TECH 3293

EBU CORE METADATA SET (EBUCore)

SPECIFICATION v. 1.8

Source: MIM

Geneva
October 2017
Introduction

'The EBUCore is a metadata specification designed for users with different needs'.

This is version 1.8 of the “EBUCore” metadata set.

EBUCore has been designed to describe audio, video and other resources for a wide range of broadcasting applications including archives, exchange and production in the context of a Service Oriented Architecture. EBUCore is based on the Dublin Core to maximise interoperability with the community of Dublin Core users such as the European Digital Library ‘Europeana’.

EBUCore 1.8 takes into account latest developments in the Semantic Web and Linked Open Data communities. A link to EBUCore RDF ontology and its documentation is provided in the “download zone”. The EBUCore RDF ontology has been updated to match EBU’s CCDM (Tech 3351) needs and improve matching with other ontologies. EBUCore RDF is listed as Linked Open Vocabulary as well as RDF-Vocab for Ruby developers.

What's new in EBUCore 1.8?

EBUCore 1.8 provides a solution for dynamic acquisition metadata, a unique representation of the ITU-R BS.2076 Audio Data Model (ADM). Now ‘props’, ‘costumes’, ‘timed text,’ 'actions and 'emotions' (among others) can be associated to scenes and persons or character. High Dynamic Range static technical metadata and Virtual Reality flags has been added to the videoFormat element.

This document provides links to the EBUCore schema and its HTML documentation. It also provides guidelines on how to use EBUCore to implement a variety of features.

More information on EBU metadata activities is provided on the EBU TECHNICAL website (http://tech.ebu.ch/metadata).

Terms and Conditions of Use

This EBUCore is freely available for all to use, but you should take note of the following:

© EBU 2017.

REDISTRIBUTION AND USE OF THIS SPECIFICATION AND ASSOCIATED RESOURCES IS PERMITTED PROVIDED THAT THE FOLLOWING CONDITIONS ARE MET:

REDISTRIBUTIONS MUST RETAIN THE ABOVE COPYRIGHT NOTICE, THIS LIST OF CONDITIONS AND THE FOLLOWING DISCLAIMER IN THE DOCUMENTATION AND/OR OTHER MATERIALS PROVIDED WITH THE DISTRIBUTION;

NEITHER THE NAME OF THE EBU NOR THE NAMES OF ITS CONTRIBUTOR(S) MAY BE USED TO ENDORSE OR PROMOTE PRODUCTS DERIVED FROM THIS SPECIFICATION AND ASSOCIATED RESOURCES WITHOUT SPECIFIC PRIOR WRITTEN PERMISSION.

DISCLAIMER: THIS SPECIFICATION AND ASSOCIATED RESOURCES IS PROVIDED BY THE COPYRIGHT OWNER "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS [SOFTWARE], EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Contents

Introduction ................................................................. 3

1. Scope ............................................................................... 7

2. Core Metadata Set ............................................................. 8
   2.1 Introduction .................................................................. 8
   2.2 Documentation .......................................................... 8
   2.3 What is new in EBUCore 1.8 ........................................ 8
   2.4 Tech. 3364 - Audio Definition Model (ADM) ................. 9

3. Implementation Guidelines / Questions & Answers ............... 11
   3.1 How do I express titles of a custom type in different languages? .................................................. 11
   3.1.1 Title ........................................................................ 11
   3.1.2 Alternative Title ...................................................... 12
   3.2 What controlled vocabularies and reference data can I use? ......................................................... 12
   3.3 Video and Audio time point references: anything fixed? ............................................................ 13
   3.4 What is the ‘part’ element? How can I use it? ................................................................. 14
      3.4.1 How can I define editorial ‘parts’ of a media resource? ....................................................... 14
      3.4.2 How can I use the ‘part’ element for dynamic (technical) metadata? .............................. 15
      3.4.3 How can I use parts to describe (programme) groups? .................................................. 16
      3.4.4 Distributed storage of media resources: where and in which format? .......................... 16
      3.4.5 Can I use the ‘part’ element to fragment my data? ....................................................... 17
      3.4.6 Can I use the ‘part’ element to localise props and artefacts? ....................................... 17
      3.4.7 Can I use the ‘part’ element to localise agents/contributors? ..................................... 18
      3.4.8 Can I use the ‘part’ element to localise text? ............................................................. 18
   3.5 How can I describe versions of programmes? ................................................................. 18
   3.6 How can I use my own technical attributes? ................................................................. 18
   3.7 How do I apply loudness? ................................................... 18
   3.8 How can I tag content in EBUCore? ........................................................................ 19
   3.9 How can I differentiate locators? .................................................................................. 19
   3.10 How can I associate a format and rights to a publication event? ........................................ 19
   3.11 When do I use labels and/or links in type/format/status? ............................................... 19
   3.12 Can I provide e.g. a display or print name in contactDetails? ........................................ 19
   3.13 Can I provide contactDetails for a group or ensemble? .................................................. 19
   3.14 Are there examples of implementation of the new audio model? ..................................... 20
   3.15 How can I extend a schedule beyond midnight? .......................................................... 20
   3.16 targetAudience, audienceLevel and audienceRating? .................................................. 20
   3.17 When should I use rating? ........................................................................................ 20
   3.18 How do I provide annotation along a timeline? .......................................................... 21
   3.19 How do I map mxf video and audio tracks to EBUCore? ............................................... 21
   3.20 How do I map MPEG video and audio tracks to EBUCore? ........................................ 24
   3.21 How do I implement EIDR or ISAN in EBUCore? ........................................................ 25
   3.22 How to define a checksum for insertion in BWF chunks? ................................................ 26
   3.23 How do I map EBUCore xml to EBUCore rdf? ........................................................... 26
   3.24 How do I use correction factors - frame rate, timecode? ............................................. 26
      3.24.1 aspectRatioType ..................................................... 26
3.24.2 rationalType ........................................................................................................... 27
3.24.3 EditUnitNumbertype ............................................................................................... 27
3.24.4 timecodeType ......................................................................................................... 27
3.25 Can I represent highly dynamic metadata ? ............................................................... 27
3.26 Can I track props, costumes and other artefacts in production ? .............................. 27
3.27 Can I represent timed text from various sources? ....................................................... 28
3.28 Can I track emotions and actions? .............................................................................. 28
3.29 How do I attach EBUCore side-car metadata in IMF ................................................ 28
3.30 More questions? ........................................................................................................ 28

4. Compliance ................................................................................................................... 28

5. Maintenance .................................................................................................................. 29

6. Download Zone ........................................................................................................... 29

7. Useful links .................................................................................................................... 29

8. Bibliography ................................................................................................................ 30

Annex A: EBUCore Metadata Set Schema .......................................................................... 31

Annex B: EBUCore and Semantic Web ............................................................................. 33

Annex C: Applying EBU Tech 3364’s data model in EBUCore ........................................ 35
  C.1 Channel based example (extract from Tech 3364) ................................................... 35
  C.2 Object based example (extract from Tech 3364) ...................................................... 37
  C.3 Scene based example (extract from Tech 3364) ...................................................... 39
EBUCore Metadata Set (EBUCore)

<table>
<thead>
<tr>
<th>EBU Committee</th>
<th>First Issued</th>
<th>Revised</th>
<th>Re-issued</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIM</td>
<td>Dec. 2008</td>
<td>November 2017</td>
<td>(v.1.8)</td>
</tr>
</tbody>
</table>

Keywords: EBUCore, Metadata, Schema, Dublin Core, P-META, Tech 3293, Radio, Television, CCDM

1. Scope

*Metadata is essential to broadcasting.*

The “EBUCore” set of metadata defined in this specification has been identified as being the minimum information needed to describe radio and television content.

"If you can't find it, you don't have it!" This should not happen in modern IT-based production environments. Metadata is the glue between production operations in particular moving towards Service Oriented Architecture and file-based production. Documenting audiovisual resources with EBUCore information is a minimum requirement corresponding to fundamental investment with guaranteed return.

This specification addresses the creation, management and preservation of audiovisual material. EBUCore facilitates programme exchanges between broadcasters or between production facilities in distributed and cloud environments. Beyond production, EBUCore can be used to describe content for distribution (broadcast, broadband Internet, mobile or hybrid delivery). EBUCore is also the default set of technical and descriptive metadata used by FIMS, the Framework of Interoperable Media Services (http://fims.tv).

The core set of metadata presented in EBUCore is the Dublin Core for media. The Dublin Core is being used as a core metadata set by librarians and museums in cultural heritage projects. The EBUCore is recommended when describing and providing access to audiovisual content and is not limited to archives.

EBUCore takes into account latest developments in the Semantic Web and Linked Open Data
EBU Core Metadata Set

EBU Core is a collection of basic descriptive and technical/structural metadata elements used to describe audiovisual content as an extension of the Dublin Core. It is fully compatible for use in Semantic Web and Service Oriented Architectures.

EBU Core is the Dublin Core for media.

The EBU Core is a living specification. It is actively maintained and enriched.

