EBU

OPERATING EUROVISION AND EURORADIO

TECH 3370

EBU-TT, PART 3
LIVE SUBTITLING APPLICATIONS

SYSTEM MODEL AND CONTENT
PROFILE FOR AUTHORING
AND CONTRIBUTION

VERSION: 1.0

SOURCE: SP/MIM - XML SUBTITLES

Geneva
May 2017

There are blank pages throughout this document. This document is paginated for two sided printing

Conformance Notation

This document contains both normative text and informative text.

All text is normative except for that in the Introduction, any section explicitly labelled as 'Informative' or individual paragraphs which start with 'Note:'.

Normative text describes indispensable or mandatory elements. It contains the conformance keywords 'shall', 'should' or 'may', defined as follows:

'Shall' and 'shall not':	Indicate requirements to be followed strictly and from which no deviation is permitted in order to conform to the document.
'Should' and 'should not':	Indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others. OR indicate that a certain course of action is preferred but not necessarily required. OR indicate that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.
'May' and 'need not':	Indicate a course of action permissible within the limits of the document.

Default identifies mandatory (in phrases containing "shall") or recommended (in phrases containing "should") presets that can, optionally, be overwritten by user action or supplemented with other options in advanced applications. Mandatory defaults must be supported. The support of recommended defaults is preferred, but not necessarily required.

Informative text is potentially helpful to the user, but it is not indispensable and it does not affect the normative text. Informative text does not contain any conformance keywords.

A conformant implementation is one which 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.

Contents

Status of this document (Informative)

This document is a stable document and may be used as reference material or cited from another document.

This document is part of a series of EBU-TT (EBU Timed Text) documents. The full list of published and planned EBU-TT documents is given below.

Part 1: EBU-TT Subtitling format definition (EBU Tech 3350)
Introduction to EBU-TT and definition of the XML based format.

Part 2: STL (Tech 3264) Mapping to EBU-TT (EBU Tech 3360)
How EBU-TT provides backwards compatibility with EBU STL.

Part 3: EBU-TT in Live Subtitling applications: system model and content profile for authoring and contributions (EBU Tech 3370)
How to use EBU-TT for the production and contribution of live subtitles.

EBU-TT WebSocket Carriage Specification (EBU Tech 3370s1)
Carriage of EBU-TT Part 3 over WebSocket (this document).

EBU-TT, Part D (EBU Tech 3380)
EBU-TT content profile for TTML that can be used for the distribution of subtitles over IP based networks.

Carriage of EBU-TT-D in ISOBMFF (EBU Tech 3381)
How EBU-TT-D can be stored using the storage format of the ISO Base Media File Format (ISO/IEC 14496-12).

EBU-TT, Part M: Metadata Definitions (EBU Tech 3390)
Definition of metadata elements and attributes for use in EBU-TT documents

EBU-TT Annotation
How EBU-TT can be used in future scenarios for ‘authoring of intent’.

EBU-TT User Guide
General guide (‘How to use EBU-TT’).

EBU-TT in Live Subtitling applications: System model and content profile for authoring and contributions

EBU Committee	*First Issued*	*Revised*	*Re-issued*
TC	2015	2017	

Keywords: EBU Timed Text, EBU-TT, Subtitle, STL, XML, W3C, TTML, DFXP, caption, encoder, live.

Scope (Informative)

The topic of live subtitle or caption authoring, routing and encoding is large. Organisations such as broadcasters, access service providers and other content providers face a variety of challenges ranging from the editorial, for example word accuracy and rate, to the technical, for example how the text data is routed from the author to the encoder, what format it should be in and how it can be configured and monitored. The classical way to address such a large "problem space" is to divide it up into more easily solvable constituent parts. This approach is taken here.

This document describes how EBU Timed Text (EBU-TT) can be used in a broadcasting environment to carry subtitles that are created in real time ("live" or from a prepared file) from an authoring station to an encoder prior to distribution, via intermediate processing units. It does this by specifying:

- a system model consisting of processing nodes that pass streams of subtitles along a chain;
- a content profile based on EBU-TT Part 1 (EBU Tech 3350) [1] specifying the data format of each document in a live subtitle stream;
- a mechanism by which content providers can model and potentially improve synchronisation between the subtitles and the audio to which they relate;
- a mechanism for interoperable management of the handover from one subtitler to the next, to generate a single output subtitle stream.
- an extension facility to allow other types of live documents to be defined in future specifications, for example for passing messages between subtitlers

This document also provides useful options for mixing the play out of prepared subtitle documents and live subtitles, a problem that arises for all broadcast channels whose output is not either 100% pre-recorded or 100% live.

Other documents will address the carriage of live subtitle streams, i.e. the requirements of protocols for passing live subtitle streams between nodes and how specific protocols meet those requirements. This document defines what such a carriage specification document should consider.

Authoring conventions, for example the use of colour to identify speakers, are not directly addressed in this document; however care has been taken to ensure that the technical solutions presented here can support a variety of conventions. System setup, configuration, management, resilience, monitoring and recovery are likewise addressed indirectly by modelling the potential architectures in the abstract and designing the data format to support those architectures.

No requirement is presented here for any subtitle presentation device used by an audience member to behave differently in the presence of live subtitles compared to prepared subtitles: rather, it is

envisaged that any processing required to manage the two cases, and the transitions between them, occurs in the broadcaster domain. However the use of SMPTE time code is not supported.

1. Introduction (Informative)

EBU-TT Live specifies how to transfer live and non-live subtitles between production systems, e.g. from an authoring station to encoder. It builds on EBU-TT Part 1, but relaxes the requirements for layout and styling. For home delivery, other specifications, such as EBU-TT Part D (EBU Tech 3380) [2], are used.

1.1 EBU-TT as exchange format for live and prepared subtitles

EBU-TT as defined in EBU-TT Part 1 is intended as a general purpose exchange format for subtitles. The workflow diagram in Part 1 is extended by this document to include live subtitle exchange.

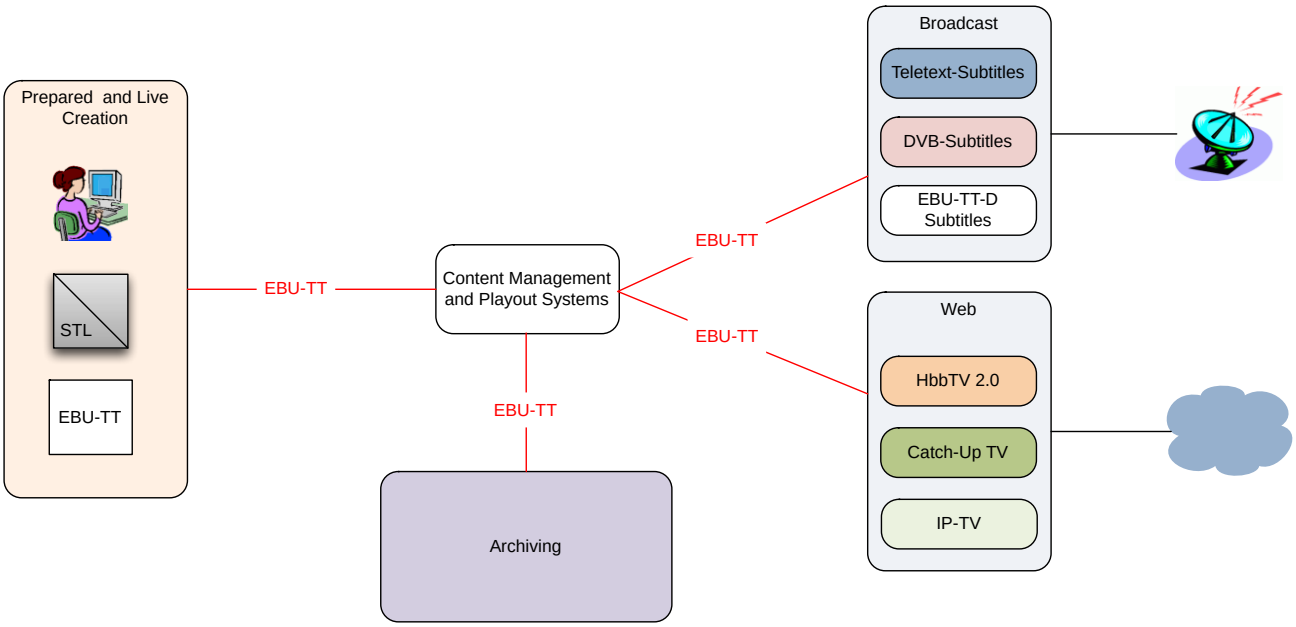


Figure 1: Subtitle workflow with EBU-TT Live

Whereas EBU-TT Part 1 focuses on document exchange for prepared subtitle documents that apply to completed programmes, EBU-TT Part 3 extends this workflow to incorporate the streaming of subtitles that may be created live in real time for programmes that could also be live. See Figure 1.

The benefits of EBU-TT Part 3 relative to other existing ways to transfer live streams of subtitles include:

- It can be used both as a source for and be generated from other EBU-TT and TTML based subtitle formats, using XML processing techniques. It is therefore possible to build a complete subtitle infrastructure based on EBU-TT throughout.
- It is an open standard.
- In general it is stateless: no prior knowledge of documents in a stream is needed to begin processing live subtitles to generate the intended output.
- It is extensible.
- It can be used in a variety of levels of complexity to suit implementation needs.
- It can be used within existing subtitle routing infrastructures, subject to constraints that could apply such as maximum data rates.

1.2 Summary of key points

The subtitles are carried in **Documents**, each of which is a valid W3C TTML document. Besides the subtitle's text it can contain styling, layout, timing and additional metadata information.

```
<tt xmlns:ebuttp="urn:ebu:tt:parameters" xmlns="http://www.w3.org/ns/ttml" … xml:lang="en"
ttp:cellResolution="40 24" ttp:clockMode="local" ttp:timeBase="clock" ebuttm:authoringDelay="5s"
ebuttp:sequenceIdentifier="testSequence_1441882303" ebuttp:sequenceNumber="1858107">
  <head>
    <styling>
      <style xml:id="s1" tts:backgroundColor="black" tts:color="lime" tts:fontSize="1c 2c"/>
    </styling>
    <layout>
      <region xml:id="r0" tts:extent="37c 4c" tts:origin="3c 20c"/>
    </layout>
  </head>
  <body begin="10:29:32.36">
    <div>
      <p region="r0">
        <span style="s1"> Hello. Have you not done something <br/>
 you should have done, for me? </span>
      </p>
    </div>
  </body>
</tt>
```

Figure 2: Example EBU-TT Part 3 document

One or more Documents together form a **Sequence**. Each Document indicates to which Sequence it belongs using a Sequence Identifier. The order is set by a Sequence Number.

```
<tt … ebuttp:sequenceIdentifier="testSequence_1441882303" ebuttp:sequenceNumber="1858107">
…
</tt>


<tt … ebuttp:sequenceIdentifier="testSequence_1441882303" ebuttp:sequenceNumber="1858108">
…
</tt>


<tt … ebuttp:sequenceIdentifier="testSequence_1441882303" ebuttp:sequenceNumber="1858109">
…
</tt>

<tt
 … ebuttp:sequenceIdentifier="testSequence_1441882303" ebuttp:sequenceNumber="1858110">
```

Figure 3: Example of sequence identifier and sequence numbers

The concept of sequence identification is separate to **service** identification. Document metadata is included to allow authors to identify the services for which the sequence is intended to carry subtitles, also known as the broadcast channel etc.

In EBU-TT Live sequences of live documents are transferred between **Nodes**. Such transfers are called **streams**. Nodes can consume, process and/or output Documents. Different types of Node can send or receive varying numbers of Streams to or from other Nodes. Some examples are shown below.

Producer
e.g. authoring station

Improver
e.g. spellchecker

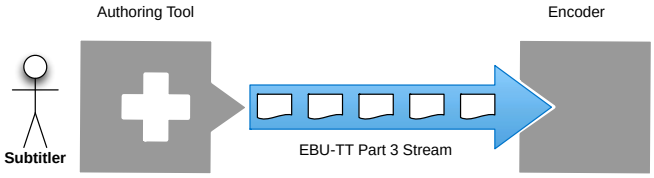
Processing Nodes output streams that differ from those at the input.

Distributing Node

Consumer
e.g. an encoder

Passive Nodes simply 'pass through' streams from input to output.

When Nodes transfer sequences between each other, we call these transfers **Streams**.



Documents can use different types of timing. Only one Document can be active at any given time. If different Documents overlap in time, the Document with the highest Sequence number 'wins'.

Explicit timing

When a Document has explicit times set (using ***begin***, ***end*** and/or ***dur*** attributes), and is available before its begin time, it becomes active at its begin time, until the next document is active, or it reaches its end.

Documents may be sent before they become active; documents may be re-evaluated later, for example to archive a corrected version of the subtitles after broadcast, or to retain precise timings from EBU-TT Part 1 documents.

Implicit timing

If no ***begin*** or ***end*** attributes are set in the Document, subtitles will be active as soon as they are received, until the next Document is active or the ***dur*** has been reached, if set.

The typical use case is sending subtitles as fast as possible for live subtitling. This simple scheme may not be optimal, because it does not support all the possible use cases, for example creating archive versions is more difficult.

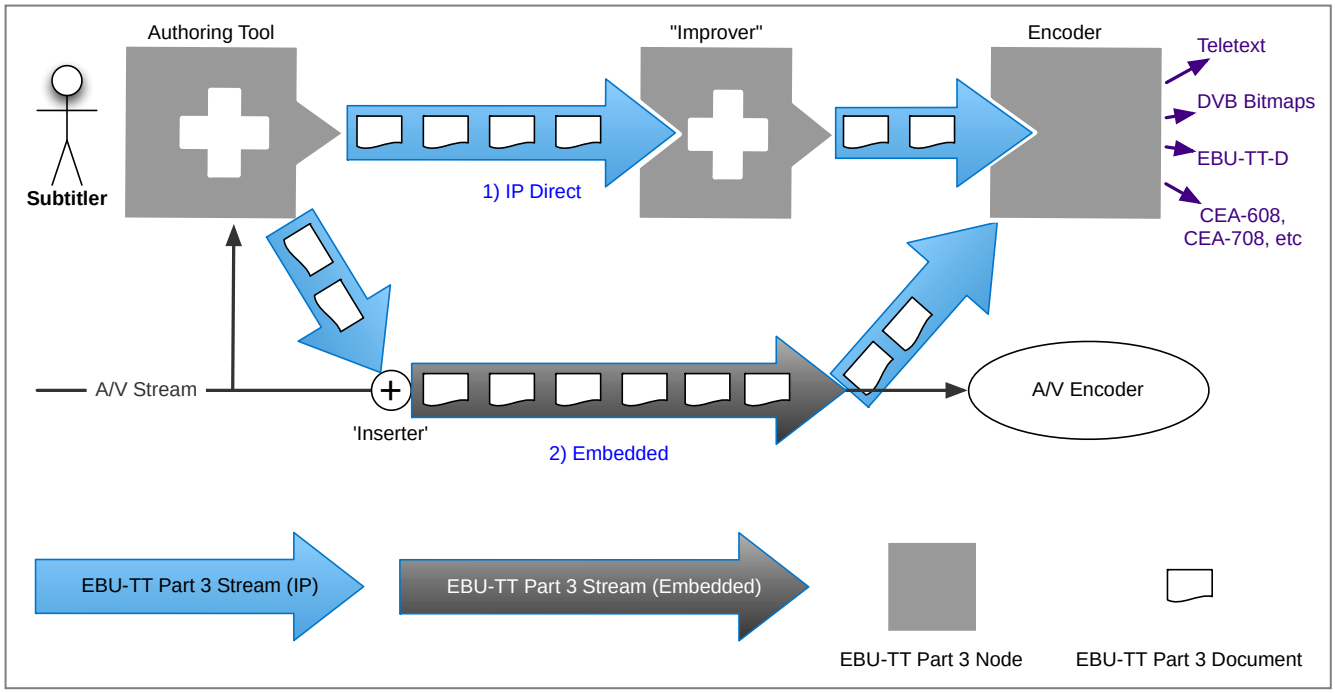


Figure 4: Schematic of simple use case showing an authoring tool generating a stream of EBU-TT Part 3 live subtitles.

Figure 4 illustrates a simple example use case in which a subtitler uses an authoring tool to create a stream of live subtitles. Those are then transferred either: 1) via a direct IP connection to an Improver and then on to an encoder; or 2) embedded into an audio/visual stream with an inserter and then de-embedded for passing to an encoder. A potential addition to this workflow would be an additional connection for example from the Improver to an Archiver to create an EBU-TT Part 1 document for later reuse.

The Improver defined in this document is a processing node that could, for example, insert a defined compensating delay, check content for correct spelling, ensure that prohibited code points are not propagated or perform other transformations to generate output suitable for encoding.

1.3 Example scenarios

The following examples represent typical real world scenarios in which documents and nodes that conform to this specification can be used, which are fully defined by this specification. Note that the scenario numbers do not increase monotonically: further example scenarios that are not fully defined by this specification are listed in Annex E, with which these form a set.

Example 1: Handover orchestration

Each subtitler in a group authors subtitles for part of a single programme; each group member takes a turn to contribute for a period of time before handing over to the next subtitler.

Each subtitler creates a distinct sequence of subtitles for their turn. Each of those input sequences has a different sequence identifier. The authoring stations emit the sequences as streams. As part of an externally orchestrated handover process a ‘handover manager’ node receives all the streams, combines them and emits a new continuous stream. This new output stream’s sequence has a different sequence identifier from each of the input sequences.

Incidentally, each subtitler may subscribe to, and view the others’ streams to assist with the handover orchestration.

Example 2: Author and correct

A pair of subtitlers authors and corrects live subtitles. The first subtitler creates a sequence using an authoring tool. The second subtitler receives a stream of that sequence and manipulates an

Improver Node that allows the sequence to be modified and then issues a new sequence with a different sequence identifier from the input sequence, for consumption downstream.

Example 4: Timing improvement

An Improver Node receives a stream and a continuous audio track in a reference time frame. The Improver analyses the audio and subtitles and creates a new sequence whose contents are time-aligned relative to the audio track's time frame, using time stamps from a common clock source. The new sequence is issued as a stream with a new sequence identifier.

Example 10: Retrospective corrections

A subtitler authors a live subtitle stream whose sequence is archived for later reuse. On noticing an error the subtitler issues a retrospectively timed correction. The archive process uses data within the sequence to apply the correction such that the error it corrects is not apparent within the generated archive document.

1.4 Document Structure

This document is structured as follows:

Section 1 comprises this Introduction.

Section 2 defines terms and introduces core concepts.

Section 3 defines the requirements for documents that conform to this specification.

Section 4 defines the requirements for nodes that conform to this specification.

2. Definitions and Concepts

2.1 Definition of terms

Author

The term "author" describes a person or system that creates a stream of live subtitle data based on observations of some other media, for example by listening to a programme audio track.

Captions and subtitles

The term "captions" describes on screen text for use by deaf and hard of hearing audiences. Captions include indications of the speakers and relevant sound effects.

