In-Browser XML Document Streaming

Cyril Concolato
Institut Mines-Télécom; Telecom ParisTech; CNRS LTCI
<cyril.concolato@telecom-paristech.fr>

Emmanouil Potetsianakis
Institut Mines-Télécom; Telecom ParisTech; CNRS LTCI
<emmanouil.potetsianakis@telecom-paristech.fr>

Abstract

Through the past few years, in-browser streaming of audiovisual content has become a commodity. Due to the diverse nature of possible audiovisual applications, there is often a need for accompanying descriptors and other metadata, such as semantic annotations, captions, etc. The metadata need to be sent in a timely fashion along with the multimedia content. The use of XML in such cases is common, and the usual approach for in-browser transmission of such data is via AJAX. Even though AJAX can be sufficient for many services, there is consideration for few offline or live scenarios. With MP4Box and MP4Box.js we are able to synchronously stream and consume XML and multimedia data, packaged in MP4 containers, with a standard browser. Accompanying XML documents can be transmitted as a whole, or progressively (in fragments). In this paper, we define the use-cases for this technology, analyze the requirements and present the mechanisms of MP4Box and MP4Box.js for XML end-to-end transmission inside the browser.

Keywords: XML, Streaming, Multimedia

1. Introduction

With the bloom of HTML5 and its <video> and <audio> tags, audiovisual (A/V) content transmission and presentation has become an essential functionality of the modern browser. In 2013, IP video traffic accounted for the 66% of the total internet traffic, and this number is expected to reach 79% by 2018, according to CISCO [1]. This growth affects XML technologies, since an important use of XML is for storing media information, subtitles, lyrics and other audiovisual content enhancements. The aforementioned complementary XML documents have to be transmitted and processed in-browser - synchronously with the main A/V content. However, such a technology, with support for both online (live and on-demand) and offline content, is yet to be widely spread.

Using the <video> element allows easy integration of A/V content, with common web technologies such as HTML, CSS and Javascript. Specifically in the case of Javascript, HTML5 defines an API consisting of Methods, Properties and Events, aiming at A/V content control. However, there are several ways to deliver the actual media to the client. As a result, the transmission of the accompanying XML documents and their proper handling on the receiving end becomes challenging.

In this paper we propose an end-to-end approach for streaming XML documents in the browser. The multimedia content along with the accompanying XML document(s) are packaged in a MP4 container. Then, they are transmitted synchronously, and the received content is made available by the browser to the web application, either live, or for offline consumption.

Synchronized In-Browser XML streaming can be applied to a plethora of use cases. From common scenarios such as subtitling and vector graphic enhancements, to more exotic, such as distributed Kinect-based applications for multimedia control [2] or digital puppetry [3].

In Section 2, we present some past and current trends in XML streaming. Followed in Section 3 by a description of our proposal for packaging XML in MP4 files. Then in Section 4 and Section 5, we detail the implementation of the packaging process with MP4Box, and the extraction mechanism in MP4Box.js, respectively. Finally in Section 6, we conclude this paper and reveal our current work on the topic.

2. Related Work

The Remote Events for XML [4] W3C Draft was produced with the purpose of DOM events transmission. The syntax defined could have been used for document streaming, besides the fact that it was focused on DOM tree enhancements. However, it did not include any notions of media synchronization or packaging. In the end, the draft was dropped due to legal implications.

Niedermeier et. al. developed a technique for compression and streaming of XML Data [6]. This schema-aware method is part of the MPEG-7 standard. XML documents are run through a binary encoding algorithm and the resulting fragments are transmitted. It also includes a text-encoding method. MPEG-7 streams can be stored in MP4 containers. But neither of the stream formats is deployed in browsers yet.

Apple HTTP Live Streaming (HLS) uses MPEG-2 Transport Streams (TS) for media packaging and ID3 Tags for metadata [5]. However, the metadata is not in an XML format, nor easily extractable within a browser. The WebVTT format also used by HLS for subtitles could facilitate timed XML data delivery, but WebVTT cannot be packaged in a single MPEG-2 TS file with the A/V content, making offline processing and content distribution difficult.