2.2 Documentation
Following best practice, the documentation has been generated from the schema.

The documentation can be summarised as follows:

- It looks very similar to what was provided in previous versions of the specification, incl. uml-like diagrams, the semantics of each element and attribute, their cardinality and associated complex types, the localisation of the element in the schema, etc.
- The documentation includes the schemas imported by EBU Core i.e. the xml stub and Dublin Core.
- The documentation is constructed around dynamic links that allow easily navigating through the schema.
- Each user can decide on the level of information detail to be displayed.

This html based documentation is attached to the specification as a zip archive to be extracted in the directory of the user's choice.

It can also be downloaded from the link provided in Section 6 “Download Zone”.

2.3 What is new in EBU Core 1.8
EBU Core 1.8 changes are listed in details in the documentation and can be categorised as follows:

- Additions based on feedback from implementers.
- Additions to facilitate mapping with SMPTECore, PBCore, EUSCreen, Europeana, MedialInfo, IMF-TV, IPTC, Movie Labs' Common Metadata, and other metadata projects in which EBU is directly involved.
- The update of the EBU extended audio model (3D and object audio) following work progress in the ITU. EBU Core is the ADM reference schema for ITU BS 2076-1.
- EBU Core 1.8 allows identifying and listing props, costumes and other artefacts used in production down to a scene. It also allows linking the use of such items by a person/character.
- Dynamic metadata can be represented in EBU Core 1.8 using effective metadata representation compatible with open source metadata extraction tools.
• Timed text from various sources (e.g. subtitling, speech to text, text recognition) can now be associated to a scene and person/character.

• Emotions and actions can now be associated to a scene and person/character.

• High Dynamic Range static technical metadata has now be added to the videoFormat, aligned with Movie Labs implementation of SMPTE ST 2086.

2.4 Tech. 3364 - Audio Definition Model (ADM)

The EBU has developed a new definition model that addresses all the richness of audio including 3D binaural and object audio.

The model is divided into two sections, the content part, and the format part. The content part describes what is contained in the audio (music, programme, accompanying sound tracks and their respective purpose), and will also describe things like the language of any dialogue, the loudness and so on. The format part describes the technical nature of the audio so it can be decoded or rendered correctly. Some of the format parts may exist before we have any audio signals, whereas the content is only completed after all the signals that form it have been generated.

While this model is based around BWF, it is not intended to be solely used for BWF files, but as a more general model. Mappings to MXF streams have also been developed and tested.

This model initially uses XML as its specification language with the EBUCore as the target schema. When it is used with BWF files, the XML can be embedded in the <axml> chunk of the file. When used with MXF, the XML can also be embedded in the header or footer partition of an MXF file. ADM metadata can also be provided in IMF as sidecar metadata.

The new model has been implemented in EBUCore as a new audioFormatExtended element in complement of the audioFormat element previously provided in earlier versions of the schema. This allows maintaining backward compatibility and graceful deprecation of the audioFormat element in favour of the new audioFormatExtended element.

The overall diagram of the model is given in Figure 1. This shows how the elements relate to each other and illustrates the split between its content and format parts.

The model has been adopted by ITU now leading its maintenance. The EBUCore schema is kept strictly aligned with approved ITU BS2076-1 changes.
Figure 2: Tech 3364 - Audio Definition Model
3. **Implementation Guidelines / Questions & Answers**

EBUCore has been developed in mind of users with different needs. EBUCore has been designed to support customisation in many different ways:

- EBUCore makes an extensive use of “type” and “format” attributes that allow flexible naming of elements such as title, description, etc.
- EBUCore provides two mechanisms for extensions:
  - The schema is systematically structured around the definition of complex types. This has been done to facilitate the “redefinition” of these complex types using restrictions or extensions via the `xs:redefine` function. This has been preferred to the use of `xs:any` allowing extensions at the instance level as `xs:redefine` is more flexible and provides more structured data, which is important for stricter validation in particular when aggregating data from different sources who may not respect the same element ordering (unless associated with a common schema for extension).
  - A set of predefined technical attributes is provided at key format extension points in the schema. These technical attributes cover the most common datatypes and structures (such as rationals). Each technical attribute is associated with a ‘type’ that allows the user to implement a parameter of his choice. When applicable, the user can additionally define the format of the data associated with this parameter.

Users can extend the EBUCore schema using proprietary definitions within their own namespace, which shall not conflict with EBUCore's and shall be used within walled garden application domains.

In all cases, it is the responsibility of the user to share the definitions of its extensions with third parties within his domain of interoperability.

Reference data and controlled vocabularies identified in the specification are proposed by default but can be extended or replaced. In order to maximise interoperability in case of e.g. exchange, it is recommended that extensions or alternative reference data be duly documented, maintained and made available to other users e.g. as open permanent resources on line.

The schema is built as an extension to the Simple Dublin Core.

The implementers are left with the choice of the method for implementation within their respective domains of application and interoperability.

User guidelines are now given that illustrate the flexibility of EBUCore schema for implementation.

### 3.1 How do I express titles of a custom type in different languages?

#### 3.1.1 Title

The MAIN title is expressed using the title element in which `dc:title` is repeated in as many languages as required using the `dc:element` and its associated `xml:lang` tag. The type of the main title can be refined as fits users’ needs.

The user can specify the geographical zones where the title can or cannot be used, as well as the date when it was attributed.

```xml
<ebucore:title startYear="2006" length="6" geographicalScope="France"
geographicalExclusionScope="Germany" typeLabel="main title in English">
  <dc:title xml:lang="en">title0</dc:title>
</ebucore:title>
```

It is also possible to express the language used to define the type of title, or an indication of the source of the title, etc.

*Note: the same applies to other elements build around a similar structure such as*
3.1.2 Alternative Title

Any other type of alternative title is expressed using alternativeTitle for which the type is defined and within which dc:title is repeated in as many languages as required using the dc:element and its associated xml:lang tag.

There can therefore be as many alternativeTitle as required type of alternativeTitle, each one grouping its expression in one or more languages.

```xml
<ebucore:alternativeTitle typeLabel="secondary title" typeLanguage="en-US" statusLabel="working title" startDate="2014-01-20">
  <dc:title xml:lang="en">Additional secondary title</dc:title>
  <dc:title xml:lang="fr">Titre additionnel et secondaire</dc:title>
</ebucore:alternativeTitle>

<ebucore:alternativeTitle typeLabel="display title" typeLanguage="en-US" statusLabel="working title" startDate="2014-01-20">
  <dc:title xml:lang="en">Additional secondary title for display</dc:title>
  <dc:title xml:lang="fr">Titre additionnel et secondaire pour affichage</dc:title>
</ebucore:alternativeTitle>
```

The need has been expressed that different types might apply to the same title. Types being defined by attributes, the solution consists of repeating the title with different types.

Note: the same applies to other elements build around a similar structure such as description, subject, rights, etc.

3.2 What controlled vocabularies and reference data can I use?

Controlled vocabularies are mainly provided in the form of lists of terms organised as Classification Schemes (CS). These CSs are structured to allow access to terms within a predefined hierarchical vocabulary list (thesaurus). Each list is uniquely identified by its namespace (URI\(^1\), in the form of a URN\(^2\) or URL\(^3\)) and ‘Alias’ for QNames. EBU namespaces are expressed in accordance to RFC5174\(^4\). A Classification Term is defined by a unique key (termID) or a Name as follows:

**Example:**

```xml
<ClassificationScheme uri="urn:ebu:metadata-cs:ContentGenreCS:2008">
  <Alias>GenreCS</Alias>
  <Term termID="3.1">
    <Name xml:lang="en">NON-FICTION / INFORMATION</Name>
    <Term key="3.1.1">
      <Name xml:lang="en">News</Name>
    </Term>
  </Term>
</ClassificationScheme>
```

It is an important implementation requirement to ensure that these resources are accessible by the metadata recipient. Classification schemes shall preferably be available as resources on the open Internet via maintained URLs. In this case URLs shall respect the following syntax:

```
URL#termID e.g. http://www.ebu.ch/metadata/cs/ebu_ContentGenreCS.xml#3.1
```

A conforming parser uses that URI to resolve the termID reference to a resource, whether physical

---

or logical. Once the termID has been resolved, the term name can be accessed (e.g. ‘News’ in the above example). The resolution method is left to the appreciation of each recipient.

URIs (URLs) can be replaced by aliases to provide a more concise, application-specific way of referring to classification terms as long as a look-up table is provided describing the relationship between Aliases and URIs.

If ‘GenreCS’ is the alias for “http://www.ebu.ch/metadata/cs/ebu_ContentGenreCS.xml”. In the above example, ‘News’ could be identified through “GenreCS#3.1”.

EBU Classification Schemes are also published in the SKOS (Simple Knowledge Organisation System) format using RDF for use and dereferencing as linked data (https://www.ebu.ch/metadata/ontologies/skos/).

3.3 Video and Audio time point references: anything fixed?

EBUCore uses three methods to identify video and audio time point references:

- a time duration according to ISO 8601 or IETF RFC 3339
- timecodes as defined by SMPTE in specification ST 12-1:2008
- a number of edit units, which are the fraction of time calculated as the inverse of the framerate for video, or the inverse of the sample rate for audio.

