EBU

OPERATING EUROVISION AND EURORADIO

TECH 3370s1

EBU-TT, PART 3 SUPPLEMENT 1 LIVE SUBTITLING APPLICATIONS: CARRIAGE OVER WEBSOCKET

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:
Carriage over WebSocket

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

Keywords: Subtitling, XML, W3C, TTML, DFXP, captions, EBU Timed Text, live, EBU-TT, encoder, WebSocket, IETF, TLS.

Scope (Informative)

This document describes how EBU-TT Live documents (EBU-TT, Part 3, EBU Tech 3370) [1] can be carried from node to node using WebSocket connections.

1. Introduction

The EBU-TT Part 3 specification [1] defines the payload data for live subtitling applications, and the requirements for specifying carriage mechanisms for transferring live subtitle sequences between nodes.

This document defines one such carriage mechanism, based on the WebSocket protocol. This protocol provides a way to establish TCP socket connections over IP networks in a "web friendly" way that uses HTTP upgrades and has a TLS encryption mode.

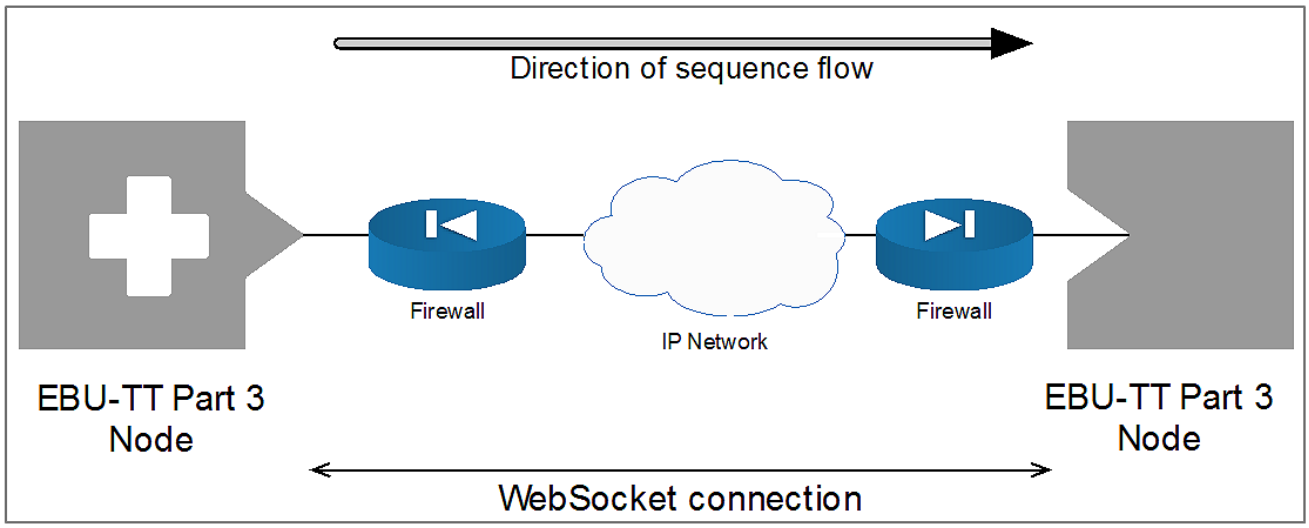

Figure 1: Conceptual diagram illustrating system model of node to node connection using WebSocket over an IP network

1.1 Document Structure

This document has the following structure:

Section 1 is this Introduction.

Section 2 defines terms and introduces core concepts.

Section 3 describes the details of the WebSocket carriage specification and meets the requirements of Annex B of EBU-TT Part 3.

2. Definitions and Concepts

2.1 Definition of terms

Terms defined in EBU-TT Part 3 carry the same definitions in this document. Additional terms are defined below:

EBU-TT Live Document

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

EBU-TT Part 3 Document

A document as defined in EBU-TT Part 3.

EBU-TT Part 3 sequence

A sequence of EBU-TT Live Documents as defined in EBU-TT Part 3.

3. Carriage Specification Details

WebSocket is a protocol that may be used to transfer a stream of EBU-TT live documents over an IP network from one node to another. WebSocket is specified as an IETF Proposed Standard, IETF RFC6455 [2].

3.1 Core networking dependencies and connection protocols

WebSocket carriage is based on TCP socket connections initiated via an HTTP upgrade request.