The term "subtitles" describes on screen text for translation purposes.

For easier reading only the term "subtitles" is used in this specification as the representation of captions and subtitles is identical here.

In this specification the term "captions" may be used interchangeably for the term "subtitles" (except where noted).

Document

A subtitle document conformant to this specification.

Document availability time

The time when a document becomes available for processing.

Document cache

The set of documents retained by a node, for example for processing. See also §2.3.1.5.1.

Document resolved begin time

The time when a document becomes active during a presentation. Note: This term is used in the same sense as "resolved begin time" is used in SMIL 2.1 [3], when applied to a document.

Document resolved end time

The time when a document becomes inactive during a presentation. Note: This term is used in the same sense as "resolved end time" is used in SMIL 2.1 when applied to a document.

Encoder

The term “encoder” refers to a system that receives a stream of live subtitle data and somehow encodes it into a format suitable for use downstream, for example EBU-TT Part D. Note that some encoders may also package the encoded output data into other types of stream e.g. MPEG DASH.

Inserter

A unit that embeds subtitle data into an audio/visual stream. This is in common use in current subtitling architectures.

Live Document

Any entity defined to be an EBU-TT Live Document by an EBU specification.

Node

A unit that creates, emits, receives or processes one or more sequences.

Node identifier

The unique identifier of a Node.

Presentation

In this document the term 'presentation' is used in the sense in which it is used in SMIL 2.1.

Presentation Processor

This term is used as defined in EBU Tech 3350: "A *Content Processor* which purpose is to layout, format, and render, i.e., to present, *Timed Text Markup Language* content by applying the presentation semantics defined in this specification."

Processing Context

The configuration and operating parameters of a node that processes a document.

Sequence

A set of related live documents each of which shares the same sequence identifier, for example the documents that define the subtitles for a single programme.

Sequence Begin

The start of the interval in which a sequence is presented is referred to as the *sequence begin*. Equivalent to the document begin in SMIL 2.1 of the first document in the sequence.

Sequence End

The end of the interval in which a sequence is presented is referred to as the *sequence end*. Equivalent to the document end in SMIL 2.1 of the last document in the sequence.

Sequence Duration

The difference between the sequence end and the sequence begin is referred to as the *sequence duration*.

Service identifier

An identifier used to uniquely identify a broadcast service, for example the HD broadcast of the broadcaster's main channel.

Stream

The transfer of a sequence.

Logical stream

A stream offered or provided by a node to zero or more nodes; identified by the source node identifier and sequence identifier.

Physical stream

A stream provided between two nodes, identified by the source node identifier, destination node identifier and sequence identifier.

2.2 System Model

The following abstract system model is defined; systems that are conformant with this specification shall meet the requirements within the system model and the node conformance section (see § 4):

Documents

- A document is a single entity conformant to this specification.
- Documents defined by this specification are live documents in the set of EBU-TT Live Documents.
- An implicitly timed document is one that contains no timing information and is considered active as soon as it has become available. It may also express a maximum duration after which it shall be deactivated.
- An explicitly timed document is one that contains timing information and whose activation time depends both on when it has become available and the timing information that it contains, resolved using its time base and a reference clock source.
- When presenting a sequence of documents, at each moment in time exactly zero or one document shall be active.
- If no document is active, or if a document with no content is active, no content shall be displayed.
- Two documents are considered identical if the result of the `fn:deep-equal` function in XFUNC [4] is true when both documents are provided as operands. Note that for valid serialised XML documents if a byte comparison of the two documents shows they are identical then `deep-equal` is expected also to return true, however there are cases where serialisations are not identical in a byte comparison but `deep-equal` correctly reports that they are identical, for example if XML comments have been added or attributes have been reordered.

Sequences

- A *sequence* is a set of related live documents, e.g. the documents that define the subtitles for a single programme.
- Sequences shall be considered distinct if they have had processing applied, even if the result of that processing is no change other than a known difference in quality, for example if the processing has checked the spelling of the text content.
- Every distinct sequence shall have a unique *sequence identifier*.
- A sequence of implicitly timed documents may be transformed, with knowledge of the documents' associated availability times, into a sequence of explicitly timed documents or a single document with explicit timings.

- A document shall be associated with exactly one sequence.
- Sequences do not have an explicit existence other than being the set of their constituent live documents; the sequence identifier shall be present within every document in order to make concrete the association with the document's sequence.
- Documents with the same sequence identifier shall contain a *sequence number*. Sequence numbers shall increase with the passage of time for each new document that is made available. Sequence numbers are used to resolve temporal overlaps: see § 2.3.1 below. In case a document is received with the same sequence identifier and sequence number as a previously received document the later document shall be discarded from processing. Note: if sequence numbers begin at 1 then, at an example nominal mean rate of 1 document per second the maximum sequence number that will fit within a 4 byte unsigned integer corresponds to a sequence duration of over 136 years. Sequences should begin at low sequence numbers such as 1 to avoid the possibility of an unwanted integer overflow.
- Every document in a sequence is by definition a valid and self-contained document conforming to this specification. NOTE: In general no knowledge of other documents is required to process it[1][2].
- Sequences may be stored: that is, the lifetime of the sequence is unbounded.

Every document in a sequence shall have an identical timing model as defined by using the same set of specified or absent values for the `ttp:timeBase` and `ttp:clockMode` attributes.

Nodes and streams

- A node is an EBU-TT Part 3 aware unit or mechanism that creates, emits, receives or processes one or more sequences.
- Nodes are identified. NOTE: the format of node identifiers is not defined here.
- A node may offer a *logical stream* as the transfer of a sequence to one or more nodes. A logical stream is identified by the source node's identifier and the sequence identifier.
- A *physical stream* is the transfer of a sequence between two nodes. It is identified by the pair of identifiers of the source node and the destination node and by the sequence identifier.
- Any number of nodes may process or consume the same logical stream: by definition the delivery of those streams requires one physical stream per destination node. NOTE: a node can provide multiple physical streams of the same sequence.
- A *processing node* emits sequence(s) that are distinct[3] from any of its input sequence(s). A creation node is a specialised processing node with zero input sequences.
- A *passive node* shall NOT modify input sequences and shall only emit sequences that are identical (including the sequence numbers) to the input sequence(s), for example nodes that are simply used for switching. A consumer node is a specialised passive node that does not emit any sequence.
- Streams are transient: that is, the lifetime of a stream is bounded by the period between the start of transfer (when the first document is transferred) and the end of transfer (when the last document is transferred). Streams may begin before the last document in the sequence has been generated - indeed it is envisaged that in normal operation documents

[1] Some specific kinds of processor may constitute exceptions, such as one that accumulates multiple documents together and combines them into a single one.

[2] For the purposes of efficiency, a processing node may store the preceding document for document comparison, e.g. when a subsequent document only changes the temporal validity of the document content.

[3] There may be little or no observable difference in content but the processing node may have the potential to modify the output stream; for example a spell-checker or profanity removal node may not make any changes most of the time but be called into action on an occasional basis. Nevertheless the output is considered to be different from the input because it has a logically different state, e.g. it is known not to contain any profanities, and it has a different sequence identifier.

within a sequence are generated dynamically and transferred in a stream with an undefined end time.

- The flow of a stream is unidirectional. Any 'back channel' reverse communication is external to the payload of the sequence[4].
- At any moment in the presentation of a sequence by a node exactly zero or one document in that sequence shall be temporally active. Note the logic defining which document is temporally active is defined in § 2.3.1 below.

Nodes are defined as abstract classes. Further detail of node behaviour is defined in § 4. See also Figure 11.

For correct temporal processing of streams, nodes are expected to maintain synchronisation between internal clocks used to compare documents' availability times with the computed times for the documents' contents.

Note that the use of SMPTE time base is prohibited in this version of this specification. This is because the combination of `ttp:timeBase="smpte"` and `ttp:markerMode="discontinuous"` implies that time expressions are merely markers that cannot be relied upon to increase monotonically. Therefore they are unsuitable for direct comparison purposes, which is a processing requirement for identifying temporal overlaps between documents. In practice any source of SMPTE timecode, for example timecode associated with video frames, even if intended to be continuous, may be subject to discontinuities for example at programme boundaries. Techniques for working around the absence of SMPTE time base are described in §2.3.1.

Figure 5 shows a UML class model illustrating the logical entities in this system model.

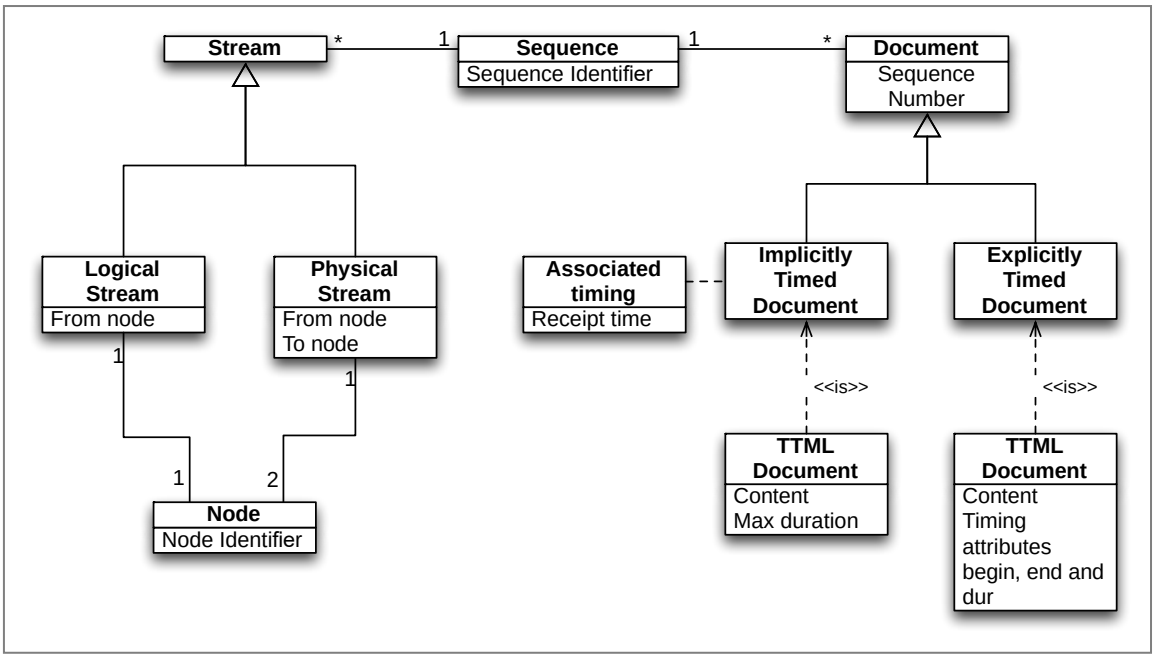


Figure 5: UML model showing system model logical entities

It is envisaged that some processing nodes will be purely software whereas others may require human intervention.

Sequence identifiers should not be used as service identifiers, such as broadcast television channel identifiers. For example consider the case of a 'simulcast' where a single video source is broadcast on more than one output service simultaneously, e.g. to support localised services or SD/HD

[4] Nodes can subscribe to multiple streams – see the Handover Manager node for example.

services. A subtitler can only observe a single media source at any moment, and generates a single sequence with a single sequence identifier, which could be subsequently processed, generating new sequences with different identifiers. Those sequences could be destined to be encoded for multiple output or destination services. Destination service identifiers may be carried as metadata within documents. Similarly the observed source service identifier may be carried as metadata within documents.

2.2.1 Other live documents

The definition of sequence deliberately allows for the possible future introduction of other kinds of live document than the document type specified here. Any such live document types will be published in EBU specifications.

Implementations of EBU-TT Live nodes should be designed to accommodate other live documents.

2.3 Timing and synchronisation

This section defines the temporal processing of a sequence of documents within a presentation, the management of delay in a live authoring environment and the use of reference clocks.

2.3.1 Document resolved begin and end times

Every document in a sequence has a time period during which it is active within a presentation, defined in EBU Tech 3350 as the Root Temporal Extent. At any single moment in time during the presentation of a sequence either zero documents or one document shall be active. The period during which a document is active begins at the document resolved begin time and ends at the document resolved end time.

An algorithm for computing the time when a document is active during a presentation has the following variables to consider:

- The document availability time, i.e. the time when each document becomes available. In a live authoring scenario it is typical for documents to become available during the presentation, soon after they have been authored. In a replay scenario all of the documents may be available prior to the presentation beginning.
- The earliest computed begin time in the document, calculated from the `begin` and `end` attribute values if present, according to the semantics of EBU Tech 3350.
- The latest computed end time in the document, calculated from the `begin` and `end` attribute values if present, according to the semantics of EBU Tech 3350.
- The value of the `dur` attribute if present.
- The document sequence number.
- Any externally specified document (de)activation times, such as the beginning of sequence presentation.

The definitions of the document resolved begin and end times below are derived from the following rules:

1. A document cannot become active until it is available, at the earliest.
2. The absence of any begin time implies that the document is active immediately;
3. The absence of any end time implies that the document remains active indefinitely;
4. The `dur` attribute on the `body` element imposes a maximum document duration relative to the point at which it became active.
5. If two documents would otherwise overlap temporally, the document with the greater sequence number supersedes the document with the lesser sequence number.

Note: the `timeContainer` attribute cannot be specified in EBU-TT Part 3 documents so `par` semantics apply as specified in EBU Tech 3350 § 10.2.4.

Note: It is not necessary in all processing contexts to resolve the document begin and end times. For example a processing node that checks text spelling only can do so without reference to the timing constructs defined in this section.

Note: In general the document time base, as specified by the `ttp:timeBase` attribute, can be transformed within a processing context to a value that can be compared with externally specified document activation times, for example the clock values when documents become available. Implementations that process documents that have `ttp:markerMode="discontinuous"` would not be able to assume the time values to be part of a monotonically increasing clock, but only as markers. This scenario is common with timecodes, i.e. when `ttp:timeBase="smpte"`. In this version of the specification support for `ttp:timeBase="smpte"` is absent; to accommodate operational scenarios in which SMPTE time code is in common use, several strategies are available including:

1. Use implicitly timed documents only, with the functional limitations implied;

2. In both the producer and any consumers of documents, use a common algorithm for converting time codes to values on a commonly available reference clock;

3. Embed values from a commonly available continuous reference clock (e.g. station clock, GPS, UTC etc) into any associated media streams and use those for synchronisation.

2.3.1.0.1 Definition of time values used for resolving document begin and end times

The rules for determining resolved begin and end times in this section require comparison of times that are potentially derived from different clock sources. For example the availability time of a document may be found by inspecting a local system clock whereas the earliest computed begin time in the document may be in a timebase relating to a different reference clock.

For the purpose of making these comparisons the following times shall be converted to values on the same timebase:

- document availability time;
- earliest computed begin time in the document;
- any externally specified document activation begin time;
- latest computed end time in the document;
- any externally specified document deactivation time.

The “earliest computed begin time” is defined as the earlier of a) the earliest computed begin time of any leaf element in the document and b) the earliest computed time corresponding to a specified `begin` attribute value on an element that either has no `end` attribute value or has an `end` attribute value that is later than the `begin` attribute value. Note that in the case that a root to leaf path contains elements all of which omit a `begin` attribute value this evaluates to the value zero on the document’s timebase.

The “latest computed end time” is defined as the latest computed end time corresponding to a specified `end` attribute value on an element that either has no specified `begin` attribute value or has an `end` attribute value that is later than the `begin` attribute value. Note that in the case that a root to leaf path contains elements all of which omit an `end` attribute this evaluates to the [SMIL] term `"undefined"`, i.e. the latest computed end time is not determined, and is effectively infinite for comparison purposes.

Note: It is syntactically permitted for an element to have a `begin` attribute value that is later than or equal to its `end` attribute value; in this case normally the element would be considered never to be active; this is why such elements are excluded from the calculation of the earliest computed begin time and the latest computed end time.

Note: The `dur` attribute is not used when computing the latest computed end time however it is used when computing the document resolved end time relative to the document resolved begin time – see §2.3.1.2 below.

See Annex B for informative worked examples.

2.3.1.1 Document resolved begin time

The document resolved begin time shall be the later of (a) the document availability time, (b) the earliest computed begin time in the document and (c) any externally specified document activation begin time, such as the beginning of sequence presentation.

See Annex C for informative worked examples.

2.3.1.2 Document resolved end time

The document resolved end time shall be the earlier of (a) the earliest document resolved begin time of all available documents in the sequence with a greater sequence number, (b) if and only if the `dur` attribute is present, the document resolved begin time plus the value of the `dur` attribute, (c) the latest computed end time in the document and (d) any externally specified document deactivation time, such as the end of sequence presentation.

See Annex C for informative worked examples.

2.3.1.3 Behaviour when no document is active

When no document is active a presentation processor shall NOT render any content.

Note: An encoder can be considered to be a specialised type of presentation processor. More generally, this applies to any consumer node.

2.3.1.4 Document creation and issuing strategies (informative)

The above rules allow for implementations to use different strategies for creating and issuing sequences. It is beyond the scope of this document to recommend the appropriate strategy for every circumstance. This section describes some of the strategies that could be employed.

The concepts of implicitly timed and explicitly timed documents, and their potential usage, are described. Further, a selection of document issuing strategies is described; this is not intended to encompass all of the possible strategies.

2.3.1.4.1 Implicitly timed documents

Documents that do not contain any `begin` or `end` attributes are described as having implicit timing (see SMIL 2.1 § 10.7.1). Such documents can contain a maximum duration, set as a `dur` attribute on the `body` element.

The absence of a begin time implies that the content is active as soon as the document becomes available. If neither an end time nor a duration is specified then the content will be active forever, or until the document becomes inactive, for example because a different document is activated in its place. See rule 5 in § 2.3.1 above.

Figure 6 shows a diagrammatic representation of the resolved begin and end times of implicitly timed documents.