A web development paradigm that can be used for in-browser metadata is Asynchronous JavaScript and XML (AJAX). More specifically, in the Comet model of AJAX, a persistent connection is established between the server and the client. By using the open request made for the connection, it is up to the server to set the event to commence the data transfer (Long-Polling technique). When the server has available data, the client receives a Push Notification. Even though AJAX has some advantages comparing to the other alternatives, it can only be used for online services (not for offline). Our implementation on the other hand, is suitable for any consumption mode - even offline.

3. The Mechanism

As mentioned in Section 1, the XML documents are packaged with the accompanying media file(s) in an MP4 container (the process is detailed in Section 4). Then, the client browser is responsible for requesting the MP4 file from the server. The playback can be achieved via *Simple Streaming*, in which we have a Progressive Download of the file, and it is the use-case of plain `<video>` tag. Alternatively, we can have *Adaptive Streaming* by utilizing the extra functionalities provided by the use of Media Source Extensions (MSE).

In respect to these media delivery methods, there are two ways XML data can be streamed:

- *One XML document per time (or time range).* In this scenario, an entire XML document is timely delivered, for consumption within a time range of the audiovisual content. This method can introduce some latency with sizable files, since it requires the browser to fetch the whole XML document in order to use it, or it may cause some overhead if the XML documents are repetitive. However, it is simple and applicable to many use cases. An example usage is the carriage of [9] subtitles in MP4 files.
- *One XML Section (of a document) per time (or time range).* In this scenario, fragments of an XML document are coupled with a time range of the audiovisual content. Fragments are delivered progressively, removing the latency of the previous approach, but requiring a progressive consumption of the XML data. In particular, since the reception of the XML document is continuous and in fragments, we cannot have a balanced XML at a given time. To remove the overhead of the previous approach, common data is delivered upfront in a document header. Finally, seeking into the document stream can be more complex than with the previous scenario. It is only possible at positions in the document that require only the header information and nothing between the header and the current position. Such position is called a Random Access Point (RAP). The storage in MP4 permits indications on where the RAPs are located in the stream, thus -in conjunction with the header stored specifically- allowing seeking.

Both of the aforementioned delivery methods can be realized with our platform, which is composed of two parts, and detailed below:

- a server side packaging solution (MP4Box)
- a client side browser tool (MP4Box.js)

3.1. MP4Box

In order to provide a complete end-to-end XML streaming solution, MP4Box can be used for the packaging of the A/V and XML content in MP4 containers (on the server side). MP4Box is part of the GPAC multimedia framework [8].

An example of XML document streaming would be Timed Text Markup Language (TTML) subtitling [9]. In live scenarios, with real-time subtitle editing, several XML documents can be packed in the A/V stream. For such cases, Timed Text support was added to the ISO Base Media File Format (ISOBMFF) MPEG-4 Part 30.

MP4Box achieves any XML stream integration, in a similar with TTML manner, by using NHML descriptor files as follows:

```
MP4Box -add test_file.nhml:lang=en media_file.mp4
```