Audiovisual entities generally embed the property of having a “Timeline”, which comes from the fact that the AV work is conceived to be played for a defined “Duration”, and all the events characteristic of the AV work itself are located on the Timeline.

The Timeline concept applies to AV ‘editorial entities’ as well as to the ‘physical entities’ incl. their ‘technical parameters’, which are the sources providing the AV material for actual realisations.

A typical application of the timeline mechanism is for identifying the location of a given AV-entity A which is a part (in time) of another AV-entity B.

As B has got its own duration D(B), we can say that A, with its own duration D(A), is located at point S of the Timeline of B.

![Figure 3: Illustration of a timeline](image)

This means that if A is located on the Timeline of B, from S to E, then E=S+D(A).

In EBUCore there are two mechanisms for expressing a position on a Timeline:

- the “Elapsed Time”, which gives the time elapsed on the Timeline of the reference entity (B in the example above) from its beginning.
  - the data type for that is a ISO 8601 duration(e.g. PT1M5.0S) or IETF RFC 3339;
  - the reference point for the elapsed time is always the beginning of the reference entity.
- the “Elapsed Units” which give the same information in terms of the number of Edit Units (which are countable)
  - this is to be preferred because it ensures that Timeline markers fall on the boundary of
the Edit Unit;
  o duration of the EditUnit must be known unambiguously and indicated, otherwise it is better to use the “Elapsed Time”.

The two mechanisms mentioned above can also be used to locate the position of an AV-entity on the Timeline of a material source.

However there are contexts, in terms of the type of source, where the information in those terms is not available or it’s possibly ambiguous. For instance, identifying the position on a clip within a video-tape in terms of “Elapsed Time” or “Elapsed Units” from the “BOT (Beginning of Tape) is very difficult in practice. The BOT position itself may not be precise enough. In those cases, typically, the position on material source (e.g. the tape) is indicated by the “TimeCode”, which is a label recorded together with the EditUnit.

Although the “TimeCode” mechanism doesn’t provide any certainty about the uniqueness of the point on the Timeline (the same TimeCode might be repeated) and neither does it provide reliable information on Duration (“TimeCode” is not constrained to be continuous), this is the way on which legacy production systems rely for editing and for saving EDLs (Editing Decision Lists).

This is the reason why EBUCore also supports the indication of TimeCodes for all the cases where the Timeline positioning deals with material sources.

However it is recommended to also provide, if available, the information in terms of elapsed time or edit units (required in IMF).

Alternatively, a user may use the user defined time and duration formats to express time and duration as a number of seconds and fractions of seconds. Other user defined formats can be used.

The pattern used for timecode respects recommendation SMPTE ST 12-1-2008 and supports the use of drop-frame or non-drop-frame timecodes:

```
(((0-1)[0-9])|(2)(0-3)):0-5[0-9]:0-5[0-9]:0-9{2,5}
```

3.4 What is the 'part' element? How can I use it?

The ‘Part’ element was originally introduced to identify ‘editorial’ segments of content within a media resource. Following work done within the EBU on acquisition metadata, the definition of the ‘part’ element has been extended to allow any form of partitioned description, editorial or technical metadata, optionally bound to a specific timeline.

Some implementers also use part to sort metadata per category e.g. one or more parts for descriptive metadata, one part for video technical metadata, another for audio technical metadata, etc. This is absolutely in conformance with the EBUCore schema.

The EBU work on the Class Conceptual Data Model (CCDM), tech 3351, has allowed extending the use of ‘part’ to a business object made of one or more components. For example, EBUCore can be used to describe a series or season, each part describing an episode of that series or season.

The ‘part’ element is extremely versatile. Sections 3.4.1 and 3.4.5 illustrate some of these possible implementations. But the use of the ‘part’ element is not limited to these examples.

An EBUCore xml instance can contains a mix of ‘part’ elements of different nature, editorial technical or else. It is just a matter for the implementer to figure a coherent identification and name convention to help parsing these elements.

3.4.1 How can I define editorial 'parts' of a media resource?

There are many different editorial reasons why ‘parts’ (or e.g. segments, sequences, scenes) could be identified within a timeline. For example, content can be split into a set of purposefully
constructed sequences designed to facilitate user navigation (e.g. such as DVD chapters). ‘Parts’ can also be identified when a particular actor appears (e.g. as the result of face recognition processing or using user labelling). It can also be used to identify, for example, news items (internal, affairs, news report, weather report...) within a news programme.

Two mechanisms allow such ‘parts’ to be identified and described in EBUCore.

The first solution consists of using the ‘part’ element, which provides a description of ‘parts’ (and parts of parts of parts...) within one metadata instance.

The second solution consists of using the ‘hasPart’ or ‘hasTrackPart’ relations pointing to objects being described on their own with separate EBUCore metadata xml files/instances for each ‘part’. This will be the approach followed when using the EBU Core RDF ontology, linking data.

3.4.2 How can I use the ‘part’ element for dynamic (technical) metadata?

In order to describe the changing value of a technical attribute over time, all that is needed is to identify that the ‘Part’ element is used for this purpose through an appropriate ‘formatId’ or ‘formatName’. Time segments are defined by sub-parts. The ‘format’ element contains the value of the technical attribute associated to a ‘start’, ‘end’ and/or ‘duration’ time points.

```xml
<ebucore:part partId="CameraMetadata">
  <ebucore:format>
    <ebucore:start>
      <ebucore:editUnitNumber editRate="60" factorDenominator="1001" factorNumerator="1000">200</ebucore:editUnitNumber>
    </ebucore:start>
    <ebucore:duration>
      <ebucore:editUnitNumber editRate="60" factorDenominator="1001" factorNumerator="1000">800</ebucore:editUnitNumber>
    </ebucore:duration>
    <ebucore:technicalAttributeString typeLabel="AutoExposureMode" typeDefinition="a value from RP224" formatLabel="Universal Label">
      06.0E.2B.34.04.01.01.0B.05.10.01.01.01.02.00.00
    </ebucore:technicalAttributeString>
    <ebucore:technicalAttributeUInt16 typeLabel="ISOSpeed" typeDefinition="ISO12232">
      800
    </ebucore:technicalAttributeUInt16>
  </ebucore:format>
</ebucore:part>

<ebucore:part partId="part_CameraMetadata_2">
  <!-- SECOND TIME SEGMENT WITH A DIFFERENT SET OF CAMERA SETTINGS -->
  <ebucore:format>
    <ebucore:start>
      <ebucore:editUnitNumber editRate="60" factorDenominator="1001" factorNumerator="1000">1000</ebucore:editUnitNumber>
    </ebucore:start>
    <ebucore:duration>
      <ebucore:editUnitNumber editRate="60" factorDenominator="1001" factorNumerator="1000">630</ebucore:editUnitNumber>
    </ebucore:duration>
    <ebucore:technicalAttributeString typeLabel="AutoExposureMode" typeDefinition="a value from RP224" formatLabel="Universal Label">
      06.0E.2B.34.04.01.01.0B.05.10.01.01.01.02.00.00
    </ebucore:technicalAttributeString>
    <ebucore:technicalAttributeUInt16 typeLabel="ISOSpeed" typeDefinition="ISO12232">
      1600
    </ebucore:technicalAttributeUInt16>
  </ebucore:format>
</ebucore:part>
```

Example - Camera parameter evolution associated to a timeline

Important note: EBUCore now offers an alternative solution specifically designed for highly dynamic metadata. See Section 3.25.
3.4.3 How can I use parts to describe (programme) groups?

EBUCore fully supports the description of groups and collections using the appropriate relations such as isMemberOf or isEpisodeOf.

Similar mechanisms can be used to identify different parts composing a media resource using e.g. hasPart.

However, it is also possible to use the "part" element as syntactically represented in the following xml snippet:

```xml
<ebuCoreMain xmlns=....>
  <coreMetadata>
    <!-- Describe here a series with series title, description, cast, etc.-->
    <!-- define the type / objectType as "series"
    <part partId="season1" partName="season a" typeLabel="Season">
      <!-- describe here the elements characterising a season -->
    </part>
    <part partId="e1" partName="episode 1" typeLabel="Episode">
      <!-- describe here the episode with title, description, cast incl. e.g. guests, etc. -->
    </part>
    <part partId="e2" partName="episode 2" typeLabel="Episode">
      <!-- describe here the episode with title, description, cast incl. e.g. guests, etc. -->
    </part>
    <!-- Etc.-->
  </coreMetadata>
</ebuCoreMain>
```

Example: description of a series with seasons and episodes

3.4.4 Distributed storage of media resources: where and in which format?

As shown below, there are two built-in mechanisms to describe where are different instantiations of the same editorial objects, possibly in different formats or using different type of storage.

A third option would consists of defining a new relation such as "isInstantiatedBy" establishing the relation between the editorial object described in an EBUCore instances with instantiations described by their own EBUCore instances.