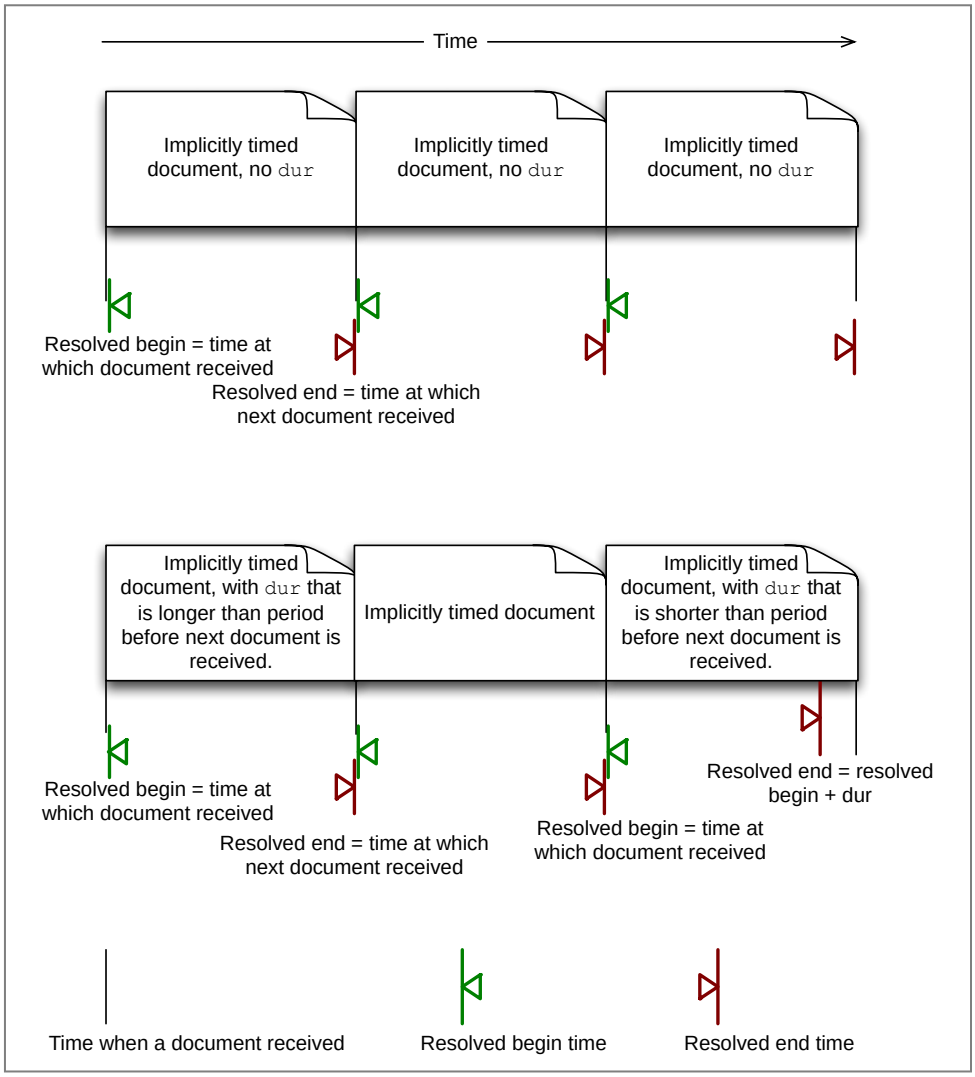


Figure 6: Diagram showing resolved begin and end times of implicitly timed documents with and without `dur` attributes

Use of documents with implicit timing restricts the available issuing strategies because it is not possible to include timed changes within them. In other words every change to the displayed content can only be contained in a new document.

It is neither possible to provide implicitly timed documents in advance of their required activation nor retrospective documents that revise earlier documents in the sequence since no time information is present within the documents to indicate such intentions.

A use case for implicitly timed documents is one where the timing for the document is defined by association with a specific frame of video or set of frames, where that association is inherent in the carriage mechanism, for example because the document is embedded in the VANC data area of an HD-SDI video stream, or in an ISOBMFF sample.

A second use case for implicitly timed documents is one where the timing does not need to be resolved with reference to a clock or other timed data at all, for example because the sequence is streamed and the recipient of the stream treats the availability of each new document in the sequence as a 'do it now' event and presents it as soon as possible. In this case the availability time could be captured from some clock source and noted alongside each document for later re-use if desired.

A document containing a missing or an empty `body` element can be used to indicate that the presentation is to be cleared while that document is active.

If implicitly timed documents are dissociated from the context that defines the timing of their presentation, the resulting sequence cannot reasonably be presented.

2.3.1.4.2 Explicitly timed documents

Documents that contain a combination of `begin` and `end` attributes are described as having *explicit timing* (see SMIL 2.1 § 10.7.1). Such documents can additionally contain a maximum duration, set as a `dur` attribute on the `body` element.

See Figure 7 for a diagrammatic representation of the resolved begin and end times of explicitly timed documents.

Use of documents with explicit timing requires that any presentation processor that needs to resolve the timing, such as an encoder, needs to be able to compute the local real time that applies in its processing context for any given time expression.

A further implication is that the time values present in the document need to be valid and reasonable in the context of use. If for example an independent reference clock is used to match document times to encoding presentation times, then the presentation processor cannot proceed without access to that reference clock.

A use case for explicitly timed documents is one where precise timing is known at document creation time. For example the document could have been created by a synchronisation process that calculates the correct alignment of the subtitles with the audio and stores that alignment within the document.

A second use case for explicitly timed documents is one where the relationship between the media time and the subtitle time needs to be stored entirely within the documents, for example because there is no embedding context from which to derive times. If there is an embedding context that imposes times, explicitly timed documents can still be used according to the document activation rules defined above.

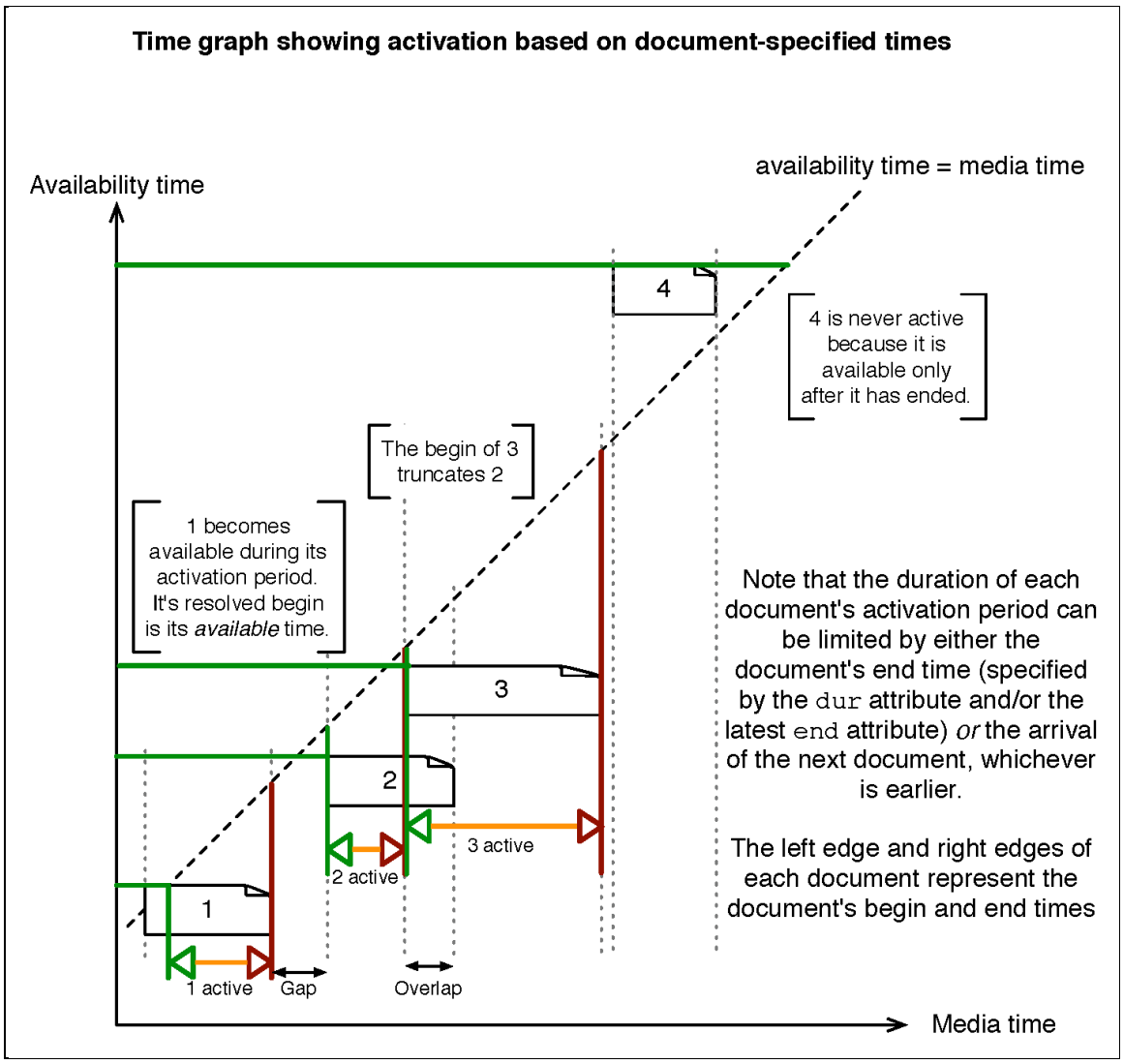


Figure 7: Diagram showing resolved begin and end times for explicitly timed documents

A third use case for explicitly timed documents is one where it is acceptable or necessary to accumulate a set of timed changes over a defined period and deliver them in a single document. For example there could be sufficient time available in the encoding and distribution system that this is considered acceptable, or if the fastest data transfer rate is limited this could be a scheme that in some cases helps to reduce the average required rate. Even if the timings are not precise absolutely, they can be precise in a relative sense within the document. For example if the time of each word issued by a subtitle author is captured in the document the relative timing per word may be precise, even though there is an overall offset for the whole document relative to the media being subtitled that may be imprecise and indeed variable.

The timings within an explicitly timed document can be used to apply retrospective changes so that the re-presentation of the sequence when all the documents are available differs from the original presentation.

The timings within an explicitly timed document can be used to signal future content if it is known. For example if a news programme contains a video package for which subtitles have been authored in advance a single document could be issued containing all of the subtitles for the package.

It is possible for explicitly timed documents to be made available before they become active. In this case it is expected that processing nodes will not discard such documents until they have become inactive, at the earliest.

It is possible to create sequences of explicitly timed documents that contain periods in which no document is active. This technique can be used to indicate that the presentation is to be cleared.

However it cannot be used in a sequence of implicitly timed documents; in such a sequence an empty document is in general required to achieve the same effect.

A second option for indicating that the presentation is to be cleared is to issue a document with appropriate `begin`, `end` or `dur` attributes on the body element that is otherwise empty of content.

2.3.1.4.2.1 Cumulative documents

It is possible to create a sequence of explicitly timed documents in which each document contains all of the content from the previous document in addition to any new content. This could be described as a cumulative document creation technique. See Figure 8.

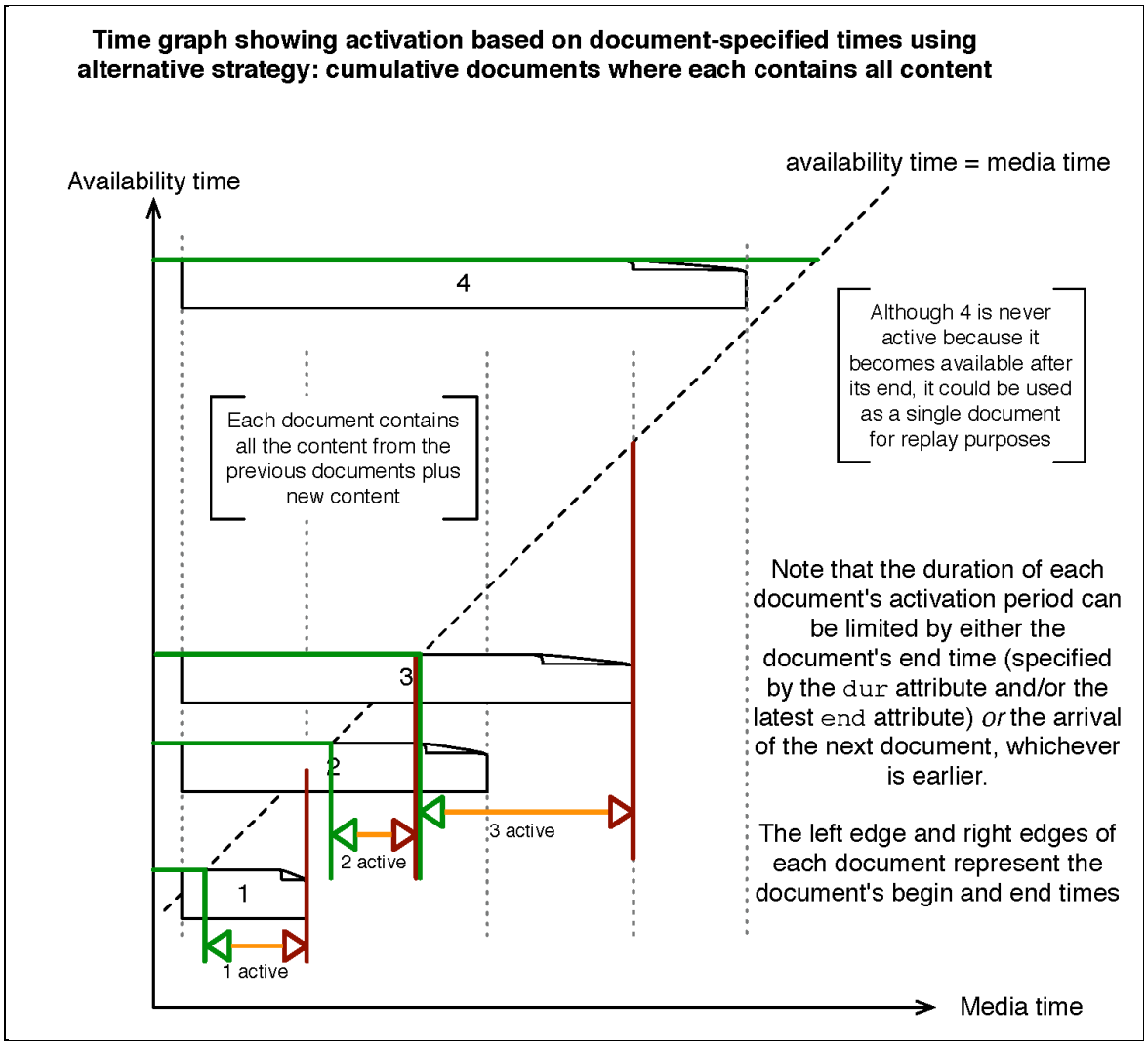


Figure 8: Timing graph showing cumulative document approach

This technique allows the entire sequence to be presented simply by activating the last document, much as though it were a prepared subtitle document. In real world implementations using this technique it could be wise to limit the duration applicable to the sequence in order to limit document sizes, memory usage, parsing time etc.

2.3.1.4.3 Mix of explicitly timed documents and implicitly timed documents

It is possible to create sequences that contain a mixture of implicitly timed and explicitly timed documents.

In this scenario implementations need to relate any external time values to the time base used within the documents. One approach is to store the document receipt timestamps using the same time base as that used within the document.

If the presentation processor does not have access to document availability times or some other context that constrains the activation period of each document then every implicit document effectively prunes all of the documents with a lower sequence number, which by definition are never activated if the sequence is re-presented.

See Figure 9 for an illustration of how mixed documents are activated.

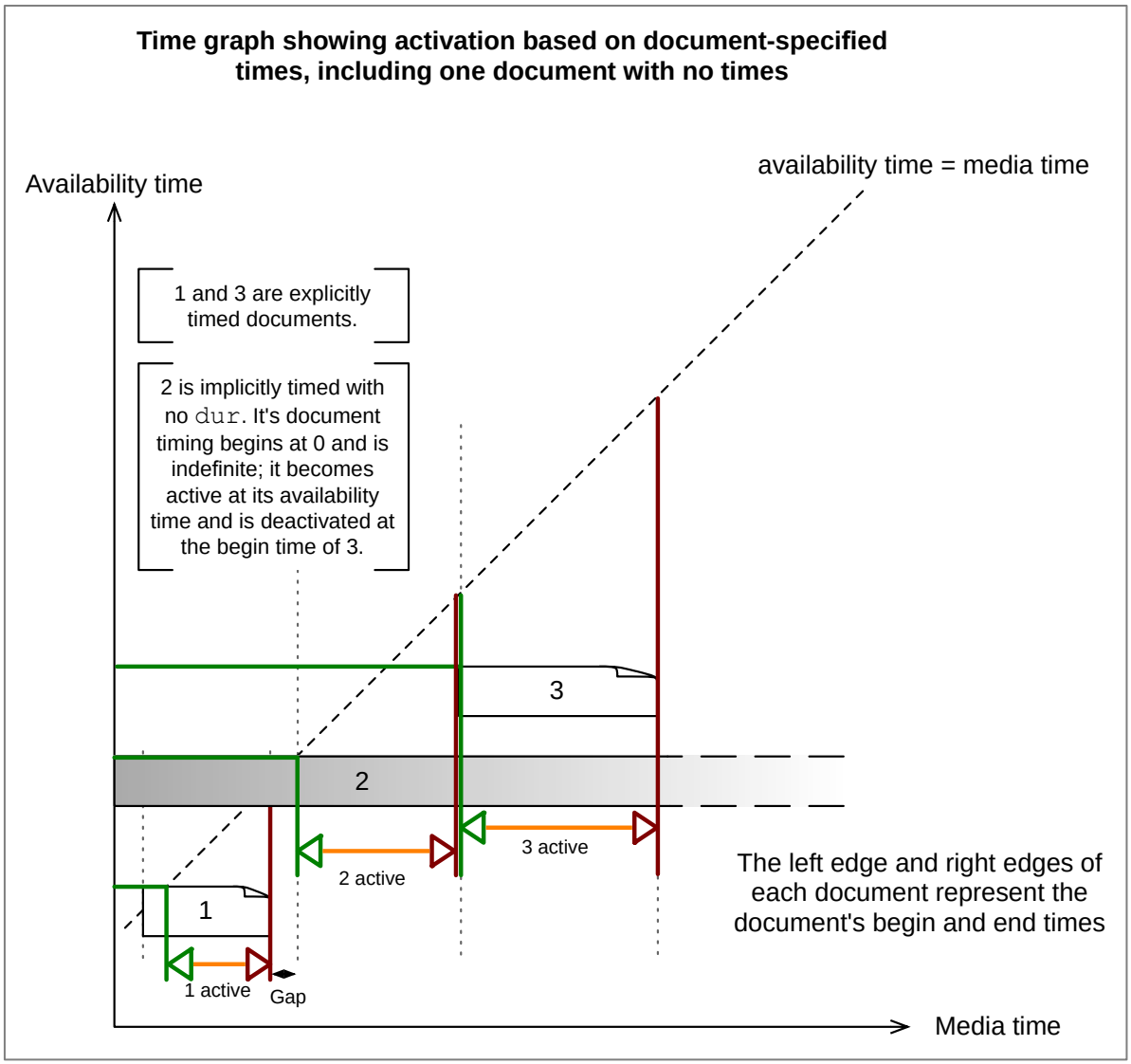


Figure 9: Timing graph showing document activation when a mix of implicitly and explicitly timed documents is in the same sequence

2.3.1.4.4 Issuing documents on every change

One strategy for issuing documents is to create a new document for every change needed to the presented content. This can allow changes to be communicated as quickly as possible without using any more data than is required. Any consumer node subscribing to the stream needs to wait until the next change before seeing any content to present; that in turn could have a negative impact on for example the recovery time if the encoder had entered a fault state before restarting its subscription.

