Key of a Variable-Length Pack is 16 bytes in length.

The Length of a Variable-Length Pack is coded by default as per ASN.1 BER length notation; long or short form as required.

The Value of a Variable-Length Pack shall be a sequence of KLV-encoded items whose total length is given by the Length field.

The Variable-Length Pack Designator is defined within the last 8 bytes in the Variable-Length Pack Key. It defines the variable length pack placement in a hierarchical structure.

Variable-Length Pack Keys shall be defined in an associated ST / RP and the Key value shall be registered with the SMPTE Registration Authority in accordance with the provisions of the accompanying ST / RP to guarantee a unique Key value.

The data structure for the encoding of Variable-Length Packs is illustrated by Figure 7.

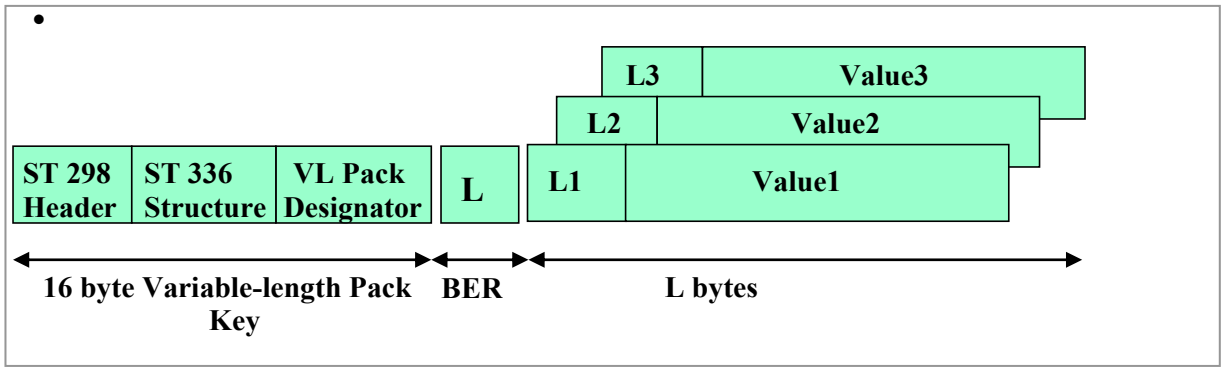


Figure 7 - KLV Coded Variable-Length Pack Structure

An example of the operation of Variable-Length Packs is given in Annex F

The 16-byte Variable-Length Pack Key is followed by the Variable-Length Pack Length (encoded using ASN.1 BER length encoding), which is followed by a number of items which shall each be pairs of Length and Value.

When non-BER encoding is used it shall be big-endian byte ordered. The full range of allowed Length field lengths shall be defined by the Registry Designator according to Table 6.

Table 6 (Normative) - Coding of Registry Designator (Byte 6) for Variable Length Pack Syntax

Byte 6 Value	Length fields	Description
0x04	ASN.1 BER short or long	Any length
0x24	1 byte	Length up to 255
0x44	2 bytes	Length up to 65535
0x64	4 bytes	Length up to 2^{32}-1

Because the items within a pack do not have a local Tag, the order of the items shall be specified by the defining ST / RP.

The relevant Variable-Length Pack ST / RP shall define the link between each data item and the corresponding Key value by providing the Key of each item in the Variable-Length Pack. This linking definition is a mechanism that gives users of this Standard the flexibility to define their own aliases for highly efficient coding. Developers of Variable-Length Packs shall register the mapping between each item in a Variable-Length Pack and the defining Key. Unlike Universal Sets and Global Sets where the Key of each item in the set can be losslessly recreated, the Key of each item in a Variable-Length Pack cannot be reconstructed without recourse to the defining ST / RP and the corresponding Register.

Variable-Length Packs can accommodate recursion, so that the Key linked to an item may identify either a single data item from a register or a data group from a Set or Pack ST / RP and the corresponding Register.

7.6 Defined-Length Packs

NOTE - The term Fixed-Length Pack has been replaced by Defined-Length Pack in a previous version of the Standard.

A Defined-Length Pack is similar to a Variable-Length Pack but does not have Length fields. Thus each item of a Defined-Length Pack comprises only a Value field. Items in a Defined-Length Pack shall appear in the defined order.

The use of Keys for Defined-Length Pack coding shall be defined by an accompanying ST / RP including a Structure Designator and an accompanying Register including a Version number.

