EBU

OPERATING EUROVISION AND EURORADIO

TECH 3370

EBU-TT, PART 3
LIVE SUBTITLING APPLICATIONS

SYSTEM MODEL AND CONTENT PROFILE FOR AUTHORING AND CONTRIBUTION

STATUS: Draft version 0.8 for review
Send your comments to subtitling@ebu.ch

Geneva
June 2015

Page intentionally left blank. 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.

Page intentionally left blank. This document is paginated for two sided printing

Contents

Page intentionally left blank. This document is paginated for two sided printing

Status of this document (Informative)

This document is a working draft and may be updated, replaced or made obsolete by other documents at any time. It is inappropriate to cite this document as other than a work in progress.

Readers are encouraged to review this document and provide feedback via subtitling@ebu.ch. All comments received before September 2015 will be considered.

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) [EBUTT1]
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
How to use EBU-TT for the production and contribution of live subtitles.

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-D (EBU Tech 3380) [EBUTTD]
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).

Page intentionally left blank. This document is paginated for two sided printing

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

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

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

Scope (Informative)

This document describes how EBU-TT [EBUTT1] can be used in a broadcasting environment to carry subtitles that are created in real time ("live") 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 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;

Future 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.

Note that though this document primarily defines the live subtitles, it may also be used within a broadcast playout environment for creating streams of subtitles whose source is a prepared EBU-TT Part 1 subtitle document, for example if the goals of temporal alignment and simple routing are achieved by inserting subtitle data directly within video streams, associating a subtitle with a specific series of frames.

No requirement is presented here for any device used by an audience member to receive and present subtitles 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.

1. Introduction (Informative)

The scope 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.

The problem area that is addressed in this document is the data format for carrying subtitles from the author to the encoder, and the possible transformations that may happen to that data on its journey, to ensure that what is received at the encoder is a suitable input for further handling.

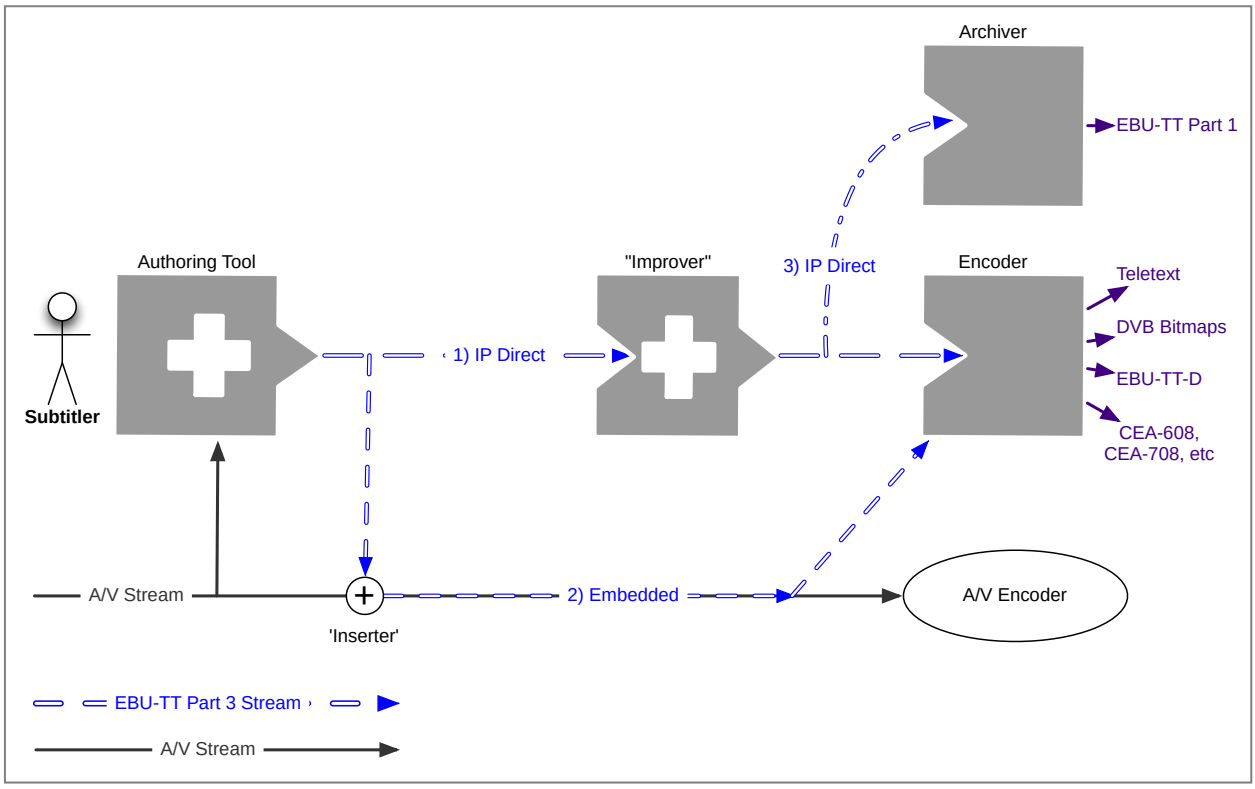

Figure 1: Schematic of simple use case showing an authoring tool generating a stream of EBU-TT Part 3 live subtitles.

Figure 1 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 in one of two possible ways: 1) via a direct IP connection to an Improver and then on to an encoder; 2) embedded into an audio/visual stream with an inserter and then de-embedded for passing to an encoder. It also shows a potential direct IP connection 3) to an archiver that creates an EBU-TT Part 1 document [EBUTT1].

The Improver is a generic processing node; it could for example insert a defined compensating delay, check content for correct spelling or perform other transformations to generate output suitable for encoding.

Editorial issues 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 editorial approaches. 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.

This document also provides useful options for the problem of 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.

This document defines a system model that describes both the temporal flow of data in ‘streams’ and a logical network of ‘nodes’ between which data may flow. Streams are transfers of sequences of documents; each document is a valid TTML document. In order to facilitate a variety of working practices, including those where style, layout and timing are not applied simultaneously with authoring, the content model defined for those TTML documents is significantly more lax than that for EBU-TT part 1, for example.

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.

1.1 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 presentation.

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], 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] 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-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.

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].

Presentation Processor
This term is used as defined in [TTML1]: "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 documents each of which shares the same sequence identifier.

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 [SMIL] 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 [SMIL] 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.

Sequences

- A *sequence* is a set of related contiguous documents, e.g. the documents that define the subtitles for a single programme.
- Every distinct sequence SHALL have a unique *sequence identifier*. Sequences SHALL be considered distinct if they have had processing applied.
- A sequence of implicitly timed documents MAY be transformed, with knowledge of the documents' associated timings, 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 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*. Every distinct document with the same sequence identifier SHALL have a different 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 § 2.4.1 below. 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 SHALL be valid and self-contained. 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 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, duplicates, forwards, processes or consumes 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 defined by the source node's identifier and the sequence identifier.
- A *physical stream* is the transfer of a sequence between two nodes. It is defined 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

[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

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 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 SHALL be temporally active. Note the logic defining which document is temporally active is defined in § 2.4.1 below.

Nodes are defined as abstract classes. Further detail of node behaviour is defined in § 4. See Figure 2.

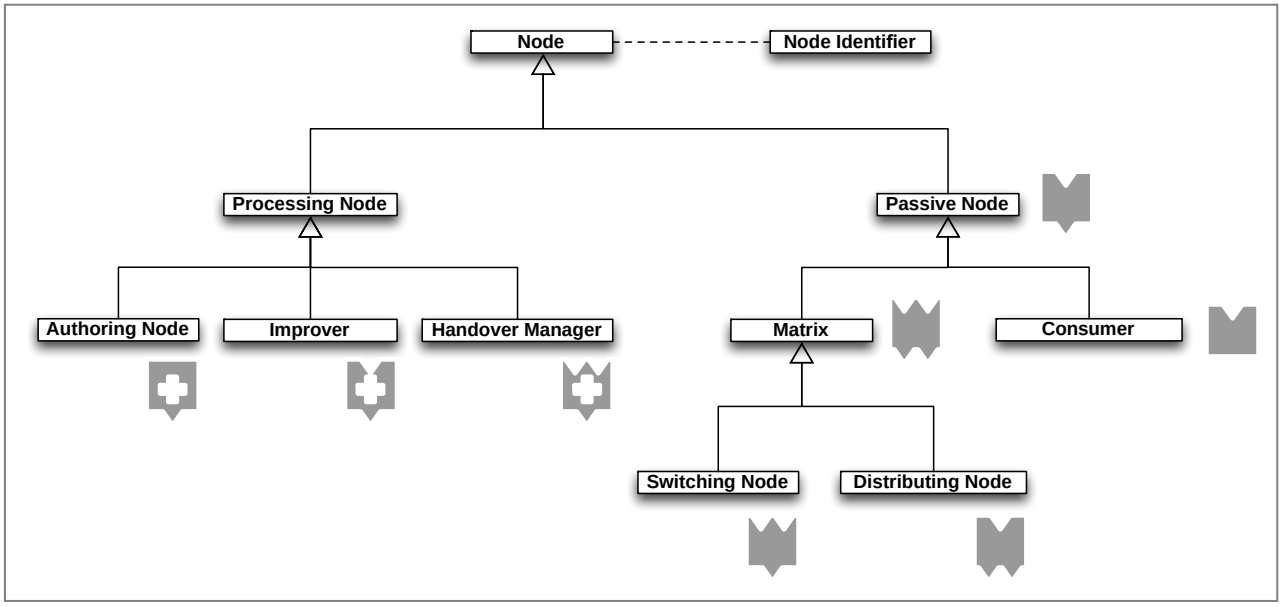

Figure 2: UML model diagram showing logical nodes and their relationships

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.