**Option 1 - Repeating the format element**

```xml
<ebuCoreMain xmlns=....>
  <coreMetadata>
    <!-- Describe here the editorial objects possibly with associated editorial parts, etc.-->
    <!-- Define the type/objectType e.g. as a programme -->
    <format formatId="instance1" formatName="instantiation 1">
      <!-- describe here the format and location of this instantiation -->
    </format>
    <locator>xxxx</locator>
    <format formatId="instance2" formatName="instantiation 2">
      <!-- describe here the format and location of this instantiation -->
    </format>
    <locator>xxxx</locator>
    <!-- Etc. -->
  </coreMetadata>
</ebuCoreMain>
```
Option 2 - Using different 'part' element for each format

```xml
<ebuCoreMain xmlns=...>
  <coreMetadata>
    <!-- Describe here the editorial objects possibly with associated editorial parts, etc.-->
    <!-- Define the type/objectType e.g. as a programme -->
    <part partId="instance1" partName="Instantiation 1">
      <format>
        <!-- describe here the format and location of this instantiation -->
        <locator>xxxx</locator>
      </format>
    </part>
    <part partId="instance2" partName="Instantiation 2">
      <format>
        <!-- describe here the format and location of this instantiation -->
        <locator>yyyy</locator>
      </format>
    </part>
  </coreMetadata>
</ebuCoreMain>
```

3.4.5 Can I use the 'part' element to fragment my data?
Yes. As an example, some implementers use the part element to clearly separate technical format metadata:

```xml
<ebuCoreMain xmlns=...>
  <coreMetadata>
    <!-- Optionally describe here the editorial object, not required if the xml document only describes an specific instantiation -->
    <format>
      <!-- provide here file and container information -->
      <fileName>filename</fileName>
      <locator>xxxxxx</locator>
    </format>
    <part partId="id23" partName="audio format information">
      <audioFormatExtended>
        <!-- describe here the audio format -->
      </audioFormatExtended>
    </part>
    <part partId="id34" partName="video format information">
      <videoFormat>
        <!-- describe here the video format -->
      </videoFormat>
    </part>
  </coreMetadata>
</ebuCoreMain>
```

This approach can apply in a variety of contexts left to the appreciation of the implementer.

3.4.6 Can I use the 'part' element to localise props and artefacts?
The elements artefact, props, costume, food allow identifying which of these objects can be seen in a scene/part. It is also possible to link an object to a person/actor/contributor appearing in the scene/part.
3.4.7 Can I use the 'part' element to localise agents/contributors?
Each part can be associated with agents/contributors with a variety of roles.

3.4.8 Can I use the 'part' element to localise text?
The textLine element allows associating text to a scene and agent in a given order and language. The source of the text can also be specified (e.g. speech to text).

3.5 How can I describe versions of programmes?
There can be many reasons why a programme is declared to be a version of a particular source (e.g. a shorter version, a different language, with or without captioning, but also available on different mediums such as a file, a tape, a disk).

The best approach to identify versions is to use relations such as hasVersion or hasSource. The relation links two instances and their respective descriptions highlighting differences such as given above as examples.

EBUCore provide a rich set of predefined relations. Users can also define their own relation as shown below:

```xml
<relation typeLabel="my relation"
  typeDefinition="my type of relation in a few words"
  note="relation for this purpose">
  <dc:relation>isConnectedTo</dc:relation>
</relation>
```

3.6 How can I use my own technical attributes?
The following example illustrates how to use the technical attribute patterns within EBUCore. The average bitrate can be expressed as:

a. Example of a user defined technicalAttributeString

```xml
<ebucore:technicalAttributeString typeLabel="averageBitrate" typeDefinition="the average bitrate" formatLabel="integer">123456789</ebucore:technicalAttributeString>
```

b. Example of a user defined a technicalAttributeLong

```xml
<ebucore:technicalAttributeLong typeLabel="myCustomTechnicalParameter">123456789</ebucore:technicalAttributeLong>
```

The choice is left to the implementer within his domain of interoperability. Some implementations use only the technicalAttributeString, which allows defining/replacing all simple datatypes with only one structure. It can also be used for more complex datatypes such as the technicalAttributeRational structure by defining a template in the format attribute, etc.

Some pre-defined technical attributes may be replaced by user defined attributes if required by the format of the data available to the implementer.

The metadata extraction application "mediaInfo" exports data as EBUCore. When technical attributes are not specifically defined, mediaInfo uses technical attributes to export technical metadata.

3.7 How do I apply loudness?
Documents EBU Tech 3343 and EBU R 128 establish a predictable and well-defined method to measure the loudness level for news, sports, advertisements, drama, music, promotions, film etc. throughout the broadcast chain and thereby helps professionals to create robust specifications for ingest, production, play-out and distribution to a multitude of platforms. EBU R 128 is based
entirely on open standards and aims to harmonise the way we produce and measure audio internationally.

The audioFormatExtended element specifies precisely how and where to apply loudness.

### 3.8 How can I tag content in EBUCore?

It is now common to 'tag' content. Tags can be issued by professionals like content creators or content providers, or by users.

In EBUCore, tags are defined as 'subjects' of typeLabel="tag" issued by 'attributors' (persons or organisations).

### 3.9 How can I differentiate locators?

The locator attribute in format now has a type attribute to specify what the locator is being used for. A locator can point to an object of typeLabel "resource", "thumbnail", "landing page", etc.

Relations can also be used to point to related external objects.

### 3.10 How can I associate a format and rights to a publication event?

The publicationHistory lists the publication events of a media resource. This allows indicating, when, where, in which format and under which rights the media resource has been published.

The format attribute now has an associated formatId attribute of type ID, which can be referenced to from the publication event formatIdRef attribute. This allows indicating which format is being used for a particular publication.

The same applies to rights using the rightsID and associated rightsIDREFS.

### 3.11 When do I use labels and/or links in type/format/status?

Several EBUCore elements and attributes propose to define their value through a “label” and/or a “link”. Links can be used to define a relationship in the sense of linked data. A typical case of linked data consists of pointing to classification scheme term via its identifier.

However, when available, the corresponding term of value should also be provided as a 'label' to facilitate mapping.

### 3.12 Can I provide e.g. a display or print name in contactDetails?

A display or print name can easily be implemented as follows:

```xml
<contactDetails>
  <name typeLabel="Display name">Mr. Best Guest for display</name>
  <name typeLabel="Print name">Mr. Best Guest for print</name>
</contactDetails>
```

### 3.13 Can I provide contactDetails for a group or ensemble?

For groups, ballets or similar non individual formation, you can use contactDetails with typeLabel="music group, orchestra dancing team, ballet team or e.g. individual” (real person by default but can be fictitious). Then use “name” to provide the name of the group, orchestra, dancing team, ballet team or individual.

You can use the other more structured way for detailing names for persons either real or fictitious (there are web resource describing the bio and life of fictitious characters). Then you can now use a link to web resource with related information, which can be of any type optionally specified.
Another option would be to use organisationDetails in the same way for groups and then use contactDetails for members of the group.

3.14 Are there examples of implementation of the new audio model?

Examples of implementation of the new EBU extended audio model implemented in the EBUCore schema through the audioFormatExtended element can be found in Annex C and in the audio model specification Tech.3364.

EBU Tech 3364 provides more details on the audio model, including examples on its mapping to BWF and MXF.

The extended audio model is build around a set of relations. These relations are clearly identified through the use of a naming convention by which elements or attributes contains 'ID', 'IDREF', 'IDREFS'. In xml, IDREF and IDREFS pair with ID to establish a relation. This can be implemented in two ways:

- xml Schema define datatypes for ID, IDREF and IDREFS. If these datatypes are used, relations must be established within the same metadata instance file, which can be seen as a design restriction but provides strong validation support.
- EBUCore doesn't use ID, IDREF and IDREFS datatypes but anyURI. This allows establishing relations across metadata instance files (e.g. allowing to separate a content description from the structural description of the audio model components, or to constitute external libraries of audio configurations). Of course, this requires that an implementer pays particular attention to the pairing of URIs, which is facilitated by the naming convention used for related elements and attributes in the EBUCore schema.

3.15 How can I extend a schedule beyond midnight?

It is common practice that a day schedule starts for example at around 6 O'clock in the morning and finishes at around the same time on the following day. The following xml snippet shows how it should be done instantiating the EBUCore schema.

```
<publicationEvent>
  <publicationDate>2014-03-12</ns2:publicationDate>
  <publicationTime>01:00:00</ns2:publicationTime>
  <scheduleDate>2014-03-11</ns2:scheduleDate>
  <publicationService…
</publicationEvent>
```

In this example, a programme is broadcast a 1 O'clock in the morning on 12th March 2014, but this publication event belong to the schedule of 11th March 2014.

3.16 targetAudience, audienceLevel and audienceRating?

Originally, targetAudience has been the common placeholder for audience rating and audience level as defined also by MPEG-7 and TV-Anytime. However, for the sake of harmonisation with North-American best practice, the audienceLevel element has been added as a sub-element of ebucore:type. It is recommended to use targetAudience or audienceLevel to identify age and population groups.

The element audienceRating has also been added for the sake of harmonisation. It shall be used to define an audience appropriate to the programme. The EBU Classification Scheme for audience rating has been deprecated. For ratings the EBU now recommends the use of http://www.movielabs.com/md/ratings/v2.0/html/Summary.html.

