An HTML5 Interactive (MPEG-A IM AF) Music Player

Giacomo Herrero[1], Panos Kudumakis[2] *, Lorenzo J. Tardón[1], Isabel Barbancho[1] and Mark Sandler[2]

[1] Universidad de Málaga, Málaga, Spain
[2] Queen Mary University of London, London, UK
giacomoherrero@gmail.com
{ibp,lorenzo}@ic.uma.es
{panos.kudumakis,mark.sandler}@eecs.qmul.ac.uk

Abstract. In this paper, the approach and implementation towards a cross-platform and cross-browser interactive music player, inspired by the MPEG-A: Interactive Music Application Format (IM AF) standard, are discussed. The approach included requirements gathering as well as a pros and cons analysis of alternative technologies considered. This is followed by the description of the IM AF player implementation using HTML5 and the related APIs used, as well as a discussion on its features and conformance to standard.

Keywords: MPEG, IM AF, Interactive Music, standards, online player.

1 Introduction

The music industry is nowadays going through a transformation. With the advent of new paradigms of recording, producing and sharing music over the Internet, an advanced audio file format, that enables users to interactively enjoy music as it best suits them, is needed.

The MPEG-A: Interactive Music Application Format (IM AF) [1] provides the users a way to mix, tweak and rearrange the different musical instruments tracks of which a song is composed, allowing them for example to only listen to the vocals (*a cappella*), or mute them (*karaoke*) [2]. Such interactivity is of outmost significance especially in relation to nowadays sharing services and social networks, where users are able to upload their own content, which instantly becomes available for other users to download, stream, and even propose their own mix and re-upload the songs, creating this way a powerful user community. In this scenario, a cross-platform and cross-browser player able to decode and play online IM AF files becomes a necessity. Therefore, in this paper the different approaches and technologies available for building such a cross-platform and cross-browser IM AF player are discussed alongside its prototype HTML5 implementation.

* Panos Kudumakis acknowledges that this work has been done during his visit at the University of Malaga in the context of the program Andalucía TECH: Campus of International Excellence.

In particular, Section 2 presents the steps considered in the design and implementation of the HTML5 IM AF player, while Section 3 discusses the results obtained and in Section 4 some conclusions are drawn.

1.1 The IM AF Standard

The IM AF standard structures the contents of a song with multiple unmixed tracks/instruments into an ISO-Base Media File Format container [3] (stored with a `.ima` file extension), along with synchronised lyrics, album artwork and a set of rules that limit the users' mixing options to a range allowed by the creator.

This container is therefore comprised of different media data formats, such as MP3, PCM or AAC for audio data, JPEG for still images and 3GPP Timed Text for synchronised lyrics [2].

The standard, besides the use of preset modes, which are predefined mixes established by the content creators, such as the *a cappella* and *karaoke* versions mentioned above, allows the grouping of different musical instruments, thereby rules could be easily applied to groups of instruments rather than individual ones, i.e., changing volume of all string instruments rather than to guitar and bass, separately.

2 Design and Implementation

2.1 Requirements

An online IM AF player, apart from the requirements for compliance to the IM AF standard, needs to also satisfy additional requirements with respect to its online implementation. These requirements are summarised in the following:

- **Reading `.ima` files.** The player shall be able to read `.ima` files and perform the basic functionalities of the format, such as change the volume of each musical instrument, display the associated pictures and show the lyrics synchronously.
- **Cross-platform.** The player should be cross-platform and cross-browser, of course within possible limits, in order to simplify the further development of the service.
- **Sustainability.** The application shall be sustainable, which means that every service and technology used, as well as the target platforms for the application, should be available in the near future.
- **Adaptability.** The player shall be able to support any device independently of their processing power capabilities, screen resolution or size. Allowing if necessary only a limited set of operations, e.g., streaming a limited number of tracks or simply a preset track.
- **Server connectivity.** The player shall be able to contact a specific server from which to pull the contents at the client's request, the contents being either the `.ima` file itself or audio tracks, pictures, lyrics and metadata individually.

2.2 Considered Options

This section discusses the alternative approaches considered for the IM AF player implementation, detailing their advantages and disadvantages, and specifying the reasons as to why they were not eventually adopted. Table 1 summarises these approaches with respect to the requirements mentioned in the previous section.