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

Figure 3 shows a UML class model illustrating the logical entities in this system model.

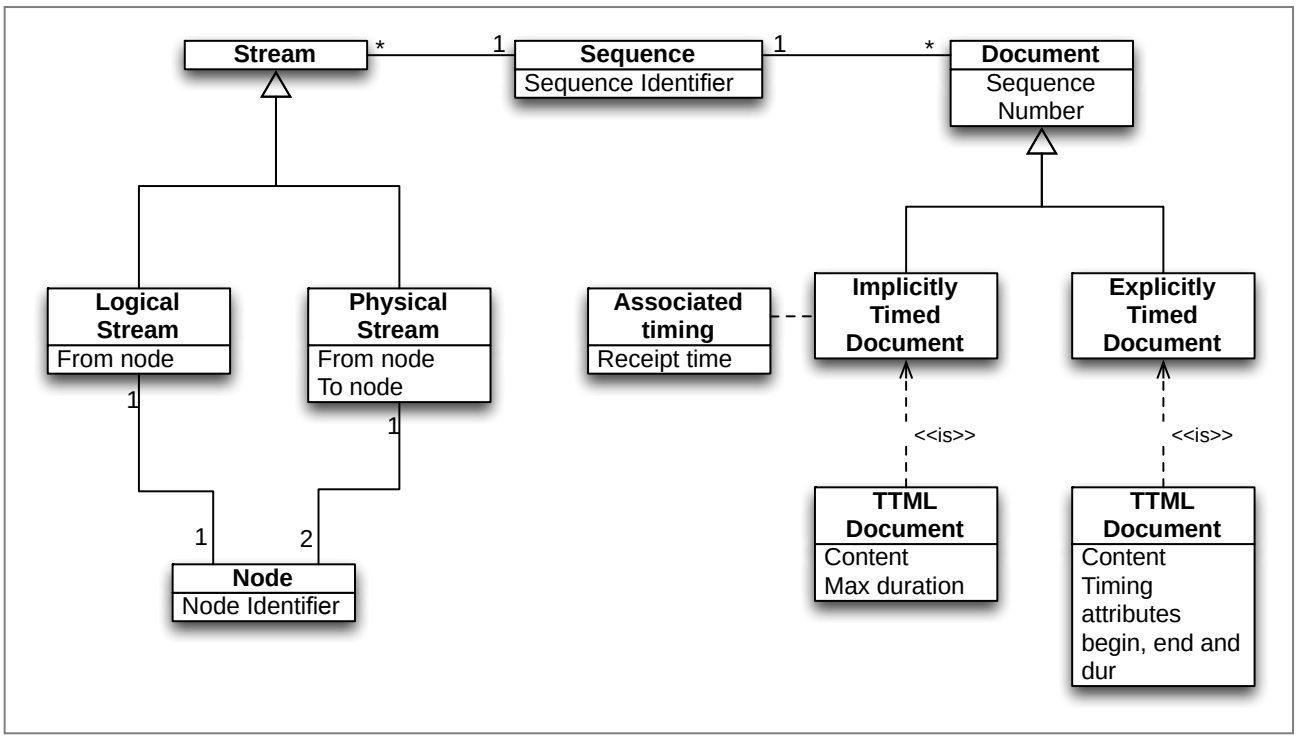

Figure 3: UML model showing system model logical entities

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 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.

2.3 Example scenarios (Informative)

The following examples represent typical real world scenarios in which documents and nodes that conform to this specification can be used.

Example 1

A team of subtitlers authors subtitles for a single programme, taking turns to contribute for a period of time before handing over to the next author.

Each subtitler creates a distinct sequence of subtitles for their turn. Each input sequence 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

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 3 – 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[5], for insertion into video, or other emission as a stream.

Example 4

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 5– 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 – 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 – not fully defined in this specification

An 'archiver' node has the task to create 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 – 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.

[5] 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.

Example 9 – not fully defined in this specification

A single broadcast channel has multiple, local downstream 'opt-outs'[6] for localisation purposes. A single subtitler authors the main stream, and other subtitlers author the 'opt-out' streams. Local downstream switching nodes facilitate the switching of the subtitle streams in time with the switch in video and audio streams.

Example 10 – not fully defined in this specification

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

2.4 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.

2.4.1 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 TTML1 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 [TTML1].
- The latest computed end time in the document, calculated from the `begin` and `end` attribute values if present, according to the semantics of [TTML1].
- 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.

[6] 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.

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 [TTML1] § 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"` cannot 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 case a presentation node has to derive comparable timing events from some other source, such as time code embedded in some related media.

Note: the `dur` attribute is not permitted in documents that have `ttp:markerMode="discontinuous"`.

2.4.1.1 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.

2.4.1.2 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) the document resolved begin time plus the value of the `dur` attribute, if present, (c) the latest computed end time in the document and (d) any externally specified document deactivation time, such as the end of sequence presentation.

2.4.1.3 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.

2.4.1.4 Document creation and issuing strategies (informative)

The rules above allow for implementations to use different strategies for creating and issuing sequences. It is beyond the scope of this document to attempt 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.

2.4.1.4.1 Implicitly timed documents

Documents that do not contain any `begin` or `end` attributes are described as having implicit timing (see [SMIL] § 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

active. 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 § 2.4.1 above.

See Figure 4 for a diagrammatic representation of the resolved begin and end times of implicitly timed documents.

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.

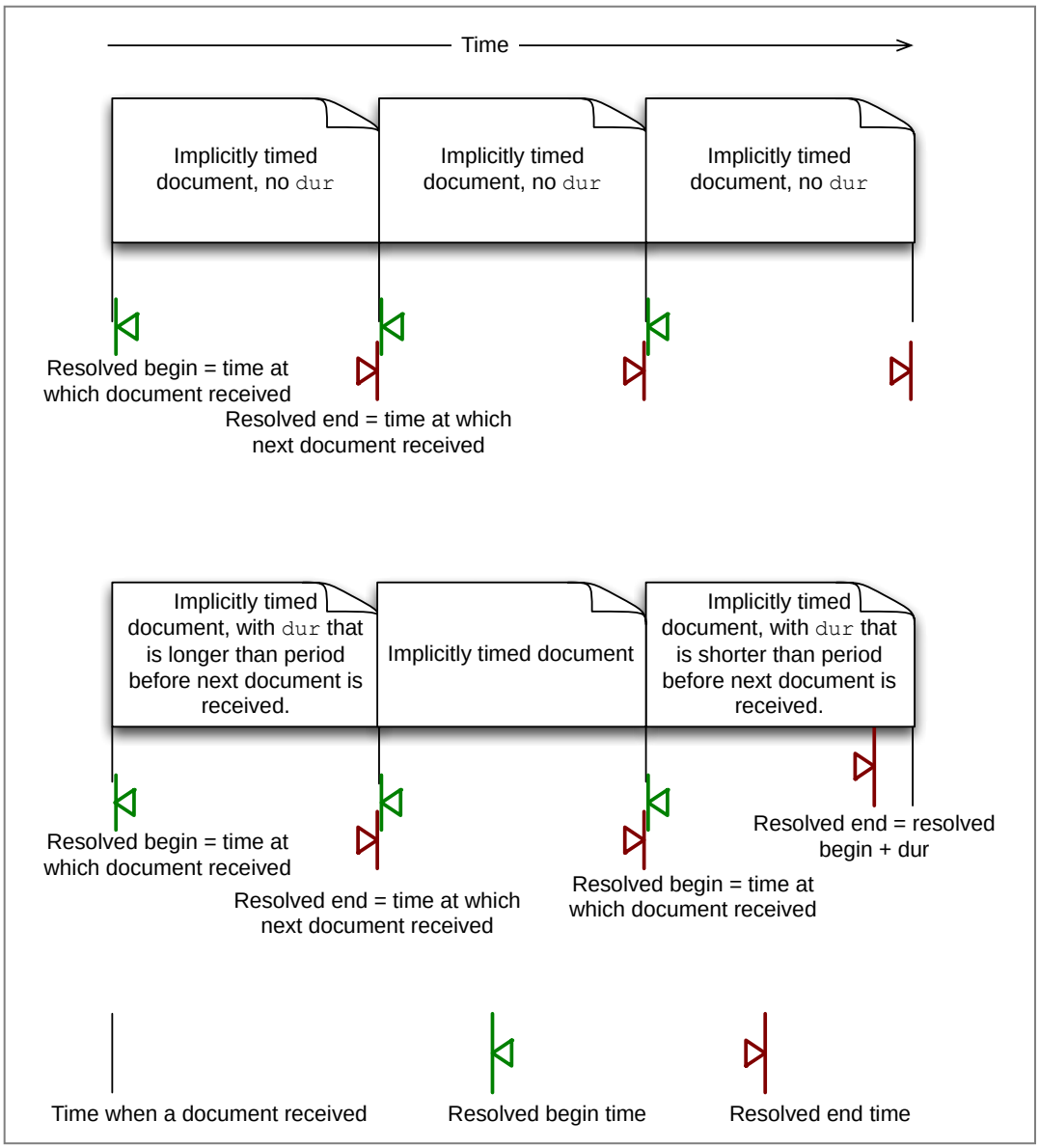

Figure 4: Diagram showing resolved begin and end times of implicitly timed documents with and without `dur` attributes

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 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.