3.17 When should I use rating?

Rating can be used to rate content (e.g. 'like' or 'not like' or using custom scales). Rating can be provided by users or organisations.
3.18 **How do I provide annotation along a timeline?**

There are two options:

- Use a part element with a timestamp and a description of type annotation with an attributor.
- Populate the new textLine element with an part or associated to a timestamp. In this implementation, timed text can be specifically associated to a scene, a timestamp and/or a person/character. See Section 3.27.

3.19 **How do I map mxf video and audio tracks to EBUCore?**

The following is an example of technical metadata extracted from a UK DPP AS-11 MXF file and converted into EBUCore using mediaInfo (https://mediaarea.net/fr/MediaInfo).

```xml
<ebucore:coreMetadata>
  <ebucore:title typeLabel="PROGRAMME TITLE">
    <dc:title>Programme Title - Post Watershed Version</dc:title>
  </ebucore:title>
  <ebucore:alternativeTitle typeLabel="SERIES TITLE">
    <dc:title>Series Title</dc:title>
  </ebucore:alternativeTitle>
  <ebucore:alternativeTitle typeLabel="EPISODE TITLE NUMBER">
    <dc:title>Episode Title / Number</dc:title>
  </ebucore:alternativeTitle>
  <ebucore:description typeLabel="SYNOPSIS">
    <dc:description>DPP test material</dc:description>
  </ebucore:description>
  <ebucore:description typeLabel="PRODUCT PLACEMENT">
    <dc:description>false</dc:description>
  </ebucore:description>
  <ebucore:contributor>
    <ebucore:contactDetails>
      <ebucore:details>
        <ebucore:emailAddress>xxx.yyy@bbc.co.uk</ebucore:emailAddress>
        <ebucore:telephoneNumber>0208 008 4566</ebucore:telephoneNumber>
      </ebucore:details>
    </ebucore:contactDetails>
    <ebucore:role typeLabel="contact"/>
  </ebucore:contributor>
  <ebucore:contributor>
    <ebucore:organisationDetails>
      <ebucore:organisationName>BBC R&D</ebucore:organisationName>
    </ebucore:organisationDetails>
    <ebucore:role typeLabel="originator"/>
  </ebucore:contributor>
  <ebucore:date>
    <ebucore:copyrighted startYear="2013"/>
  </ebucore:date>
  <ebucore:type>
    <ebucore:genre typeDefinition="Test Material"/>
  </ebucore:type>
  <ebucore:format>
    <ebucore:videoFormat videoFormatName="AVC">
      <ebucore:width unit="pixel">1920</ebucore:width>
      <ebucore:height unit="pixel">1080</ebucore:height>
      <ebucore:frameRate factorNumerator="25000" factorDenominator="1000">25</ebucore:frameRate>
      <ebucore:aspectRatio typeLabel="display">
        <ebucore:factorNumerator>16</ebucore:factorNumerator>
        <ebucore:factorDenominator>9</ebucore:factorDenominator>
      </ebucore:aspectRatio>
      <ebucore:videoEncoding typeLabel="High 4:2:2 Intra@L4.1"/>
    </ebucore:videoFormat>
  </ebucore:format>
</ebucore:coreMetadata>
```
<ebucore:bitRate>113664000</ebucore:bitRate>
<ebucore:scanningFormat>interlaced</ebucore:scanningFormat>
<ebucore:videoTrack trackId="1001" trackName="V1"/>
<ebucore:technicalAttributeString typeLabel="ActiveFormatDescription">5</ebucore:technicalAttributeString>
<ebucore:technicalAttributeString typeLabel="Standard">Component</ebucore:technicalAttributeString>
<ebucore:technicalAttributeString typeLabel="ColorSpace">YUV</ebucore:technicalAttributeString>
<ebucore:technicalAttributeString typeLabel="ChromaSubsampling">4:2:2</ebucore:technicalAttributeString>
<ebucore:technicalAttributeString typeLabel="colour_primaries">BT.709</ebucore:technicalAttributeString>
<ebucore:technicalAttributeString typeLabel="transfer_characteristics">BT.709</ebucore:technicalAttributeString>
<ebucore:technicalAttributeString typeLabel="matrix_coefficients">BT.709</ebucore:technicalAttributeString>
<ebucore:technicalAttributeString typeLabel="colour_range">Full</ebucore:technicalAttributeString>
<ebucore:technicalAttributeInteger typeLabel="StreamSize">1700924815</ebucore:technicalAttributeInteger>
<ebucore:technicalAttributeInteger typeLabel="BitDepth">10</ebucore:technicalAttributeInteger>
<ebucore:technicalAttributeBoolean typeLabel="CABAC">false</ebucore:technicalAttributeBoolean>
<ebucore:technicalAttributeBoolean typeLabel="MBAFF">true</ebucore:technicalAttributeBoolean>
<ebucore:comment typeLabel="VideoComments">RP 2027 AVC-Intra compatible encoded</ebucore:comment>
<ebucore:videoFormat>
<ebucore:audioFormat audioFormatName="PCM"/>
<ebucore:audioEncoding typeLabel="PCM"/>
<ebucore:codec>
<ebucore:codecIdentifier>
<dc:identifier>0D01030102060100</dc:identifier>
</ebucore:codecIdentifier>
</ebucore:codec>
<ebucore:audioTrackConfiguration typeLabel="EBU R 123: 16c"/>
<ebucore:sampleSize>24</ebucore:sampleSize>
</ebucore:audioFormat>
<!-- other audio channel formats -->
<ebucore:containerFormat containerFormatName="MXF"/>
<ebucore:containerEncoding formatLabel="MXF"/>
<ebucore:technicalAttributeString typeLabel="AS11ShimName">UK DPP HD</ebucore:technicalAttributeString>
<ebucore:technicalAttributeString typeLabel="FormatProfile">OP-1a</ebucore:technicalAttributeString>
<ebucore:technicalAttributeString typeLabel="FormatSettings">Closed / Complete</ebucore:technicalAttributeString>
<ebucore:technicalAttributeString typeLabel="WritingApplication">BBC bmx 0.1.2.0.2</ebucore:technicalAttributeString>
3.20 How do I map MPEG video and audio tracks to EBUCore?

The following is an example of technical metadata extracted from an MPEG transport stream (TS) file and converted into EBUCore using mediaInfo (https://mediaarea.net/fr/MediaInfo).

```xml
<ebucore:format>
  <ebucore:videoFormat videoFormatName="MPEG Video" videoFormatVersionId="Version 2">
    <ebucore:width unit="pixel">720</ebucore:width>
    <ebucore:height unit="pixel">576</ebucore:height>
    <ebucore:framerate factorNumerator="25000" factorDenominator="1000">25</ebucore:frameRate>
    <ebucore:aspectRatio typeLabel="display">
      <ebucore:factorNumerator>4</ebucore:factorNumerator>
      <ebucore:factorDenominator>3</ebucore:factorDenominator>
    </ebucore:aspectRatio>
    <ebucore:videoEncoding typeLabel="MPEG-2 Video Main Profile @ Main Level"/>
    <ebucore:codec>
      <ebucore:codecIdentifier>
        <dc:identifier>2</dc:identifier>
      </ebucore:codecIdentifier>
    </ebucore:codec>
    <ebucore:bitRate>3614266</ebucore:bitRate>
    <ebucore:bitRateMax>15000000</ebucore:bitRateMax>
    <ebucore:scanningFormat>interlaced</ebucore:scanningFormat>
    <ebucore:scanningOrder>top</ebucore:scanningOrder>
    <ebucore:videoTrack trackId="120"/>
  </ebucore:videoFormat>
  <ebucore:audioFormat audioFormatName="MPEG Audio" audioFormatVersionId="Version 1">
    <ebucore:audioEncoding typeLabel="MPEG-1 Audio Layer II" typeLink="http://www.ebu.ch/metadata/cs/ebu_AudioCompressionCodeCS.xml#7.2"/>
    <ebucore:codec>
      <ebucore:codecIdentifier>
        <dc:identifier>4</dc:identifier>
      </ebucore:codecIdentifier>
    </ebucore:codec>
    <ebucore:samplingRate>48000</ebucore:samplingRate>
    <ebucore:bitRate>256000</ebucore:bitRate>
    <ebucore:bitRateMode>constant</ebucore:bitRateMode>
    <ebucore:audioTrack trackId="130" trackLanguage="fr"/>
    <ebucore:channels>2</ebucore:channels>
  </ebucore:audioFormat>
</ebucore:format>

<!—other audio channels and associated formats—>
```
3.21 How do I implement EIDR or ISAN in EBUCore?

