2025-01-17

Withdrawal of SMPTE ST 2045

A document should be Withdrawn only if there is a significant possibility of its use causing harm. A Withdrawn document shall still be made available and offered for sale by the Society, but it shall be prefaced by a cover page explaining its current status including a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

This Standard has been withdrawn and its content is no longer endorsed by the Society. This action has been taken because it is judged that there is a significant possibility that use of the document may cause harm. SMPTE ST 2045 Register Interchange Format was an early attempt (pre-2010) to define an XML representation for UL based registers defined by SMPTE but was ultimately not used by SMPTE to publish metadata registers as XML - a separate approach was developed much later which is the format that SMPTE has used for many years with wide support. Readers of this document are cautioned that the information contained in the document is not in compliance with extant practice, process, and tooling for XML publication of SMPTE UL registers at SMPTE-RA.org.

SMPTE STANDARD SMPTE

Register Interchange Format

Table of Contents

Approved
September 3, 2010

Foreword

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

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

SMPTE ST 2045 was prepared by Technology Committee 30MR.

Intellectual Property

At the time of publication no notice had been received by SMPTE claiming patent rights essential to the implementation of this Standard. 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 Introduction (Informative)

This standard defines a common class model and XML encoding for the UL-based registers defined by SMPTE. The corresponding defining documents of these registers need to specify rules about how to extend the common class model to support the specific registers requirements.

At the time of writing this document the defining documents of the UL-based registers are under revision. Future releases of the RIF document family will further specify the encodings rules for each SMPTE UL-based register for the RIF representation according to this standard.

Requirements for register entries, both leaves and nodes, are contained in the defining document for these registers. This standard defines the common administrative and identification metadata required to manage the contents of the UL registers.

This standard starts with an overview of the generic common class model. Sections 6.2 and 6.3 define the scalar data types and the compound data types respectively. Section 6.4 defines the class identifiers, followed by Section 6.5 where all RIF classes are specified including their properties. The specification how to encode the contents of registers as XML documents is given in Section 7. The rules defining how to extend the data model to support specific registers can be found in Section 8. Annex A provides a base XML schema definition. In Annex B and Annex C informative examples are given which illustrate the support for SMPTE 395M and SMPTE 335M.

2 Scope

This standard specifies a generic data model to represent the contents of the UL-based SMPTE registers. Examples of such registers are SMPTE 335M, SMPTE 395M or SMPTE 400M.

This generic data model also provides elements to hold administrative information used for the maintenance of the UL-based SMPTE registers. It includes rules how to extend the data model to represent the contents of specific UL-based SMPTE registers.

This standard also specifies how to encode the contents of individual registers as XML documents.

This standard does not define the relationships between the different registers. The data model is limited to defining the attributes that can be used by different SMPTE registers.

3 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.

4 Normative References

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

IETF RFC 3061, A URN Namespace of Object Identifiers

IETF RFC 4122, A Universally Unique IDentifier (UUID) URN Namespace

ISO/IEC 10646:2003, Information technology – Universal Multiple-Octet Coded Character Set (UCS)

SMPTE 298M-2009, Universal Labels for Unique Identification of Digital Data

SMPTE 335M-2001, Metadata Element Dictionary Structure

SMPTE 336M-2007, Data Encoding Protocol Using Key-Length-Value

SMPTE 395M-2003, Metadata Groups Registry

SMPTE 400M-2004, SMPTE Labels Structure

SMPTE 2029-2009, Uniform Resource Names for SMPTE Resources

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

Namespaces in XML, W3C Recommendation

XML Schema Part 1: Structures, W3C Recommendation

XML Schema Part 2: Datatypes, W3C Recommendation

5 Definition of Acronyms and Terms

Acronyms, terms and data types used in the RIF specification are defined in this section.

5.1 Acronyms and Terms

Abstract Class: A Class used within the definition of an inheritance hierarchy that cannot be instantiated as an Object. Only non-abstract Subclasses of an Abstract Class can be instantiated as Objects.

Aggregation: A form of Association that defines a whole-part relationship between a whole Object and a part Object or multiple part Objects. See Composition.

Class: A category of Objects with defined properties and defined behavior.

Composition: A stronger form of Aggregation in which parts may only belong to one whole and may only exist as part of the whole. Composition is used to combine simple Objects into more complex ones.

KLV: Key Length Value. The binary encoding protocol defined in SMPTE 336M.

Object: An instance of a Class.

Reference: A relationship between Objects (e.g. StrongRef).

RIF: Register Interchange Format.

RIF Base XML-Schema: The core part of the RIF XML-Schema which is provided in Annex A This XML-Schema has to be extended in order to support specific UL-based SMPTE registers.

Set: A grouping of KLV items defined in SMPTE336M.

StrongRef: Strong Reference. A Composition relationship between Objects implemented as an aggregation by reference between Objects using identifiers (e.g. the value of an InstanceUID). Strong References are a one to one relationship. They are typed. This means that the definition of the aggregation identifies the kinds of Objects (encoded using KLV or XML encoding) which may be the target of the reference.

Subclass: A Class that is directly or indirectly derived through Inheritance from another Class. The other Class is also called a Superclass of the Subclass.

UL: SMPTE Universal Label (see SMPTE 298M).

UL Item Designator: The last 8 bytes of a SMPTE Universal Label. They uniquely identify an entry in a SMPTE register. The Item Designator is a reference to the SMPTE label inside the register that is identified by the UL Designator. SMPTE registers relevant to this specification are SMPTE 335M, SMPTE 400M and SMPTE 395M.