2.4.1.4.2 Explicitly timed documents

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

See Figure 5 for a diagrammatic representation of the resolved begin and end times of explicitly timed documents.

Time graph showing activation based on document-specified times

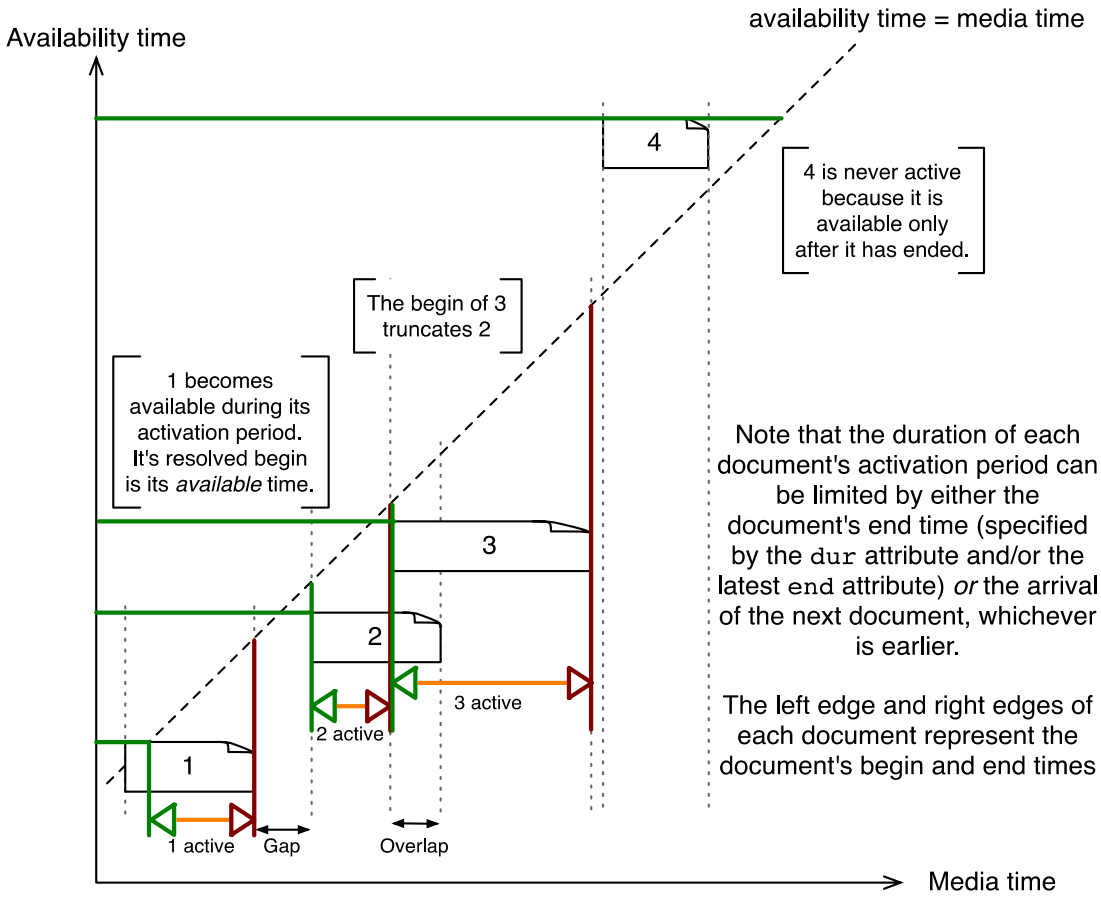

Figure 5: Diagram showing resolved begin and end times for explicitly timed documents

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. Where the marker mode indicates

that an event model is being used, by the `ttp:markerMode` attribute having the value `"discontinuous"`, the presentation processor needs to be able to derive comparable timing events from some other source, such as time code embedded in some related media.

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.

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. This is distinct from a sequence of implicitly timed documents where 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.

2.4.1.4.2.1 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 6, overleaf.

Time graph showing activation based on document-specified times using alternative strategy: cumulative documents where each contains all content

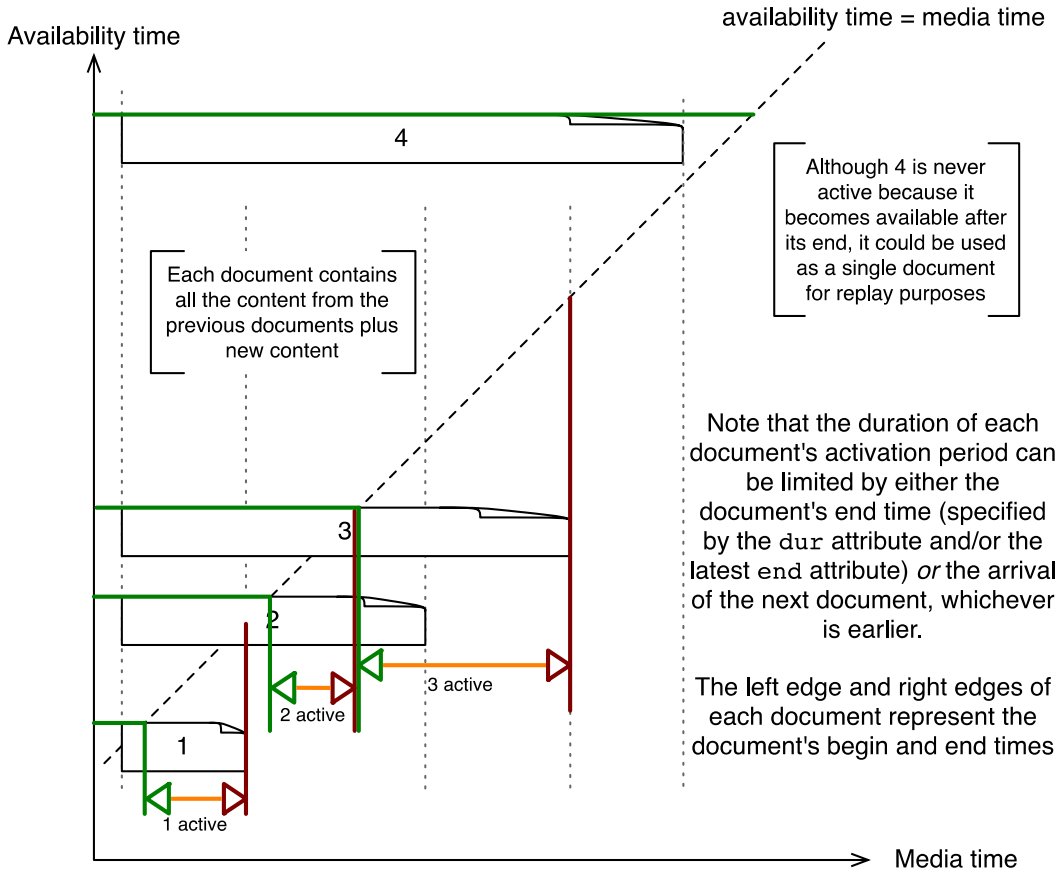

Figure 6: Timing graph showing cumulative document approach

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 to limit document sizes and parsing time.

2.4.1.4.3 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 7 for an illustration of how mixed documents are activated.