The NHML file details the integration of the XML documents in the MP4 track. For the command mentioned before, we can use a file as the one shown in Figure 1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<NHNTStream version="1.0" timeScale="1000" trackID="1"
            mediaType="meta" mediaSubType="metx"
            xml_namespace="http://example.namespace.org">
   <NHNTSample DTS="0" isRAP="yes" mediaFile="first.xml"/>
   <NHNTSample DTS="10000" isRAP="yes" mediaFile="second.xml"/>
   <NHNTSample DTS="20000" isRAP="yes" mediaFile="last.xml" duration="10000"/>
</NHNTStream>
```

Figure 1. Sample NHML file for packaging of multiple documents

With the above specifics, the metadata is inserted in an MP4 track with ID "1". This track will contain media of type 'meta' (i.e. metadata) and subtype 'metx' (i.e. in XML format). With this 'metx' configuration, we have one XML document per sample, as indicated in Figure 2, with here 3 samples made of 3 different documents. Other supported mediaTypes for XML packaging are shown in Table 1. The 'xml_namespace', is the namespace of the packaged XML document. The start time (measured in timeScale units) of each sample is is given by the DTS attribute, set to 0 for the first one. If multiple samples are used, for each sample, the DTS difference between two consecutive samples is used to compute its duration, but the last sample should have the duration attribute set. All samples are marked as RAP since they contain a self-contained XML document.

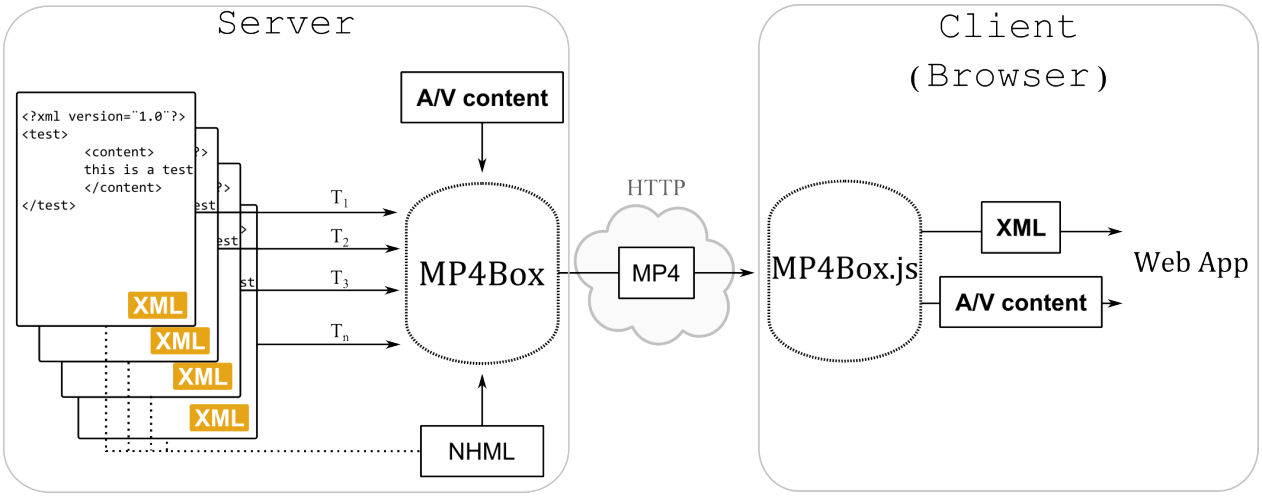


Figure 2. Streaming of multiple XML documents

Table 1. XML-suitable mediaTypes and mediaSubTypes

media-Type	mediaSub-Type	Definition	Usage
meta	metx	XML Metadata	One XML document per sample
	mett	Text (including XML) Metadata	XML documents and fragments
subt	stpp	XML Subtitle	One XML document per sample
	sbtt	Text (including XML) Subtitle	XML documents and fragments
text	stxt	Text (including XML) Stream	XML documents and fragments

NHML also considers the fragmented packaging of a XML document, for progressive consumption (Figure 3). The desired fragments are defined either in terms of document element tags (referenced by their "id" or "xml:id" attributes), or sample position and size. For the tag approach, the 'xmlFrom' field indicates the location of the first tag to copy from the document, while 'xmlTo' the last one. Alternatively, if we want to define the fragment in position and size, 'dataLength' defines its size, and 'mediaOffset' the position of the first byte. The 'isRAP' field is used to specify if a fragment is suitable for RAP. In order to achieve seeking with RAPs, the header of the document must be declared. Figure 4, shows an example use of NHML for progressive XML streaming, using XML elements (tags) selection.

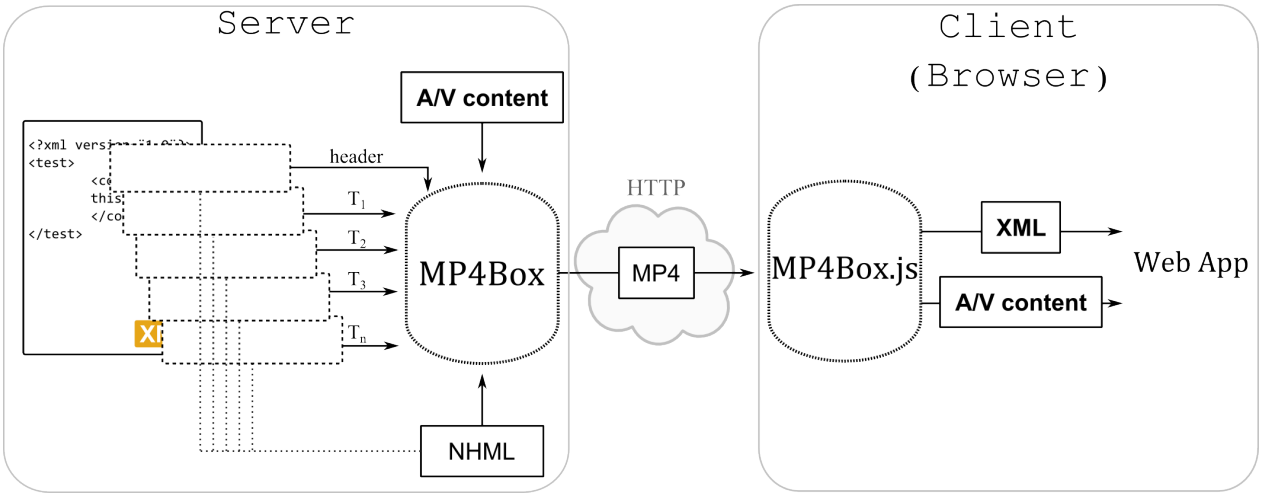


Figure 3. Progressive streaming of single XML document