UL Designator: The first 8 bytes of a SMPTE Universal Label. They identify the register, its minor and major versions, and may also convey other information that is used by the KLV encoding protocol. The register is the normative reference for the values of these 8 bytes.

UUID: Universally Unique Identifier according to IETF RFC 4122.

6 Generic Class Model

6.1 **Overview** (Informative)

SMPTE 336M defines the elements that are needed to exactly define entries in all Universal Label-based SMPTE registers. The upper part of the UML class diagram in Figure 1 illustrates this common base which includes the definition of the abstract classes “RifBaseClass”, “AdministrationBaseClass”, “RegisterAdministrationBaseClass”, “EntryAdministrationBaseClass”, “Entry”, “Leaf” and CompoundEntryElementBaseClass. Also it includes the definition of the concrete classes “Register”, “Node” and ApplicationInformation.

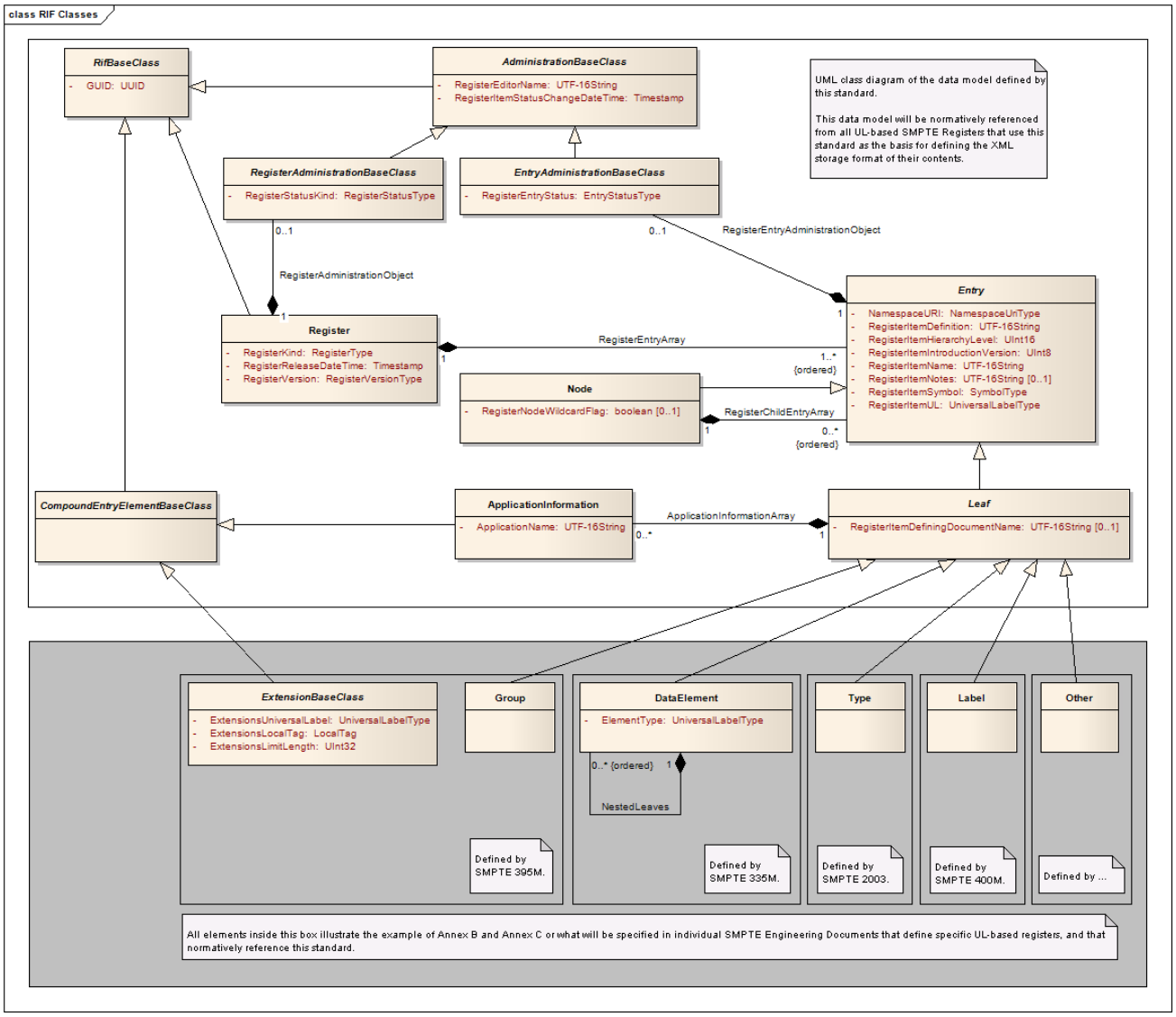

Figure 1 – UML class diagram of the common class model

Each individual Register extends the abstract base class “Leaf” to represent its own, register-specific information. This is illustrated in the lower part of Figure 1.

6.2 Scalar Data Types

This section defines the scalar data types used by elements of the RIF classes. The corresponding XML encodings of the type values are defined in Section 7.2.