WebSocket does not in itself impose any data rate constraints; such constraints are imposed by the underlying network infrastructure. Data payloads are delivered in packets of limited size; these do not impact the maximum size of any EBU-TT Live document that can be sent. The TCP protocol ensures that all data is delivered in order. Operational implementations should ensure that there is sufficient data rate capacity available to carry the volume EBU-TT Live data needed for the application without imposing growing delay.

Such a growing delay would manifest as increasingly late availability times of EBU-TT Live documents, which in turn may truncate the visibility of presented text or delay its appearance.

3.2 Synchronisation impacts and/or thresholds

The latency of WebSocket and TCP socket connections is generally dependent on the network infrastructure used. The protocol guarantees that packets are all delivered, and delivered in the same order in which they were sent. This guarantee can impose transient additional latency if network conditions change. For example failure to receive acknowledgement of delivery of a packet causes further data effectively to be held back until that packet can be successfully delivered.

Such an increase in latency would manifest as an increased availability time for EBU-TT Live documents, which in turn may truncate the visibility of presented text or delay its appearance.

Techniques available for handling this include careful management of the network conditions, for example setting up static routes or using software defined networks. If such techniques are not locally practical then careful consideration should be given to decide if WebSocket is indeed an

appropriate carriage mechanism to use[1].

3.3 Information Security

See [2] §10. Security Considerations for further general details.

3.3.1 Encryption

Use of the `wss:` URI (Uniform Resource Identifier) [3] scheme requires the connection to be encrypted using TLS (Transport Layer Security).

3.3.2 Authentication

From [2] §10.5 WebSocket Client Authentication:

> The WebSocket server can use any client authentication mechanism available to a generic HTTP server, such as cookies, HTTP authentication, or TLS authentication.

3.3.3 Error checking and correction

The WebSocket protocol does not offer any inherent error checking and correction. As stated in [2] §10.7 Handling of Invalid Data, both sending and receiving nodes shall validate documents and should close the connection if invalid data is received.

To avoid issues with encoding of data with Text data type all documents shall be encoded as UTF-8.

3.3.4 Crossing organisational boundaries

The WebSocket protocol facilitates transition across organisational boundaries with firewalls etc. by using ports that are commonly open for HTTP; communication can pass through web proxy servers. Use of the secure web socket protocol provides a straightforward to use mechanism for encrypting communication between organisations when the intermediate network is not under direct control of either party, such as the open internet.

3.4 Endpoint cardinality

WebSocket is a point to point full duplex protocol, with connections being defined by two single end points.

Many implementations offer a 'broadcast' mode in which multiple connections may be made to or from the same endpoint, and the same data is sent over each connection. Such an implementation could be used to provide a Distributing Node for example.

Generally the performance of such implementations is dependent on network capacity and processing power; for EBU-TT Live applications, practical experience shows that small numbers of connections (for example, fewer than 10) do not impose any significant additional latency on current hardware, however care should be taken to ensure that specific applications work in the applicable operational environment.

EBU-TT Live requires only unidirectional flow of data. The reverse channel may be used for application specific purposes if and only if both nodes can validate such application specific data; in general this is a risky strategy since unexpected data causes connection closure – see §3.5 below. The reverse channel shall not be used for providing an EBU-TT Live stream back from a Handover Node; such a stream shall be made available using a separate connection.

Implementations should document how many simultaneous connections each node supports.

[1] Other network based protocols such as those based on UDP trade guaranteed delivery for guaranteed latency; in some such protocols it is possible for data loss to occur.

Since connections can be initiated from either the Node that sends EBU-TT Part 3 sequences or a node that receives them, and it is operationally important to know which connection "direction" applies. Implementations shall document whether they accept incoming connections or make outgoing connections.

A WebSocket Distributing Node that supports incoming connections for both receiving and sending can be used to mediate between implementations that make outgoing connections and those that accept incoming connections by providing a 'receiving' incoming connection and offering 'sending' connection(s) and passing through documents from one to the other without modifying them. See Figure 2 below.