As defined in collaboration with EIDR, this is how it is recommended to encode EIDR identifiers into EBUCore metadata instances:

```
<ebucore:identifier typeLabel="EIDR" typeLink="http://www.ebu.ch/metadata/cs/ebu_IdentifierTypeCodeCS.xml#3.11">
</ebucore:identifier>
```

Term 3.11 points to the EBU Classification Scheme for identifiers containing the following information:

```
<Term termID="3.11">
  <Name xml:lang="en">EIDR</Name>
  <Definition xml:lang="en">Canonical Representation of an Entertainment Identifier Registry (EIDR) Identifier, a universal unique identifier for movie and television assets.</Definition>
  <Reference>Section 2.2 of EIDR: ID FORMAT</Reference>
  <ChangeComment>Clarified that the Canonical Representation is used.</ChangeComment>
  <ChangeVersionDate>2015-05-05</ChangeVersionDate>
  <ValidityFlag>1</ValidityFlag>
  <Link xml:lang="en">
    <URL>http://eidr.org/documents/EIDR_ID_Format_v1.2.pdf</URL>
    <LinkName>EIDR: ID FORMAT</LinkName>
  </Link>
</Term>
```

Note: A similar approach can be followed to encode ISAN or other identifiers into EBUCore metadata instances.
3.22 How to define a checksum for insertion in BWF chunks?
As defined in collaboration with “The Music Producers Guild” this is how it is recommended to represent checksums in an EBUCore snippet for insertion in BWF metadata chunks.

```xml
<format>
  <hash>
    <hashValue>
      d131dd02c5e6eeec4 693d9a0698afff95c 2fca85712467eeb 4004583eb0f7f89
      5ad3406e9f6b302 83e4888323d1b15a 085125e8f7c99f 091d6c4289373c5b
      d8823e3156348f5b 0e93c69391cd65a75a6712467eeb 4004583eb0f7f89
      5ad3406e9f6b302 83e4888323d1b15a 085125e8f7c99f 091d6c4289373c5b
      d8823e3156348f5b 0e93c69391cd65a6712467eeb 4004583eb0f7f89
      5ad3406e9f6b302 83e4888323d1b15a 085125e8f7c99f 091d6c4289373c5b
      d8823e3156348f5b 0e93c69391cd65a6712467eeb 4004583eb0f7f89
      5ad3406e9f6b302 83e4888323d1b15a 085125e8f7c99f 091d6c4289373c5b
      d8823e3156348f5b 0e93c69391cd65a6712467eeb 4004583eb0f7f89
      5ad3406e9f6b302 83e4888323d1b15a 085125e8f7c99f 091d6c4289373c5b
      d8823e3156348f5b 0e93c69391cd65a6712467eeb 4004583eb0f7f89
      5ad3406e9f6b302 83e4888323d1b15a 085125e8f7c99f 091d6c4289373c5b
      d8823e3156348f5b 0e93c69391cd65a6712467eeb 4004583eb0f7f89
      5ad3406e9f6b302 83e4888323d1b15a 085125e8f7c99f 091d6c4289373c5b
    </hashValue>
    </hashFunction typeLabel="MD5" typeLink="https://dbpedia.org/page/MD5">
  </hash>
</format>
```

3.23 How do I map EBUCore xml to EBUCore rdf?
The EBUCore schema inherently addresses the classes and properties that can be found in EBUCore RDF.

An EBUCore description is about content e.g. a programme, a group of programme. Several of these types of programmes are predefined in the ontology as subclasses of the EditorialObject. Descriptive metadata such as e.g. title or description are associated with the EditorialObject.

Each EditorialObject is instantiated into MediaResources and Essences. This corresponds to what is described in the format element of the EBUCore schema, including all technical properties. The EBUCore schema allows describing the different formats in which content is available, which would correspond to different MediaResources or Essences.

Many other classes are mapped directly from the schema into the ontology, e.g. persons/contacts, organisations, events, locations, props, emotions, etc.

In order to facilitate future mappings, it is recommended to associate identifiers as early in the process as possible and when applicable to formats/@formatId, entities/@entityId, locations/@locationId, events/@eventId, etc. This will be useful when defining URIs to identify instances of classes in RDF.

3.24 How do I use correction factors – frame rate, timecode…?
The following complex types of the EBUCore schema contain factorNumerator and factorDenominator attributes allowing an alternative representation of rational values:

- aspectRatioType
- rationalType (frameRate, technicalAttributeRational)
- editUnitNumberTypetimecodeType

This is how the correction factors should be used.

3.24.1 aspectRatioType
An aspect ratio can be expressed as float, e.g. 1.7777. It can also be expressed as the “16 by 9” ratio. The EBUCore schema supports the use of a technicalAttributeFloat (1.777) or a technicalAttributeString in which this ratio can be expressed in many possible ways, e.g. “16 by 9”, “16:9”, “16/9”.

Alternatively, the aspectRatioType defines the ratio by its numerator and denominator; factorNumerator=”16” and factorDenominator = “9”.

```xml
</format>
```
3.24.2 rationalType
In this case the rational value in an integer, which can be weighed by a factorNumerator and factorDenominator.

Examples of possible framerate expressions:
- framerate=60 with factorNumerator =1000 and factorDenominator =1001

Other possible examples but not recommended:
- framerate=1 with factorNumerator =60000 and factorDenominator =1001

3.24.3 EditUnitNumbertype
In this case, the integer expressed a number of Edit Units. The value of each Edit Unit is based on the framerate for video (e.g. 60 Hz) or sampleRate (48000 Hz) for audio, so called EditRate in the EditUnitNumberType, which is weighed by a factorNumerator and factorDenominator.

3.24.4 timecodeType
In order to be able accurately convert timecode into real time, timecode should be annotated with an edit rate and drop frame attributes. For example, a timecode of 01:00:00;00 at 29.97 frames per second dropframe should be represented as follows:

```
<timecodeStart editRate="30" factorNumerator="1000" factorDenominator="1001"
    dropframe="true">
  01:00:00;00
</timecodeStart>
```

Note that while dropframe should always be denoted by a ';' in between the seconds and frames field, this is not always done in practice, so it is better to make it explicit, i.e. setting the timecode/@dropframe attribute to 'true'.

A timecode of 01:00:00:00 at 25 frames per second can be represented as follows:

```
<timecodeStart editRate="25">
  01:00:00:00
</timecodeStart>
```

3.25 Can I represent highly dynamic metadata?
Highly Dynamic metadata is typically acquisition technical metadata that can potentially change frame after frame. However, the following solution applies to any form of highly dynamic metadata.

EBUCore supports two alternative formats for highly dynamic acquisition technical metadata instantiating the acquisitionData element:

- representation of each technical parameter, which value varies along a line of time segments.
- representation of time segments during which each available parameter value is provided.
- Dynamic technical metadata can for example be extracted using the mediaInfo application (open source, licence free, https://mediaarea.net/fr/MediaInfo/Download/Source).
- Best practice is to generate separate EBUCore files for dynamic metadata. For technical dynamic metadata, the structure of the xml instance would be:
- ebuCoreMain/coreMetadata/format/acquisitionData

3.26 Can I track props, costumes and other artefacts in production?
EBUCore now allows to identify and list props, costumes and other artefacts down to a scene or timestamp and establish relations to e.g. persons/characters using these items. A relation to a
person/character can be done simple through the entityId.

### 3.27 Can I represent timed text from various sources?

TextLine is a new element that allows to link text to parts/segments/scenes or a timestamp. Text can be referenced from various sources such as subtitling, speech to text, text extraction, etc. Text can also be associated to a Person/Character in the scene. A relation to a person/character can be done simple through the entityId. This comes in complement to other sources of timed text such as W3C TTML or EBU-TT.

### 3.28 Can I track emotions and actions?

EBUCore now allows to identify emotions and actions in scenes and associated persons/characters. A relation to a person/character can be done simple through the entityId. A precise timestamp can be used to locate emotions and actions.

### 3.29 How do I attach EBUCore side-car metadata in IMF

An answer to that question is being studied for the next release of EBUCore.

### 3.30 More questions?

The guidelines and questions presented in this specification address queries received from a variety of EBUCore implementers. EBUCore can sustain more scenarios.

If you have additional questions on how to use EBUCore, please forward your queries to metadata@ebu.ch. You will receive personalised advice and answers will enrich this section of the EBUCore specification, with your permission.

### 4. Compliance

EBUCore doesn’t pretend to cover everyone’s needs in detail. EBUCore is an open framework allowing each user to adapt it to their own needs!

EBUCore is governed by Creative Commons’ Attribution-NonCommercial-ShareAlike3.0 Unported (CC BY-NC-SA 3.0).

You are free: to Share—to copy, distribute and transmit the work to Remix—to adapt the work incl. under your own namespace under the following conditions:

- **Attribution** — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Noncommercial** — You may not use this work for commercial purposes.
- **Share Alike** — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

Noncommercial: EBUCore can be adapted and integrated into commercial products but shall not be subject to a specific financial license.

EBUCore is flexible and adaptable in nature. For example, thanks to the extended nature of the ‘Part’ element, description can be implemented in different ways. Each implementer will define best practice, incl. additional compliance rules specific to its implementation and own domain of interoperability.
5. **Maintenance**

The EBU Core Metadata Set is maintained by the EBU and suggestions for corrections or additions can be made by mailing to (metadata@ebu.ch).

Contributions will be subject to peer review by the metadata experts participating in EBU metadata Strategic Programmes and projects (http://tech.ebu.ch/groups/mim).