Table 1 – Scalar Data Types

Data Type	Definition
Boolean	*A logical 'true' or 'false'.*
Byte	*A unit of information storage.*
EntryStatusType	*Enumerated value. The approval status of an entry.*
Int16	*Signed 16-bit integer.* *Note: An integer in the range -32768 to 32767.*
LocalTag	*Local Tag as defined in SMPTE 336M.*
NamespaceURIType	*String type that conforms to XML language syntax.*
RegisterStatusType	*Enumerated value. The published status of a register.*
RegisterType	*Enumerated value. The type of a register.*
StrongRef	*Strong Reference. A Composition relationship between Objects implemented as an aggregation by reference between Objects using UUIDs as identifiers for sub-classes of RifBaseClass and using UL for sub-classes of Entry. Strong References are a one to one relationship. They are typed. This means that the definition of the aggregation identifies the kinds of Objects which may be the target of the reference.*
SymbolType	*A symbol is a name that conforms to computer language syntax restrictions, and it is intended for use in computer languages such as the Extensible Markup Language (XML). To enable the use of symbols in a wide range of computer languages, a symbol shall be a string composed only of the characters A-Z, a-z, 0-9, and _, and it shall begin with an alpha character (A-Z, a-z) or an underscore (_).*
UInt8	*Unsigned 8-bit integer.* *Note: An integer in the range 0 to 255.*
UInt16	*Unsigned 16-bit integer.* *Note: An integer in the range 0 to 65535.*
UInt32	*Unsigned 32-bit integer.* *Note: An integer in the range 0 to 4294967295.*
UnitsOfMeasureType	*Enumerated value. A list of the units of measure in order to ensure consistency. E.g. "meters", "seconds", "degree Celsius".*
UniversalLabel	*SMPTE Universal Label as defined in SMPTE 298M.*
UTF-16String	*A variable-length string that is able to hold any valid sequence of Unicdode symbols.* *Note: The encoding is defined in ISO/IEC 10646:2003, Annex Q. RFC-2781 also contains a description of the encoding.*
UUID	*Universally unique identifier according to ITEF RFC 4122.*

6.3 Compound Data Types

This section defines the compound data types used by elements of the RIF classes. The corresponding XML encodings of the type values are defined in Section 7.3.

Table 2 – Compound Data Types

Data Type	Definition
Array	*A compound type comprising multiple individual elements where the elements are ordered and the type is defined. The order of the individual Array elements in the Array is significant. An Array is identified as an Array of Type, where Type is the defined type of the elements of the Array.*
Batch	*A compound type comprising multiple individual elements where the elements are unordered and unique and the type is defined. The order of the individual Batch elements in a Batch is not significant. RIF encoders shall not place duplicate elements into a Batch. RIF decoders shall be able to successfully decode RIF instances which do contain Batches with repeated Batch element values. A Batch is identified as a Batch of Type, where Type is the defined type of the elements of the Batch.*
Timestamp	*A time and date item according to the Gregorian calendar comprising Year: [Int16], Month: [UInt8], Day: [UInt8], Hour: [UInt8], Minute: [UInt8], Second: [UInt8] and mSec/4: [UInt8]. A value of '0' for every field identifies a timestamp value of 'unknown'. This value should not be used unless unavoidable. Timestamp values shall be specified according to UTC time.*
RegisterVersionType	*A compound data type comprising the major version number [UInt8] and the minor version number [UInt8] of the register.*

6.4 Class UL Definitions

Class IDs shall be Universal Labels. All RIF classes shall be registered in the SMPTE Groups register. The following tables specify the Universal Label values for the classes that are defined in this document.

Table 3 – Common Universal Label for all RIF classes

Byte No.	Description	Value (hex)	Meaning
1	*Object Identifier*	*06h*	
2	*Label size*	*0Eh*	
3	*Designator*	*2Bh*	*ISO, ORG*
4	*Designator*	*34h*	*SMPTE*
5	*Register Category Designator*	*02h*	*Sets & packs*
6	*Register Designator:*	*7fh*	*SMPTE 395M-defined default value for undefined KLV codings.*
7	*Structure Designator*	*01h*	*Set/Pack Dictionary*
8	*Version Number*	*01h*	*Register Version 1*
9	*Item Designator*	*0Ch*	*Compound*
10	*Item Designator*	*01h*	*Data Models*
11	*Item Designator*	*01h*	*Dictionary Representation*
12	*Application*	*01h*	*Register Interchange Format (RIF)*
13	*Structure version*	*01h*	*RIF Version 1 compatible classes*
14	*Set Kind (1)*	*xxh*	*RIF class Definition (see)*
15	*Set Kind (2)*	*yyh*	*RIF class Definition (see)*
16	*Reserved*	*00h*	*Reserved*

The definition of bytes 14 and 15 of the Universal Labels for all classes is given in .

Table 4 – Universal Label Byte 14 and Byte 15 values for all RIF classes

Class Name	Byte 14	Byte 15
AdministrativeBaseClass	*01h*	*01h*
CompoundEntryElementBaseClass	*01h*	*02h*
Entry	*01h*	*03h*
EntryAdministration	*01h*	*04h*
Leaf	*01h*	*05h*
Node	*01h*	*06h*
Register	*01h*	*07h*
RegisterAdministration	*01h*	*08h*
RifBaseClass	*01h*	*09h*

6.5 RIF Class Definitions

This section defines the common classes that are shared between all UL-based SMPTE registers.

For all items of each class, the class definition tables normatively define the item name, its type, the Universal Label (UL), the required status and the definition of the item. The required status can be either required (req) or optional (opt).

6.6 RIF Base Class

Table 5 – RIF Base Class

Item Name	Type	UL	Symbol	Req ?	Definition
GUID	*UUID*	*06.0E.2B.34. 01.01.01.01. 01.01.15.02. 00.00.00.00*	*GUID*	*Req*	*Unique identifier of this instance.*

This shall be an abstract class.