```
<?xml version="1.0" encoding="UTF-8" ?>
<NHNTStream version="1.0" timeScale="1000" trackID="1"
            mediaType="meta" mediaSubType="mett"
            mediaFile="document.xml" headerEnd="elt1.start"/>
   <NHNTSample DTS="0"    isRAP="yes" xmlFrom="elt1.start" xmlTo="elt1.end"/>
   <NHNTSample DTS="1000" isRAP="no"  xmlFrom="elt1.end"   xmlTo="elt3.end"/>
   <NHNTSample DTS="2000" isRAP="yes" xmlFrom="elt3.end"   xmlTo="elt10.end"/>
</NHNTStream>
```

Figure 4. Sample NHML file for fragmented packaging

3.2. MP4Box.js

In order to achieve the proposed XML document transmission method, on the client side, we developed a tool that is able to extract the accompanying data from the MP4 container. Since the XML fetching happens inside the browser, *MP4Box.js* was developed in javascript. This way, it can be integrated in any web application.

MP4Box.js decouples the accompanying XML documents from the A/V stream and utilizes the `<track>` HTML element to synchronize (and possibly render) the data. The `<track>` element is used inside the `<video>` or `<audio>` tag, as a child element, and holds timed text data. The pre-defined data types that `<track>` can host are set in the "kind" attribute, which can take values of: subtitles, captions, descriptions, chapters or metadata. Metadata is a special case, since there is no predefined rendering and accompanying scripts can utilize the data as desired, by using cue events for synchronization.

An example usage of MP4Box.js setup in order to parse XML samples is shown in Figure 5. An MP4Box instance is created and with `setExtractionOptions`, its parameters are the track "id" and the "user" parameter for the `onSample` callback - typically a `TextTrack` object. The "options" parameter is used to define if the sample

array should start with a RAP sample and the total number of samples to receive for the callback. In turn, `onSample` returns an Array of samples, for track with "id", when called by "user".