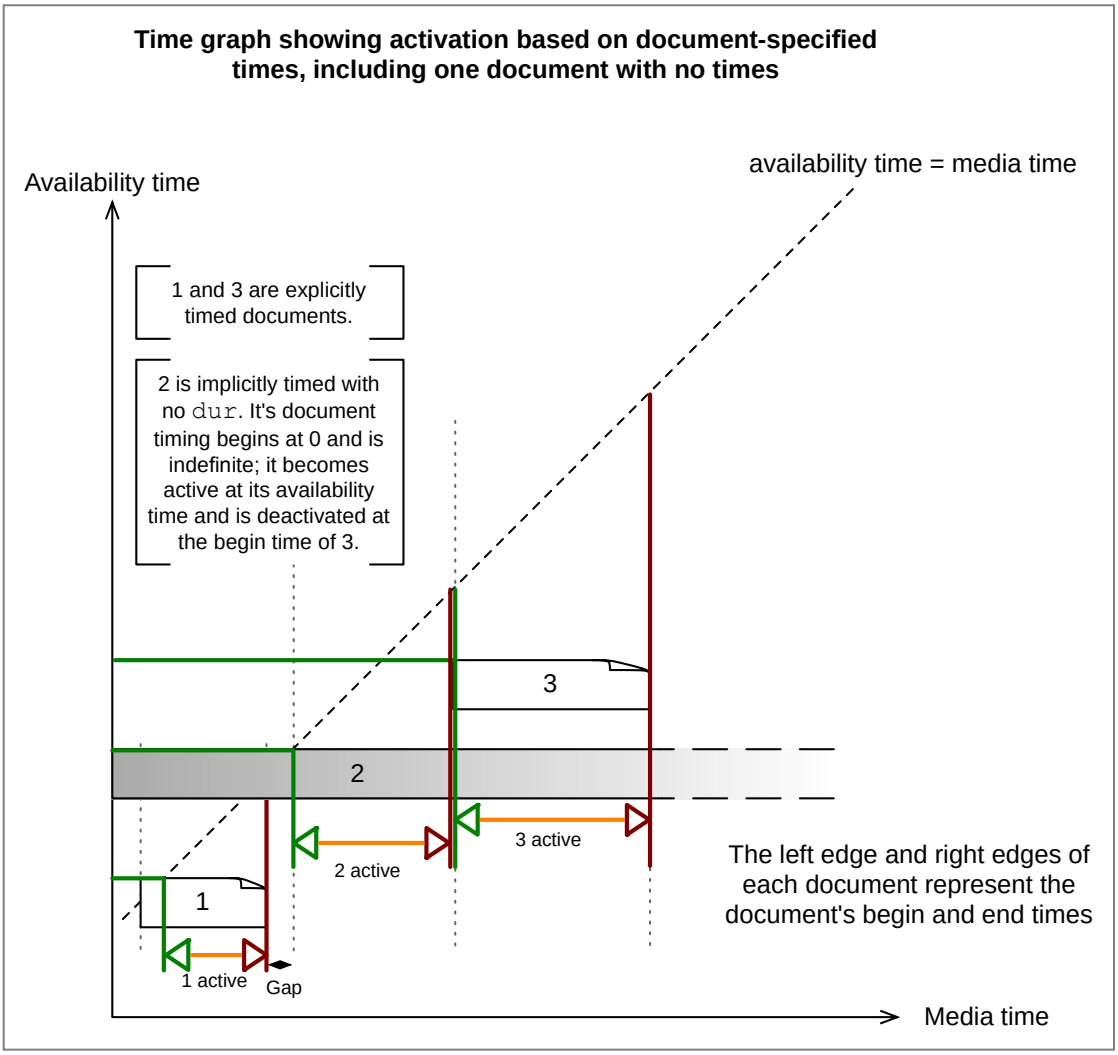

Figure 7: Timing graph showing document activation when a mix of implicitly and explicitly timed documents is in the same sequence

2.4.1.4.4 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.

2.4.1.4.5 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.

2.4.1.4.6 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.

2.4.2 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 pre-defined duration, encoded, packaged and distributed via a content delivery network.

In the context of live created subtitles, there is a variable authoring delay[7], 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 8 and 9.

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.

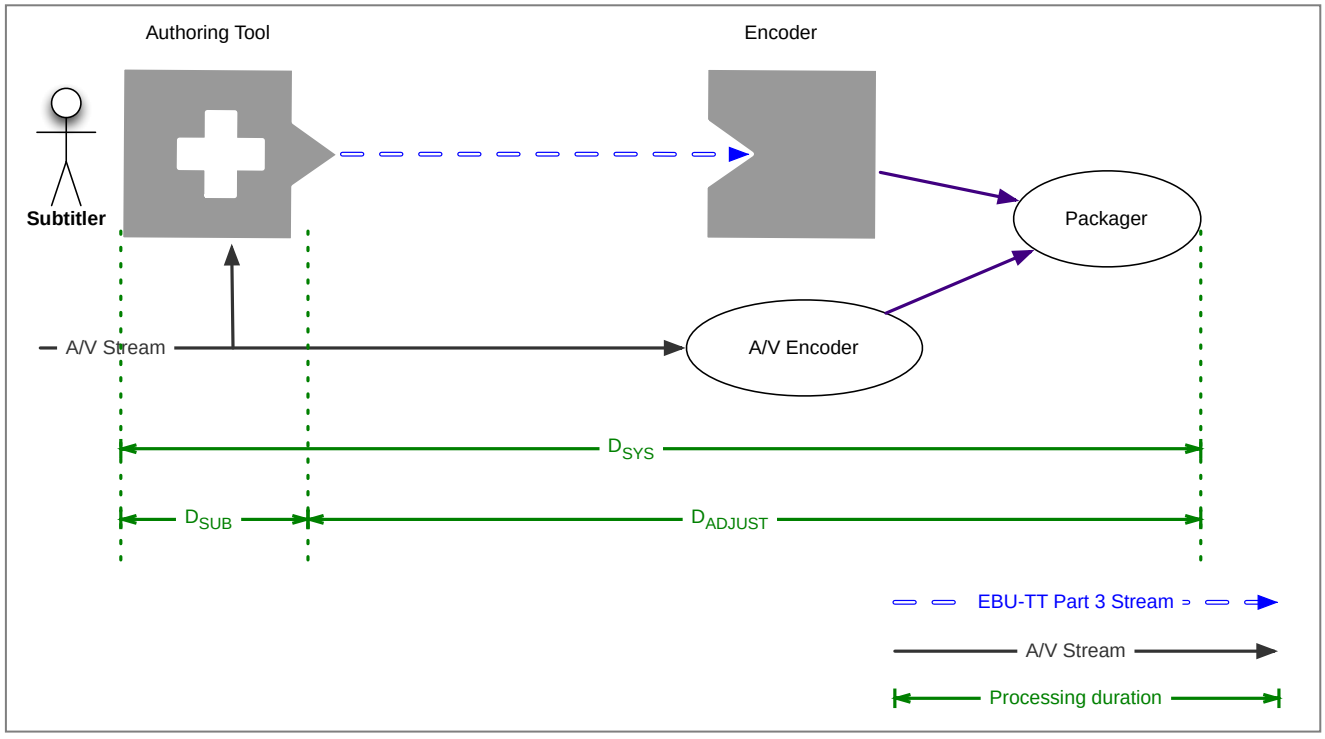

Figure 8: Use case diagram showing timing and delay concepts

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

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 $D_{SUB} + D_{ADJUST} = D_{SYS}$.

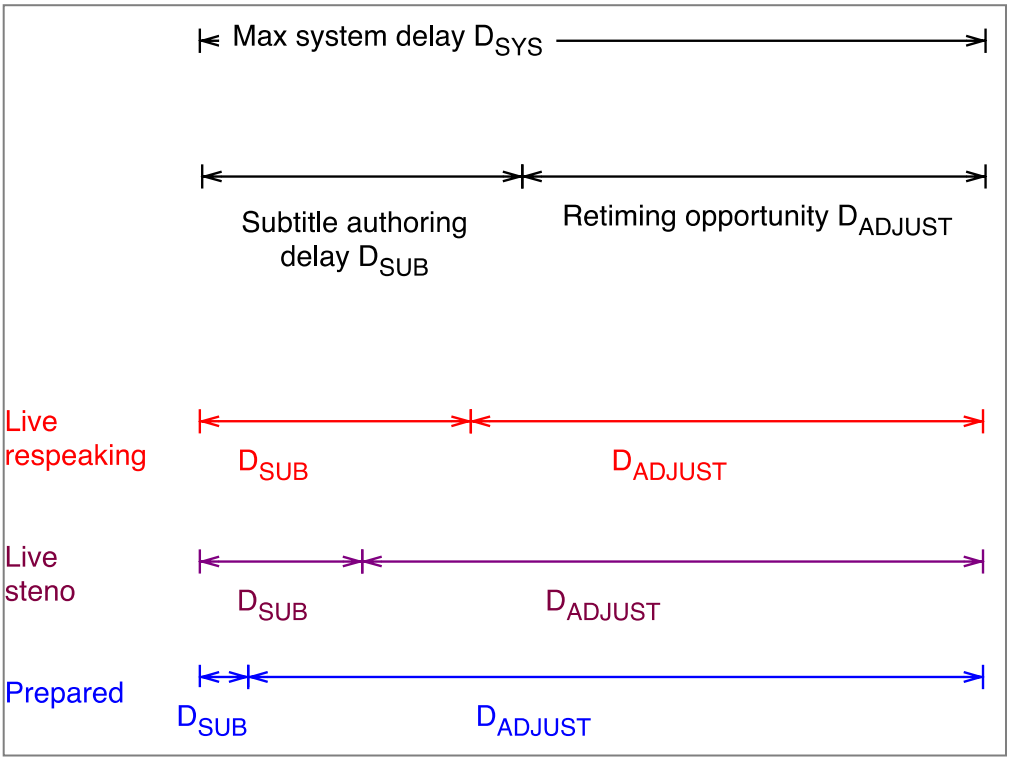

Figure 9: Opportunity for adjustable retiming of subtitles

The minimum value of D_{ADJUST} within a real world live authoring environment is 0[8] 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 9. Note that in some environments a solution for frequently late subtitle presentation could be to add a delay 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.

2.4.3 ebuttm:authoringDelay

The `ebuttm:authoringDelay` metadata attribute, 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 Authoring Node 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.

[8] 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 combined authoring interval is typically:

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

2.4.4 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.

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

1. A Delay node is a processing node. Therefore the output sequence SHALL have a different sequence identifier from the input sequence.
2. A Delay node MAY emit explicitly timed documents within a sequence. In this case the Delay node SHOULD not delay emission of the stream. NOTE: in this scenario it is recommended that when `ttp:timeBase="clock"` the time values within the documents relate to a reference clock that is available to downstream nodes for example consumer nodes.
3. A Delay node MAY emit implicitly timed documents within a sequence. In this case the Delay node SHALL delay emission of the stream by a period equivalent to the adjustment value.
4. A Delay node SHALL NOT emit an output sequence with reordered subtitles. NOTE: it is

possible that delay functionality is combined with other processing in a single node, for example an accumulator; hence this requirement 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 Delay node, though such a relationship would be expected for the simplest conceivable Delay.

5. A Delay node SHALL NOT update the value of `ebuttm:authoringDelay`.
6. A Delay node SHOULD add an `ebuttm:trace` element to the document metadata to indicate that the delay has been added.

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.

2.4.5 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"`, or when `ttp:timeBase="smpte"` is used, 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: future versions of this specification are likely to include local time offset and begin date semantics.

2.5 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 10.