This class shall use the symbol “RIFBaseClass”.

6.6.1 Register

Table 6 – Register

Item Name	Type	UL	Symbol	Req ?	Definition
All items in 6.6	*See 6.6*	*See 6.6*	*See 6.6*	*See 6.6*	*See 6.6*
Register Release Date Time	*Timestamp*	*06.0E.2B.34.01.01.01.0C.07.02.01.01.01.06.00.00*	*RegisterReleaseDateTime*	*Req*	*The release (publication) date of the register.*
Register Kind	*RegisterType*	*06.0E.2B.34.01.01.01.0C.02.10.02.02.01.00.00.00*	*RegisterKind*	*Req*	*The kind of the register.*
Register Version	*RegisterVersionType*	*06.0E.2B.34.01.01.01.0C.02.10.02.02.02.00.00.00*	*RegisterVersion*	*Req*	*The version of the register.*
Register Entry Array	*StrongRefArray (Entry)*	*06.0E.2B.34.01.01.01.0C.06.01.01.04.06.11.00.00*	*RegisterEntryArray*	*Req*	*Array of strong references to Entry objects. The sequence of entries shall be according to ascending order of their UL Item Designator values. For the ordering, the most significant byte shall be byte 9 of the Universal Label, the least significant byte shall be byte 16 of the Universal Label.*
Register Administration Object	*StrongRef (Register Administration)*	*06.0E.2B.34.01.01.01.0C.06.01.01.04.02.11.00.00*	*RegisterAdministrationObject*	*Opt*	*Strong reference to one concrete class of Register Administration object Base Class.*

This class shall use the symbol “Register”.

6.6.2 Entry

Table 7 – Entry

Item Name	Type	UL	Symbol	Req ?	Definition
Register Item UL	*UniversalLabelType*	*06.0E.2B.34.01.01.01.0C.02.10.02.03.05.00.00.00*	*UniversalLabel*	*Req*	*Unique ID of this register entry.*
Register Item Name	*UTF-16String*	*06.0E.2B.34.01.01.01.0C.02.10.02.03.01.00.00.00*	*RegisterItemName*	*Req*	*The name of the leaf or node.*
Register Item Definition	*UTF-16String*	*06.0E.2B.34.01.01.01.0C.02.10.02.03.02.00.00.00*	*RegisterItemDefinition*	*Req*	*The detailed and unambiguous definition of the data element (leaf) or class/subclass (node).*

Item Name	Type	UL	Symbol	Req ?	Definition
Register Item Symbol	*SymbolType*	*06.0E.2B.34.01.01.01.0C.02.10.02.03.03.00.00.00*	*RegisterItemSymbol*	*Req*	*The symbol is a name that conforms to computer language syntax restrictions, and it is intended for use in computer languages such as Extensible Markup Language (XML).*
Namespace URI	*NamespaceURIType*	*06.0E.2B.34.01.01.01.08.01.02.01.05.01.00.00.00*	*NamespaceURI*	*Req*	*Defines the scope over which symbols are unique.*
Register Item Introduction Version	*UInt8*	*06.0E.2B.34.01.01.01.0C.02.10.02.03.07.00.00.00*	*RegisterItemIntroductionVersion*	*Req*	*This field records the version number of the register which first recorded the allocation of a data element or class/subclass description against its UL.*
Register Item Hierarchy Level	*UInt8*	*06.0E.2B.34.01.01.01.0C.02.10.02.03.08.00.00.00*	*RegisterItemHierarchyLevel*	*Req*	*This indicates the level of an entry in the class hierarchy of the SMPTE UL-based register. It is calculated from the position of the last active byte of the item designator.*
Register Item Notes	*UTF-16String*	*06.0E.2B.34.01.01.01.0C.02.10.02.03.06.00.00.00*	*RegisterItemNotes*	*Opt*	*Provides additional information that may assist in the interpretation and correct application of the data element (leaf) or class/subclass (node).*
Register Entry Administration Object	*StrongRef (Administration)*	*06.0E.2B.34.01.01.01.0C.06.01.01.04.02.12.00.00*	*RegisterEntryAdministrationObject*	*Opt*	*Strong reference to one concrete subclass of Entry AdministrationBase Class object.*

This shall be an abstract class.

This class shall use the symbol “Entry”.

6.6.3 Node

Table 8 – Node

Item Name	Type	UL	Symbol	Req ?	Definition
All items in 6.6.2	*See 6.6.2*	*See 6.6.2*	*See 6.6.2*	*See 6.6.2*	*See 6.6.2*
Register Child Entry Array	*StrongRefArray (Entry)*	*06.0E.2B.34. 01.01.01.0C. 06.01.01.04. 06.14.00.00*	*RegisterChildEntryArray*		*Array of strong references to Entry objects. The sequence of entries shall be according to ascending order of their UL Item Designator values. For the ordering, the most significant byte shall be byte 9 of the Universal Label, the least significant byte shall be byte 16 of the Universal Label.*
Register Node Wildcard Flag	*Boolean*	*06.0E.2B.34. 01.01.01.0C. 02.10.02.03. 09.00.00.00*	*RegisterNodeWildcardFlag*	*Opt*	*Provides information that entries exist under this node which are using wildcards. These wildcard entries shall be expanded always when exchanging SMPTE registers. This wildcard flag is an annotation that the entries under this node were created according to a pattern. An omitted wildcard flag shall have the meaning of 0 (false).*

This class shall use the symbol “Node”.