The Key of a Defined-Length Pack is 16 bytes in length.

The Length of a Defined-Length Pack is coded as per ASN.1 BER length notation; long or short form as required.

The Value of a Defined-Length Pack shall be a sequence of items whose total length is given by the Length field.

Individual items in a Defined-Length Pack occur in a defined order, and each item has a defined length. Individual items within the pack may have length values which need to be determined by parsing the item, thus resulting in a pack with a defined yet variable overall length. There is no requirement in this standard for Defined-Length Packs to have fixed, constant length values.

The Defined-Length Pack Designator is defined within the last 8 bytes in the Defined-Length Pack Key. It defines the Defined-Length pack placement in a hierarchical structure. Defined-Length Pack Keys shall be defined in an associated ST / RP and the Key value shall be registered with the SMPTE Registration Authority in accordance with the provisions of the accompanying ST / RP to guarantee a unique Key value.

The data structure for the encoding of Defined-Length Packs is illustrated by Figure 8.
An example of the operation of Defined-length Packs is given in Annex G .

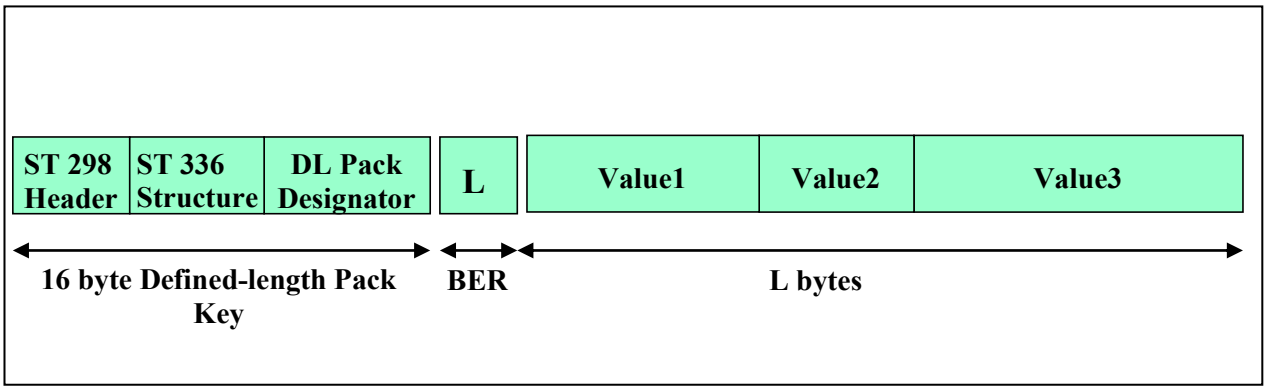


Figure 8 – KLV coded Defined-Length Pack Structure

Because the items within a Defined-Length Pack do not have a Local Tag, the order of the items shall be specified by the defining ST / RP.

A relevant Defined-Length Pack ST / RP shall define the link between each data item and the corresponding Key value by providing the Key of the defining item. This linking definition is a mechanism that gives users of this Standard the flexibility to define their own aliases for highly efficient coding. Developers of Defined-Length Packs shall register the mapping between each item in a Defined-Length Pack and the defining Key. Unlike Universal Sets and Global Sets where the Key of each item in the set can be losslessly recreated, the Key of each item in a Defined-Length Pack cannot be reconstructed without recourse to the defining ST / RP and the corresponding Register.

Defined-Length Packs can accommodate recursion, so that the Key linked to an item may identify either a single data item from a register or a data group from a Set or Pack ST / RP and the corresponding Register.

NOTE – In many cases, groups can be coded as Universal Sets, Local Sets, Variable Length Packs and Defined-Length Packs without changing the values of the individual metadata items within the group. In each case, for a given group, only byte 6 of the group key will change and bytes 9 to 16 would remain the same.

8 Wrappers and Containers

Wrappers and containers shall be identified by byte 5 having the value of 0x03.

Wrappers and Containers differ from Sets and Packs in that they do not necessarily employ an overall KLV data construct for the entire contents of the Wrapper or Container. It is recommended that individual parts of a Wrapper or Container encode data using the KLV coding protocol, but these parts may be bound together by other techniques. In some cases, a Wrapper or Container may employ an overall KLV construct in certain applications (such as a streaming interface) but employ another technique in other applications (such as a storage container). In these cases, the Wrapper or Container is not redefined as a Set or a Pack but retains the definition as a Container or Wrapper for consistency of identification.