2.3.1.4.5 Issuing documents periodically

An alternative strategy for issuing documents is to create a new document at regular intervals containing the timed content since the previous document. The interval defines the longest delay in communicating changes, additional to any other authoring delays. Conversely it also defines the maximum time until a new subscriber can begin to process content.

In extremis the interval would be the smallest unit of time available. For example a new document could be created and associated with every frame of video.

If the desired behaviour is for the presentation to change (e.g. new subtitles appearing) at times that are not coincident with the interval boundaries, then explicitly timed documents would be needed since by definition there is no way to represent such changes within implicitly timed documents.

In the case of prepared subtitle content the interval could be the duration of the prepared content.

2.3.1.4.6 Issuing documents on every change with a minimum repeat duration

A blend of the above two strategies could be useful to control data rates and set limits on recovery times in the event of source switching or other fault recovery. Such a blended strategy would be one in which a new document is created on every change, but if a predefined interval expires during which no change occurs, a new document is created regardless.

2.3.1.5 Implementation and Operational Considerations

2.3.1.5.1 Pruning ever-extending history

The model presented here allows infinite rewriting of history backwards in time. A new document can be sent that supersedes an arbitrary set of previous documents including a partially superseded document and has a document resolved begin time that falls at any point before, within or after the previously received sequence times. This presents a potential implementation difficulty, since it is impossible in general to know how many documents to retain, and when it is safe to discard documents. Typically it is useful for operational software designed to run continuously to be able to manage its data usage requirements to avoid growing forever.

This problem is however specific to the type of node and the task that it needs to perform. For example some passive nodes need to keep only a transient set of documents, since it is not expected to do more than minimal buffering, say for the time it takes for a distributing node to emit each received document to all subscribers.

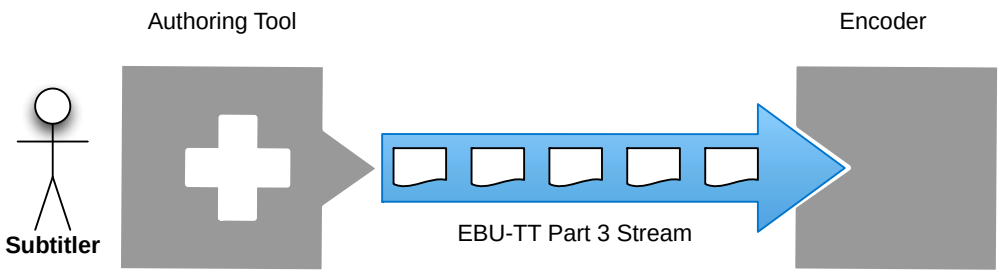
Processing nodes may need to retain a longer history depending on what they are doing. A delay node needs to keep documents for at least the period by which it is applying a delay. The retention semantics for improver nodes or synthesiser nodes need to be defined based on the functions that they are designed to perform.

Consumer nodes similarly need to have some kind of defined retention semantics. One useful strategy for an encoder is to discard everything that has already been encoded into an output format. For example an EBU-TT Part D encoder configured to output an EBU-TT Part D document every 5 seconds could be configured regularly to discard all documents whose resolved end times are earlier than the begin time of the next required output document.

Note: The set of documents that a node retains is defined as the *document cache*.

2.3.1.5.2 Synchronising clocks and handling non-zero transit times

Consider a simple system in which a subtitler authors explicitly timed subtitles using a Producer Node, and the resulting sequence is streamed immediately to a Consumer node, for example to encode into a downstream format.



If the producer node issues documents with the intent of them being presented immediately, and therefore specifies a begin time equal to its perceived time 'now' perhaps with an end time 3 seconds later and sends the document, and it takes a non-zero time D_{at} until the document becomes available then the effective duration for which the subtitles in the document are shown will be reduced by D_{at}. This is because the document resolved begin time will be the availability time, but the document resolved end time is the specified end time in the document.

This scenario can impact the effective reading speed needed to read the text, but is opaque to the subtitle author, who cannot in general know how long it will take for each document to become available. A similar effect is shown diagrammatically in Figure 7 above, where the beginning of document 1 is truncated by late availability.

The above scenario includes an implicit assumption that the two nodes have somehow reached a sufficiently close value for the current time, i.e. that they are synchronised. Another cause of such a problem is if that assumption is invalid and the two nodes are in fact not synchronised relative to each other. For example, if a document's duration is 1s but the consumer node's clock runs 1s or more later than the producer node's clock, then even if the real D_{at} were 0, the consumer node would consider it to be 1s late and the document's content would never be encoded.

Strategies for identifying that two nodes are not synchronised closely enough include:

- Specifying the document creation date and time using `ebuttm:documentCreationDate` placed as a descendant of `/tt:tt/tt:head/tt:metadata`
- Specifying the document emission time by adding to the document an XML comment such as:

```
<!-- ebutt3:emission_time: 12:13:14.15 -->
```

Note: XML comments are excluded from the `deep-equals` comparison of XML elements; therefore two otherwise identical documents that contain different comments are considered to be identical by the test defined in §2.2. As a result, passive nodes can add such comments without breaking passive node conformance requirements.

Note: No formal syntax for this or any other XML comment is defined in this document.

Using either or both of these methods, some potential problems can be identified. In scenarios where the relationship between creation time, emission time and availability time can be modelled consistently, variations over time can highlight issues.

These data can also reveal if document times were in the future when the document was created and in the past when it became available downstream, which can cause truncation of the beginning of subtitle presentation as described above. They can also provide some information about where the delay might have occurred.

Strategies for dealing with these practical challenges include:

- Ensuring all nodes use clocks synchronised to the same time reference (for example an NTP server, a GPS receiver etc).
- If using a local time, specifying the time server's URL in the `ebuttp:referenceClockIdentifier` parameter in documents, and using it.
- Explicitly adding a delay to the begin times of documents either before emitting them or downstream (perhaps using a Retiming Delay node), where that delay is greater than or equal to D_{at}. Note that this scenario only applies to explicitly timed documents, and a Retiming Delay node that performs this function is not expected both to adjust document times and to delay emission of the adjusted documents, since that would simply reintroduce a new availability time delay.

In this discussion, the term "clock" is used to indicate the time source that is used to convert

between real time events such as documents becoming available and times in the document's timebase. There is no requirement that this is directly related to any system clock.

2.3.2 Management of delay in a live authoring environment

Within a typical broadcast workflow there are a variety of delays. These are imposed by the need to perform processing of content, for example transcoding, or by the need to synchronise media relative to other media or defined time structures such as frame boundaries, GOP structures etc. Many of these delays are significant, for example in an MPEG-DASH distribution model the video needs to be accumulated into segments of a predefined duration, encoded, packaged and distributed via a content delivery network.

In the context of live created subtitles, there is a variable authoring delay[5], which is the time between the subtitler hearing the audio and the authoring tool outputting the relevant subtitles. In some environments there are unavoidable delays in the chain, for example for video processing, which are long enough to permit some re-alignment of subtitles relative to the audiovisual media, so that the combined output as experienced by the audience has reduced or even zero latency. See Figures 10 and 11.

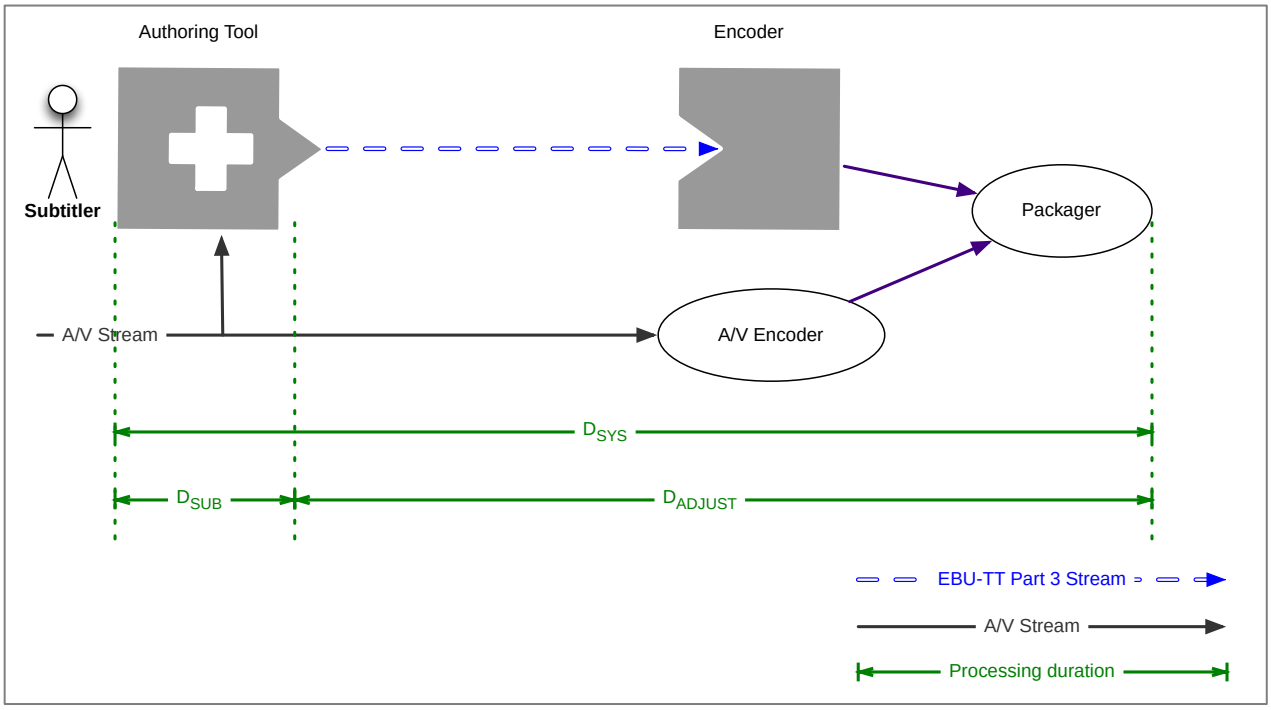


Figure 10: Use case diagram showing timing and delay concepts

The likely size of this authoring delay depends on the technique used to create the subtitles and the author's speed. For example the delay might be longer for subtitles created using live respeaking or stenography than for subtitles prepared in advance and cued manually or automatically.

Additionally, any propagation or processing delays introduced by nodes in the chain need to be taken into account, and could in general result in the resolved document begin times being later than desired or intended.

If the total required delay is D_{SYS}, being the time between the A/V signal being observed by the subtitler and the desired moment of emission or availability of that A/V signal, and the subtitle authoring delay is D_{SUB}, then it follows that for correct timing an adjustment delay D_{ADJUST} must be imposed within the subtitle chain such that $D_{SUB} + D_{ADJUST} = D_{SYS}$.

[5] Real world experiments in the UK suggest this can be around 5 - 10 seconds.

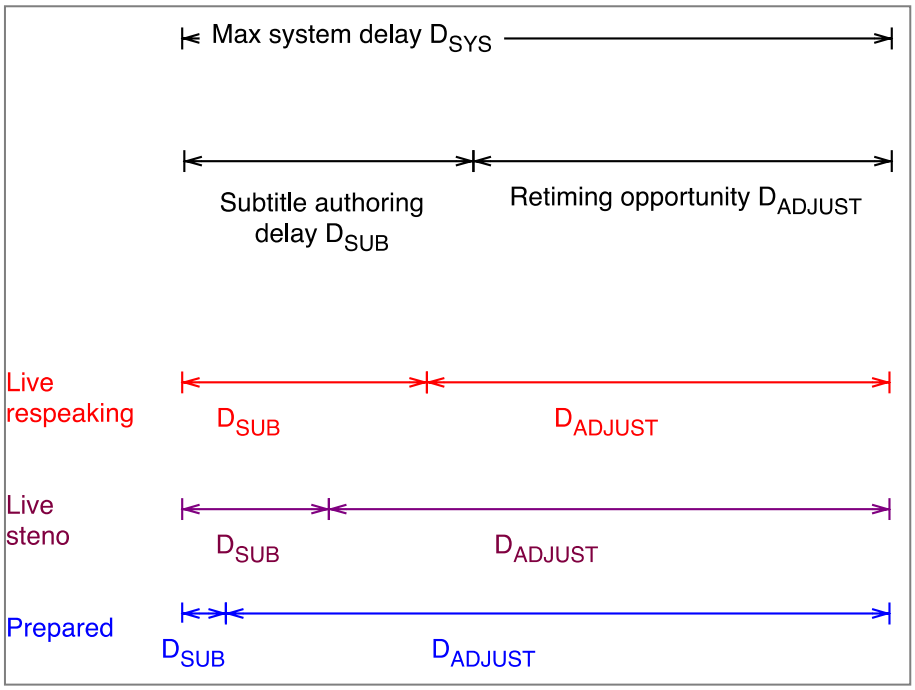


Figure 11: Opportunity for adjustable retiming of subtitles

The minimum value of D_{ADJUST} within a real world live authoring environment is 0[6] because when D_{SUB} exceeds D_{SYS} no correction is possible. This indicates that there is a window of opportunity of duration D_{SYS} – D_{SUB} within which the D_{ADJUST} may be varied to achieve correct synchronisation. See Figure 11. Note that in some environments a solution for frequently late subtitle presentation could be to add a delay (e.g. to the video) to increase the value of D_{SYS}; in scenarios where encoders are able to output accurate subtitle timings further synchronisation options could be available downstream, e.g. in client devices – such specification is out of scope of this document.

Note: if an authored sequence is intended for multiple services, care should be taken in case those services have differing D_{SYS} values, for example because one is encoded as SD video and another is HD, and their encoders have different latencies.

2.3.3 ebuttm:authoringDelay

The `ebuttm:authoringDelay` metadata attribute, defined in EBU-TT Part M (EBU Tech 3390) [5], applied to `tt:tt`, may be used to capture the equivalent of D_{SUB}, where it is known or can be estimated.

The value of `ebuttm:authoringDelay` is intended to express the latency associated with the authoring process and has no direct correlation to the timing values of a document. Specifically this means that even if a process applies a delay (e.g. a delay node), the information that is carried in `ebuttm:authoringDelay` is left unchanged.

The time taken (as measured from the original spoken word) for a given subtitle to be transmitted from an Producer is a composite of the time taken to author (if the workflow is live) and the time taken for the device, application or operator to release the subtitle.

The combined authoring interval is typically:

- sub-frame for an automated prepared file play-out process;

[6] It is conceivable that a Delay node that emits explicitly timed documents could apply a negative D_{ADJUST} offset; an encoder receiving documents that have already expired could reasonably discard them with no output; however this technique could be useful for creating correctly timed *archive* documents.

- the operator's reaction time for live-cued subtitles;
- the 'listening and thinking' time of the subtitler plus the subtitling applications' processing time for stenography or re-speaking in live origination.

The authoring time is a variable for any human-driven process. It can vary in-session for re-speaking and stenography as a function of operator performance, media content type, language etc. The calculation or estimation of D_{SUB} can therefore facilitate an accurate 'improvement' process; it can also enable any downstream Improver Nodes to modify the timing applied to automatically or manually cued prepared subtitles.

Note: the value of D_{SUB} is likely to be higher for processes involving more human cognitive processes and lower for more automated processes; in the case that there is high certainty of upcoming subtitles they may even be sent in advance, resulting in a negative value. In the cases of a very low value Authoring Delay it is probable that any Improver will not try and improve the alignment of the subtitles – the low value signifies that the subtitles are already pretty close in time alignment.

Note: a small or negative value of `ebuttm:authoringDelay` can be considered to have greater associated confidence than a larger positive value. An Improver Node intended to resynchronise the subtitles could use this proxy for a confidence indicator as a heuristic.

2.3.4 Delay nodes

An Improver Node could apply an adjustment delay to the subtitles as one part of a solution for achieving resynchronisation. We will refer to such a node as a Delay node. The value of the delay could be derived using one of a variety of techniques, including potentially:

- heuristics based on `ebuttm:authoringDelay`, and other metadata in the stream e.g. the method used to author the subtitles etc.;
- knowledge of the typical delay within the broadcast chain;
- a comparative measurement using audio analysis to establish a varying value adjustment delay based on the actual subtitles and speech within the programme content.

Note: It is out of scope of this document to mandate the use of specific techniques; furthermore it is expected that the relative success of these techniques will depend on programme content, the level of variability in the chain and the quality of implementation of each technique.

Note: any node that receives and emits streams is likely to incur some real world processing delay; a Delay node is intended to apply a controlled adjustment delay.

Two types of Delay Node for applying a delay are specified:

- A Buffer Delay Node buffers each received Document and emits it after a non-negative delay offset period. Since this is essentially equivalent to a longer carriage latency no modification to the documents is required. The Buffer Delay Node increases the availability time of the documents received by Nodes receiving the sequence (downstream). It is primarily intended for delaying implicitly timed documents.
- A Retiming Delay Node modifies the times within each Document and issues them without further emission delay as part of a new sequence with a new sequence identifier. The times are modified such that all of the computed begin and end times within the document are increased by a non-negative delay offset period. The Retiming Delay Node is primarily intended for delaying explicitly timed documents.

Note: If it is operationally required to use both types of delay node then a chain of nodes can be constructed in which both a Buffer Delay Node and a Retiming Delay Node

exist “in series” with each other.

2.3.4.1 Buffer Delay Node

From a stream and sequence perspective, the following behaviours of a Buffer Delay node are defined:

1. A Buffer Delay node is a passive node. Therefore the output documents shall be identical to the input documents.
2. A Buffer Delay node shall delay emission of the stream by a period not less than the non-negative offset period

The offset period shall be non-negative; in the context of a buffer delay node a negative offset period would require documents to be emitted before they had arrived.

2.3.4.2 Retiming Delay Node

From a stream and sequence perspective, the following behaviours of a Retiming Delay node are defined:

1. A Retiming Delay node is a processing node. Therefore the output sequence shall have a different sequence identifier from the input sequence.
2. A Retiming Delay node shall modify each document to result in the document’s computed times being offset by the non-negative offset period. NOTE: In general this results in implicitly timed documents being converted to explicitly timed documents, since all but a zero offset period requires at a minimum a `begin` attribute on an element for example the `body` element. This behaviour may be surprising.
3. A Retiming Delay node should not emit an output sequence with reordered subtitles.
4. A Retiming Delay node shall not update the value of `ebuttm:authoringDelay`.
5. A Retiming Delay node should add an `ebuttm:appliedProcessing` element to the document metadata to indicate that the delay has been added.

The offset period shall be non-negative; in the context of a retiming delay node applying a negative offset could result in documents having negative begin times, which is not permitted in TTML.

Note: It is possible that delay functionality is combined with other processing in a single node, for example an accumulator; hence the requirement not to reorder is expressed in terms of subtitles not documents: there is for example no requirement that there is a 1:1 relationship between input and output documents from a Retiming Delay node, though such a relationship would be expected for the simplest conceivable Retiming Delay.

Note: If varying the delay offset, take care to manage the other timings to avoid inadvertently changing the displayed order of subtitles; for example one strategy could be to treat delay offset changes as target values that are arrived at over a fixed period, so instead of jumping from 10s to 4s in one step an implementation could gradually reduce the offset from 10s to 4s over, say, a 6s period. Another strategy when the delay varies is to let the node (or a downstream node) to apply its own logic, which may result in skipping documents to achieve synchronisation.