6.6.4 Leaf

Table 9 – Leaf

Item Name	Type	UL	Symbol	Req ?	Definition
All items in 6.6.2	*See 6.6.2*	*See 6.6.2*	*See 6.6.2*	*See 6.6.2*	*See 6.6.2*
Application Information Array	*StrongRefArray (Application Information)*	*06.0E.2B.34. 01.01.01.0C. 06.01.01.04. 06.13.00.00*	*ApplicationInformationArray*	*Opt*	*This field is an informative listing of some known applications that use a particular data element.*
Register Item Defining Document Name	*UTF-16String*	*06.0E.2B.34. 01.01.01.0C. 02.10.02.03. 04.00.00.00*	*RegisterItemDefiningDocumentName*	*Opt*	*Indicates the SMPTE document which defines this entry .*

This shall be an abstract class.

This class shall use the symbol “Leaf”.

6.6.5 Administration Base Class

Table 10 – Administration Base Class

Item Name	Type	UL	Symbol	Req ?	Definition
All items in 6.6	*See 6.6*	*See 6.6*	*See 6.6*	*See 6.6*	*See 6.6*
Register Editor Name	*UTF-16String*	*02.10.02.02.03*	*RegisterEditorName*	*Req*	*Provides the name of the person who administrates a register or entry.*
Register Item Status Change DateTime	*Timestamp*	*07.02.01.01.01.07*	*RegisterItemStatusChangeDateTime*	*Req*	*The timestamp of the last change of status of a register or entry.*

This shall be an abstract class.

This class shall use the symbol “AdministrationBaseClass”.

Concrete administrative classes that inherit from this base class shall not be used for interchange and shall not be part of published versions of the Universal Label-based SMPTE registers. These concrete administrative classes are exclusively intended for SMPTE internal management purposes.

Note: This document does not define concrete administrative classes. The concrete administrative classes will be defined in a future part of the RIF document family.

6.6.6 Register Administration Base Class

Table 11 – Register Administration Base Class

Item Name	Type	UL	Symbol	Req ?	Definition
All items in 6.6.5	*See 6.6.5*	*See 6.6.5*	*See 6.6.5*	*See 6.6.5*	*See 6.6.5*
Register Status Kind	*RegisterStatusType*	*06.0E.2B.34.01.01.01.0C.02.10.02.02.04.00.00.00*	*RegisterStatusKind*	*Req*	*Indicates the status of a register.*

This shall be an abstract class.

This class shall use the symbol “RegisterAdministration”.

6.6.7 Entry Administration Base Class

Table 12 – Entry Administration Base Class

Item Name	Type	UL	Symbol	Req ?	Definition
All items in 6.6.5	*See 6.6.5*	*See 6.6.5*	*See 6.6.5*	*See 6.6.5*	*See 6.6.5*
Register Entry Status	*EntryStatusType*	*06.0E.2B.34.01.01.01.0C.02.10.02.03.0A.00.00.00*	*RegisterEntryStatus*	*Req*	*Indicates the status of an entry.*

This shall be an abstract class.

This class shall use the symbol “EntryAdministrationBaseClass”.

6.6.8 Compound Entry Element Base Class

Table 13 – Compound Entry Element Base Class

Item Name	Type	UL	Symbol	Req ?	Definition
All items in 6.6	*See 6.6*	*See 6.6*	*See 6.6*	*See 6.6*	*See 6.6*

This shall be an abstract class.

This class shall use the symbol “CompoundEntryElementBaseClass”.

6.6.9 Application Information

Table 14 – Application Information

Item Name	Type	UL	Symbol	Req ?	Definition
All items in 6.6.8	*See 6.6.8*	*See 6.6.8*	*See 6.6.8*	*See 6.6.8*	*See 6.6.8*
Application Name	*UTF-16String*	*06.0E.2B.34. 01.01.01.02. 05.20.07.01. 03.00.00.00*	*Application Name*	*Req*	*Indicates the name of the application which uses the particular data element.*

This class shall use the symbol “ApplicationInformation”.

7 XML Encoding Rules

XML documents based on this standard shall comply with the XML document syntax and XML document structure defined in Annex A of this specification. Extensions to SMPTE UL-based registers shall be expressed according to the extension rules defined in this document.

The XML encodings that are defined below take precedence over the binary type of string encodings inside XML files.

7.1 Namespaces

The RIF Schema shall use the XML namespace URI defined in this document.

Table 15 – Namespace URI definitions

Structural Part of the RIF Schema	Namespace URI	Namespace Prefix (informative)
RIF root element	*http://www.smpte-ra.org/schemas/2045*	*rif*
RIF data types definitions	*http://www.smpte-ra.org/schemas/2045/types*	*types*
RIF dictionary element definitions	*http://www.smpte-ra.org/schemas/2045/dictionary*	*dictionary*
AdministrationBaseClass and all its subclasses	*http://www.smpte-ra.org/schemas/2045/groupsAdmin*	*groupsAdmin*
RIFBaseClass, Register, Entry, Node, Leaf and CompoundEntryElementBase.	*http://www.smpte-ra.org/schemas/2045/groupsContent*	*groupsContent*

7.2 Scalar Data Types

RIF defines the following basic data types that are shared between all UL-based SMPTE registers.

Note: Enumerated values (restriction of xs:string) are defined in the SMPTE enumerations register.