Simple Wrappers and Containers are defined as embedding all the data into a single framework with no external references. Simple Wrappers and Containers shall be identified by byte 6 having the value 0x01.

Complex Wrappers and Containers are defined by frameworks where individual data items may be included in a file by reference rather than embedding. Complex Containers and Wrappers can be more efficient and are suited to local environments where references can be easily resolved. Complex Wrappers and Containers shall be identified by byte 6 having the value 0x02.

The definition of individual Wrapper and Container specifications is outside the scope of this standard and will be found in other ST / RPs.

9 SMPTE Labels

SMPTE Labels shall be identified by byte 5 having the value of 0x04. SMPTE Labels shall not be used as the Key in KLV coding. SMPTE Labels may be used as a value within a KLV coding triplet or within any other coding construct.

SMPTE Labels shall be used to identify any object whose meaning is entirely conveyed by the SMPTE Administered UL itself. SMPTE Labels can be used to identify essence coding schemes, provide unique identification of parametric values, identify metadata constructs and more.

Within Wrappers and Containers and sometimes even in Sets, there is sometimes the need to identify aspects of the data contents which are not identified by the Set, Wrapper or Container Key. Such an aspect can be identified by including a SMPTE Label in the Set, Wrapper or Container as a data item. It is necessary to define the presence of a SMPTE Label at a high level in the SMPTE UL structure so that decoders are aware that the item is a SMPTE Label and not the Key of a KLV coded triplet.

The Label Designator is defined within the last 8 bytes in the Label Key. It defines the Label placement in a hierarchical structure. Label Keys shall be defined in an associated ST / RP and the Key value shall be registered with the SMPTE Registration Authority in accordance with the provisions of the accompanying ST / RP to guarantee a unique Key value.

A SMPTE Label is illustrated in Figure 9.
An example of the operation of Labels is given in Annex H

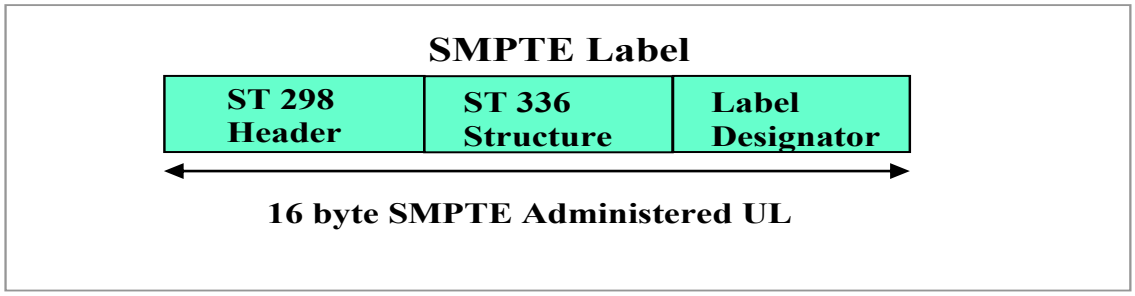


Figure 9 – SMPTE Administered UL for SMPTE Labels

10 Registered Private Information

Registered private information shall be identified with the Registry Category field of the SMPTE administered UL set to 0x05. The purpose of this category is to provide a standard, unambiguous means to carry information that is registered with an external agency, and where that information is not desired to be publicly registered in a SMPTE register, or not desired to be qualified as either metadata or essence.

Refer to SMPTE RP 225 for the authoritative definition of the remainder of the UL fields when the Registry Category Designator is set to 0x05.

This registered private information does not define a SMPTE administered UL as either a Key or a SMPTE Label. This Category does not refer to “Class 13” or “Class 14” metadata – these classes are defined in the controlling standards for metadata elements, types, groups, labels, essence.