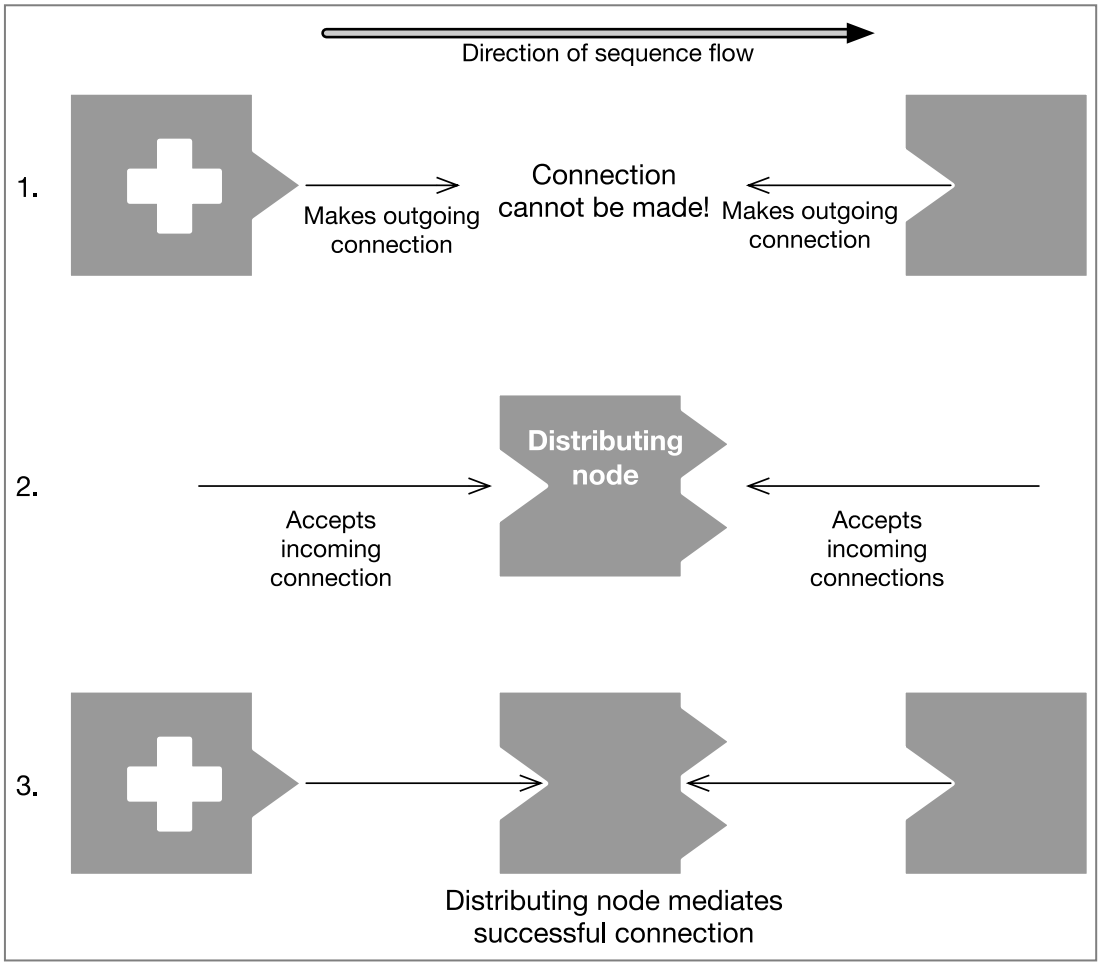

Figure 2: Showing how a Distributing node can be used to mediate between nodes with mutually incompatible connection initiation directions

Similarly a "connecting" node can mediate in the other direction by simultaneously making two outgoing connections, one to a "sender" and the other to a "receiver" and passing documents from the sender to the receiver without modifying them.

3.5 Connection lifecycle management

A WebSocket connection is initiated when a node dereferences a URI with a `ws:` or `wss:` URI scheme whose address resolves to a second node that supports the WebSocket Protocol, and the WebSocket opening handshake is successfully completed. A WebSocket connection is closed when either node initiates the closing handshake.

The WebSocket protocol and underlying TCP layer maintains established connections without further application support being required; no keep-alive messages are required by this carriage mechanism.

Nodes should close connections if unexpected or invalid data is received. Implementations shall support EBU-TT Part 3 documents. Implementations shall not close connections if valid EBU-TT live documents are received. Implementations should document any other data formats that they support.

3.6 Channel routing

This carriage mechanism does not specify or recommend any channel routing mechanisms.