Table 16 – XML encoding rules for scalar data types

Data Type	Value encoding
Boolean	*xs:Boolean. See Annex A – dataTypes/ rif_types.xsd, simpleType "Boolean".*
Byte	*Restriction of xs:hexBinary. See Annex A – dataTypes/ rif_types.xsd, simpleType "Byte".*
EntryStatusType	*Restriction of xs:string. As defined in SMPTE 335M valid values are "workInProgress", "balloting", "editorial", "approved", or "deprecated". See Annex A – dataTypes/ rif_types.xsd, simpleType "EntryStatusType".*
LocalTag	*Restriction of xs:hexBinary. See Annex A – dataTypes/ rif_types.xsd, simpleType "LocalTag".*
NamespaceUriType	*Restriction of xs:string. See Annex A – dataTypes/ rif_types.xsd, simpleType "NamespaceUriType".*
RegisterStatusType	*Restriction of xs:string. See Annex A – dataTypes/ rif_types.xsd, simpleType "RegisterStatusType".*
RegisterType	*Restriction of xs:string. As defined in SMPTE 336M valid values are: "DataElementDictionaries", "EssenceDictionaries", "ControlDictionaries", "TypesDictionaries", "Groups", "WrappersAndContainers", "Labels", "RegisteredPrivateInformation". See Annex A – dataTypes/ rif_types.xsd, simpleType "RegisterType".*
StrongRef	*See 7.4.1*
SymbolType	*Restriction of xs:string. See Annex A – dataTypes/ rif_types.xsd, simpleType "SymbolType".*
UInt8	*xs:unsignedByte. See Annex A – dataTypes/ rif_types.xsd, simpleType "UInt8".*
UInt16	*xs:unsignedShort. See Annex A – dataTypes/ rif_types.xsd, simpleType "UInt16".*
UInt32	*xs:unsignedInt. See Annex A – dataTypes/ rif_types.xsd, simpleType "UInt32".*
UnitsOfMeasureType	*Restriction of xs:string. See Annex A – dataTypes/ rif_types.xsd, simpleType "UnitsOfMeasureType".*
UniversalLabel	*Restriction of xs:string. See Annex A – dataTypes/ rif_types.xsd, simpleType "UniversalLabel", according to SMPTE 2029-2009.*
UTF-16String	*Restriction of xs:string. See Annex A – dataTypes/ rif_types.xsd, simpleType "UTF16CharString".* *Unicode characters coded with 16-bits, i.e. 2-byte characters.*
UUID	*Restriction of xs:string. See Annex A – dataTypes/ rif_types.xsd, simpleType "UUID". Universally unique identifier according to ITEF RFC 4122.*

7.3 Compound Data Types

Table 17 – XML encoding rules for compound data types

Data Type	Value encoding
Array of Type	*Extension of Array (See Annex A – dataTypes/ rif_types.xsd, complexType "Array") such that the extension defines a sequence of Element of Type.*
Batch of Type	*Extension of Batch (See Annex A – dataTypes/ rif_types.xsd, complexType "Batch") such that the extension defines a sequence of Element of Type.*
StrongRefArray	*All concrete instances of array of strong references shall extend this abstract base class according to the rules defined in 7.4.2.* *See Annex A – dataTypes/ rif_types.xsd, complexType "StrongRefArray".*
StrongRefBatch	*All concrete instances of batch of strong references shall extend this abstract base class according to the rules defined in 7.4.2.* *See Annex A – dataTypes/ rif_types.xsd, complexType "StrongRefBatch".*
RegisterVersionType	*See Annex A – dataTypes/ rif_types.xsd, complexType "RegisterVersionType".*
Timestamp	*See Annex A – dataTypes/ rif_types.xsd, complexType "Timestamp".*

Compound data types shall be defined by using XML <xs:complex> types which use internally scalar data types that are defined in Section 6.3.

7.4 References

Strong references shall be represented in RIF compliant XML instances by using a reference mechanism. For this reason three abstract base classes "StrongRefHolder", "StrongRefArray" and "StrongRefBatch" are defined for the XML representations of Strong References.

7.4.1 StrongRef

For the object allowed to be referenced from an instance of a SMPTE 335M entry that is a Strong Reference, the base class StrongRefHolder shall be extended and one declaration shall be added to the complex type definition (to the xs:choice node). This declaration shall use the XML ref mechanism to reference the tag name of the class (e.g. groupsAdmin:EntryAdministration).

Table 18 – Example – Strong Reference between objects

```
<xs:element name="EntryAdministration">
        <xs:complexType>
                <xs:complexContent>
                        <xs:extension base="rifTypes:StrongRefHolder">
                                <xs:choice>
                                        <xs:element ref="groupsAdmin:EntryAdministration"/>
                                </xs:choice>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:element>
```

7.4.2 Array of StrongRef and Batch of StrongRef

For each object allowed to be referenced from an instance of a SMPTE 335M entry that is a Array of Strong References or Batch of Strong References, the base classes StrongRefArray and StrongRefBatch shall be extended and one declaration shall be added to the complex type definition (to the xs:choice node). This declaration shall use the XML ref mechanism to reference the tag name of the class (e.g. groupsContent:Group or groupsContent:Node).

Table 19 – Example – Array of Strong References between objects

```
<xs:element name="RegisterEntryArray">
        <xs:complexType>
                <xs:complexContent>
                        <xs:extension base="rifTypes:StrongRefArray">
                                <xs:choice minOccurs="0" maxOccurs="unbounded">
                                        <xs:element ref="groupsContent:Group"/>
                                        <xs:element ref="groupsContent:Node"/>
                                </xs:choice>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:element>
```

7.5 XML Document Structure

For XML conformance reasons one XML root node is introduced which encloses one or many “Register” nodes. The name of this root node shall be “SmpteRegisters”.