Annex A Example Usage of the SMPTE UL (Informative)

Table A.1 – Expanded Example of SMPTE UL Fields for Metadata Encoding

Byte No.	Value (hex)	Example Explanation	Reference
1	0x06	UL Object Identifier	SMPTE ST 298
2	0x0E	UL Size	SMPTE ST 298
3	0x2B	Concatenation of Designators ISO, ORG	SMPTE ST 298
4	0x34	SMPTE Designator	SMPTE ST 298
5	0x01	SMPTE Dictionaries Designator	SMPTE ST 336
6	0x01	Metadata Register Designator	SMPTE ST 336
7	0x01	Structure Standard Reference No.	SMPTE ST 336
8	0x01	Standard Version Number	SMPTE ST 336
9	0x01	Metadata Class: Identifiers & Locators	SMPTE ST 335 and associated register
10	0x01	Globally Unique Identifiers	SMPTE ST 335 and associated register
11	0x11	ISO Identifiers	SMPTE ST 335 and associated register
12	0x01	ISO Audio-Visual Number (ISAN)	SMPTE ST 335 and associated register
13	0x00	Unused	SMPTE ST 335 and associated register
14	0x00	Unused	SMPTE ST 335 and associated register
15	0x00	Unused	SMPTE ST 335 and associated register
16	0x00	Unused	SMPTE ST 335 and associated register

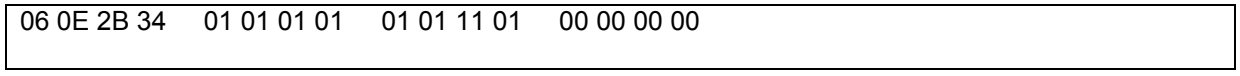


Figure A.1 – Example SMPTE UL Fields for Metadata Encoding

NOTE – Hex-encoded bytes; 0x removed and bytes separated for improved readability.

Annex B Example of the KLV Encoding of a Single Metadata Item (Informative)

Table B.1 shows an example of the fields that comprise the KLV protocol used for an individual Value encoding of a human-assigned Main Title for a video segment. NOTE – Hex-encoded bytes; 0x removed and bytes separated for improved readability.

Table B.1 – Example of KLV Individual Value Encoding of Metadata

Key (Hex encoded)	06 0E 2B 34 01 01 01 01 01 05 02 00 00 00 00 00
Description	Main Title (ISO 7-bit char)
Length (Binary) [Hex]	0x10
Value (7-bit char)	Yesterdays World

Annex C Example of a Universal Set (Informative)

In the example Universal Metadata Set in Table C.1 below, the three items of Main Title, Version Title, and Source Organization can be in any order and each entry is self-contained with its own individual Universal KLV. Similarly, if one or more of the items of the defined Universal Set is missing, the remaining, valid parts of the Universal Set can still be recovered because each has its own KLV combination.

NOTE – Hex-encoded bytes; 0x removed and bytes separated for improved readability.

Table C.1 – Example of KLV Universal Set Encoding of Metadata

Universal Set UL	06 0E 2B 34 02 01 01 01 0F 01 01 01 00 00 00 00
Universal Set Length	0x5E
Universal Key 1	06 0E 2B 34 01 01 01 01 01 05 02 00 00 00 00 00
Description	Main Title (ISO 7-bit char)
Length 1	0x10
Value 1	Yesterdays World
Universal Key 2	06 0E 2B 34 01 01 01 02 01 05 08 00 00 00 00 00
Description	Version Title (ISO 7-bit char)
Length 2	0x15
Value 2	International Edition
Universal Key 3	06 0E 2B 34 01 01 01 01 02 01 01 00 00 00 00 00
Description	Source Organization (ISO 7-bit char)
Length 3	0x06
Value 3	WXYZ15

Annex D Example of a Global Set (Informative)

In the example Global Metadata Set in Table D.1 below, the three items of Main Title, Version Title, and Source Organization can be in any order and each entry is self-contained with its own individual Global Tag-Length-Value. Similarly, if one or more of the items of the defined Global Set is missing, the remaining, valid parts of the Global Set can still be recovered because each has its own Tag-Length-Value combination.

NOTE – Hex-encoded bytes; 0x removed and bytes separated for improved readability.

Table D.1 –Example of KLV Global Set Encoding of Metadata

Global Set UL	06 0E 2B 34 02 02 08 01 06 0E 2B 34 01 01 01 00
Global Set Length	0x3D
Global Tag 1	01 01 05 02 00
Original Key	06 0E 2B 34 01 01 01 01 01 05 02 00 00 00 00 00
Description	Main Title (ISO 7-bit char)
Length 1	0x10
Value 1	Yesterdays World
Global Tag 2	02 01 05 08 00
Original Key	06 0E 2B 34 01 01 01 02 01 05 08 00 00 00 00 00
Description	Version Title (ISO 7-bit char)
Length 2	0x15
Value 2	International Edition
Global Tag 3	01 02 01 01 00
Original Key	06 0E 2B 34 01 01 01 01 02 01 01 00 00 00 00 00
Description	Source Organization (ISO 7-bit char)
Length 3	0x06
Value 3	WXYZ15

Annex E Example of a Local Set (Informative)

In the example Local Metadata Set in Table E.1 below, the three items of Main Title, Version Title, and Source Organization can be in any order and each entry is self-contained with its own individual Local Tag-Length-Value. Similarly, if one or more of the items of the defined Local Set is missing, the remaining, valid parts of the Local Set can still be recovered because each has its own Local Tag-Length-Value combination.