```
mp4box = new MP4Box();
mp4box.setExtractionOptions(id, aTextTrack, options);
mp4box.onSamples = function (id, user, samples) {
 var sample;
 var parser;
    for(var i in samples) {
      sample = samples[i];
      if(samples.description.type === "metx") {
        parser = new XMLSubtitlein4Parser();
        var sampleDocument = parser.parseSample(sample).document;
         user.addCue(transformDocToCue(sample, sampleDocument));
      }
      else if(sample. description.type == "mett") {
        parser = new Textin4Parser();
         var sampleText = parser.parseString(sample);
         user.addCue(transformTextToCue(sample, sampleText));
      }
   }
}
```

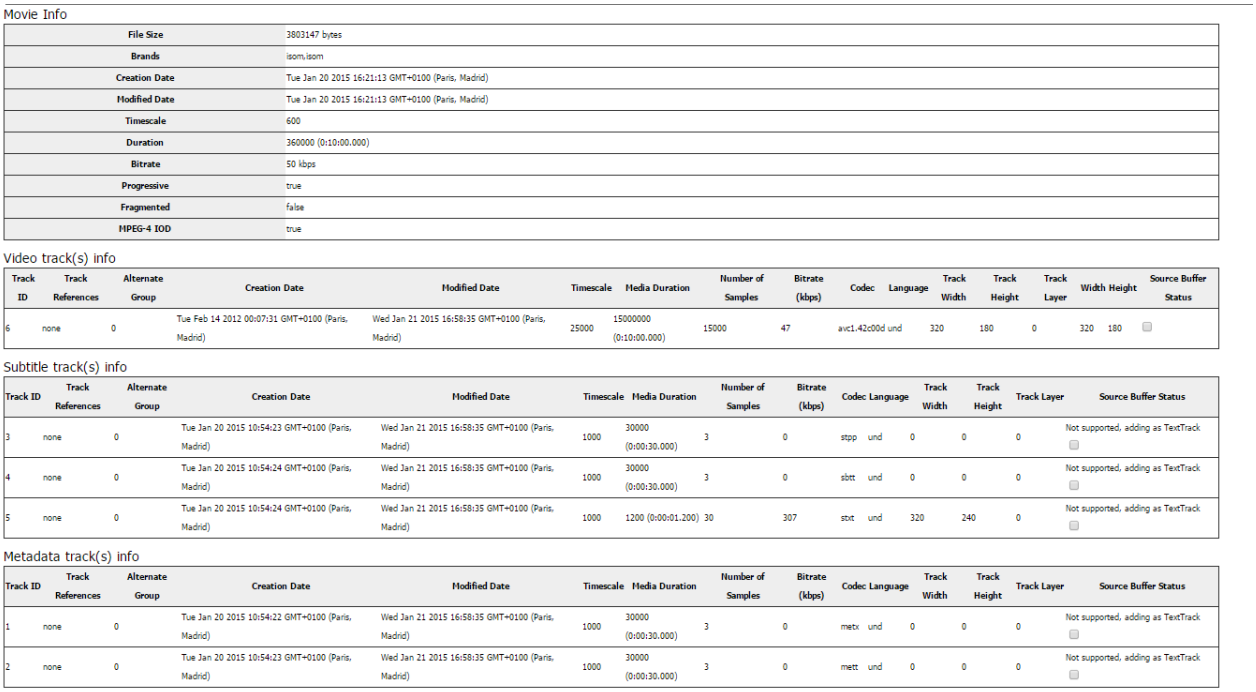
Figure 5. MP4Box.js code extract

Each sample that is extracted from the stream contains the XML data and the packaging information (timestamps, isRAP, etc) - set with MP4Box via the NHML descriptors. The actual XML data is stored in the field "data" of the sample, as an ArrayBuffer. Figure 6 shows the extracted first sample (XML document), as defined in Figure 1.