2.3.5 Reference clocks

Some broadcast environments do not relate time expressions to a real world clock such as UTC but to some other generic reference clock such as a studio timecode generator. When `ttp:timebase="clock"` is used and `ttp:clockMode="local"`, the `ebuttp:referenceClockIdentifier` parameter may be specified on the `tt:tt` element to identify the source of this reference clock to allow for correct synchronisation.

Note that for real time processing of EBU-TT Part 3 documents, correct dereferencing of the external clock is a processing requirement, therefore the `referenceClockIdentifier` is defined as

a parameter attribute in the `ebuttp:` parameter namespace. This is in contrast to an EBU-TT Part 1 document where a `referenceClockIdentifier` element is available in the `ebuttm:` metadata namespace of EBU-TT Part M. A Part 1 document created as an archive version of a sequence of Part 3 documents may preserve the value of the `ebuttp:referenceClockIdentifier` parameter attribute in a `ebuttm:referenceClockIdentifier` element.

2.4 Handover

In a live subtitle authoring environment it is common practice for multiple subtitlers to collaborate with each other in the creation of subtitles for a single programme. From an encoder perspective, it is desirable to manage only a single stream of live subtitles. To mediate between the streams that each subtitler creates we will refer to a *Handover Manager* node. See Figure 12.

The Handover Manager subscribes to a set of sequences and selects documents from one sequence at a time, switching between sequences dependent on parameters within the documents. It then emits a new sequence of documents representing the time-interleaved combination of subtitles from each of the authors, where each output document is derived from an input document from the selected sequence. See also Figure 13 for an example of handover sequences.

The Handover Manager node shall use a 'who claimed control most recently' algorithm for selecting the sequence, based on a control token parameter within each document.

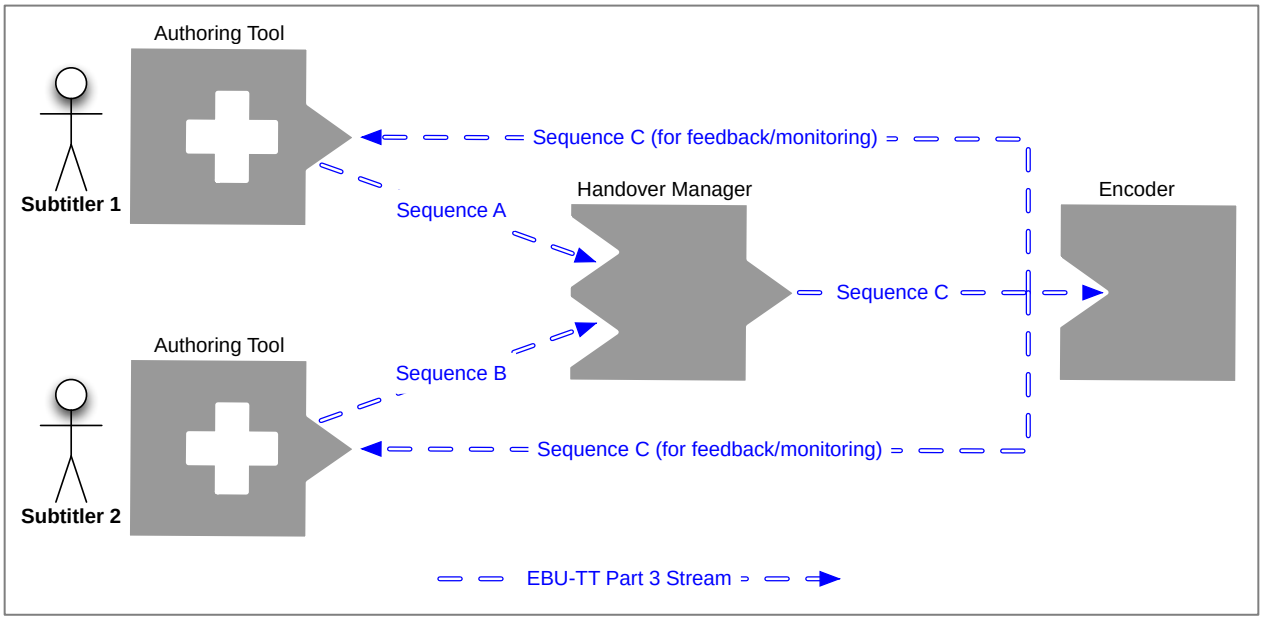


Figure 12: Use case showing a Handover Manager selecting between Sequences A and B and emitting Sequence C

Authoring tools can subscribe to the output stream from the Handover Manager; this makes the control token parameter values visible to them to permit each to direct the Handover Manager to switch to their output; it also facilitates monitoring.

Note: Other schemes for directing handover are possible, for example the control token could be derived from a separate mediation source or even the clock.

2.4.1 Authors Group parameters

The following parameters on the `tt:tt` element are provided to facilitate handover:

The `ebuttp:authorsGroupIdentifier` is a string that identifies the group of authors from which a particular Handover Manager can choose. A Handover Manager should be configured to

subscribe to all the streams whose documents have the same `ebuttp:authorsGroupIdentifier` except for its own output stream. Within a single sequence, all documents that contain the element `ebuttp:authorsGroupIdentifier` shall have the same `ebuttp:authorsGroupIdentifier`.

The Handover Manager may include within each document in its output sequence the parameter attributes `ebuttp:authorsGroupIdentifier` and `ebuttp:authorsGroupControlToken` from the currently selected input sequence. The Handover Manager includes within each output document the EBU-TT Part M metadata attribute `ebuttm:authorsGroupSelectedSequenceIdentifier` set to the value of the source sequence's identifier for that document. This is so that each subscriber to the Handover Manager's output stream can know the current status, including any subscribed subtitle authoring stations.

The Handover Manager's normative behaviour is defined in the algorithm in §2.4.3 below.

When present in a document, the `ebuttp:authorsGroupControlToken` is a number that the Handover Manager uses to identify which sequence to select: when a document is received with a higher value `ebuttp:authorsGroupControlToken` than that most recently received in the currently selected sequence the Handover Manager switches to that document's sequence, i.e. it emits a document in its output sequence corresponding to and derived from the received document with the new control token without delay.

Having selected a sequence, the Handover Manager emits further documents derived from that sequence until a new sequence is selected.

Note: This means that the control token value can be lowered after taking control, by setting the control token value in a new document in the selected sequence to a lower number. Therefore the control token value does not need to increase forever.

Regardless of the selected sequence, the Handover Manager does not emit any documents derived from input sequence documents that do not contain both the parameters `ebuttp:authorsGroupIdentifier` and `ebuttp:authorsGroupControlToken`.

Note: Care should be taken if the carriage mechanism does not guarantee delivery of every document in the sequence in case a document intended to take control is lost. One strategy for avoiding this would be for the subtitle authoring station to observe the Handover Manager's output and verify that control has been taken before lowering the control token value. Another strategy would be to maintain the high control token value and duplicate it in each document in the sequence until the sequence switch has been verified through another mechanism.

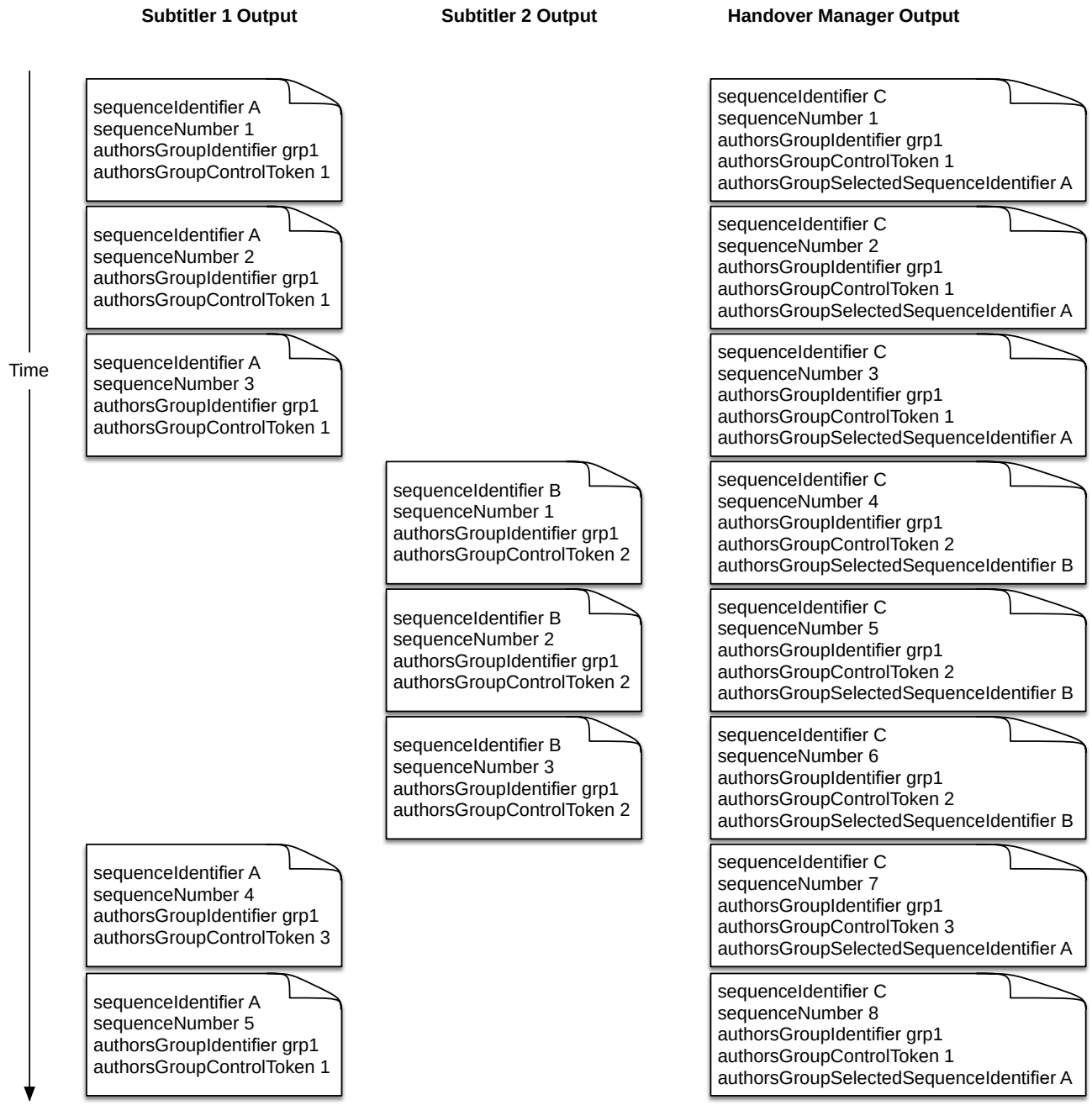


Figure 13: Sample sequences demonstrating Handover

2.4.2 Handover Manager algorithm

The Handover Manager node uses the following 'who claimed control most recently' algorithm for selecting the sequence, based on the control token parameter present within each document.

The Handover Manager shall maintain all of the following variables:

- a token *T* equal to the value of the `ebuttp:authorsGroupControlToken` of the most recently emitted document, or a null value if no document has been emitted.
- a record S_s of the sequence identifier of the source sequence used to generate the most recently emitted document, known as the selected sequence, initially a null value until a sequence is selected.
- a record *AG* of the configured authors group identifier.
- an output sequence identifier S_o.

When new incoming data is received, the Handover Manager shall:

1. If the input data is a valid document that contains all of: 1) a present and valid `ebuttp:authorsGroupIdentifier` equal to AG; and 2) a present and valid `ebuttp:authorsGroupControlToken` then treat the document as the "input document" and execute the following sub-steps in the stated order:
    a. If *T* is null (and therefore by definition S_s is also null), or if *T* is not null and the input document's `ebuttp:authorsGroupControlToken` is greater than *T*, then select the input document's sequence by executing the following sub-steps in any order:
        i. set *T* to the input document's `ebuttp:authorsGroupControlToken`.
        ii. set S_s to the input document's sequence identifier.
    b. If and only if the input document's sequence identifier equals S_s (i.e. the input document's sequence is selected), then execute the following sub-steps in the stated order:
        i. generate an output document based on the input document, setting the output document's sequence identifier to S_o and allocating a valid sequence number greater than the most recently emitted sequence number, and adding an `ebuttm:authorsGroupSelectedSequenceIdentifier` attribute set to S_s;
        ii. emit the output document;
2. Otherwise do not emit a document.

Note: Any implementation that provides the above algorithm's outputs given any input documents satisfies the requirements; it is not required that implementations use code that exactly matches the steps.

This algorithm may be extended in implementations for example to set specific behaviour when basing an output document on the input document, or to define emission rules for other types of live document. See also § 2.4.3 below for practical considerations.

2.4.3 Practical considerations for Handover Manager implementations

A Handover Manager is likely to be used to switch between multiple contributors of live subtitles for a single service or broadcast channel. In this situation each author might be able to configure within their production system some metadata to be included in the EBU-TT Live sequence such as the `ebuttm:broadcastServiceIdentifier` element. In production environments a Handover Manager should check that the input streams are consistent with each other, taking into account possible "emergency" scenarios where a live subtitle author needs to step in at short notice and may not have the opportunity to make such configurations.

The Handover Manager produces a single sequence derived from multiple input sequences. However §2.2 requires that every document in a sequence has an identical timing model. This implies that practical arrangements need to be made to ensure that the Handover Manager's output sequence is conformant. These could include arranging for all input sequences to have the same timing model, or including within the Handover Manager a timing model converter that can set the output documents' timing model correctly independently of the input documents' timing models. Such a converter might need access to external time sources.

2.5 Describing facets of the subtitle content

In a chain of processing nodes each might modify the subtitle content to suit some particular need. For example the subtitler's priority could be to issue new documents as quickly as possible without considering spelling, grammatical correction, profanities etc. Alternatively the subtitler could be issuing a sequence that is known not to be suitable without modification for all downstream platforms: one encoder could be able to emit Unicode code points; another could be restricted to ISO 8859-x. A combination of these scenarios is possible. In order to complete the processing needed to ensure that a stream is suitable, the system model presented here proposes that a series of Improver nodes would be used to perform the necessary processing.

However from a Consumer node's perspective it is not always desirable to rely on the node

configuration being correct without further information. Additionally for compliance monitoring automated processes could be used to assess conformance against some rule set without necessarily enforcing it or making modifications to the text content in the sequence.

Some knowledge of the processing applied can be obtained at a document level from the debug `ebuttm:appliedProcessing` element of EBU-TT Part M (see § 2.6) however this is coarse-grained. To indicate for a particular piece of content some aspect of its editorial or technical quality, for example if it has been spellchecked, profanity checked, had its code point set reduced for downstream compatibility, had colours applied, been positioned etc. the `ebuttm:facet` element of EBU-TT Part M can be applied. To indicate the document level summary the the `ebuttm:documentFacet` element of EBU-TT Part M can be applied.

2.6 Tracing and debugging

The model presented here allows for multiple Processing Nodes to receive and emit streams. In real world scenarios it can be useful to log the activity that generated a document for audit or debugging purposes, for example to check that the correct configurations have been applied.

The `ebuttm:appliedProcessing` element of EBU-TT Part M permits such logging. If present, an `appliedProcessing` element shall describe in text the action that generated the document, in the `action` attribute, and an identifier that performed that action, in the `generatedBy` attribute. The action can be derived from a classification scheme not specified here. The `generatedBy` node identifier is an URI and is also not further defined here.

Optionally the `ebuttm:appliedProcessing` element can identify the node that supplied the source content for the action, using the `sourceId` attribute.

The `ebuttm:appliedProcessing` element may contain text content providing any further logging information.

3. Document Conformance

This section defines the requirements for documents that conform to this specification.

3.1 Generic Constraints

The EBU-TT Part 3 format defines constraints for an XML document instance. A valid EBU-TT Part 3 XML document shall comply with the generic constraints in § 3.1 and the document structure defined in § 3.2.

3.1.1 Namespaces

The following external namespaces from the W3C TTML 1.0 [6] specification shall be used for the TTML elements and attributes in EBU-TT Part 3:

Name	Prefix	Value
TT	tt:	http://www.w3.org/ns/ttml
TT Parameter	ttp:	http://www.w3.org/ns/ttml#parameter
TT Style	tts:	http://www.w3.org/ns/ttml#styling
TT Metadata	ttm:	http://www.w3.org/ns/ttml#metadata

The following namespaces shall be used for the assignment of XML Schema datatypes:

Name	Prefix	Value
XML Schema	xs:	http://www.w3.org/2001/XMLSchema

The following namespaces shall be used for the EBU-TT Part 3 specific vocabulary:

Name	Prefix	Value
EBU-TT Metadata	ebuttm:	urn:ebu:tt:metadata
EBU-TT Styling	ebutts:	urn:ebu:tt:style
EBU-TT Datatypes	ebuttdt:	urn:ebu:tt:datatypes
EBU-TT Parameters	ebuttp:	urn:ebu:tt:parameters

Note: Although any prefix can be used to bind the namespaces in an XML document the use of the prefixes listed above is recommended.

If attributes in this document are defined without prefix they are not in any namespace.

3.1.2 Extensibility

EBU-TT Part 3 documents may be extended as defined by EBU-TT Part 1.

3.1.3 Compatibility with TTML 1.0 timing model

The timing model in EBU-TT Part 3 is compatible with the TTML 1.0 timing model. The additional rules concerning the temporal activation of documents defined in § 2.3.1 constitute constraints on the Document Processing Context as defined in TTML 1.0.

3.1.3 Conformance signalling

If the element `ebuttm:conformsToStandard` is being used within a document instance to signal conformance to this specification it shall have the value "urn:ebu:tt:live:2017-05".

3.2 Document Structure and Content Profile

This specification bases document compliance on EBU-TT Part 1 (Tech 3350) [1] with some changes. Only the changes are described here. These changes are either modifications to the cardinality of elements or attributes or additional elements and attributes. Any elements and attributes omitted from this section are defined identically to EBU-TT Part 1.

The formal definition of how the EBU-TT Part 3 specification uses EBU-TT-, TTML- and XML-vocabulary is presented in tabular form. When using this specification, the definition of the use of an element or attribute shall be interpreted relative to the position defined in the table.

The order of content in this specification shall be as in TTML 1.0 for elements defined in TTML 1.0 and as in EBU-TT Part 1 for elements defined in EBU-TT Part 1. Elements defined in this specification shall immediately follow all siblings defined in EBU-TT Part 1 and precede any siblings defined in other EBU-TT specifications.

Example:

The definition of the use of the `ttp:timebase` attribute in § 3.2.2.1 “tt:tt” specifies only the permitted syntax of the `dur` attribute in relation to the value of the `ttp:timebase` element; all other definition of the `ttp:timebase` element is as defined in EBU-TT Part 1.

Definitions used within this section:

Type: Constraints of the Information structure of an XML element or XML attribute. The type can be further constrained through Enumerations and normative text in the description.