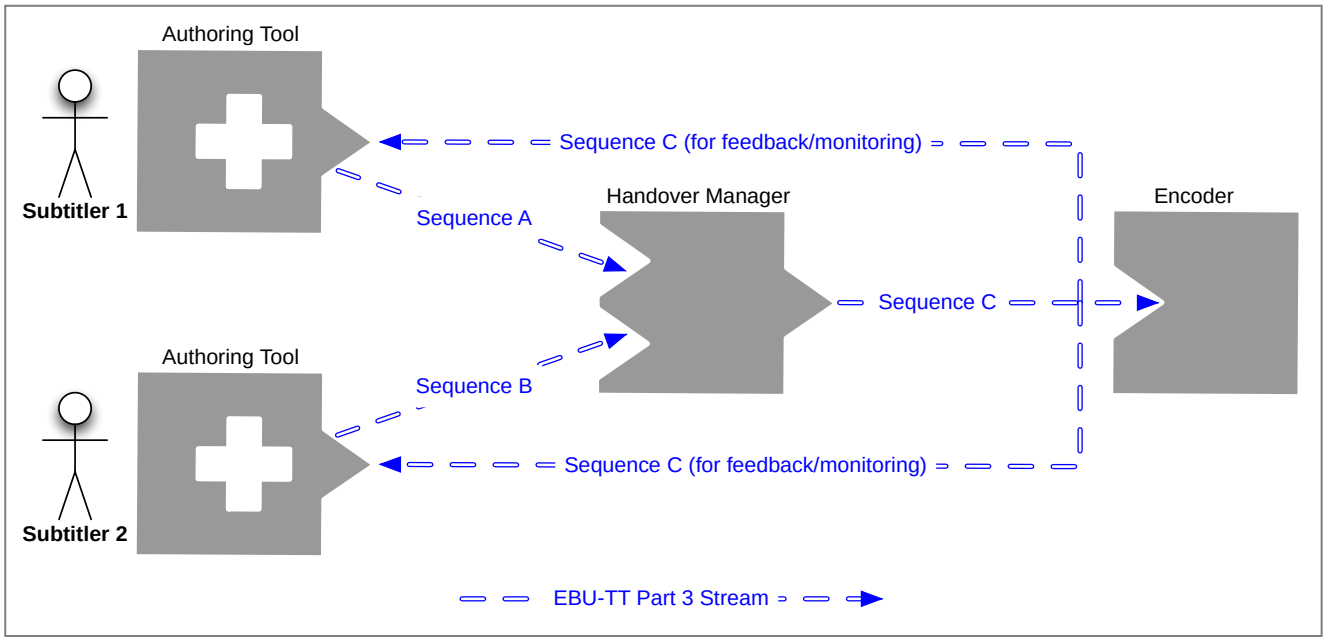

Figure 10: Use case showing a Handover Manager selecting between Sequences A and B and emitting Sequence C

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. See also Figure 11 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.

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.

2.5.1 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. All documents within a sequence that contain the element `ebuttp:authorsGroupIdentifier` SHALL have the same `ebuttp:authorsGroupIdentifier`.

The `ebuttp:authorsGroupControlToken` is a number that the Handover Manager uses to identify which sequence to select: when a sequence is received with a higher value `ebuttp:authorsGroupControlToken` than the currently selected sequence the Handover Manager SHALL switch to that sequence.

The `ebuttp:authorsGroupControlRequest` MAY be used to indicate to other subtitle authors within the same group the current author's intent, i.e. if they are continuing to author, or are requesting that another author should take over control.

The Handover Manager MAY include the parameters `ebuttp:authorsGroupIdentifier`, `ebuttp:authorsGroupControlToken` and `ebuttp:authorsGroupControlRequest` within its output sequence. This is so that each subtitle authoring station can know the current status, if subscribed to the Handover Manager's output stream.

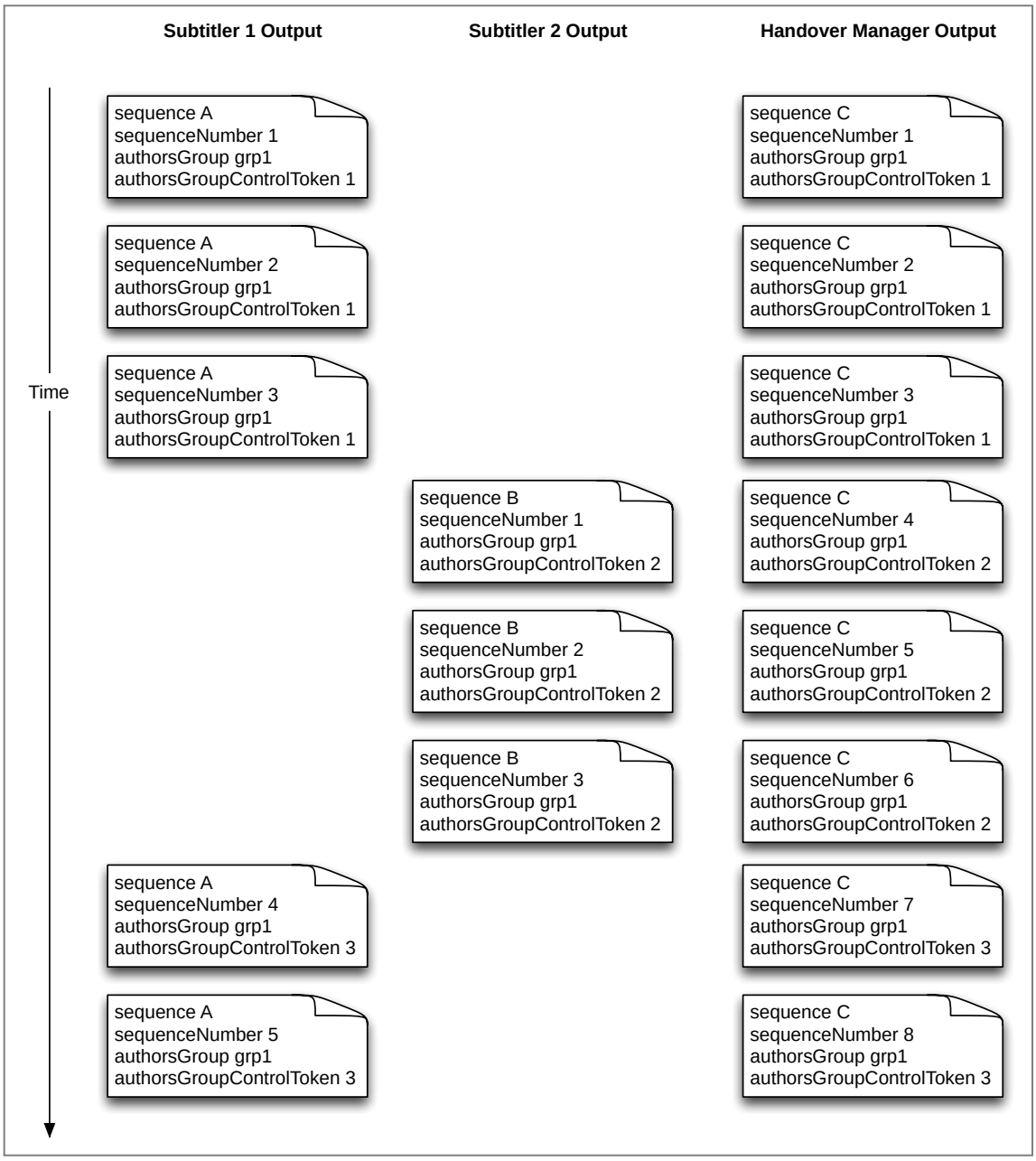

Figure 11: Sample sequences demonstrating Handover (omitting `ebuttp:authorsGroupControlRequest` for simplicity)

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:trace` element permits such logging. If present, a `trace` 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 a URI and is also not further defined here.

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

The ebuttm:trace 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 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

The following EBU-TT Part 3 elements may contain zero or one `tt:metadata` child element(s):

- `tt:head`
- `tt:styling`
- `tt:style`
- `tt:layout`
- `tt:region`
- `tt:body`
- `tt:div`
- `tt:p`
- `tt:span`
- `tt:br`

If an element has a `tt:metadata` as child element, `tt:metadata` shall appear before all other child elements that are defined for this element by EBU-TT Part 3 (see § 3 “Document Structure”).

Every `tt:metadata` element may be extended by zero or more elements as children. These

elements, their attributes and their XML Content shall neither be in a namespace defined by the TTML 1.0 specification nor in a namespace defined in the EBU-TT Part 3 specification.

Exceptions to this rule are as follows:

1) `tt:metadata` as child element of `tt:head` shall have one `ebuttm:documentMetadata` element
2) `tt:metadata` as child element of `tt:head` may have zero or more `ebuttm:binaryData` elements.
3) `tt:metadata` as child element of `tt:head` may have zero or more `ttm:agent` elements.
4) any `tt:metadata` element may have zero or one `ttm:title` element(s) and/or zero or one `ttm:desc` element(s).

3.1.3 Initial values

[TTML1] defines initial values for certain attributes that act as fall-back values in case a value cannot be computed from a specified value in the document. The EBU-TT Part 3 specification does not override these initial values and for any TTML 1.0 attribute that is used in an EBU-TT Part 3 document the initial value as specified in TTML 1.0 shall apply.

Note: To clarify the intention of the author of an EBU-TT Part 3 document it is recommended that attributes and their values be explicitly specified rather than relying on their initial values.

3.1.4 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 § 2.4.1 constitute constraints on the Document Processing Context as defined in TTML 1.0.

3.1.5 Unicode support

EBU-TT Part 3 processing and transformation engines SHOULD support Unicode characters and the Unicode bidirectional algorithm [UAX9].