NOTE – Hex-encoded bytes; 0x removed and bytes separated for improved readability.

Table E.1 – Example of KLV Local Set Encoding of Metadata

Local Set UL	06 0E 2B 34 02 03 01 01 0F 01 01 01 00 00 00 00
Local Set Length	0x31
Local Tag 1	0x01
Original Key	Specified by defining Standard or RP
Description	Main Title (ISO 7-bit char)
Length 1	0x10
Value 1	Yesterdays World
Local Tag 2	0x02
Original Key	Specified by defining Standard or RP
Description	Version Title (ISO 7-bit char)
Length 2	0x15
Value 2	International Edition
Local Tag 3	0x03
Original Key	Specified by defining Standard or RP
Description	Source Organization (ISO 7-bit char)
Length 3	0x06
Value 3	WXYZ15

Annex F Example of a Variable-Length Pack (Informative)

In the example Variable-Length Pack in Table F.1 below, the three items of Main Title, Version Title, and Source Organization have to be in the order specified and all be present. If one or more of the items of the defined Variable-Length Pack is missing, the remaining, valid parts of the Variable-Length Pack cannot be recovered.

NOTE – Hex-encoded bytes; 0x removed and bytes separated for improved readability.

Table F.1 –Example of KLV Variable-Length Pack Encoding of Metadata

VL Pack UL	06 0E 2B 34 02 04 01 01 0F 01 01 01 00 00 00 00
VL Pack Length	0x2E
Original Key	Specified by defining Standard or RP
Description	Main Title (ISO 7-bit char)
Length 1	0x10
Value 1	Yesterdays World
Original Key	Specified by defining Standard or RP
Description	Version Title (ISO 7-bit char)
Length 2	0x15
Value 2	International Edition
Original Key	Specified by defining Standard or RP
Description	Source Organization (ISO 7-bit char)
Length 3	0x06
Value 3	WXYZ15

Annex G Example of a Defined-Length Pack (Informative)

In the example Defined-Length Pack in Table G.1 below, the two items of Copy Count, and Main Title have to be in the order and of the length specified. If one or more of the items of the defined Defined-Length Pack is missing, the remaining, valid parts of the Defined-Length Pack cannot be recovered. The Main Title value is a variable length text string whose end is determined by the overall pack length value. If a variable length value is followed by another value, a mechanism must be defined to identify the end of the variable length value; for example, specifying that it is a zero-terminated string.

NOTE – Hex-encoded bytes; 0x removed and bytes separated for improved readability.

Table G.1 –Example of KLV Defined-Length Pack Encoding of Metadata

DL Pack UL	06 0E 2B 34 02 05 01 01 0F 01 01 02 00 00 00 00
DL Pack Length	0x11
Original Key	Specified by defining ST / RP
Description	Copy Count (UInt8)
Value 2	0x03
Original Key	Specified by defining ST / RP
Description	Main Title (ISO 7-bit char)
Value 3	Yesterdays World

Annex H Example of a Label (Informative)

Label UL	06 0E 2B 34 04 01 01 01 04 01 01 01 01 01 00 00
Description	ITU-R BT470 Transfer Characteristic

NOTE – Hex-encoded bytes; 0x removed and bytes separated for improved readability.

Annex I ASN.1 BER Length Coding (Informative)

NOTE – The term “byte” used in this document is synonymous with the term “octet” used in ISO/IEC 8825-1.

The following section is quoted from ISO/IEC 8825-1:

“8.1.3.3 For the definite form, the length octets shall consist of one or more octets, and shall represent the number of octets in the contents octets using either the short form (see 8.1.3.4) or the long form (see 8.1.3.5) as a sender’s option.

NOTE – The short form can only be used if the number of octets in the contents octets is less than or equal to 127.”

“8.1.3.4 In the short form, the length octets shall consist of a single octet in which bit 8 is zero and bits 7 to 1 encode the number of octets in the contents [Value] octets (which may be zero), as an unsigned binary integer with bit 7 as the most significant bit.