Enumeration: Enumerated values that shall be used for certain elements or attributes of type `xs:string`.

Cardinality: How often an element or attribute may be used inside the corresponding parent element. If the lower bound is greater than 0 (e.g. “1..1” or “1..*”) the element or attribute is mandatory at this position of the document structure. If the lower bound is equal to 0 (e.g. “0..1” or “0..*”) the element or attribute is optional at this position of the document structure.

Position: The position of an element or attribute in an EBU-TT Document as XPATH [7] expression starting with the document root "/". A default namespace of "http://www.w3.org/ns/ttml" is assumed and the prefix "tt" is omitted in the XPATH expression.

TTML: The URL to the specific chapter in the TTML 1.0 specification where the attribute or element is defined. The normative constraints of TTML 1.0 apply unless they are further constrained by this specification.[7]

Rows highlighted in grey indicate required attributes.

3.2.1 Elements and attributes whose cardinality differs

The following table shows the entities whose cardinality differs relative to [EBUTT1]. The entity names are provided in the style of an XPath [XPATH].

Entity path and name	Entity type	Cardinality		Notes
		Tech 3350	Here	
tt:tt/@ttp:markerMode	Attribute	0..1	0..0	ttp:timeBase="smpte" is also prohibited.
tt:tt/tt:head/tt:styling	Element	1..1	0..1	Styling may be omitted if unknown
tt:tt/tt:head/tt:layout	Element	1..1	0..1	Layout may be omitted if unknown

3.2.2 Newly introduced and modified elements and attributes

Entity name	Entity type	Location
`ttp:timeBase` (applicable to `"dur"` attribute)	Attribute	`tt:tt`
`ebuttp:sequenceIdentifier`	Attribute	`tt:tt`
`ebuttp:sequenceNumber`	Attribute	`tt:tt`
`ebuttp:authorsGroupIdentifier`	Attribute	`tt:tt`
`ebuttp:authorsGroupControlToken`	Attribute	`tt:tt`
`ebuttp:referenceClockIdentifier`	Attribute	`tt:tt`
`dur`	Attribute	`tt:body`

The introduced entities are defined in further detail below.

[7] 1 At time of publication the links are based on the TTML version specified at [TTML1].

3.2.2.1 tt:tt

The following attributes are defined by this specification for inclusion on the `tt:tt` element.

EBU-TT Part 3 uses the following parameters from EBU-TT Part 1 to give information on how the timing information in an EBU-TT document should be interpreted. If present, these attributes shall be specified on the `tt:tt` element.

ttp:timeBase (attribute)

Type	`xs:string`
Enumeration	"media" | "clock"
Cardinality	1..1
Position	/tt
TTML	http://www.w3.org/TR/ttml1/#parameter-attribute-timeBase
Description	The `ttp:timeBase` element is as defined in EBU-TT Part 1 with two modifications: • the addition that all time expressions of `dur` attributes shall denote a relative coordinate on the same timeline as the `begin` and `end` attributes. • the restriction that the value "smpte" is not permitted.

ebuttp:sequenceIdentifier (attribute)

Type	`xs:string minLength="1"`
Cardinality	1..1
Position	/tt
Description	The sequence to which every document belongs shall be identified using the `ebuttp:sequenceIdentifier` attribute. The data type is constrained to be a non-empty string, i.e. with a restriction defined as `minLength="1"` in WC3 XML Schema Definition Language (XSD 1.1 Part 2) [8]. Note: It is possible for legal sequence identifier values to be used in a context within which they could cause difficulties. An example of this is if the sequence identifier contains a "/" character and is being used to form part of a URL. In such cases one approach is to escape the string before using it within that context and to de-escape it on extraction from that context before using it as a sequence identifier.

ebuttp:sequenceNumber (attribute)

Type	`xs:positiveInteger`
Cardinality	1..1
Position	/tt
Description	Every non-identical document with the same `ebuttp:sequenceIdentifier` shall be uniquely numbered using the `ebuttp:sequenceNumber` attribute. Processors shall discard documents whose pair of `ebuttp:sequenceIdentifier` and `ebuttp:sequenceNumber` are identical to those in its document cache. In this case processors may issue a warning if the two documents are not identical. The availability time of the (not discarded) document shall not be changed due to such a discard. See also §2.3.1.5.1. Note: It is not considered an error to issue an identical document more than once; this pattern may be used for example when inserting a subtitle document in ancillary data associated with every frame of a video stream, which pattern could facilitate editing.

The parameters `ebuttp:authorsGroupIdentifier` and `ebuttp:authorsGroupControlToken` are provided to facilitate handover between subtitle authors, using semantics defined for the Handover Manager node in § 2.4.

ebuttp:authorsGroupIdentifier (attribute)

Type	`xs:string minLength="1"`
Cardinality	0..1
Position	/tt
Description	The data type is constrained to be a non-empty string, i.e. with a restriction defined as `minLength="1"` in XSD 1.1 Part 2. Identifies the group of authors whose sequences relate to the same content and amongst which a Handover Manager should select documents when generating its output sequence.

ebuttp:authorsGroupControlToken (attribute)

Type	xs:positiveInteger
Cardinality	0..1
Position	/tt
Description	The control token used to direct a Handover Manager to select an input sequence from a particular authors group. The input sequence whose document has the greatest value `ebuttp:authorsGroupControlToken` value is selected for output.

ebuttp:referenceClockIdentifier (attribute)

Type	xs:anyURI
Cardinality	0..1
Position	/tt
Description	Allows the reference clock source to be identified. Permitted only when `ttp:timeBase="clock"` AND `ttp:clockMode="local"`. Note: This attribute differs from the metadata element `ebuttm:referenceClockIdentifier` in EBU-TT Part M because it is expected to affect processing rather than simply being a record of the clock source.

3.2.2.2 tt:body

Type	Element content
Cardinality	0..1
Position	/tt
TTML	http://www.w3.org/TR/ttml1/#document-structure-vocabulary-body
Description	The `tt:body` element is defined as in EBU-TT Part 1 with the attributes in the following tables also permitted or modified. A document that contains a `tt:body` element with no content shall be treated as being active as defined by the semantics described in § 2.3.1, and shall cause no content to be presented while it is active. Note: TTML 1.0 § 9.3.2 Intermediate Synchronic Document Construction specifies that empty elements are pruned, and therefore play no part in the creation of intermediate synchronic documents; nevertheless this specification interprets the timing attributes present on `tt:body` as defining the document activation period, even if presentation of that document generates no intermediate synchronic documents by that algorithm.

begin (attribute)

Type	`ebuttdt:mediaTimingType | ebuttdt:clockTimingType`
Cardinality	0..1
Position	/tt/body
TTML	http://www.w3.org/TR/ttml1/#timing-attribute-begin
Description	Start point of a temporal interval associated with a `tt:body` element. If the timebase is "media" the type shall be `ebuttdt:mediaTimingType`. If the timebase is "media" the time expression should be the offset from a syncbase of "00:00:00.0". If the timebase is "clock" the type shall be `ebuttdt:clockTimingType`. Note: This attribute is identical to its definition in EBU-TT Part 1 with the restriction that the type `ebuttdt:smpteTimingType` is not used.

end (attribute)

Type	`ebuttdt:mediaTimingType | ebuttdt:clockTimingType`
Cardinality	0..1
Position	/tt/body
TTML	http://www.w3.org/TR/ttml1/#timing-attribute-end
Description	End point of a temporal interval associated with a `tt:p` element. If the timebase is "media" the type shall be `ebuttdt:mediaTimingType`. If the timebase is "media" the time expression should be the offset from a syncbase of "00:00:00.0". If the timebase is "clock" the type shall be `ebuttdt:clockTimingType`. Note: This attribute is identical to its definition in EBU-TT Part 1 with the restriction that the type ebuttdt:smpteTimingType is not used.

dur (attribute)

Type	`ebuttdt:mediaTimingType | ebuttdt:clockTimingType`
Cardinality	0..1
Position	/tt/body
TTML	http://www.w3.org/TR/ttml1/#timing-attribute-dur
Description	The maximum duration of the document relative to the resolved begin time, as defined in TTML 1.0. See also § 2.3.1 for further details of the semantics of document activation and the resolution of begin and end times for each document in a sequence. If the timebase is "media" the type shall be `ebuttdt:mediaTimingType`. If the timebase is "clock" the type shall be `ebuttdt:clockTimingType`.

3.2.2.3 tt:div

Type	Element content
Cardinality	0..*
Position	/tt//(body|div)
TTML	http://www.w3.org/TR/ttml1/#document-structure-vocabulary-div
Description	The `tt:div` element is defined as in EBU-TT Part 1 with the attributes in the following tables modified as described.

begin (attribute)

Type	`ebuttdt:mediaTimingType | ebuttdt:clockTimingType`
Cardinality	0..1
Position	/tt/body//div
TTML	http://www.w3.org/TR/ttml1/#timing-attribute-begin
Description	Start point of a temporal interval associated with a `tt:div` element. If the timebase is "media" the type shall be `ebuttdt:mediaTimingType`. If the timebase is "media" or "clock" the time expression should be the offset from a syncbase of the begin time of the closest ancestor that specifies a begin time. If no ancestor specifies a begin time and the timebase is "media" the time expression should be the offset from a syncbase of "00:00:00.0". If the timebase is "clock" the type shall be `ebuttdt:clockTimingType`. Note: This attribute is identical to its definition in EBU-TT Part 1 with the restriction that the type ebuttdt:smpteTimingType is not used.

end (attribute)

Type	`ebuttdt:mediaTimingType | ebuttdt:clockTimingType`
Cardinality	0..1
Position	/tt/body//div
TTML	http://www.w3.org/TR/ttml1/#timing-attribute-end
Description	End point of a temporal interval associated with a `tt:div` element. If the timebase is "media" the type shall be `ebuttdt:mediaTimingType`. If the timebase is "media" or "clock" the time expression should be the offset from a syncbase of the begin time of the closest ancestor that specifies a begin time. If no ancestor specifies a begin time and the timebase is "media" the time expression should be the offset from a syncbase of "00:00:00.0". If the timebase is "clock" the type shall be `ebuttdt:clockTimingType`. Note: This attribute is identical to its definition in EBU-TT Part 1 with the restriction that the type ebuttdt:smpteTimingType is not used.

3.2.2.4 tt:p

Type	Element content
Cardinality	0..*
Position	/tt/body//div
TTML	http://www.w3.org/TR/ttml1/#content-vocabulary-p
Description	The `tt:p` element is defined as in EBU-TT Part 1 with the attributes in the following tables modified as described.

begin (attribute)

Type	`ebuttdt:mediaTimingType | ebuttdt:clockTimingType`
Cardinality	0..1
Position	/tt/body//div/p
TTML	http://www.w3.org/TR/ttml1/#timing-attribute-begin
Description	Start point of a temporal interval associated with a `tt:p` element. If the timebase is "media" the type shall be `ebuttdt:mediaTimingType`. If the timebase is "media" or "clock" the time expression should be the offset from a syncbase of the begin time of the closest ancestor that specifies a begin time. If no ancestor specifies a begin time and the timebase is "media" the time expression should be the offset from a syncbase of "00:00:00.0". If the timebase is "clock" the type shall be `ebuttdt:clockTimingType`. Note: This attribute is identical to its definition in EBU-TT Part 1 with the restriction that the type ebuttdt:smpteTimingType is not used.

end (attribute)

Type	`ebuttdt:mediaTimingType | ebuttdt:clockTimingType`
Cardinality	0..1
Position	/tt/body//div/p
TTML	http://www.w3.org/TR/ttml1/#timing-attribute-end
Description	End point of a temporal interval associated with a `tt:p` element. If the timebase is "media" the type shall be `ebuttdt:mediaTimingType`. If the timebase is "media" or "clock" the time expression should be the offset from a syncbase of the begin time of the closest ancestor that specifies a begin time. If no ancestor specifies a begin time and the timebase is "media" the time expression should be the offset from a syncbase of "00:00:00.0". If the timebase is "clock" the type shall be `ebuttdt:clockTimingType`. Note: This attribute is identical to its definition in EBU-TT Part 1 with the restriction that the type ebuttdt:smpteTimingType is not used.

3.2.2.5 tt:span

Type	Element content
Cardinality	0..*
Position	/tt/body//p
TTML	https://www.w3.org/TR/ttml1/#content-vocabulary-span
Description	The `tt:span` element is defined as in EBU-TT Part 1 with the attributes in the following tables modified as described.

begin (attribute)

Type	`ebuttdt:mediaTimingType | ebuttdt:clockTimingType`
Cardinality	0..1
Position	/tt/body//div/p/span
TTML	http://www.w3.org/TR/ttml1/#timing-attribute-begin
Description	Start point of a temporal interval associated with a `tt:span` element. If the timebase is "media" the type shall be `ebuttdt:mediaTimingType`. If the timebase is "media" or "clock" the time expression should be the offset from a syncbase of the begin time of the closest ancestor that specifies a begin time. If no ancestor specifies a begin time and the timebase is "media" the time expression should be the offset from a syncbase of "00:00:00.0". If the timebase is "clock" the type shall be `ebuttdt:clockTimingType`. Note: This attribute is identical to its definition in EBU-TT Part 1 with the restriction that the type ebuttdt:smpteTimingType is not used.

end (attribute)

Type	`ebuttdt:mediaTimingType | ebuttdt:clockTimingType`
Cardinality	0..1
Position	/tt/body//div/p/span
TTML	http://www.w3.org/TR/ttml1/#timing-attribute-end
Description	End point of a temporal interval associated with a `tt:span` element. If the timebase is "media" the type shall be `ebuttdt:mediaTimingType`. If the timebase is "media" or "clock" the time expression should be the offset from a syncbase of the begin time of the closest ancestor that specifies a begin time. If no ancestor specifies a begin time and the timebase is "media" the time expression should be the offset from a syncbase of "00:00:00.0". If the timebase is "clock" the type shall be `ebuttdt:clockTimingType`. Note: This attribute is identical to its definition in EBU-TT Part 1 with the restriction that the type ebuttdt:smpteTimingType is not used.

3.3 Datatypes

EBU-TT Part 3 defines no specific datatypes to restrict the content of attributes or textual Element content.

EBU-TT Part 3 also uses any datatypes referenced and defined in EBU-TT Part 1 andEBU-TT Part M.

Note: If a datatype is applied to an attribute that was taken from TTML 1.0 the restriction of the datatype is equal to the definition in TTML 1.0 or it is a further restriction of the content as defined in TTML 1.0. Therefore all values that conform to the EBU-TT Part 3 datatypes also conform to the values allowed in TTML 1.0. However it is possible to create a value that conforms to the TTML 1.0 definitions but does not conform to the EBU-TT Part 3 datatypes.

4. Node Conformance

This section defines the requirements for nodes that conform to this specification.

Node conformance is defined in terms of their behaviours only. The nodes defined here are not the only permitted nodes; the purpose of defining them is to define minimal expectations to support interoperability of node implementations.

Nodes are defined as abstract classes. See Figure 14 for a UML representation of the abstract node class structure.

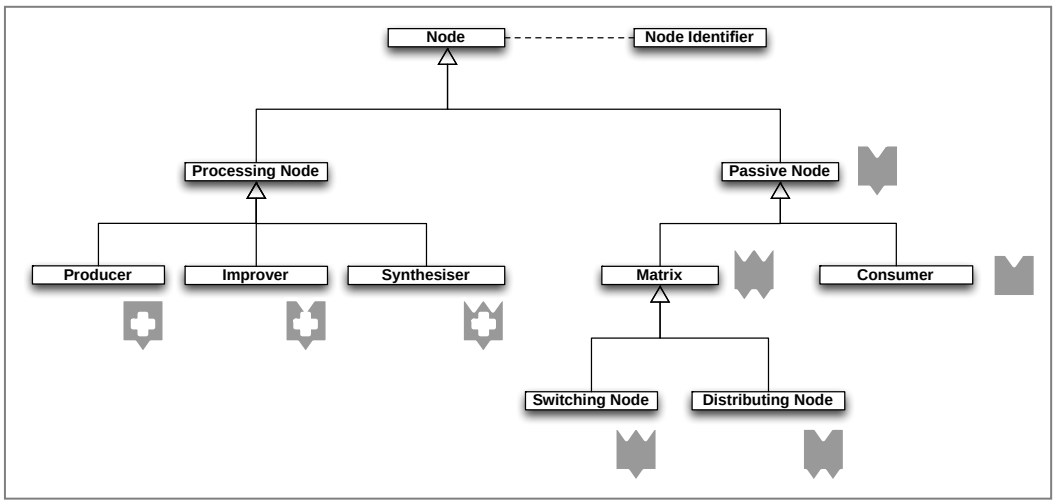


Figure 14: UML model diagram showing logical nodes and their relationships

4.1 Generic Node Classes

This section defines the behaviours of nodes. The nodes are specified as abstract classes that inherit behaviour from their parent. This section is structured according to the inheritance tree, so a subsection inherits the definitions of its parent section within the document. For example a Processing Node *is* a Node.

4.1.1 Node

A node is a logical processing unit that consumes or emits a sequence.

Note: physical implementations could combine the actions of multiple nodes.

4.1.1.1 Processing Node

A processing node emits a sequence that can differ from any of its inputs.

A processing node may consume one or more sequences. The sequence that it emits shall be differently identified to all of its input sequences.

4.1.1.1.1 Producer Node

A Producer consumes no sequences.

Note: An authoring station is a concrete example of a Producer node.

4.1.1.1.2 Improver Node

An Improver consumes exactly one sequence.

4.1.1.1.2.1 Delay

A Delay imposes a specified delay between its input sequence and its output sequence.

4.1.1.1.2.1.1 Buffer Delay

A Buffer Delay imposes an emission delay according to the semantics defined in section § 2.3.4.1.

4.1.1.1.2.1.2 Retiming Delay

A Retiming Delay imposes a delay by adjusting document times according to the semantics defined in section §2.3.4.2.

4.1.1.1.3 Synthesiser Node

A Synthesiser consumes one or more sequences and outputs a new sequence in some way derived from those input sequences.

4.1.1.1.3.1 Handover Manager

A Handover Manager synthesises an output sequence based on a combination of its input sequences, according to the semantics defined in section § 2.4.

4.1.1.2 Passive Node

A Passive Node receives one or more sequences. Any documents that it emits are identical to the document(s) that it receives.

4.1.1.2.1 Matrix

A matrix receives any number of sequences and emits any number of sequences.

4.1.1.2.1.1 Switching Node

A switching node receives one or more sequences and emits one of the received sequences.

4.1.1.2.1.2 Distributing Node

A distributing node receives one sequence and emits it as any number of streams.

4.1.1.2.4 Consumer

A consumer receives one sequence and emits zero sequences.

Note: An encoder is a concrete example of a Consumer node.