The ability to represent registers such that strong references are either resolved or not resolved, leads to two possible representations of the same data in XML documents: The hierarchical structure and the table structure (Universal-Label-ordered structure).

7.5.1 Hierarchical Structure

In hierarchical structure, the RegisterEntryArray XML element shall contain only the root Node XML element of the register. All other Entries shall be contained as XML elements in the ChildEntryArray of their parent Node.

Note 1: This results in a nesting of node elements.

Note 2: The value of the UL Item Designator of the register’s root element is 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00.

Note 3: XML instances using the hierarchical model could enable human readers to more easily understand the logical structure of the register.

In order to apply the hierarchical model, all “_REF” elements shall be resolved such that the target metadata object is represented in its place.

7.5.2 Table Structure

In table structure, the ChildEntryArray XML element of the Node class shall not be present. All Entries shall be encoded as a flat list of sibling XML elements within the RegisterEntryArray XML element of the Register class.

Note: XML instances using the table structure could be more efficient to process for computers. The hierarchical information (which is expressed in the hierarchical model by nesting node elements), is still available in the flat model: This information can be found in the Universal Label.

7.5.3 Hybrid Models

An XML instance shall follow either the hierarchical structure or the table structure. Hybrid structures shall not be used.

8 Rules for Extending the Data Model to Support Specific Registers

Each defining document of the SMPTE UL-based registers shall define the extensions of the Leaf class for the representation of the register in the Register Interchange Format. The following sections define the rules for extending the XML schema to support specific registers.

8.1 Namespaces

For each individual register the corresponding defining document shall define a unique namespace for all the register specific sub-class of Leaf and all register specific sub-classes of CompoundEntryElementBaseObject.

8.2 Scalar Data Types

For the definition of the extended LSeaf class, the scalar data types defined in Section 6.2 shall be used where applicable. If the defining document of the SMPTE UL-based register defines new scalar data types that are not part of this RIF standard, these additional data types shall be added to the Sections 6.2 and 7.2.

8.3 Compound Data Types

For the definition of the extended leaf class, the compound data types defined in Section 6.3 shall be used where applicable. If the defining documents of the SMPTE UL-based registers defines new compound data types that are not part of this RIF standard, these additional data types shall be added to the Sections 6.3 and 7.3. These new compound data types shall be concrete subclasses of the “Compound Entry Element Base Object” and shall be referenced from the leaf extension property via the strong reference mechanism.

8.4 References

Strong References, Array of Strong References and Batches of Strong References shall be added by defining an appropriate data type that extends “StrongRefHolder”, “StrongRefArray” and “StrongRefBatch”, respectively. The provisions are defined in Sections 7.4.1 and 7.4.2.

Annex A Base XML Schema Definition (Normative)

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

Table A.1 – Directory and File Structure of the RIF-XML-Schema

```
Annex-A
¦
+---rif
¦   +---registers.xsd
¦
+---groups
¦   +---rif_admin.xsd
¦   +---rif_content.xsd
¦
+---dictionary
¦   +---rif_dictionary.xsd
¦
+---dataTypes
    +---rif_types.xsd
```

Annex B Class Model and XML Schema Extension Example for SMPTE 395M (Informative)

In the following example, the RIF data model is extended in order to support the SMPTE 395M groups register.

This illustrative example has been written based on a draft of SMPTE 395M. The elements and their definitions are not intended to match any published version of the SMPTE 395M standard, but to be a meaningful example. Readers are advised to check the latest published version of SMPTE 395M for the precise definition of the register and the appropriate SMPTE document for a definition of the XML encoding of that register's contents.

B.1 Class Model Extension

B.1.1 Group

Table B.1 – Group

Item Name	Type	UL Designator	Symbol	Req ?	Definition
All items in 6.6.4	*See 6.6.4*	*See 6.6.4*	*See 6.6.4*	*See 6.6.4*	*See 6.6.4*
BaseType	*UniversalLabel*	*TBD*	*BaseType*	*Req*	*This item lists the UL of the parent group from which this group inherits all mandatory items and all optional items, if any.*
Abstract	*Boolean*	*TBD*	*Abstract*	*Req*	*This item identifies if a group can be instantiated or if it is an abstract superclass that serves as the base type for other group definitions within the register.*
Allowed KLV Syntax	*Batch of Byte*	*TBD*	*AllowedKLVSyntax*	*Req*	*This item lists those values of byte 6 of the Universal Label which are valid for this group designator.*
Mandatory Extensions	*StrongRefArray (MandatoryExtension)*	*TBD*	*MandatoryExtensions*	*Req*	*This array of extensions details the elements of the group which, in addition to the mandatory elements inherited from all base classes of the groups, shall be present in every instance of the group.*
Optional Extensions	*StrongRefArray (OptionalExtension)*	*TBD*	*OptionalExtensions*	*Req*	*This array of extensions details the elements of the group which, in addition to the optional elements inherited from all base classes of the group, may be present in some instances of the group.*
Group Contents	*StrongRefArray (GroupContents)*	*TBD*	*GroupContents*	*Req*	*This array of extensions contains the complete and ordered list of all mandatory and optional elements of the group.*

This class uses the symbol “Group”.