Example

L=38 can be encoded as 00100110_2”

“8.1.3.5 In the long form, the length octets shall consist of an initial octet and one or more subsequent octets. The initial octet shall be encoded as follows:

a) bit 8 shall be one;

b) bits 7 to 1 shall encode the number of subsequent octets in the length octets, as an unsigned binary integer with bit 7 as the most significant bit;

c) the value 11111111_2 shall not be used.

NOTE – This restriction is introduced for possible future extensions

Bits 8 to 1 of the first subsequent byte, followed by bits 8 to 1 of the second subsequent byte, followed in turn by bits 8 to 1 of each further byte up to and including the last subsequent byte, shall be the encoding of an unsigned binary integer equal to the number of bytes in the Value field, with bit 8 of the first subsequent byte as the most significant bit.

NOTE – This is sometimes known as ‘big-endian byte order’.

Example

L=201 can be encoded as:

Byte 1 = 10000001_2, Byte 2 = 11001001_2 [b7 …….b0]

Annex J ASN.1 BER Encoding of an Object Identifier Value (Informative)

NOTE – The term “byte” used in this document is synonymous with the term “octet” used in ISO/IEC 8825-1.

The following section is quoted from ISO/IEC 8825-1:

8.19 Encoding of an object identifier value

8.19.1 The encoding of an object identifier value shall be primitive.

8.19.2 The contents octets shall be an (ordered) list of encodings of subidentifiers (see 8.19.3 and 8.19.4) concatenated together.

Each subidentifier is represented as a series of (one or more) octets. Bit 8 of each octet indicates whether it is the last in the series: bit 8 of the last octet is zero; bit 8 of each preceding octet is one. Bits 7 to 1 of the octets in the series collectively encode the subidentifier. Conceptually, these groups of bits are concatenated to form an unsigned binary number whose most significant bit is bit 7 of the first octet and whose least significant bit is bit 1 of the last octet. The subidentifier shall be encoded in the fewest possible octets, that is, the leading octet of the subidentifier shall not have the value 80_{16}.

8.19.3 The number of subidentifiers (N) shall be one less than the number of object identifier components in the object identifier value being encoded.

8.19.4 The numerical value of the first subidentifier is derived from the values of the first *two* object identifier components in the object identifier value being encoded, using the formula:

$$(X*40) + Y$$

where X is the value of the first object identifier component and Y is the value of the second object identifier component.

NOTE – This packing of the first two object identifier components recognizes that only three values are allocated from the root node, and at most 39 subsequent values from nodes reached by X = 0 and X = 1.

The numerical value of the i*th* subidentifier, (2 ≤ i ≤ N) is that of the (i + 1)*th* object identifier component.

EXAMPLE

An **OBJECT IDENTIFIER** value of:

{joint-iso-itu-t 100 3}

which is the same as:

{2 100 3}

has a first subidentifier of 180 and a second subidentifier of 3. The resulting encoding is:

OBJECT IDENTIFIER	Length	Contents
06_{16}	03_{16}	813403_{16}

Bibliography (Informative)

Standards defining ST 336 Key allocations:

SMPTE ST 335:2012 - Metadata Element Dictionary Structure
SMPTE ST 395:2014 - Metadata Groups Register
SMPTE ST 400:2012 – SMPTE Labels Structure
SMPTE RP 225:2005 - Registered Private Information in KLV
SMPTE ST 2003:2012 - Types Dictionary Structure
SMPTE ST 2088:201x - Essence Element Key Register Structure

Related topics:

ISO/IEC 8824-1:2008 (ITU-T X.690), Information Technology – Abstract Syntax Notation One (ASN.1) – Specification of Basic Notation
SMPTE/EBU “Task Force for Harmonized Standards for the Exchange of Program Material as Bitstreams” (TFHS) 1997-1998
SMPTE ST 377-1:2011 - Material Exchange Format (MXF) — File Format Specification
SMPTE ST 379-1:2009 - Material Exchange Format (MXF) — MXF Generic Container
SMPTE ST 379-2:2010 - Material Exchange Format (MXF) — Constrained MXF Generic Container
SMPTE ST 380:2004 - Television — Material Exchange Format (MXF) — Descriptive Metadata Scheme-1
SMPTE EG 377-3-2013, Material Exchange Format (MXF) — Engineering Guideline