5. References

[1]	**EBU Tech 3350**	EBU-TT Part 1 v1.2 Subtitling format definition https://tech.ebu.ch/publications/Tech 3350
[2]	**EBU Tech 3380**	EBU-TT, Part D Subtitling Distribution Format https://tech.ebu.ch/publications/Tech 3380
[3]	**SMIL 2.1**	Synchronized Multimedia Integration Language (SMIL 2.1), W3C Rec. http://www.w3.org/TR/2005/REC-SMIL2-20051213/
[4]	**XPATH FUNCTIONS 3.0**	XPath and XQuery Functions and Operators 3.0 http://www.w3.org/TR/2014/REC-xpath-functions-30-20140408/
[5]	**EBU Tech 3390**	EBU-TT, Part M Metadata specification https://tech.ebu.ch/publications/Tech 3390
[6]	**TTML 1.0**	Timed Text Markup Language (TTML) 1.0 (Second Edition), W3C Rec. http://www.w3.org/TR/2013/REC-ttml1-20130924/
[7]	**XPATH 3.0**	XPath 3.0 https://www.w3.org/TR/2014/REC-xpath-30-20140408/
[8]	**XSD 1.1 Part 2**	W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/

6. Bibliography

EBU R 95	Recommendation on Safe areas for 16:9 television production. http://tech.ebu.ch/publications/r095
EBU R 133	Recommendation on transport of subtitles inside and outside MXF files http://tech.ebu.ch/publications/r133
EBU Tech 3360	EBU-TT, Part 2 Mapping EBU-STL (Tech 3264) to EBU-TT subtitle files http://tech.ebu.ch/publications/tech3360
ISO 3166	Codes for the representation of names of countries and their subdivisions.
ISO 8859_x	8-bit single-byte coded graphic character sets (16 parts)
MPEG-DASH	ISO/IEC 23009-1; Codec-agnostic adaptive bitrate HTTP-based streaming
Open Stand	Open Stand Principles https://open-stand.org/about-us/principles/
RFC 3066	H Alvestrand, ed. RFC 3066: Tags for the Identification of Languages 1995. http://www.ietf.org/rfc/rfc3066.txt
SMPTE-12M-1:2008	"SMPTE Standard for Television -- Time and Control Code"
SMPTE ST 2052-1:2010	"SMPTE Standard for Television -- Timed Text Format (SMPTE-TT)
UAX9	Mark Davis. Unicode Standard Annex #9. Unicode Bidirectional Algorithm. http://unicode.org/reports/tr9/
UAX15	Mark Davis. Unicode Standard Annex #15. Unicode Normalization Forms. http://unicode.org/reports/tr15/
XML 1.0	Tim Bray, et al. Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Rec. 26 November 2008. http://www.w3.org/TR/2008/REC-xml-20081126/
XML Schema Part 2	Paul Biron and Ashok Malhotra, XML Schema Part 2: Datatypes, W3C Rec. 28 October 2004. http://www.w3.org/TR/xmlschema-2/
XPATH 3.0	XPath 3.0 https://www.w3.org/TR/2014/REC-xpath-30-20140408/
SMIL 2.1	Synchronized Multimedia Integration Language (SMIL 2.1), W3C Rec. http://www.w3.org/TR/2005/REC-SMIL2-20051213/

Annex A: Overview document structure (Informative)

The following is a syntactic representation of the EBU-TT Part 3 document model. It is derived from the syntactic representation of TTML 1.0 and the definition of the reduced XML Infoset in TTML 1.0 and is aligned where appropriate with the equivalent section of [EBUTT1].

ELEMENT INFORMATION ITEMS

```
<tt:tt
    xml:space = ("default"|"preserve")>
    ttp:timeBase = ( "media" | "clock") #REQUIRED
    ttp:clockMode = ( "local" | "gps" | "utc")
    xml:lang = ("" | <xs:language>) #REQUIRED
    ttp:cellResolution = <ebuttdt:cellResolutionType>
    tts:extent = <ebuttdt:extentType>  /* Restricted to length metric "px" */
    ebuttp:sequenceIdentifier = xs:string minLength="1" #REQUIRED
    ebuttp:sequenceNumber = xs:positiveInteger #REQUIRED
    ebuttp:authorsGroupIdentifier = xs:string minLength="1"
    ebuttp:authorsGroupControlToken = xs:positiveInteger
    ebuttp:referenceClockIdentifier = xs:anyURI
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content: tt:head, tt:body?
</tt:tt>

<tt:head
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content: tt:metadata?, ttm:copyright?, tt:styling?, tt:layout?
</tt:head>

<ttm:copyright
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content: <xs:string>
</ttm:copyright>

<tt:metadata
    xml:id = <xs:ID>
    xml:lang = ("" | <xs:language>)
    xml:space = ("default"|"preserve")
    {any attribute in the TT Metadata namespace}
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content: ({any element in TT Metadata or EBU-TT Metadata namespace} | {any
element not in any TT or EBU-TT namespace})*
</tt:metadata>

<tt:styling
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content: tt:metadata?, tt:style+
</tt:styling>
```

```
<tt:style
    xml:id = <xs:ID> #REQUIRED
    style = <xs:IDREFS>
    tts:direction = ( "ltr" | "rtl" )
    tts:fontFamily = <ebuttdt:fontFamilyType>
    tts:fontSize = <ebuttdt:fontSizeType>
    tts:lineHeight = ("normal" | <ebuttdt:lengthType>)
    tts:textAlign = ( "left" | "center" | "right" | "start" | "end" )
    tts:color = <ebuttdt:colorType>
    tts:backgroundColor = <ebuttdt:colorType>
    tts:fontStyle = ( "normal" | "italic" )
    tts:fontWeight = ( "normal" | "bold" )
    tts:textDecoration = ( "none" | "underline" )
    tts:unicodeBidi = ( "normal" | "embed" | "bidiOverride" )
    tts:wrapOption = ( "wrap" | "noWrap" )
    tts:padding = <ebuttdt:paddingType> // deprecated
    ebutts:multiRowAlign = ("start" | "center" | "end" | "auto")
    ebutts:linePadding = <ebuttdt:linePaddingType>
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    />

<tt:layout
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content: tt:metadata?, tt:region+
</tt:layout>

<tt:region
    xml:id = <xs:ID> #REQUIRED
    tts:origin = <ebuttd:originType> #REQUIRED
    tts:extent = <ebuttd:extentType> #REQUIRED
    style = <xs:IDREFS>
    tts:displayAlign = ( "before" | "center" | "after" )
    tts:padding = <ebuttdt:paddingType>
    tts:writingMode = ("lrtb"| "rltb" | "tbrl" | "tblr" | "lr" | "rl" | "tb")
    tts:showBackground = ("always" | "whenActive")
    tts:overflow = ("visibile" | "hidden")
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content: tt:metadata?
</tt:region>

<tt:body
    style = <xs:IDREFS>
    begin = (<ebuttdt:mediaTimingType> | <ebuttdt:clockTimingType>)
    end = (<ebuttdt:mediaTimingType> | <ebuttdt:clockTimingType>)
    dur = (<ebuttdt:mediaTimingType> | <ebuttdt:clockTimingType>)
    {any attribute in the TT Metadata namespace}
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content: tt:metadata?, tt:div+
</tt:body>

<tt:div
    xml:id = <xs:ID>
    region = <xs:IDREF>
    style = <xs:IDREFS>
    begin = (<ebuttdt:mediaTimingType> | <ebuttdt:clockTimingType>)
    end = (<ebuttdt:mediaTimingType> | <ebuttdt:clockTimingType>)>
    xml:lang = ("" | <xs:language>)
```

```
    {any attribute in the TT Metadata namespace}
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content: tt:metadata?, tt:div*, tt:p*
</tt:div>

<tt:p
    xml:id = <xs:ID> #REQUIRED
    xml:space = ("default" | "preserve")
    xml:lang = ("" | <xs:language>)
    region = <xs:IDREF>
    style = <xs:IDREFS>
    begin = (<ebuttdt:mediaTimingType> | <ebuttdt:clockTimingType>)
    end = (<ebuttdt:mediaTimingType> | <ebuttdt:clockTimingType>)
    {any attribute in the TT Metadata namespace}
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content (Mixed): tt:metadata?, (tt:span | tt:br)*
</tt:p>

<tt:span
    xml:id = <xs:ID>
    xml:space = ("default" | "preserve")
    xml:lang = ("" | <xs:language>)
    style = <xs:IDREFS>
    begin = (<ebuttdt:mediaTimingType> | <ebuttdt:clockTimingType>)
    end = (<ebuttdt:mediaTimingType> | <ebuttdt:clockTimingType>)
    {any attribute in the TT Metadata namespace}
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content (Mixed): tt:metadata?, tt:span*, tt:br*
</tt:span>

<tt:br
    {any attribute in the TT Metadata namespace}
    {any attribute in the EBU-TT Metadata namespace}
    {any attribute not in default, any TT namespace or any EBU-TT namespace}
    >
    Content: tt:metadata?
</tt:br>
```

EXPRESSIONS

```
<ebuttdt:cellResolutionType>
    : <columns> <whiteSpace> <rows>

<columns> | <rows>
    : <digit>* <digitGreaterZero> <digit>*

<ebuttdt:colorType>
    : <color> as defined in TTML 1

<ebuttdt:extentType>
    : <width> <whiteSpace> <height>

<width> | <height>
    : <ebuttdt:lengthType>

<ebuttdt:fontFamilyType>
    : <familyName> | <genericFamilyName> as defined in TTML 1
```

```
<ebuttdt:fontSizeType>
    : <ebuttdt:lengthType> <whiteSpace> <ebuttdt:lengthType>?

<ebuttdt:lengthType>
    : <scalar>
    | <percentage>

<scalar>
    : <number> <units>

<percentage>
    : <number> "%"

<number>
    : <sign>? <non-negative-number>

<sign>
    : "+" | "-"

<non-negative-number>
    : <non-negative-integer>
    | <non-negative-real>

<non-negative-integer>
    : <digit>+

<non-negative-real>
    : <digit>* "." <digit>+

<units>
    : "px"    /* abbreviation of "pixel" */
    | "c"     /* abbreviation of "cell" */

<ebuttdt:lineHeightType>
    : "normal" | <ebuttdt:lengthType>           /* length >= 0 */

<ebuttdt:originType>
    : <x-coord> <whiteSpace> <y-coord>

<x-coord> | <y-coord>
    : <ebuttdt:lengthType>

<ebuttdt:paddingType>
    : <all-edges>
    | <beforeAndAfter> <whiteSpace> <startAndEnd>
    | <before> <whiteSpace> <startAndEnd> <whiteSpace> <after>
    | <before> <whiteSpace> <end> <whiteSpace> <after> <whiteSpace> <start>

<all-edges> | <before> | <end> | <after> | <start> | <beforeAndAfter> |
<startAndEnd>
    : <ebuttdt:lengthType>

<ebuttdt:linePaddingType>
    : <non-negative-number> "c"

<ebuttdt:noTimezoneDateType>
    : <xs:date> with no timezone specified

<hh> | <mm> | <ss>
    : <digit> <digit>
```

```
<ebuttdt:mediaTimingType>
    : <full-clock-value>
    | <timecount-value>

<ebuttdt:clockTimingType>
    : <limited-clock-value>
    | <timecount-value>

<ebuttdt:delayTimingType>
    : <sign> <timecount-value>

<limited-clock-value>
    : <hh> ":" <mm> ":" <ss> ("." <digit>+)?

<full-clock- value>
    : <hhh> ":" <mm> ":" <ss> ("." <digit>+)?

<hhh>
    : <digit> <digit>+

<timecount-value>
    : <digit>+ ("." <digit>+)? <metric>

<metric>
    : "h"   /* hours */
    | "m"   /* minutes */
    | "s"   /* seconds */
    | "ms"  /* milliseconds */

<digit>
    : "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<digitGreaterZero>
    : "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<whiteSpace> /*(space, carriage return, line feed, tab)*/
      : (#x20 | #x9 | #xD | #xA)+
```

Annex B: Examples of computed document times (informative)

This section describes by example how to calculate the earliest computed begin and the latest computed end times for documents that might not have intuitive results. This illustrates the definition of these terms in §2.3.1.0.1.

These are illustrative examples only and not intended to imply an implementation algorithm: any algorithm that achieves the same outcome can be considered conformant.

Example 1: Untimed (i.e. implicitly timed) document

```
<?xml version="1.0" ?>
<tt ebuttp:sequenceIdentifier="testSequence001" ebuttp:sequenceNumber="1"
ttp:clockMode="local" ttp:timeBase="clock" xml:lang="en-GB"
xmlns:ebuttp="urn:ebu:tt:parameters" xmlns="http://www.w3.org/ns/ttml"
xmlns:ttp="http://www.w3.org/ns/ttml#parameter"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <head/>
  <body>
    <div>
      <p xml:id="ID005">
        <span>Some example text...</span>
        <br/>
        <span>And another line</span>
      </p>
    </div>
  </body>
</tt>
```

Earliest Computed Begin Time: 0s

Latest Computed End Time: “undefined”

In this case, there is no specified begin time; more precisely there are multiple untimed paths to leaf elements, i.e. the two `span` elements and the `br`. Each of those has a computed begin time which is the default of 0 (zero). Therefore the earliest computed begin time is 0s.

Similarly the computed end time is unspecified, so the latest computed end time is considered in SMIL terms to be “undefined”.

Example 2: End time not defined by body element

```
<?xml version="1.0" ?>
<tt ebuttp:sequenceIdentifier="testSequence001" ebuttp:sequenceNumber="2"
ttp:clockMode="local" ttp:timeBase="clock" xml:lang="en-GB"
xmlns:ebuttp="urn:ebu:tt:parameters" xmlns="http://www.w3.org/ns/ttml"
xmlns:ttp="http://www.w3.org/ns/ttml#parameter"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
  </head/>
  <body begin="10s">
    <div begin="1s" end="4s">
      <p xml:id="ID005">
        <span>Some example text...</span>
        <br/>
        <span>And another line</span>
      </p>
    </div>
  </body>
</tt>
```

Earliest Computed Begin Time: 10s

Latest Computed End Time: 14s

In this case, the earliest computed begin time of the document is specified by the `body` element as 10s; in this case where there are nested timed elements the computed begin time of the div is 11s, which is an offset of 1s relative to the syncbase of 10s defined by the `body`'s begin time.

The latest computed end time is defined not by the `body`, which does not specify an end time, but by its child `div` element, so the latest computed end time is 14s, calculated as the `div`'s 4s end time relative to the syncbase of 10s specified by the `body`'s begin time.

Example 3: Begin time not defined by body element

```
<?xml version="1.0" ?>
<tt ebuttp:sequenceIdentifier="testSequence001" ebuttp:sequenceNumber="3"
ttp:clockMode="local" ttp:timeBase="clock" xml:lang="en-GB"
xmlns:ebuttp="urn:ebu:tt:parameters" xmlns="http://www.w3.org/ns/ttml"
xmlns:ttp="http://www.w3.org/ns/ttml#parameter"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <head/>
  <body end="10s">
    <div begin="1s" end="4s">
      <p xml:id="ID005">
        <span>Some example text...</span>
        <br/>
        <span>And another line</span>
      </p>
    </div>
  </body>
</tt>
```

Earliest Computed Begin Time: 1s

Latest Computed End Time: 10s

In this example, the timing of the `div` element being relative to the timing of its parent element (`body`) which sets no begin time, and therefore has a syncbase of 0s. The earliest specified begin time is the `div`'s time, which is computed as 1s; therefore the earliest computed begin time is 1s. The `body` does however specify the latest end time, which is the latest computed end time, being 10s.

Example 4: End time truncated by body element

```
<?xml version="1.0" ?>
<tt ebuttp:sequenceIdentifier="testSequence001" ebuttp:sequenceNumber="4"
ttp:clockMode="local" ttp:timeBase="clock" xml:lang="en-GB"
xmlns:ebuttp="urn:ebu:tt:parameters" xmlns="http://www.w3.org/ns/ttml"
xmlns:ttp="http://www.w3.org/ns/ttml#parameter"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <head/>
  <body end="10s">
    <div begin="5s" end="15s">
      <p xml:id="ID005">
        <span>Some example text...</span>
        <br/>
        <span>And another line</span>
      </p>
    </div>
  </body>
</tt>
```

Earliest Computed Begin Time: 5s

Latest Computed End Time: 10s

In this example, as in example 3, the timing of the div element being relative to the timing of its parent element (`body`) which sets no begin time, and therefore has a syncbase of 0s. The earliest specified begin time is the `div`'s time, which is computed as 5s. This is the earliest computed begin time, 5s. The `body` does however specify the latest end time, which is 10s. This is later than the end time of the `div`, however the `body` ending truncates the end time of the `div`; the `div` is active from 5s to 10s only. The latest computed end time is 10s.

Example 5: Excluding elements that begin after they have ended

```
<?xml version="1.0" ?>
<tt ebuttp:sequenceIdentifier="testSequence001" ebuttp:sequenceNumber="5"
ttp:clockMode="local" ttp:timeBase="clock" xml:lang="en-GB"
xmlns:ebuttp="urn:ebu:tt:parameters" xmlns="http://www.w3.org/ns/ttml"
xmlns:ttp="http://www.w3.org/ns/ttml#parameter"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <head/>
  <body>
    <div>
      <p xml:id="ID004" begin="2s" end="1s">
        <span>This text is never visible</span>
      </p>
      <p xml:id="ID005" begin="5s" end="8s">
        <span>Some example text...</span>
        <br/>
        <span>And another line</span>
      </p>
    </div>
  </body>
</tt>
```

Earliest Computed Begin Time: 5s

Latest Computed End Time: 8s

In this example, the first `p` element is timed but the begin time is after the end time, so it is excluded from the calculation of the earliest computed begin time and latest computed end time. The timing of the `p` element therefore sets the earliest computed begin time as 5s and the latest computed end time as 8s for this document.

Example 6: Mix of timed and untimed paths

```
<?xml version="1.0" ?>
<tt ebuttp:sequenceIdentifier="testSequence001" ebuttp:sequenceNumber="5"
ttp:clockMode="local" ttp:timeBase="clock" xml:lang="en-GB"
xmlns:ebuttp="urn:ebu:tt:parameters" xmlns="http://www.w3.org/ns/ttml"
xmlns:ttp="http://www.w3.org/ns/ttml#parameter"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <head/>
  <body>
    <div>
      <p xml:id="ID004">
        <span>This text is always visible</span>
      </p>
      <p xml:id="ID005" begin="5s" end="8s">
        <span>Some example text...</span>
        <br/>
        <span>And another line</span>
      </p>
    </div>
  </body>
</tt>
```

Earliest Computed Begin Time: 0s

Latest Computed End Time: “undefined”

In this contrived example, the first `p` element is not timed and the second `p` element has times. The untimed leaf span containing “This text is always visible” has a computed begin time of 0s – this is the earliest computed begin time. It also has a computed end time equivalent to “undefined”, which is the latest computed end time.

The *specified* begin and end times on the second `p` are included in the search for the earliest computed begin time and the latest computed end time, but for the begin time it is not the “winner” because 5s is not earlier than 0s, and for the end time it is also not the “winner” because “undefined”, being equivalent to infinity, is considered later than 8s. The children `span` and `br` elements of the second `p` share the same computed times as the `p`, so taking them into account as leaf nodes does not change the calculation.