6. **Download Zone**

<table>
<thead>
<tr>
<th>Filename</th>
<th>Doc. Description</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.ebu.ch/metadata/schemas/EBUCore/ebucore.xsd">http://www.ebu.ch/metadata/schemas/EBUCore/ebucore.xsd</a></td>
<td>XML Schema</td>
<td>ebucore.xsd</td>
</tr>
<tr>
<td><a href="https://www.ebu.ch/metadata/schemas/EBUCore/documentation/ebucore.html">https://www.ebu.ch/metadata/schemas/EBUCore/documentation/ebucore.html</a></td>
<td>HTML documentation</td>
<td>Navigate through the online documentation starting from the ebuCoreMain element.</td>
</tr>
<tr>
<td><a href="https://www.ebu.ch/metadata/ds/EBUcs/ebuCoreMainDocumentation/EBUCore/ebucoreCS/EBU_cs_p.zip">https://www.ebu.ch/metadata/ds/EBUcs/ebuCoreMainDocumentation/EBUCore/ebucoreCS/EBU_cs_p.zip</a></td>
<td>EBU Classification Schemes</td>
<td>Periodically updated list of EBU Classification Schemes in xml</td>
</tr>
<tr>
<td><a href="https://www.ebu.ch/metadata/ontologies/ebucore/ebucore.rdf">https://www.ebu.ch/metadata/ontologies/ebucore/ebucore.rdf</a> (ontology)</td>
<td>RDF Schema</td>
<td>ebusc.rdf</td>
</tr>
<tr>
<td><a href="https://www.ebu.ch/metadata/ontologies/skos/EBU_cs_skos_p.zip">https://www.ebu.ch/metadata/ontologies/skos/EBU_cs_skos_p.zip</a></td>
<td>EBU SKOS Classification Schemes</td>
<td>Periodically updated list of EBU Classification Schemes in RDF</td>
</tr>
</tbody>
</table>

7. **Useful links**

- AES (http://www.aes.org)
- Dublin Core (http://dublincore.org)
- EBU Metadata (http://tech.ebu.ch/metadata/)
- EBU Loudness (http://tech.ebu.ch/docs/tech/tech3343.pdf)
- EBU BWF (http://tech.ebu.ch/docs/tech/tech3285.pdf)
- EBU ADM (http://tech.ebu.ch/docs/tech/tech3364.pdf)
- EUScreen (http://www.euscreen.eu)
- FIMS (http://fims.tv)
- IOC - International Olympic Committee (http://www.olympic.org/uk/sports/)
- W3C SKOS (http://www.w3.org/2004/02/skos/)
- ISO (http://www.iso.org)
- ITU
- IETF
  - RFC 3339 (Date and time on the Internet): http://tools.ietf.org/html/rfc3339
- IANA MIME Type: http://www.iana.org/assignments/media-types/
- Thesaurus of Geographic Names: http://www.getty.edu/research/tools/vocabulary/tgn/index.html
- Mediainfo https://mediaarea.net/fr/MediaInfo/Download/Source
8. Bibliography

- EBU Technical Information I36-2003 - Metadata Implementation considerations for Broadcasters
- EBU Tech 3293-2001 - Core Metadata Set for Radio Archives
- EBU Tech 3295 - P-META Metadata Library
- EBU Tech 3343 - Practical guidelines for Production and Implementation of EBU R 128
- EBU Tech 3351 - Class Conceptual Data Model
- EBU Tech 3364 - Audio Definition Model (ADM)
- ITU-R BS.2076-0 - Audio Definition Model (ADM)
Annex A: EBUCore Metadata Set Schema

The EBUCore Metadata schema is used to generate compliant EBUCore Metadata instances formed of an ebuCoreMain document.

The ebuCoreMain document contains several attributes required to contribute to OAI (Open Archive Initiative) for metadata harvesting. These attributes include the name of the schema (in case the schema location urn would not be present), the version of the schema used to generate the document, the date and time of last modification of the document and a unique identifier associated to the document. The name of the contributing archive is given by the metadata provider’s organisation name or ID.

Resource related information is provided by instantiating the coreMetadata element.

The reference schema is available from the download links in §6 (Download Zone) of this document.
Annex B: EBUCore and Semantic Web

The EBUCore RDF schema (so called EBUCore ontology) is a semantic alternative to the EBUCore XML schema. The EBUCore ontology has been designed to provide classes and properties in complement to the minimum set defined in the EBU Class Conceptual Data Model (Tech 3351 - CCDM). The business object and resource names have been selected to offer maximum harmonisation with other related ontologies. The hierarchy of the EBUCore RDF class model reflects the audiovisual objects that can be met in audiovisual production. The ontology offers all the properties defined in EBUCore XML schema still providing additional expressivity.

The EBUCore RDF ontology can be accessed from the following locations as permanent dereferencable resources. The server has been setup to allow client requests for xml/rdf or html content with a priority to the documentation.


More information on server setups can be found at the following address (Best Practice Recipes for Publishing RDF Vocabularies). The EBU server has been setup following Recipe 3 with a default to the html documentation.

[http://www.w3.org/TR/swbp-vocab-pub/](http://www.w3.org/TR/swbp-vocab-pub/)

Important Notes:

- The EBUCore ontology is now mostly RDF, which allows to instantiate properties with resources or literals. This approach offers great interoperability with various communities.
- SubClasses inherit properties from upper classes;
- Custom classes can be added;
- Custom properties can also be added but adding classes and using predefined EBUCore properties is recommended;
- Ids are intrinsic properties of classes (defined by their URIs) but additional identifiers can also be provided;
- Types, formats, status (see the attribute groups in the XML schema) shall be implemented through the definition of sub-classes (e.g. manufacturer would be defined as a sub-class of organisation if the defined as such in the organisation typeGroup attributes);
- Terms that can be derived from Classification Schemes, such as Genre or Role, shall be treated as Classes. If expressed in SKOS, such Classes are sub-classes of the SKOS Concept Class. If a term identifier (e.g. a SKOS Concept) is not available, additional properties associated to the Class can be used through a blank node. The EBUCore imports the SKOS ontology. In this implementation, e.g. Genre is declared as a subclass of skos:Concept.

The EBUCore RDF ontology is referenced as a Linked Open Vocabulary (Linked Open Vocabulary) ([http://lov.okfn.org/dataset/lov/vocabs/ebucore](http://lov.okfn.org/dataset/lov/vocabs/ebucore)) and also as RDF-Vocab for Ruby developers ([https://github.com/ruby-rdf/rdf-vocab](https://github.com/ruby-rdf/rdf-vocab)).
Annex C: Applying EBU Tech 3364’s data model in EBUCore

C.1 Channel based example (extract from Tech 3364)

The most common use of audio is still channel-based, where tracks within a file each represent a static audio channel. This example demonstrates how to define two tracks, streams and channels; and a pack for stereo. The track and stream definitions are for PCM audio. Two objects are defined, both stereo, but containing different content so there are 4 tracks used. This example uses a programme called ‘Documentary’ containing ‘Music’ and ‘Speech’ each defined as separate stereo objects.

The format-related elements in this example represent a tiny subset of the standard reference set of definitions. In practice, this XML code would be part of the standard reference file and would not have to be included in the BWF file. All that would be required is a <chna> chunk with the references to the audioTrackFormats and audioPackFormats and any extra XML required for audioObject, audioContent and audioProgramme.

Table C.1: Summary of model elements

<table>
<thead>
<tr>
<th>Element</th>
<th>ID</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audioTrackFormat</td>
<td>AT_00010001_01</td>
<td>PCM_FrontLeft</td>
<td>Defines track as PCM</td>
</tr>
<tr>
<td>audioTrackFormat</td>
<td>AT_00010002_01</td>
<td>PCM_FrontRight</td>
<td>Defines track as PCM</td>
</tr>
<tr>
<td>audioStreamFormat</td>
<td>AS_00010001</td>
<td>PCM_FrontLeft</td>
<td>Defines stream as PCM</td>
</tr>
<tr>
<td>audioStreamFormat</td>
<td>AS_00010002</td>
<td>PCM_FrontRight</td>
<td>Defines stream as PCM</td>
</tr>
<tr>
<td>audioChannelFormat &amp;</td>
<td>AC_00010001</td>
<td>FrontLeft</td>
<td>Describes channel as front left with a position and speaker reference</td>
</tr>
<tr>
<td>audioBlockFormat</td>
<td>AB_00010001_00000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>audioChannelFormat &amp;</td>
<td>AC_00010002</td>
<td>FrontRight</td>
<td>Describes channel as front right with a position and speaker reference</td>
</tr>
<tr>
<td>audioBlockFormat</td>
<td>AB_00010002_00000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>audioPackFormat</td>
<td>AP_00010002</td>
<td>Stereo</td>
<td>Defines a stereo pack referring to two channels.</td>
</tr>
</tbody>
</table>
<?xml version="1.0" encoding="UTF-8"?>
<ebuCoreMain xmlns="urn:ebu:metadata:schema:ebuCore_2014"
xsi:schemaLocation="urn:ebu:metadata:schema:ebuCore_2014 EBU_CORE_20131014/EBU_CORE_20131014.xsd"
<coreMetadata>