1. **Plug-in.** The first approach considered was the development of a browser plug-in, either starting from scratch or making use of the standalone reference software IM AF player [1] as a starting point. However, this approach would only be feasible for desktop browsers, that allow the user to download a third-party plug-in. For mobile devices this approach would be either quite complex (Android) or simply not possible (iOS) [4]. In such case, a separate standalone application for each platform would be needed.
2. **Adobe Flash.** The next alternative approach considered was the implementation of the IM AF player as a Flash application. This approach would have partially solved the problem of requiring the user to install external plug-ins, since most users have already Adobe Flash in their browsers.
   Adobe Flash provides a powerful framework for audio processing and most of the online tools regarding audio used to be implemented in Flash ActionScript. However, Adobe Flash is being lately substituted by more interoperable approaches, and nowadays an increasingly number of platforms are adopting JavaScript and HTML5 instead of Adobe Flash, iOS being one of them. Thus, Adobe Flash does not satisfy neither the sustainability requirement, nor the cross-platform requirement, since as with the previous plug-in approach, a standalone application for iOS would have had still be needed to be implemented.

	`.ima` files	Cross-platform	Sustainability	Adaptability	Server connectivity
Plug-in	✓	×	✓	✓	✓
Adobe Flash	✓	×	×	✓	✓
HTML5	✓	✓	✓	✓	✓

Table 1: Requirements fulfillment versus alternative technologies considered towards the IM AF prototype implementation.

2.3 HTML5 Approach

With respect to the requirements mentioned in Section 2.1 and the aforementioned discussion on the alternative options considered for a cross-platform and cross-browser IM AF player implementation, eventually the HTML5 approach was adopted. This approach with its pros and cons is discussed in the following.

HTML5 (and its complementing CSS3 [5] and JavaScript) is a relatively recently standardised set of technologies that has seen an enormous increase in popularity in the last years and, although some services are still reluctant to

adopt it, more and more platforms are switching to HTML5 support rather than Flash.

The disadvantages of the HTML5 technology are:

(a) The availability of Application Programming Interfaces (APIs) for each platform, depending on the browser manufacturer's good will.
(b) The processing power available to applications depends on the browser rendering engine.

The advantages of the HTML5 technology are:

(a) As opposed to the other approaches considered earlier, HTML5 is a newly created technology, which is an indication that is highly likely to be continued to be supported by browsers for some time in the near future. As a result, it satisfies the sustainability requirement.
(b) Possibility to be used for both front-end and back-end development, with the help of other technologies such as AJAX, JSON, etc. The connectivity with servers is extremely straight-forward.
(c) Availability of several very powerful standard APIs focused on audio processing and file management, such as Web Audio API [6] and File API [7], respectively. Web Audio API is in fact the framework on which this IM AF prototype player mostly relies. This is because the Web Audio API includes the audio engine necessary to mix, extract data, apply effects, etc. to the basic audio file formats (MP3, PCM, AAC, etc.).

Table 2 summarises the availability of these APIs used for the HTML5 IM AF prototype player implementation on the most commonly used browsers. Consequentially, it can be noted that the HTML 5 IM AF player is currently supported on Chrome and Safari on OSX, Chrome for Windows and Safari on iOS. Support of the player on Chrome on Android and Firefox is also expected in the near future.

These HTML5 cross-browser capabilities allow for an extremely easy adaptation of the player to different platforms, with only a small number of tweaks in the CSS3 style files.

	HTML5	Web Audio API	File API
Chrome	Supported	Supported	Supported
Safari	Supported	Supported	Supported
Firefox	Supported	Near Future	Supported
Safari iOS	Supported	Supported	Supported
IE	Supported	Not Supported	Supported
Android Browser	Supported	Not Supported	Partial Support
Chrome Android	Supported	Not Supported	Supported

Table 2: Availability of the technologies (Web Audio API and File API) used for the HTML5 IM AF prototype player implementation on the most commonly used nowadays browsers.