Various schemes for maintaining and querying registries of, and brokering access to, available data sources (which could be EBU-TT Live sequences) are feasible and in use.

It would be reasonable to consider schemes that are either coincident with those intended for wider broadcast applications, such as the AMWA NMOS Discovery & Registration API [4], and that are aligned with the EBU-sponsored Joint Taskforce on Networked Media Reference Architecture [5].

Alternatively schemes that are more targeted towards a generic WebSocket infrastructure, such as the Web Application Messaging Protocol [6] may be more appropriate for some applications.

One practical operational consideration for implementations is the design of the WebSocket URI. See [2] §12 and [3] for further details. The following examples are illustrative only:

`ws://sequence.organisation.tld:port/` is a URI that uses the DNS system to resolve the host "`sequence.organisation.tld`" at port "`port`" to a single specified sequence.

`ws://organisation.tld:port/sequence` is a URI that uses the DNS system to resolve the host "`organisation.tld`" at port "`port`" which may offer multiple resources (some of which may offer EBU-TT Live sequences), and requests the specific resource "sequence".

`ws://organisation.tld:port/sequence/subscribe` might be a URI that uses the DNS system to resolve the host "`organisation.tld`" at port "`port`", and requests a subscription to the resource "`sequence`". An alternative but not equivalent formulation of this URI could be `ws://organisation.tld:port/sequence?subscribe`

`ws://organisation.tld:port/sequence/publish` might be a URI that uses the DNS system to resolve the host "`organisation.tld`" at port "`port`", and requests to *publish* data matching the resource "`sequence`", i.e to provide that data to any subscribers. An alternative but not equivalent formulation of this URI could be `ws://organisation.tld:port/sequence?publish`

`ws://1.2.3.4:9000/` is a URI that specifies a specific IP address and port without requiring resolution using the DNS system.

3.6.1 Sequence Identifier Encoding in URIs

The data type of `ebuttp:sequenceIdentifier` is `xs:string` with a minimum length of 1. It can therefore contain characters that are reserved in URIs, as defined in [6] §2.2. Prior to inserting into a URI the sequence identifier shall be percent-encoded exactly once. On extraction from a URI, and prior to use as a sequence identifier, the value shall be percent-decoded exactly once.

3.7 Stability

WebSocket and TCP guarantee delivery of data in the same order in which it was sent at the expense of latency if required.

Constant latency can only be achieved by managing the underlying network infrastructure.

Given a reliable network infrastructure a connection remains open indefinitely until either node closes it.

5. References

[1] **EBU Tech 3370** EBU-TT, Part 3 Live Subtitling applications http://tech.ebu.ch/publications/tech3370

[2] **IETF RFC6455** The WebSocket Protocol. https://tools.ietf.org/html/rfc6455, updated by https://tools.ietf.org/html/rfc7936

[3] **IETF RFC3986** Uniform Resource Identifier (URI): Generic Syntax https://tools.ietf.org/html/rfc3986

[4] **ANDR API** AMWA NMOS Discovery and Registration API; https://github.com/AMWA-TV/nmos

[5] **JT-NM Reference Architecture** Reference Architecture v1.0 http://jt-nm.org/RA-1.0/JT-NMReferenceArchitecturev1.0%20150904%20FINAL.pdf

[6] **WAMP** The Web Application Messaging Protocol; http://wamp-proto.org/

6. Bibliography

EBU Tech 3264 Specification of the EBU Subtitling data exchange format. http://tech.ebu.ch/publications/tech3264

EBU Tech 3350 EBU TT, Part 1; Subtitling format definition. http://tech.ebu.ch/publications/tech3350

EBU Tech 3360 EBU TT, Part 2 Mapping EBU STL (Tech 3264) to EBU TT subtitle files http://tech.ebu.ch/publications/tech3360

EBU Tech 3380 EBU TT, Part D Subtitling Distribution Format https://tech.ebu.ch/publications/tech3380

EBU Tech 3390 EBU TT, Part M Metadata definitions. http://tech.ebu.ch/publications/tech3390

EBU R 133 Recommendation on transport of subtitles inside and outside MXF files http://tech.ebu.ch/publications/r133

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)

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/

TLS Transport Layer Security 1.3 (draft); https://github.com/tlswg/tls13-spec