3.1.6 White space handling

The `xml:space` attribute MAY be added to a `tt:tt`, `tt:p` or `tt:span` element to indicate the authors' intent in the use of white space (spaces, tabs, and blank lines).

In accordance with the W3C XML 1.0 Specification [XML1], the value "default" signals that the default white-space processing modes of the processing application are acceptable for this element; the value "preserve" indicates the intent that applications preserve all of the white space.

This declared intent is considered to apply to all elements within the content of the element where it is specified, unless it is overridden with another instance of the `xml:space` attribute.

3.2 Document Structure and Content Profile

This specification bases document compliance on EBU-TT Part 1 (Tech 3350) [EBUTT1] 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 (Tech 3350).

Please note that at time of publication of this document version 1.1 of EBU Tech 3350 is in the final phase of specification. EBU-TT Part 3 Live anticipates some of the changes made in this update and may base a future version of EBU Tech 3370 on EBU Tech 3350 version 1.1.

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 in the document instance.

Example:
The definition of the use of the `xml:id` attribute in § 3.1.2.1 "Style" specifies only the use of the `xml:id` attribute on the `tt:style` element.

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.

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
		Tech3350	Here	
`tt:tt/tt:body`	Element	1..1	0..1	Body may be omitted to indicate that all content should be removed, as a 'clear' instruction in an implicit processing context.
`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
`//(tt:body|tt:div)/@begin`	Attribute	0..0	0..1	
`//(tt:body|tt:div)/@end`	Attribute	0..0	0..1	
`//tt:p/@begin`	Attribute	1..1	0..1	
`//tt:p/@end`	Attribute	1..1	0..1	

3.2.2 Newly introduced elements and attributes

The following entities are introduced in this specification. The entity names are qualified with their namespace, if applicable. The location is the parent element within which the entity has been introduced.

Entity name	Entity type	Location
`ttp:timeBase` (value `"clock"`)	Attribute	`tt:tt`
`ttp:clockMode`	Attribute	`tt:tt`
`ebuttm:sequenceIdentifier`	Attribute	`tt:tt`
`ebuttm:sequenceNumber`	Attribute	`tt:tt`
`ebuttm:authoringDelay`	Attribute	`tt:tt`
`ebuttp:authorsGroupIdentifier`	Attribute	`tt:tt`
`ebuttp:authorsGroupControlToken`	Attribute	`tt:tt`
`ebuttp:authorsGroupControlRequest`	Attribute	`tt:tt`
`ebuttp:referenceClockIdentifier`	Attribute	`tt:tt`
`dur`	Attribute	`tt:body`
`ebuttm:originalSourceServiceIdentifier`	Element	`ebuttm:documentMetadata`
`ebuttm:intendedDestinationServiceIdentifier`	Element	`ebuttm:documentMetadata`
`ebuttm:trace`	Element	`ebuttm:documentMetadata`
`ebuttm:documentCreationMode`	Element	`ebuttm:documentMetadata`
`ebuttm:localTimeOffset`	Element	`ebuttm:documentMetadata`
`ebuttm:documentBeginTime`	Element	`ebuttm:documentMetadata`

The introduced entities are defined in further detail below.

3.2.2.1 tt:tt element

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

EBU-TT Part 3 uses the following parameters from [TTML1] 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	“smpte” | “media” | “clock”
Cardinality	1..1
Description	The timebase defines the time coordinate system for all time expressions in EBU-TT. If the timebase is “smpte” time expressions of `begin` and `end` attributes of the subtitle content shall be interpreted in the time coordinate system of SMPTE 12M-1-2008 and shall be of type `ebuttdt:smpteTimingType`. Additionally if the timebase is “smpte” the attributes `ttp:markerMode` and `ttp:dropMode` shall be specified on the `tt:element`. If the timebase is “media” then all time expressions of `begin` and `end` attributes of the subtitle content shall denote a coordinate on the time line of a media object and shall be of type `ebuttdt:mediaTimingType`. Note: The timebase “media” is intended to use the playtime of the associated video as reference. If the timebase is “clock” then all `begin` and `end` and `dur` attributes of the subtitle content shall denote a coordinate in some real-world time line and shall be of type `ebuttdt:clockTimingType`.

ttp:clockMode (attribute)

Type	`xs:string`
Enumeration	"local" | "gps" | "utc"
Cardinality	0..1
Description	`ttp:clockMode` specifies the interpretation of time expressions when `ttp:timebase` is set to clock. The attribute shall be specified when the value of the `ttp:timebase` attribute is “clock”. The semantics of the values “local”, “gps” and “utc” are defined in TTML 1.0.

The sequence to which every document belongs SHALL be identified using the `ebuttm:sequenceIdentifier` attribute.

ebuttm:sequenceIdentifier (attribute)

Type	`xs:string`
Cardinality	1..1
Description	Specifies the sequence to which this document belongs.

Every document with the same `ebuttm:sequenceIdentifier` SHALL be uniquely numbered using the `ebuttm:sequenceNumber` attribute.

ebuttm:sequenceNumber (attribute)

Type	`xs:positiveInteger`
Cardinality	1..1
Description	Specifies the document number within the sequence to which this document belongs.

The authoring delay may be indicated using the `ebuttm:authoringDelay` attribute. NOTE: this delay may be estimated or measured, and is intended to represent the duration between when the subtitle authoring tool received the instantaneous media for which subtitles were authored and the moment that the authoring tool emitted those subtitles.

ebuttm:authoringDelay (attribute)

Type	`ebuttdt:delayTimingType`
Cardinality	0..1
Description	Specifies the authoring delay associated with the timed content within this document.

The parameters `ebuttp:authorsGroupIdentifier`, `ebuttp:authorsGroupControlToken` and `ebuttp:authorsGroupControlRequest` are provided to facilitate handover between subtitle authors, using semantics defined for the Handover Manager node in § 2.5.

ebuttp:authorsGroupIdentifier (attribute)

Type	`xs:string`
Cardinality	0..1
Description	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
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:authorsGroupControlRequest (attribute)

Type	ebuttdt:delayTimingType
Enumeration	'request' | 'continuing'
Cardinality	0..1
Description	Allows an author to signal their intent to continue providing a stream, or to request that another author begins creating output with a higher value `ebuttp:authorsGroupControlToken`.

ebuttp:referenceClockIdentifier (attribute)

Type	xs:anyURI
Cardinality	0..1
Description	Allows the reference clock source to be identified. Permitted only when `ttp:timeBase="clock"` AND `ttp:clockMode="local"` OR when `ttp:timeBase="SMPTE"`.

3.2.2.2 ebuttm:documentMetadata element

The following additional metadata elements are defined by this specification. If used these elements shall be the first children of the `ebuttm:documentMetadata` element in the order documented below, after any metadata elements defined by EBU-TT Part 1 Tech3350.

ebuttm:documentCreationMode (element)

Type	`xs:string`
Enumeration	`"live" | "prepared"`
Cardinality	0..1
Description	The ebuttm:documentCreationMode identifies the overall workflow used to create the content in the document. The value "live" is intended to signify content that was originated out in real-time at the time of transmission. The value "prepared" is intended to signify content that has been prepared prior to transmission.

ebuttm:documentBeginDate (element)

Type	`xs:date`
Cardinality	0..1
Description	The value of `ebuttm:documentBeginDate` shall be the corresponding date of creation of the earliest begin time expression (i.e. the begin time expression that is the first coordinate in the document time line, if present). The `ebuttm:documentBeginDate` may be used to establish a synchronisation point between the document timeline and a referenced real world time line (as may be identified by `ebuttm:localTimeOffset` and/or `ebuttm:clockIdentifier`). A timezone SHALL NOT be specified. The timezone of `ebuttm:documentBeginDate` is the same timezone that is applied to the earliest `begin` time expression in the document, if present.

ebuttm:localTimeOffset (element)

Type	`xs:string`
Cardinality	0..1
Description	Specifies the timezone that applies to `begin` and `end` attributes when `ttp:timebase` is "clock" and `ttp:clockMode` is "local". When `ttp:timebase` is "clock" and `ttp:clockMode` is "local" the `ebuttm:localTimeOffset` is optional and may be present in an EBU-TT document. The timezone is specified as defined in ISO 8601. If a timezone is specified on `ebuttm:localTimeOffset` and `ebuttm:documentBeginDate` is included then `ebuttm:documentBeginDate` SHALL be processed in the timezone specified by `ebuttm:localTimeOffset`. Note: Some examples: "Z" - Universal Time (UTC) +hh:mm (e.g. +01:00) – local time zone is hh hours and mm minutes ahead of UTC -hh:mm (e.g. -08:00) – local time zone is hh hours and mm minutes behind UTC

The `ebuttm:originalSourceServiceIdentifier` MAY be used to identify the stream of audio-visual content that was used as the source for authoring the document.

ebuttm:originalSourceServiceIdentifier (element)

Type	`xs:string`
Cardinality	0..1
Description	Identifier of the audio-visual content that was used as the source for authoring the document.

The `ebuttm:intendedDestinationServiceIdentifier` MAY be used to identify the streams of destination audio-visual content for which the document is intended to apply.

ebuttm:intendedDestinationServiceIdentifier (element)

Type	`xs:string`
Cardinality	0..1
Description	Identifier of the audio-visual content that is the intended destination for the document.

ebuttm:trace (element)

Type	`xs:string`
Cardinality	0..*
Description	The trace element MAY be used to indicate processing that has been applied during the creation process of a document; multiple trace elements MAY be used to indicate a chain of processing.