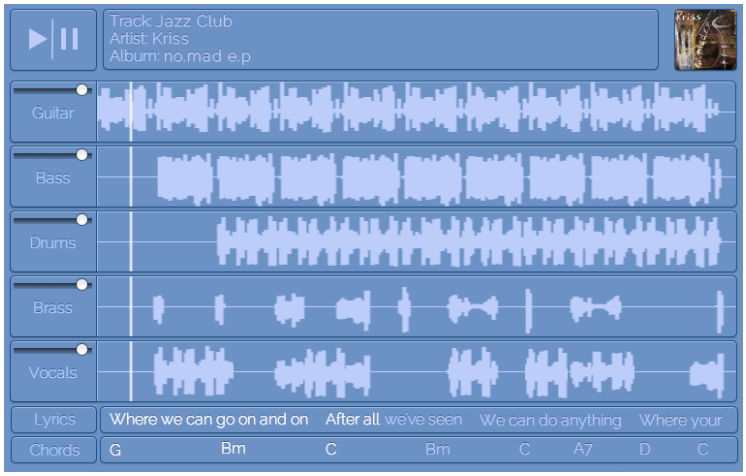


Fig. 1: Graphical User Interface of the HTML5 IM AF online player prototype.

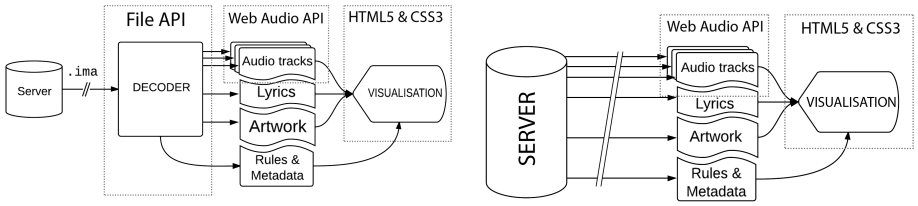
Figure 1 shows the graphical user interface of the online HTML5 IM AF player. The waveforms of each musical instrument track are plotted alongside their volume sliders enabling easy tracks' mixing by the user. At the top right corner a thumbnail picture related the album's artwork is shown; the synchronised lyrics and chords are also shown at the bottom.

2.4 Client- or Server-side Decoding

Regarding the decoding of the IM AF container, two possibilities have been considered, each one with its advantages and disadvantages: a) processing and parsing the file directly in the users' browser (client-side approach) or b) parsing the file on a server and sending to the client the individual audio, text, and metadata files (server-side approach).

In the client-side approach some of the available processing resources at the client is consumed in parsing and decoding the IM AF file. This is a drawback for the mobile devices case since may limit the number of musical instrument tracks able to be decoded at the same time. This approach is shown in Figure 2a.

Performing the IM AF file decoding on the server side, as is shown in Figure 2b, solves the possible limited processing power problem, i.e., on mobile devices. That is, whenever the application requests a song, the server parses the IM AF file and sends the resulting audio, text and metadata files to the client.



(a) (b)

Fig. 2: (a) Client-side and (b) server-side IM AF decoding diagrams with their respective technologies used.

2.5 APIs

- **Web Audio API.** This technology is used in the prototype IM AF implementation presented in this paper for handling all audio events. This is a high-level JavaScript API [6], which aims to enhance web browsers with some of the capabilities that can be found in modern recording and producing desktop applications, as well as in modern game audio engines. Up until recently, audio on the web has been fairly overlooked, relying primarily on external plug-ins as Flash or QuickTime. Even though the `audio` element was recently introduced in HTML5, it is not powerful enough when it comes to heavy signal processing and mixing.
  The Web Audio API is based on a routing graph model, where audio nodes (`AudioNode` interface) are connected together in a modular fashion, mimicking traditional analog ways of handling music processing, with the intention of simplifying its use for music-oriented application developers [6].
  The Web Audio API allows, therefore, to handle several audio formats, depending on the browser [8], such as MP3, OGG and PCM, and perform to a great extent operations such as: extracting sample raw data, buffering, mixing signals, filtering, compression, delays, etc.
  As can be seen in Figure 3, the primary interface of this API is the `AudioContext` interface, which contains all conections between `AudioNode`s. Such nodes can be input nodes, that pull the files upon a server request (see XMLHttpRequest [9]), as well as processing nodes and output nodes, that render the processed sound.

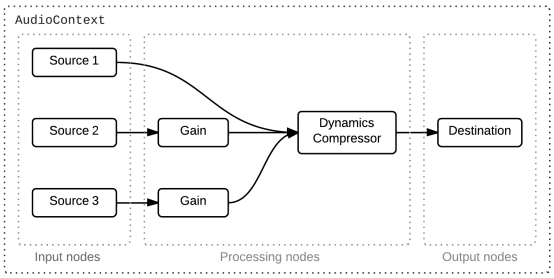


Fig. 3: Example diagram of the Web Audio API routing system.