```
{
    "track_id":1,
    "description": [Box],
    "is_rap":true,
    "timescale":1000,
    "dts":0,
    "cts":0,
    "duration":1000,
    "size":41,
    "data": [ArrayBuffer]
}
```

Figure 6. JSON representation of a sample

A screenshot of a webpage containing information on the available tracks of a mp4 file is in Figure 7. MP4Box.js is used to extract the details of one video track and five tracks with XML data - one for each mediaSubType. More information on the usage and features of MP4Box.js can be found at the github repository of GPAC[1].



Movie Info

File Size	3803147 bytes
Brands	isom,isom
Creation Date	Tue Jan 20 2015 16:21:13 GMT+0100 (Paris, Madrid)
Modified Date	Tue Jan 20 2015 16:21:13 GMT+0100 (Paris, Madrid)
Timescale	600
Duration	360000 (0:10:00.000)
Bitrate	50 kbps
Progressive	true
Fragmented	false
MPEG-4 IOD	true

Video track(s) info

Track ID	Track References	Alternate Group	Creation Date	Modified Date	Timescale	Media Duration	Number of Samples	Bitrate (kbps)	Codec	Language	Track Width	Track Height	Track Layer	Width	Height	Source Buffer Status
6	none	0	Tue Feb 14 2012 00:07:31 GMT+0100 (Paris, Madrid)	Wed Jan 21 2015 16:58:35 GMT+0100 (Paris, Madrid)	25000	15000000 (0:10:00.000)	15000	47	avc1.42c00d	und	320	180	0	320	180	

Subtitle track(s) info

Track ID	Track References	Alternate Group	Creation Date	Modified Date	Timescale	Media Duration	Number of Samples	Bitrate (kbps)	Codec	Language	Track Width	Track Height	Track Layer	Source Buffer Status
3	none	0	Tue Jan 20 2015 10:54:23 GMT+0100 (Paris, Madrid)	Wed Jan 21 2015 16:58:35 GMT+0100 (Paris, Madrid)	1000	30000 (0:00:30.000)	3	0	stpp	und	0	0	0	Not supported, adding as TextTrack
4	none	0	Tue Jan 20 2015 10:54:24 GMT+0100 (Paris, Madrid)	Wed Jan 21 2015 16:58:35 GMT+0100 (Paris, Madrid)	1000	30000 (0:00:30.000)	3	0	sbtt	und	0	0	0	Not supported, adding as TextTrack
5	none	0	Tue Jan 20 2015 10:54:24 GMT+0100 (Paris, Madrid)	Wed Jan 21 2015 16:58:35 GMT+0100 (Paris, Madrid)	1000	1200 (0:00:01.200)	30	307	stxt	und	320	240	0	Not supported, adding as TextTrack

Metadata track(s) info

Track ID	Track References	Alternate Group	Creation Date	Modified Date	Timescale	Media Duration	Number of Samples	Bitrate (kbps)	Codec	Language	Track Width	Track Height	Track Layer	Source Buffer Status
1	none	0	Tue Jan 20 2015 10:54:22 GMT+0100 (Paris, Madrid)	Wed Jan 21 2015 16:58:35 GMT+0100 (Paris, Madrid)	1000	30000 (0:00:30.000)	3	0	metx	und	0	0	0	Not supported, adding as TextTrack
2	none	0	Tue Jan 20 2015 10:54:23 GMT+0100 (Paris, Madrid)	Wed Jan 21 2015 16:58:35 GMT+0100 (Paris, Madrid)	1000	30000 (0:00:30.000)	3	0	mett	und	0	0	0	Not supported, adding as TextTrack

Figure 7. Stream info output of MP4Box.js

4. Conclusion and Future Work

In this paper we have presented the mechanisms of MP4Box and MP4Box.js. An end-to-end solution for XML document streaming inside the browser. We have explained the process of packaging in mp4 containers on the distribution side, as well as the XML data extraction mechanism for the client.

In the future we plan on adding support for specific XML senarios, such as SVG. SVG files can be currently transmitted as a whole inside an MP4 file. However, there is consideration for progressive streaming of SVG files [10]. In that case, a fragmented file can be progressively transported and rendered.

[1] https://github.com/gpac/mp4box.js/

Bibliography

[1] Cisco Visual Networking Index: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018. 2014. http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[2] Potetsianakis, E.; Ksylakis, E.; Triantafyllidis, G.: A Kinect-based framework for better user experience in real-time audiovisual content manipulation. Telecommunications and Multimedia (TEMU), 2014 International Conference on , vol., no., pp.238,242, 28-30 July 2014. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6917767&isnumber=6917722

[3] Alberto Pacheco: Digital Puppet Ver. 0.4. 2014. http://podcast.itch.edu.mx/marioneta/index.v4.waves.html

[4] Berjon, Robin: Remote Events for XML (REX) 1.0. Working Draft, 2006, W3C. http://www.w3.org/TR/rex/

[5] R. Pantos: HTTP Live Streaming. Internet Draft, 2014, IETF. http://tools.ietf.org/html/draft-pantos-http-live-streaming-14/

[6] Niedermeier, U., Heuer, J., Hutter, A., Stechele, W., Kaup, A.: An MPEG-7 tool for compression and streaming of XML data. In Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on (Vol. 1, pp. 521-524). IEEE.

[7] Colwell Aaron, et. al.: Media Source Extensions. Candidate Recommendation, 17 July 2014, W3C http://www.w3.org/TR/2014/CR-media-source-20140717

[8] Le Feuvre, Jean and Concolato, Cyril and Moissinac, Jean-Claude: GPAC: Open Source Multimedia Framework. Proceedings of the 15th International Conference on Multimedia. MULTIMEDIA '07. New York, NY, USA: ACM, 2007, pp. 1009–1012 http://doi.acm.org/10.1145/1291233.1291452

[9] Adams, Glenn: Timed Text Markup Language 1. Recommendation, 24 September 2013, W3C. http://www.w3.org/TR/ttaf1-dfxp/

[10] Concolato, Cyril: SVG Streaming. Editor's Draft, 04 June 2013, W3C. http://www.w3.org/SVG/modules/streaming/