Example 7: dur on body

```
<?xml version="1.0" ?>
<tt ebuttp:sequenceIdentifier="testSequence001" ebuttp:sequenceNumber="5"
ttp:clockMode="local" ttp:timeBase="clock" xml:lang="en-GB"
xmlns:ebuttp="urn:ebu:tt:parameters" xmlns="http://www.w3.org/ns/ttml"
xmlns:ttp="http://www.w3.org/ns/ttml#parameter"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <head/>
  <body dur="5s">
    <div>
      <p xml:id="ID005" begin="5s" end="12s">
        <span>Some example text...</span>
        <br/>
        <span>And another line</span>
      </p>
    </div>
  </body>
</tt>
```

Earliest Computed Begin Time: 5s

Latest Computed End Time: 12s

In this example, the specified begin and end times on the `p` element set both the earliest computed begin time and the latest computed end time respectively.

The `dur` attribute on the body element is *not* included in the computation of these times; however it will have an effect later when the document resolved end time is calculated: it will certainly truncate the visible duration of the `p` element since it is set to 5s which is less than the 7s duration of the `p` element.

Example 8: End and dur on body

```
<?xml version="1.0" ?>
<tt ebuttp:sequenceIdentifier="testSequence001" ebuttp:sequenceNumber="5"
ttp:clockMode="local" ttp:timeBase="clock" xml:lang="en-GB"
xmlns:ebuttp="urn:ebu:tt:parameters" xmlns="http://www.w3.org/ns/ttml"
xmlns:ttp="http://www.w3.org/ns/ttml#parameter"
xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <head/>
  <body end="10s" dur="5s">
    <div>
      <p xml:id="ID005" begin="4s" end="8s">
        <span>Some example text...</span>
        <br/>
        <span>And another line</span>
      </p>
    </div>
  </body>
</tt>
```

Earliest Computed Begin Time: 4s

Latest Computed End Time: 10s

In this example, the specified begin time on the `p` element sets the earliest computed begin time and the `body`'s specified end time sets the latest computed end time.

The `dur` attribute on the body element is *not* included in the computation of these times; however it will have an effect later when the document resolved end time is calculated: it will truncate the visible duration of the `body` element since it is set to 5s which is less than the 6s duration implied by the difference between the begin time of the `p` element (4s) and the end time of the `body` element (10s).

Annex C: Examples of document resolved begin and end times (informative)

This section describes by example how to calculate the document resolved begin and end times for documents in the context of a sequence. This illustrates the definition of these terms in §2.3.1.1 and §2.3.1.2.

These are illustrative examples only and not intended to imply an implementation algorithm: any algorithm that achieves the same outcome can be considered conformant.

Consider the effect as each document arrives in turn and is added to our contrived sequence, with the following assumptions:

- local clock time base;
- there is an externally specified document activation begin time of 10:00:00;
- there is an externally specified document deactivation time of 10:30:00;
- every document that arrives has the same sequence identifier;
- we begin with an empty sequence.

Example 1: Untimed document 1 arrives at 10:00:03

This document has sequence number 1. The availability time is 10:00:03. As per Example 1 in Annex B, the earliest computed begin time is 0s (00:00:00) and the latest computed end time is "indefinite".

The document resolved begin time according to §2.3.1.1 is the later of a) 10:00:03, b) 00:00:00 and c) 10:00:00, which is 10:00:03.

To resolve the end time we need to inspect the other documents in the sequence. At this stage there are no other documents in the sequence, so we can ignore it for this first example. This example has no `dur` attribute, so there is no need to take into account the document resolved begin time and the `dur` attribute. The latest computed end time is "indefinite" and there is an externally specified document deactivation time of 10:30:00. The document resolved end time according to §2.3.1.2 is the earlier of c) "indefinite" and d) 10:30:00, which is 10:30:00.

Therefore our view of document activation is currently:

Sequence number	Document resolved begin time	Document resolved end time
1	**10:00:03**	**10:30:00**

Where changed or new values are in bold.

Example 2: Untimed document 2 arrives at 10:00:07

This document has sequence number 2. The availability time is 10:00:07. Again, the earliest computed begin time is 0s and the latest computed end time is "indefinite".

The document resolved begin time according to §2.3.1.1 is the later of a) 10:00:07, b) 00:00:00 and c) 10:00:00, which is 10:00:07.

In this case there is another document in the sequence, which could come into play when resolving the document end time. However on closer inspection there are no documents that have a greater sequence number. Again, this document has no `dur` attribute. The latest computed end time is "indefinite" and there is an externally specified document deactivation time of 10:30:00. The

document resolved end time according to §2.3.1.2 is the earlier of c) “indefinite” and d) 10:30:00, which is 10:30:00.

However the arrival of this document affects the document resolved end time of the first received document, which had sequence number 1. Re-examining its document resolved end time, there is now a document with a greater sequence number, document 2 and its document resolved begin time is 10:00:07. So the document resolved end time of document 1 is now the lesser of a) 10:00:07, c) “indefinite” and d) 10:30:00, which is 10:00:07.

Therefore our view of document activation after document 2 has become available is:

Sequence number	Document resolved begin time	Document resolved end time
1	10:00:03	**10:00:07**
2	**10:00:07**	**10:30:00**

Where changed or new values are in bold.

Example 3: Timed document 3 arrives at 10:00:10

This document has sequence number 3, availability time 10:00:10. The earliest computed begin time is 10:00:11 and the latest computed end time is 10:00:16. There is no `dur` attribute.

The document resolved begin time according to §2.3.1.1 is the later of a) 10:00:10, b) 10:00:11 and c) 10:00:00, which is 10:00:11.

The document resolved end time according to §2.3.1.2 is the earlier of c) 10:00:16 and d) 10:00:30, which is 10:00:16.

Adding this to the sequence potentially affects the document resolved end times of documents 1 and 2. Looking at document 1, which is already truncated by document 2, since document 2’s document resolved begin time is earlier than document 3’s document resolved begin time, there is no change. However document 2 is affected by document 3 and it now has a document resolved end time equal to the document resolved begin time of document 3.

Therefore our view of document activation after document 3 has become available is:

Sequence number	Document resolved begin time	Document resolved end time
1	10:00:03	10:00:07
2	10:00:07	**10:00:11**
3	**10:00:11**	**10:00:16**

Where changed or new values are in bold.

Example 4: Timed document 3 arrives again at 10:00:12

This document has sequence number 3, availability time 10:00:12. The earliest computed begin time is 10:00:11 and the latest computed end time is 10:00:16. There is no `dur` attribute.

Since we have seen a document with this sequence number before we discard the new one and do not modify any of the times. The availability time of document 3 remains 10:00:10.

Therefore our view of document activation after the second document 3 has arrived and been discarded is:

Sequence number	Document resolved begin time	Document resolved end time
1	10:00:03	10:00:07
2	10:00:07	10:00:11
3	10:00:11	10:00:16

Where changed or new values are in bold.

Example 5: Timed document 5 arrives at 10:00:14

This document has sequence number 5, availability time 10:00:14. We have not yet seen document 4, but we proceed regardless. The earliest computed begin time is 10:00:13, which is earlier than the availability time, and the latest computed end time is 10:00:17. There is no `dur` attribute.

The document resolved begin time according to §2.3.1.1 is the later of a) 10:00:14, b) 10:00:13 and c) 10:00:00, which is 10:00:14.

The document resolved end time according to §2.3.1.2 is the earlier of c) 10:00:17 and d) 10:00:30, which is 10:00:17.

Adding this to the sequence potentially affects the document resolved end times of documents 1, 2 and 3. Looking at document 1, which is already truncated by document 2, since document 2's document resolved begin time is earlier than document 5's document resolved begin time, there is no change. Similarly for document 2 whose document resolved end time is already earlier than document that of document 3. However document 3 is affected by document 5 and it now has a document resolved end time equal to the document resolved begin time of document 5.

Therefore our view of document activation after document 5 has become available is:

Sequence number	Document resolved begin time	Document resolved end time
1	10:00:03	10:00:07
2	10:00:07	10:00:11
3	10:00:11	**10:00:14**
5	**10:00:14**	**10:00:17**

Where changed or new values are in bold.

Example 6: Timed document 4 arrives at 10:00:15

This document has sequence number 4, availability time 10:00:15. We have already seen document 5, but we proceed regardless. The earliest computed begin time is 10:00:12, which is earlier than the availability time, and the latest computed end time is 10:00:16. There is no `dur` attribute.

The document resolved begin time according to §2.3.1.1 is the later of a) 10:00:15, b) 10:00:12 and c) 10:00:00, which is 10:00:15.

Document 5 has already been received and has a document resolved begin time of 10:00:14. The document resolved end time for document 4 according to §2.3.1.2 is the earlier of a) 10:00:14, c) 10:00:16 and d) 10:00:30, which is 10:00:14.

Observe that the resolved end time is earlier than the resolved begin time. On first glance there could be two choices: discard, or process the remaining documents according to the semantics of §2.3.1.1 and §2.3.1.2. In this case however document 5 begins before document 4 in any case, so document 4 is completely ignored in favour of document 5 and there is no change to make.

Therefore our view of document activation after document 4 has become available is:

Sequence number	Document resolved begin time	Document resolved end time
1	10:00:03	10:00:07
2	10:00:07	10:00:11
3	10:00:11	10:00:14
4	**-**	**-**
5	10:00:14	10:00:16

Where changed or new values are in bold.

Example 7: Timed document 6 with dur arrives at 10:00:16

This document has sequence number 6, availability time 10:00:16. The earliest computed begin time is 10:00:17, and the latest computed end time is 10:00:25. There is a `dur` attribute with value `"5s"`.

The document resolved begin time according to §2.3.1.1 is the later of a) 10:00:16, b) 10:00:17 and c) 10:00:00, which is 10:00:17.

The document resolved end time according to §2.3.1.2 is the earlier of b) 10:00:17 + 5s = 10:00:22, c) 10:00:25 and d) 10:00:30, which is 10:00:22.

Observe that the presence of the `dur` attribute has constrained the active duration of the document even though the latest computed end time (which as defined in this specification excludes `dur`) is set to a later value by some `end` attribute on a content element within the document.

None of the previously received documents' resolved end times are affected by document 6.

Therefore our view of document activation after document 4 has become available is:

Sequence number	Document resolved begin time	Document resolved end time
1	10:00:03	10:00:07
2	10:00:07	10:00:11
3	10:00:11	10:00:14
4	**-**	**-**
5	10:00:14	10:00:16
6	**10:00:17**	**10:00:22**

Where changed or new values are in bold.

Observe that there is a gap between the end of document 5 and the beginning of document 6 during which no document is active: this implies that there is no presented content during this period from 10:00:16 to 10:00:17.

Annex D: Requirements for Carriage Specifications

This document does not define carriage mechanisms for streaming sequences between nodes.

A specification described as an "EBU-TT Part 3 Carriage Specification" is conformant if it complies with the requirements described in this annex.

Authors of such carriage specification are encouraged to inform EBU for example by email to subtitling@ebu.ch.

Core networking dependencies and connection protocols

Carriage specification documents shall describe the core networking dependencies and protocols, i.e. if the mechanism is based on IP with TCP, IP with UDP, VBI in SDI, VANC in HD-SDI etc. and shall reference any related standards.

It is recommended that carriage specifications should NOT depend on non-standard connection protocols, i.e. those that do not conform to common definitions of open standards such as those developed to conform to the principles of [OPENSTD].

Where the networking dependencies impose constraints, those constraints shall be described. For example a carriage specification for EBU-TT Live over VBI embedded in SDI would impose a maximum data rate constraint. Another example could be the need for, and impact of packetisation if used by a networking protocol.

Synchronisation impacts and/or thresholds

Carriage specification documents shall describe any interactions between the carriage mechanism and the synchronisation of live subtitles. For example an embedded mechanism such as VANC in HD-SDI could maintain frame-based synchronisation wherever it is routed, meaning that the synchronisation is defined when the subtitles are embedded rather than when they are received further downstream.

Information Security

Carriage specification documents shall describe:

- what provision is made for supporting information security requirements including but not limited to authentication, encryption and error checking mechanisms;
- under what circumstances the mechanisms described can be considered acceptable for crossing organisational boundaries;
- any known impacts or dependencies on other information security technologies, for example is the mechanism transparent to firewalls or does it need special configuration, is there a framework for allowing future authentication and encryption mechanisms to be used, etc.

Endpoint cardinality

Carriage specification documents shall describe whether they natively support point to point delivery, or point to multipoint delivery, or both.

If a carriage specification makes use of a duplex reverse channel for delivery monitoring or fault identification then that mechanism shall be described including the technique used, e.g. acknowledgement messages and their format, the time impact of such techniques, and the expected behaviour in case a fault is identified. Note that reverse channels are not a requirement in general.

Connection lifecycle management

Carriage specification documents shall either a) define the lifecycle of any connections, or b) reference standards that define the lifecycle of any connections. The lifecycle here refers to the initiation, establishment, ongoing maintenance, planned closure and error handling of the connections.

Note: If carriage specifications do not need use any form of connection this requirement is relaxed.

Channel routing

Carriage specifications may define registries or other ways to associate a) streams with the services, channels, programmes, languages etc. for which they are intended and b) nodes that can provide particular streams. This information could be used operationally to automate the routing of subtitle streams from authors to encoders. If such mechanisms are included the metadata model used should be described or referenced, and any impact on synchronisation should be described.

Note: the channel routing mechanisms could include a description of how to switch between input streams in a content dependent way.

Stability

Carriage specifications shall describe or reference the expected level of connection stability, and operational approaches for maintaining that stability. For example how the mechanism handles dropped connection fault conditions, whether data delivery is guaranteed or some documents may be lost as a matter of course, how the latency characteristics of the mechanism may vary, how version compatibility issues are managed, what statistics and monitoring are available etc.

Note: A TCP socket based protocol would exhibit guaranteed data delivery behaviour at the possible expense of delivery timing; conversely a UDP based protocol would exhibit minimal delivery timing at the possible expense of data loss. These characteristics affect the stability of the operational connections, and are need to be described so that stable systems can be engineered.

Interoperability

It is recommended that carriage mechanisms should be defined with interoperability in mind. Interfaces should not need to be hardware or vendor specific, should ideally publish details of available services on request, should self-describe their external interfaces, and should permit any relevant configuration of the services available to be published. Where carriage mechanisms multiplex EBU-TT Part 3 data with other data, e.g. by embedding in an HD-SDI signal, the mechanism shall be transparent to devices that are unable to process EBU-TT Part 3 data, i.e. the presence of EBU-TT Part 3 data shall NOT cause a fault in compliant devices, for example by extending beyond fixed size data windows, including bytes with special meaning, mis-representing the EBU-TT Part 3 data as an incompatible format etc.

Serial data communication protocols such as RS232 and RS422 are considered unsuitable for meeting this need and are deprecated.

Annex E: Not fully defined example scenarios (Informative)

The following examples represent typical real world scenarios in which documents and nodes that conform to this specification can be used, which this specification does not fully define. Note that the scenario numbers form a set with those in the § 1.1 and therefore do not increase monotonically.

Example 3: Prepared subtitle playback – not defined in this specification

A broadcast playout provider is tasked with playing out a prepared EBU-TT Part 1 document alongside a video programme, and inserting the subtitles into the video stream for downstream routing. A playout automation system initiates playback of the EBU-TT Part 1 document alongside the video content and a 'playback' node transforms the EBU-TT Part 1 document into a sequence of EBU-TT Part 3 subtitle documents, one per video frame[8], for insertion into video, or other emission as a stream.

Example 5: Speaker based styling – not fully defined in this specification

An Improver Node receives a sequence in which each document has no styling data. The Improver analyses speaker ("agent") metadata within each document, and uses that in combination with a predefined style set to add the appropriate styles and style references to each document before issuing it as a new sequence with a new sequence identifier.

Example 6: Transcoding for distribution – not fully defined in this specification

An encoder receives a stream for a specific service, having been configured to subscribe to the node that emits a stream whose documents includes that service's destination service identifier. It accumulates documents in the sequence from the stream and transcodes them to the required output encoding, perhaps time-bounded samples of EBU-TT-D for packaging and distribution in MPEG-DASH, or DVB bitmaps for multiplexing in an MPEG-2 transport stream.

Example 7: Creation of an archive document – not fully defined in this specification

An 'archiver' node creates archive documents for each programme on a service. It receives a stream for a specific service, having been configured to subscribe to the node that emits a stream whose documents includes that service's destination service identifier. It accumulates those documents from the sequence that have the same broadcast programme identifier and combines them into a single EBU-TT Part 1 document for each programme, for storage in an archive.

Example 8: Resilient failover – not fully defined in this specification

A playout facility receives multiple physical streams (for redundancy) from different nodes, for a specific service, each with the same sequence identifier, since they originated from a single node. The facility will normally be configured to accept one of these streams for insertion into the video output, but on failure of that stream will select in turn an alternate stream in an order of preference.

Example 9: Subtitle localisation for opt-outs – not fully defined in this specification

A single broadcast channel has multiple, local downstream 'opt-outs'[9] for localisation purposes. A single subtitler authors the main stream, and other subtitlers author the 'opt-out' streams. Local

[8] NOTE: A similar alternative is to insert one EBU-TT Part 3 document per change in subtitles, or to insert an EBU-TT Part 3 document when a maximum period of time has elapsed since the previous document: this is an implementation choice that could be determined by the maximum permitted recovery time for the system, for example.

[9] An 'opt-out' is where a service overwrites part of another service's output for a limited period of time, for example to deliver local news for a particular region.

downstream switching nodes facilitate the switching of the subtitle streams in time with the switch in video and audio streams.

Annex F: Summary of document conformance differences relative to Part 1 (Informative)

The format for EBU-TT Part 3 is based on EBU-TT Part 1 with some relaxations and some additions. Processors need to be able to receive and parse those documents, and possibly to create new documents and stream those.

The conformance rules for documents defined in Part 3 are in § 3. Here's a quick summary of the differences compared to Part 1:

- The parameter attributes `sequenceIdentifier` and `sequenceNumber` are required on the `tt` element.
- `styling` and `layout` can be omitted, allowing them to be added later.
- The `dur` attribute can be used on `body`.
- In case a studio or other clock is used, it can be identified with a `referenceClockIdentifier` parameter.
- Frame based timing is not permitted; the use of `ttp:timeBase="smpte"` is prohibited; time expressions are not permitted to use frames components.
- The authoring delay between the subtitle hearing the programme audio and the subtitles being generated can be signalled if known using the `authoringDelay` metadata attribute.
- A set of parameters is available to support handover between different subtitlers working on the same programme output. They are `ebuttp:authorsGroupIdentifier` and `ebuttp:authorsGroupControlToken`. Handover Managers have to populate the metadata attribute `ebuttm:authorsGroupSelectedSequenceIdentifier` to indicate from which input sequence each document in the output was derived.