- **File API.** The File API [7] is the framework used to handle all binary data. It represents a uniform way to access local and online data in HTML5. It consists of a series of interfaces that perform various tasks. Amongst the more important are: `FileList`, which represents an array of the individual files selected, and `Blob`, which represents the raw binary data that can be read through the `FileReader` interface allowing access to a range of bytes. The File API is designed to be also used in conjuction with XMLHttpRequest to pull file requests from servers.

3 Results and Discussion

The resulting HTML5 IM AF player implementation consists therefore of the following basic functionalities: a) requesting the tracks from a server (server-side approach), b) plotting their waveforms alongside the volume sliders for easy mixing by the user, c) displaying the synchronised lyrics when available and d) the picture thumbnail related to the album's cover. A working demonstration can be found at www.giacomoherrero.com/clientside and the code is available at https://code.soundsoftware.ac.uk/projects/imafplayer

A number of tests have been performed to find out the number of tracks that the application is able to decode and play at the same time. As can be seen in Table 3, both desktop browsers supported (Chrome and Safari) allow up to 16 simultaneously decoded tracks. Under Safari for iOS, the maximum number of tracks is limited to 8 (only in the server-side mode).

It comes as no surprise that these results are in fully agreement to the maximum number of simultaneously decoded tracks specified in the IM AF standard [2], where for applications on mobile devices (brands `im01` to `im04`) is up to 8 tracks, and for desktop (`im11`) and high-end processing devices (`im21`) 16 and 32 tracks, respectively .

For the desktop version, a client-side decoding system has been implemented, being able to simultaneosly decode up to 14 tracks (brand `im11`), although the speed is considerably reduced.

With regards of future work, we intend to further develop the IM AF player, refining its capabilities and implementing an improved version of the client-side approach for mobile devices.

It is also our goal that the player, eventually, would be integrated into an online file sharing service, allowing the users to stream, download and share IM AF files creating a powerful user community.

	2 tracks	4 tracks	6 tracks	8 tracks	16 tracks	32 tracks
Chrome	✓	✓	✓	✓	✓	×
Safari	✓	✓	✓	✓	✓	×
Safari iOS	✓	✓	✓	✓	×	×

Table 3: Number of simultaneously decoded audio tracks supported by each browser. Tests performed on MacBook 2.4 GHz Intel Core 2 Duo with 2 GB RAM and OS X 10.8.4.

4 Conclusions

In this paper it is proposed and described a cross-platform and cross-browser prototype implementation of an online HTML5 IM AF player. Alternative approaches such as: a) the implementation of a downloadable browser plug-in and, b) an Adobe Flash plug-in have been considered before the HTML5 approach adopted.

By doing so, we have introduced the basic concepts of the MPEG-A: Interactive Music Application Format standard, highlighting some of its features.

The working HTML5 IM AF prototype player performs seamlessly in some of the most used operating systems and browsers, including Safari 6.0 for both iOS and OSX, and Chrome for both OSX and Windows.

References

1. ISO/IEC 23000-12:2010, *Information technology - Multimedia application format (MPEG-A) – Part 12: Interactive Music Application Format.*
2. I. Jang, P. Kudumakis, M. Sandler, and K. Kang, "The MPEG Interactive Music Application Format Standard," *IEEE Signal Processing Magazine*, vol. 28, pp. 150–154, Jan. 2011.
3. ISO/IEC 14496-12:2008, *International Standard. Information Technology - Coding of audio-visual objects. Part 12: ISO Base Media File Format.*
4. Apple, Inc., *"Safari Web Content Guide"*, Sep. 2012.
5. W3C, ""What is CSS?" – `http://www.w3.org/Style/CSS/`, last accessed: Jun., 2013."
6. W3C, ""Web Audio API", W3C Working Draft, Dec., 2012, Editor: Chris Rogers – `http://www.w3.org/TR/webaudio/`."
7. Mozilla, ""File API" – `http://www.w3.org/TR/FileAPI/`."
8. Mozilla, ""Media formats supported by the HTML audio and video elements" – `https://developer.mozilla.org/en-US/docs/HTML/Supported_media_formats`."
9. W3C, ""XMLHttpRequest", W3C, Working Draft. Dec., 2012. Editors: Julian Aubourg, Jungkee Song and Hallvord R.M. Steen – `http://www.w3.org/TR/XMLHttpRequest/`."