<format>

<!-- audio programme -->
<audioProgramme audioProgrammeID="APG_1001" audioProgrammeName="Documentary">
  <audioContentIDRef>ACN_1001</audioContentIDRef>
  <audioContentIDRef>ACN_1002</audioContentIDRef>
</audioProgramme>

<!-- audio content -->
<audioContent audioContentID="ACN_1001" audioContentName="Music">
  <audioObjectIDRef>AQ_1001</audioObjectIDRef>
  <loudnessMetadata>
    <integratedLoudness>-28.0</integratedLoudness>
  </loudnessMetadata>
</audioContent>

<audioContent audioContentID="ACN_1002" audioContentName="Speech">
  <audioObjectIDRef>AQ_1002</audioObjectIDRef>
  <loudnessMetadata>
    <integratedLoudness>-23.0</integratedLoudness>
  </loudnessMetadata>
</audioContent>

<!-- audio object -->
<audioObject audioObjectName="Music" start="00:00:00,00000">
  <audioPackFormatIDRef>AP_00010002</audioPackFormatIDRef>
  <audioTrackUIDRef>ATU_00000001</audioTrackUIDRef>
</audioObject>

<audioObject audioObjectName="Speech" start="00:00:00,00000">
  <audioPackFormatIDRef>AP_00010002</audioPackFormatIDRef>
  <audioTrackUIDRef>ATU_00000003</audioTrackUIDRef>
</audioObject>

<!-- audio pack -->
<audioPackFormat audioPackFormatName="Stereo" typeDefinition="DirectSpeakers">
  <audioChannelFormatIDRef>AC_00010001</audioChannelFormatIDRef>
  <audioChannelFormatIDRef>AC_00010002</audioChannelFormatIDRef>
</audioPackFormat>

<!-- audio channel -->
<audioChannelFormat audioChannelName="FrontLeft" typeLabel="0001" typeDefinition="DirectSpeakers">
  <speakerLabel>M+30</speakerLabel>
  <position coordinate="azimuth">30.0</position>
  <position coordinate="elevation">0.0</position>
  <position coordinate="distance">1.0</position>
</audioChannelFormat>

<audioChannelFormat audioChannelName="FrontRight" typeLabel="0001" typeDefinition="DirectSpeakers">
  <speakerLabel>M-30</speakerLabel>
  <position coordinate="azimuth">-30.0</position>
  <position coordinate="elevation">0.0</position>
  <position coordinate="distance">1.0</position>
</audioChannelFormat>

<!-- audio stream -->
<audioStreamFormat audioStreamFormatName="PCM_FrontLeft" formatLabel="0001" formatDefinition="PCM">
C.2 Object based example (extract from Tech 3364)

To demonstrate how the ADM can be used in object-based audio here is a simple example using a single object. This example uses multiple audioBlockFormats within an audioChannelFormat to describe the dynamic properties of an object called “Car”. The audioBlockFormats uses the start and duration attributes to frame the time dependent metadata, thus allowing the object’s position to move in space.

Table C.2: Summary of model elements

<table>
<thead>
<tr>
<th>Element</th>
<th>ID</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audioTrackFormat</td>
<td>AT_00031001_01</td>
<td>PCM_Car1</td>
<td>Defines track as PCM</td>
</tr>
<tr>
<td>audioStreamFormat</td>
<td>AS_00031001</td>
<td>PCM_Car1</td>
<td>Defines stream as PCM</td>
</tr>
<tr>
<td>audioChannelFormat</td>
<td>AC_00031001</td>
<td>Car1</td>
<td>Describes channel as an object type containing 3 blocks with different positional metadata in each.</td>
</tr>
<tr>
<td>audioBlockFormat</td>
<td>AB_00031001_00000001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AB_00031001_00000002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AB_00031001_00000003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>audioPackFormat</td>
<td>AP_00031001</td>
<td>Car</td>
<td>Defines a pack referring to one channel.</td>
</tr>
<tr>
<td>audioObject</td>
<td>AO_1001</td>
<td>Car</td>
<td>Object for ‘Car, stereo format’</td>
</tr>
<tr>
<td>audioContent</td>
<td>ACN_1001</td>
<td>Cars</td>
<td>‘Cars’ content</td>
</tr>
<tr>
<td>audioProgramme</td>
<td>APG_1001</td>
<td>CarsSounds</td>
<td>Programme ‘CarsSounds’ containing ‘Cars’ content</td>
</tr>
</tbody>
</table>
<?xml version="1.0" encoding="UTF-8"?>
<ebuCoreMain
xmlns="urn:ebu:metadata-schema:ebuCore_2014"
xmlls:instance="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ebu:metadata-schema:ebuCore_2014 EBU_CORE_20131014/EBU_CORE_20131014.xsd"
xmlns:dc="http://purl.org/dc/elements/1.1/">
<coreMetadata>
<format>
  <audioFormatExtended>
    <!-- audio programme -->
    <audioProgramme audioProgrammeId="APG_1001" audioProgrammeName="CarsSounds">
      <audioContentIDRef>ACN_1001</audioContentIDRef>
    </audioProgramme>

    <!-- audio content -->
    <audioContent audioContentID="ACN_1001" audioContentName="Cars">
      <audioObjectIDRef>AQ_1001</audioObjectIDRef>
      <loudnessMetadata>
        <integratedLoudness>-23.0</integratedLoudness>
      </loudnessMetadata>
    </audioContent>

    <!-- audio object -->
    <audioObject audioObjectId="AQ_1001" audioObjectName="Car">
      <audioPackFormatIDRef>AP_00031001</audioPackFormatIDRef>
      <audioTrackUIDRef>ATU_00000001</audioTrackUIDRef>
    </audioObject>

    <!-- audio pack -->
    <audioPackFormat audioPackFormatID="AP_00031001" audioPackFormatName="Car" typeLabel="0003" typeDefinition="Objects">
      <audioChannelFormatIDRef>AC_00031001</audioChannelFormatIDRef>
      <audioTrackFormatIDRef>AS_00031001_AT_01</audioTrackFormatIDRef>
    </audioPackFormat>

    <!-- audio channel -->
    <audioChannelFormat audioChannelFormatID="AC_00031001" audioChannelFormatName="Car1" typeLabel="0003" typeDefinition="Objects">
      <audioBlockFormat audioBlockFormatID="AB_00031001_00000001" rtime="00:00:00.00000" duration="00:00:05.00000">
        <position coordinate="azimuth">-22.5</position>
        <position coordinate="elevation">6.0</position>
        <position coordinate="distance">1.0</position>
      </audioBlockFormat>

      <audioBlockFormat audioBlockFormatID="AB_00031001_00000002" rtime="00:00:05.00000" duration="00:00:10.00000">
        <position coordinate="azimuth">-24.5</position>
        <position coordinate="elevation">6.0</position>
        <position coordinate="distance">1.0</position>
      </audioBlockFormat>

      <audioBlockFormat audioBlockFormatID="AB_00031001_00000003" rtime="00:00:15.00000" duration="00:00:20.00000">
        <position coordinate="azimuth">-26.5</position>
        <position coordinate="elevation">7.0</position>
        <position coordinate="distance">0.9</position>
      </audioBlockFormat>
    </audioChannelFormat>

    <!-- audio stream -->
    <audioStreamFormat audioStreamFormatID="AS_00031001" audioStreamFormatName="PCM_Car1" formatLabel="0001" formatDefinition="PCM">
      <audioChannelFormatIDRef>AC_00031001</audioChannelFormatIDRef>
      <audioTrackFormatIDRef>AS_00031001_AT_01</audioTrackFormatIDRef>
    </audioStreamFormat>
  </audioFormatExtended>
</format>
</coreMetadata>
C.3 Scene based example (extract from Tech 3364)

The other main type of audio is scene-based where the audio channels are representing Ambisonic/HOA components. Their use is very similar to that of the channel-based approach with the main difference being the parameters used within audioBlockFormat. This example shows a simple 1st order Ambisonic (using the N3D method) configuration using 4 channels mapped onto 4 tracks. Like the channel-based approach, the format elements would be defined in a standard reference file so in practice would not need to be included in the BWF file itself.

<table>
<thead>
<tr>
<th>Table C.3: Summary of model elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>audioTrackFormat</td>
</tr>
<tr>
<td>audioTrackFormat</td>
</tr>
<tr>
<td>audioTrackFormat</td>
</tr>
<tr>
<td>audioTrackFormat</td>
</tr>
<tr>
<td>audioStreamFormat</td>
</tr>
<tr>
<td>audioStreamFormat</td>
</tr>
<tr>
<td>audioStreamFormat</td>
</tr>
<tr>
<td>audioStreamFormat</td>
</tr>
<tr>
<td>audioChannelFormat &amp; audioBlockFormat</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>audioChannelFormat &amp; audioBlockFormat</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>audioChannelFormat &amp; audioBlockFormat</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>audioChannelFormat &amp; audioBlockFormat</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>audioPackFormat</td>
</tr>
</tbody>
</table>
<coreMetadata>

</coreMetadata>