B.1.2 Extension Base Class

Table B.2 – Extension Base Class

Item Name	Type	UL Designator	Symbol	Req ?	Definition
All items in 6.6.8	*See 6.6.8*	*See 6.6.8*	*See 6.6.8*	*See 6.6.8*	*See 6.6.8*
Extensions Universal Label	*UniversalLabel*	*TBD*	*ExtensionsUniversalLabel*	*Req*	*This item identifies an element of the group. Its value is the Universal Label identifier of the item in the SMPTE metadata element dictionary defined by SMPTE 335M.*
Extensions Local Tag	*LocalTag*	*TBD*	*ExtensionsLocalTag*	*Opt*	*This item specifies the default local tag for local set syntax KLV encoding.*
Extensions Limit Length	*UInt32*	*TBD*	*ExtensionsLimitLength*	*Opt*	*This item specifies the element length which shall be used in defined length pack syntax.*

This shall be an abstract class.

This class uses the symbol “ExtensionBaseClass”.

B.1.3 Mandatory Extension

Table B.3 – Mandatory Extension

Item Name	Type	UL Designator	Symbol	Req ?	Definition
All items in B.1.2	*See B.1.2*	*See B.1.2*	*See B.1.2*	*See B.1.2*	*See B.1.2*

This class uses the symbol “MandatoryExtension”.

B.1.4 Optional Extension

Table B.4 – Optional Extension

Item Name	Type	UL Designator	Symbol	Req ?	Definition
All items in B.1.2	*See B.1.2*	*See B.1.2*	*See B.1.2*	*See B.1.2*	*See B.1.2*

This class uses the symbol “OptionalExtension”.

B.1.5 Group Contents

Table B.5 – Group Contents

Item Name	Type	UL Designator	Symbol	Req ?	Definition
All items in B.1.2	*See B.1.2*	*See B.1.2*	*See B.1.2*	*See B.1.2*	*See B.1.2*
Extension Required	*Bool*	*TBD*	*ExtensionRequired*	*Req*	*This item identifies if the element of the group is optional or required. A value of 1 (true) has the meaning that the element is an entry of the Mandatory Extensions of this group or its base type. A value of 0 (false) means that the element is an entry of the Optional Extensions of this group or its base type.*

This class uses the symbol “GroupContents”.

B.2 RIF-Schema Extension

For this example all “SMPTE 395M classes” are defined in the new “S395M.xsd” file which can be found in the “groups” subfolder. In addition, all new dictionary elements are added to the “rif_dicitonary.xsd” file.

The XML schema instance example file “SMPTE_395M.xml” can be found in the root directory.

Table B.6 – Directory and File Structure of the SMPTE 395M RIF-XML-Schema example

```
Annex-B
¦
¦
+---rif
¦   +---registers.xsd
¦
+---groups
¦   +---S395M.xsd
¦   +---rif_admin.xsd
¦   +---rif_content.xsd
¦
+---dictionary
¦   +---rif_dictionary.xsd (edited)
¦
+---dataTypes
¦   +---rif_types.xsd
¦
+---examples
¦   +---SMPTE_395M_flat_representation.xml (XML schema instance example)
¦   +---SMPTE_395M_hierarchical_representation.xml (XML schema instance example)
```

Annex C Class Model and XML Schema Extension Example for SMPTE 335M (Informative)

In the following example, the RIF data model is extended in order to support the SMPTE 335M dictionary register.

This illustrative example has been written based on a draft of SMPTE 335M. The elements and their definitions are not intended to match any published version of the SMPTE 335M standard, but to be a meaningful example. Readers are advised to check the latest published version of SMPTE 335M for the precise definition of the register and the appropriate SMPTE document for a definition of the XML encoding of that register's contents.

C.1 Class Model Extension

C.1.1 Data Element

Table C.1 – Data Element

Item Name	Type	UL Designator	Symbol	Req ?	Definition
All items in 6.6.4	*See 6.6.4*	*See 6.6.4*	*See 6.6.4*	*See 6.6.4*	*See 6.6.4*
Element Type	*UniversalLabel*	*TBD*	*ElementType*	*Req*	*The type Universal Label identifies the type description in the Types Register.*
Nested Leaves	*StrongRefArray (DataElement)*	*TBD*	*NestedLeaves*	*Opt*	*Note: RP210 contains "leaves under leaves". This entry supports this by establishing a strong reference relationship from a data element which is a leaf to one or more other data elements which are also leaves.*

This class uses the symbol "DataElement".

C.2 RIF-Schema Extension

For this example all "SMPTE 335M classes" are defined in the new "S335M.xsd" file which can be found in the "groups" subfolder.

Two XML schema instance example file "SMPTE_335M_flat_representation.xml" and "SMPTE_335M_hierarchical_representation.xml" can be found in the root directory.

Table C.2 – Directory and File Structure of the SMPTE 335M RIF-XML-Schema example

```
Annex-C
¦
+---rif
¦  +---registers.xsd
¦
+---groups
¦  +---S335M.xsd
¦  +---rif_admin.xsd
¦  +---rif_content.xsd
¦
+---dictionary
¦  +---rif_dictionary.xsd
¦
+---dataTypes
¦   +---rif_types.xsd
¦
+---examples
¦    +---SMPTE_335M_flat_representation.xml (XML schema instance example)
¦    +---SMPTE_335M_hierarchical_representation.xml (XML schema instance example)
```

Annex D Bibliography (Informative)

IETF RFC 2781, UTF-16, an encoding of ISO 10646

UML, Unified Modeling Language, http://www.uml.org/