The `ebuttm:trace` element may have the following attributes:

action (attribute)

Type	`xs:string`
Cardinality	1..1
Description	The action being noted by this trace element.

generatedBy (attribute)

Type	`xs:anyURI`
Cardinality	1..1
Description	The identifier for the node that performed the action being noted by this trace element.

sourceId (attribute)

Type	`xs:anyURI`
Cardinality	0..1
Description	The identifier for the node that supplied the source sequence from which the document was derived.

3.2.2.3 tt:body element

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

Timing information MAY be set using any combination of the following attributes:

- `begin`
- `end`
- `dur`

These attributes are used as defined in TTML 1. See also § 2.4.1 for further details of the semantics of document activation and the resolution of begin and end times for each document in a sequence.

The `dur` attribute SHALL NOT be used in documents that set `ttp:markerMode="discontinuous"`.

A document that contains a `tt:body` element with no content SHALL be treated as being active as defined by its timing attributes, and SHALL cause no content to be presented while it is active.

Note: [TTML1] § 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.

3.2.2.4 tt:div element

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

Timing information MAY be set using any combination of the following attributes:

- `begin`
- `end`

These attributes are used as defined in [TTML1]. See also § 2.4.1 for further details of the semantics of document activation and the resolution of begin and end times for each document in a sequence.

3.2.3 Modified attributes

The attributes `begin` and `end` SHALL have the type `ebuttdt:smpteTimingType`, `ebuttdt:mediaTimingType` or `ebuttdt:clockTimingType`, depending on the value of the `ttp:timeBase` parameter on the `tt:tt` element (see § 3.2.2.1 above).

Note: The `ebuttdt:clockTimingType` type is not present in [EBUTT1].

3.3 Datatypes

EBU-TT Part 3 defines specific datatypes to restrict the content of attributes or textual Element content.

EBU-TT Part 3 also uses any datatypes referenced and defined in [EBUTT1].

Note: If a datatype is applied to an attribute that was taken from [TTML1] 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.

3.3.1 ebuttdt:clockTimingType

The value of `ebuttdt:clockTimingType` shall be a Full-Clock -value or a Timecount-value.

A **Full-Clock-value** shall have the format *hh:mm:ss* followed by an optional *fraction*. Values of hours, minutes and seconds less than 10 shall be padded to two digits with a leading zero.

A *fraction* shall have a leading "." followed by a non-negative integer.

Examples for Full-Clock-values

- 02:30:03 = 2 hours, 30 minutes and 3 seconds
- 01:00:10.25 = 1 hour, 10 seconds and 250 milliseconds

A **Timecount-value** shall have the format:

Timecount followed by an optional *fraction* and a *symbol for the time metric*.

A Timecount shall be a non-negative integer.

A *fraction* shall have a leading "." followed by a non-negative integer.

A *symbol for time metric* shall be one of the following:

- "h" for hours
- "m" for minutes
- "s" for seconds
- "ms" for milliseconds

Examples for Timecount values:

3.2h = 3.2 hours = 3 hours and 12 minutes

45m = 45 minutes

30s = 30 seconds

5ms = 5 milliseconds

"frame" and "tick" based metrics shall not be used in a time expression of `ebuttdt:clockTimingType`.

3.3.2 ebuttdt:delayTimingType

The content shall be constrained to a signed (positive or negative) number with an optional decimal fraction, followed by a time metric being one of:

- "h" (hours)
- "m" (minutes)
- "s" (seconds)
- "ms" (milliseconds)

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 also Figure 2 above for a UML representation of the abstract node class structure.

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 Authoring Node

An authoring node consumes no sequences.

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 according to the semantics defined in section § 2.4.4.

4.1.1.1.3 Handover Manager

A Handover Manager consumes multiple sequences, according to the semantics defined in section § 2.5.

4.1.1.2 Passive Node

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

4.1.1.2.1 Matrix

A matrix receives multiple sequences and emits multiple sequences.

4.1.1.2.1.1 Switching Node

A switching node receives multiple 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 multiple 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. Bibliography

[EBUTT1]	**EBU Tech 3350**	EBU-TT Part 1 Subtitling format definition https://tech.ebu.ch/docs/tech/tech3350.pdf
[EBUTTD]	**EBU Tech3380**	EBU-TT-D Subtitling Distribution Format https://tech.ebu.ch/docs/tech/tech3380.pdf
[TTML1]	**TTML 1.0**	Timed Text Markup Language (TTML) 1.0 (Second Edition), W3C Recommendation http://www.w3.org/TR/2013/REC-ttml1-20130924/
[UAX9]	**UAX9**	Mark Davis. Unicode Standard Annex #9. Unicode Bidirectional Algorithm. http://unicode.org/reports/tr9/
[XML1]	**XML 1.0**	Tim Bray, et al. Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation, 26 November 2008. http://www.w3.org/TR/2008/REC-xml-20081126/
[SMIL]	**SMIL 2.1**	Synchronized Multimedia Integration Language (SMIL 2.1), W3C Recommendation. http://www.w3.org/TR/2005/REC-SMIL2-20051213/
[XPATH]	**XPATH 2.0**	XPath 2.0 (Second Edition) http://www.w3.org/TR/xpath20/

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.

ELEMENT INFORMATION ITEMS

```
<tt:tt
    ttp:timeBase = ( 'media' | 'smpte' | 'clock') #REQUIRED
    ttp:clockMode = ( 'local' | 'gps' | 'utc')
    xml:lang = ('' | <xs:language>) #REQUIRED

ttp:frameRate = <xs:positiveInteger>
    ttp:frameRateMultiplier = <ebuttdt:frameRateMultiplierType>
    ttp:markerMode = 'discontinuous'
    ttp:dropMode = ('nonDrop' | 'dropNTSC' | 'dropPal')
    ttp:cellResolution = <ebuttdt:cellResolutionType>
    tts:extent = <ebuttdt:extentType>
    xml:space = ('default'|'preserve')
    ebuttm:sequenceIdentifier = xs:string #REQUIRED
    ebuttm:sequenceNumber = xs:positiveInteger #REQUIRED
    ebuttm:authoringDelay = <ebuttdt:delayTimingType>
    ebuttp:authorsGroupIdentifier = xs:string
    ebuttp:authorsGroupControlToken = xs:positiveInteger
    ebuttp:authorsGroupControlRequest = ('request' | 'continuing')
    ebuttp:referenceClockIdentifier = xs:anyURI
    >
    Content: tt:head, tt:body?
</tt:tt>

<tt:head>
    Content: tt:metadata?, tt:styling?, tt:layout?
</tt:head>

<tt:metadata>
    Content:
        ebuttm:documentMetadata? Required in tt/head/metadata,
        ebuttm:binaryData* Only permitted in tt/head/metadata,
        ttm:agent* Only permitted in tt/head/metadata
        ttm:title?,
        ttm:desc?
        {Any element not in a TTML namespace and not in an EBU-TT namespace except
where explicitly permitted in this specification}
</tt:metadata>

<ebuttm:documentMetadata>
    Content:
        ebuttm:documentEbuttVersion, !-- Needs further work re conformsToStandard
& part 1
        ebuttm:documentIdentifier?,
        ebuttm:documentOriginatingSystem?,
        ebuttm:documentCopyright?,
        ebuttm:documentReadingSpeed?,
        ebuttm:documentTargetAspectRatio?,
        ebuttm:documentTargetActiveFormatDescriptor?,
        ebuttm:documentIntendedTargetBarData?,
        ebuttm:documentIntendedTargetFormat?,
        ebuttm:documentOriginalProgrammeTitle?,
        ebuttm:documentOriginalEpisodeTitle?,
        ebuttm:documentTranslatedProgrammeTitle?,
        ebuttm:documentTranslatedEpisodeTitle?,
```

```
        ebuttm:documentTranslatorsName?,
        ebuttm:documentTranslatorsContactDetails?
        ebuttm:documentSubtitleListReferenceCode?,
        ebuttm:documentCreationDate?,
        ebuttm:documentRevisionDate?,
        ebuttm:documentRevisionNumber?,
        ebuttm:documentTotalNumberOfSubtitles?,
        ebuttm:documentMaximumNumberOfDisplayableCharacterInAnyRow?,
        ebuttm:documentStartOfProgramme?,
        ebuttm:documentCountryOfOrigin?,
        ebuttm:documentPublisher?,
        ebuttm:documentEditorsName?,
        ebuttm:documentEditorsContactDetails?,
        ebuttm:documentUserDefinedArea?,
        ebuttm:documentCreationMode?,
        ebuttm:localTimeOffset?,
        ebuttm:documentBeginTime?,
        ebuttm:originalSourceServiceIdentifier?,
        ebuttm:intendedDestinationServiceIdentifier*,
        ebuttm:trace*
</ebuttm:documentMetadata>

<ebuttm:documentEbuttVersion>
    Content: 'v1.0'
</ebuttm:documentEbuttVersion>

<ebuttm:documentIdentifier>
    Content: <xs:string>
</ebuttm:documentIdentifier>

<ebuttm:documentOriginatingSystem>
    Content: <xs:string>
</ebuttm:documentOriginatingSystem>

<ebuttm:documentCopyright>
    Content: <xs:string>
</ebuttm:documentCopyright>

<ebuttm:documentReadingSpeed>
    Content: <xs:positiveInteger>
</ebuttm:documentReadingSpeed>

<ebuttm:documentTargetAspectRatio>
    Content: <xs:string>
</ebuttm:documentTargetAspectRatio>

<ebuttm:documentTargetActiveFormatDescriptor>
    Content: <xs:string>
</ebuttm:documentTargetActiveFormatDescriptor>

<ebuttm:documentIntendedTargetBarData
    position = ('topBottom' | 'leftRight') #REQUIRED
    lineNumberEndOfTopBar = <xs:nonNegativeInteger>
    lineNumberStartOfBottomBar = <xs:nonNegativeInteger>
    pixelNumberEndOfLeftBar = <xs:nonNegativeInteger>
    pixelNumberStartOfRightBar = <xs:nonNegativeInteger> >
    Content: <xs:string>
</ebuttm:documentIntendedTargetBarData>

<ebuttm:documentIntendedTargetFormat
    link = <xs:anyURI> >
    Content: <xs:string>
</ebuttm:documentIntendedTargetFormat>
```

```
<ebuttm:documentOriginalProgrammeTitle>
    Content: <xs:string>
</ebuttm:documentOriginalProgrammeTitle>

<ebuttm:documentOriginalEpisodeTitle>
    Content: <xs:string>
</ebuttm:documentOriginalEpisodeTitle>

<ebuttm:documentTranslatedProgrammeTitle>
    Content: <xs:string>
</ebuttm:documentTranslatedProgrammeTitle>

<ebuttm:documentTranslatedEpisodeTitle>
    Content: <xs:string>
</ebuttm:documentTranslatedEpisodeTitle>

<ebuttm:documentTranslatorsName>
    Content: <xs:string>
</ebuttm:documentTranslatorsName>

<ebuttm:documentTranslatorsContactDetails>
    Content: <xs:string>
</ebuttm:documentTranslatorsContactDetails>

<ebuttm:documentSubtitleListReferenceCode>
    Content: <xs:string>
</ebuttm:documentSubtitleListReferenceCode>

<ebuttm:documentCreationDate>
    Content: <xs:date>
</ebuttm:documentCreationDate>

<ebuttm:documentRevisionDate>
    Content: <xs:date>
</ebuttm:documentRevisionDate>

<ebuttm:documentRevisionNumber>
    Content: <xs:nonNegativeInteger>
</ebuttm:documentRevisionNumber>

<ebuttm:documentTotalNumberOfSubtitles>
    Content: <xs:nonNegativeInteger>
</ebuttm:documentTotalNumberOfSubtitles>

<ebuttm:documentMaximumNumberOfDisplayableCharacterInAnyRow>
    Content: <xs:nonNegativeInteger>
</ebuttm:documentMaximumNumberOfDisplayableCharacterInAnyRow>

<ebuttm:documentStartOfProgramme>
    Content: <ebuttdt:smpteTimingType>
</ebuttm:documentStartOfProgramme>

<ebuttm:documentCountryOfOrigin>
    Content: <xs:string>
</ebuttm:documentCountryOfOrigin>

<ebuttm:documentPublisher>
    Content: <xs:string>
</ebuttm:documentPublisher>

<ebuttm:documentEditorsName>
    Content: <xs:string>
</ebuttm:documentEditorsName>
```

```
<ebuttm:documentEditorsContactDetails>
    Content: <xs:string>
</ebuttm:documentEditorsContactDetails>

<ebuttm:documentUserDefinedArea>
    Content: <xs:string>
</ebuttm:documentUserDefinedArea>

<ebuttm:binaryData
    textEncoding = 'BASE64' #REQUIRED
    binaryDataType = <xs:string> #REQUIRED
    fileName = <xs:string> >
    Content: <xs:string>
</ebuttm:binaryData>

<ebuttm:documentCreationMode>
    Content: "live" | "prepared"
</ebuttm:documentCreationMode>

<ebuttm:documentBeginDate>
    Content: <xs:date> <!-- Timezone not permitted -->
</ebuttm:documentBeginDate>

<ebuttm:localTimeOffset>
    Content: <xs:string>
</ebuttm:localTimeOffset>

<ebuttm:originalSourceServiceIdentifier>
    Content: <xs:string>
</ebuttm:originalSourceServiceIdentifier>

<ebuttm:intendedDestinationServiceIdentifier>
    Content: <xs:string>
</ebuttm:intendedDestinationServiceIdentifier>

<ebuttm:trace
    action = <xs:string> #REQUIRED
    generatedBy = <xs:anyURI> #REQUIRED
    sourceId = <xs:anyURI> >
    Content: <xs:string>
</ebuttm:trace>

<tt:styling>
    Content: tt:metadata?, tt:style*
</tt:styling>

<tt:style
    xml:id = <xs:ID> #REQUIRED
    style = <xs:IDREFS>
    tts:backgroundColor = <ebuttdt:colorType>
    tts:color = <ebuttdt:colorType>
    tts:direction = ( 'ltr' | 'rtl' )
    tts:fontFamily = <ebuttdt:fontFamilyType> As defined in TTML 1.0 [3], section
8.2.8
    tts:fontSize = <ebuttd:fontSizeType>
    tts:fontStyle = ( 'normal' | 'italic' )
    tts:lineHeight = ( 'normal' | <ebuttd:lengthType>)
    tts:fontWeight = ( 'normal' | 'bold' )
    tts:padding = <ebuttdt:paddingType>
    tts:textAlign = ( 'left' | 'center' | 'right' | 'start' | 'end' )
    tts:textDecoration = ( 'none' | 'underline' )
    tts:unicodeBidi = ( 'normal' | 'embed' | 'bidiOverride' )
    ebutts:multiRowAlign = ('start' | 'center' | 'end' | 'auto')
    ebutts:linePadding = <ebuttdt:linePaddingType> >
```

```
    Content: tt:metadata?
</tt:style>

<tt:layout>
    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')>
    Content: tt:metadata?
</tt:region>

<tt:body
    xml:id = <xs:ID>
    begin = (<ebuttd:smpteTimingType> | <ebuttdt:mediaTimingType> |
<ebuttdt:clockTimingType>)
    end = (<ebuttd:smpteTimingType> | <ebuttdt:mediaTimingType> |
<ebuttdt:clockTimingType>)
    dur = (<ebuttd:smpteTimingType> | <ebuttdt:mediaTimingType> |
<ebuttdt:clockTimingType>)
    ttm:agent = <xs:IDREFS>
    ttm:role = As defined in TTML 1.0 [3], section 12.2.2>
    Content: tt:metadata?, tt:div+
</tt:body>

<tt:div
    xml:id = <xs:ID>
    style = <xs:IDREFS>
    begin = (<ebuttd:smpteTimingType> | <ebuttdt:mediaTimingType> |
<ebuttdt:clockTimingType>)
    end = (<ebuttd:smpteTimingType> | <ebuttdt:mediaTimingType> |
<ebuttdt:clockTimingType>)>
    region = <xs:IDREF>
    xml:lang = ('' | <xs:language>)
    ttm:agent = <xs:IDREFS>
    ttm:role = As defined in TTML 1.0 [3], section 12.2.2>
    Content: tt:metadata?, (tt:div+ | (tt:div*, tt:p+))
</tt:div>

<tt:p
    xml:id = <xs:ID> #REQUIRED
    begin = (<ebuttd:smpteTimingType> | <ebuttdt:mediaTimingType> |
<ebuttdt:clockTimingType>)
    end = (<ebuttd:smpteTimingType> | <ebuttdt:mediaTimingType> |
<ebuttdt:clockTimingType>)
    style = <xs:IDREFS>
    region = <xs:IDREF>
    xml:lang = ('' | <xs:language>)
    xml:space = ('default'|'preserve')
    ttm:agent = <xs:IDREFS>
    ttm:role = As defined in TTML 1.0 [3], section 12.2.2>
    Content (Mixed): tt:metadata?, (tt:span|tt:br)*
</tt:p>

<tt:span
    xml:id = <xs:ID>
    begin = (<ebuttd:smpteTimingType> | <ebuttdt:mediaTimingType> |
<ebuttdt:clockTimingType>)
```

```
    end = (<ebuttd:smpteTimingType> | <ebuttdt:mediaTimingType> |
<ebuttdt:clockTimingType>)
    style = <xs:IDREFS>
    xml:lang = ('' | <xs:language>)
    xml:space = ('default'|'preserve')
    ttm:agent = <xs:IDREFS>
    ttm:role = As defined in TTML 1.0 [3], section 12.2.2>
    Content (Mixed): tt:metadata?, tt:br*
</tt:span>

<tt:br
    ttm:role = As defined in TTML 1.0 [3], section 12.2.2>
    Content: tt:metadata?
</tt:br>
```

EXPRESSIONS

```
ebuttdt:CellResolutionType
    : columns rows

columns | rows
    : <xs:positiveInteger>

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

ebuttdt:extentType
    : width height

width | height
    : <ebuttdt:lengthType>

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

ebuttdt:fontSizeType
    : <ebuttdt:lengthType> <ebuttdt:lengthType>?

ebuttdt:frameRateMultiplierType
    : numerator denominator

numerator | denominator
    : <xs:positiveInteger>

ebuttdt:lengthType
    : scalar
    | percentage

scalar
    : number units

percentage
    : number "%"

sign
    : "+" | "-"

number
    : sign? non-negative-number

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 y-coord

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

ebuttdt:paddingType
    : all-edges
    | before end after start

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

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

ebuttdt:smpteTimingType
    : hh ":" mm ":" ss ":" ff

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

ebuttdt:mediaTimingType
    : full-clock-value
    | timecount-value

full-clock-value
    : hh ":" mm ":" ss ("." <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"

ebuttdt:clockTimingType
    : full-clock-value
    | timecount-value

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