iWebDJ Online DJ Mixer

technical information **(rev. P / iWebDJ4.9X)**

This document contains all technical information related to the iWebDJ technologies.

Summary :

Part 1 : iWebDJ Mixer overview

You can integrate the iWebDJ mixer on your website to allow your users to mix your music !

This allows you to showcase your music, create a marketing campaign, create online DJ contests, create an online DJ service, create a DJ radio…

The iWebDJ mixer can easily be customized and rebranded. We can do the customization for you or you can customize it by yourself, the source code is given.

To install the iWebDJ mixer, you only need to copy a set of files on your server (see chapter 2).

All features of the iWebDJ mixer

Mixing / DJ features	Other features
- double-player with crossfader	- versatile API
- beatsync (simply the best in the market !)	- multi-threaded (super robust performance, no audio cut)
- automix technology (amazing song transitions)	- topnotch loudness normalization
- playback speed control with keylock (tone not altered !)	(adjust the perceived volume based on a human-ear model)
- mix recording / live-broadcast / sharing	- music anti-theft technology (optional)
- realistic vinyl scratch engine	- advanced music loading module
- seamless loop and seek (beatjump)	- basic playback function (play, seek, volume…)
- realtime audio FX	- advanced buffering / seeking management
- multiple cue points	- pro-class 3 bands equalizer (+low/high pass filter)
- sampler	- special filter for bass enhancement
- external midi decks (beta)	- amazingly crisp waveform
- headphone cueing (with Y-mono-splitter)	- beat grid editor (optional)

iWebDJ mixer API

The iWebDJ API allows to establish a bidirectional communication with the iWebDJ mixer. You can do pretty much what you want with the API such as controlling the playback or load a song. By default, the API is accessible via Javascript and AS3. The API can also be extended according to your needs.

iWebDJ mixer GUI

The graphical interface allows to interact with the mixer. It contains graphical elements and code. We provide a generic GUI which can be totally customized to match your brand. The source code of the GUI is provided, you can then modify it as much as you want. You can also develop your own GUI from scratch, or connect to the GUI of your existing app via the iWebDJ API.

HTTP(s) vs RTMP

All iWebDJ technologies rely on HTTP(s) because it is far superior to RTMP for audio streaming.

5 advantages of HTTP(s) over RTMP

1. the audio data can be manipulated (equalizer, automix, dj features…)
2. songs are bufferized in the browser cache (robust to brief connection loss over wifi, 3/4G…)
3. cheaper infrastructure on the server side (no rtmp servers, less servers)
4. CDN friendly, especially if the files are delivered into several small pieces
5. a RTMP stream is super easy to rip (with rtmpdump) and this can't be fixed

Compatible platforms

The iWebDJ mixer is compatible with :

- Win/Mac/Linux all browsers (with flash)
- Android devices (with flash)
- Win8/10/RT devices (with flash)

Part 2 : iWebDJ Mixer on your website

To use the iWebDJ mixer on your website, you just need a package containing all files needed (with source).

You will get exactly the iWebDJ demo page (http://iwebdj.com/demo) on your website but with your music.

The iWebDJ package

The package contains all needed files and source to install the iWebDJ mixer on your website, more precisely it contains :

- the mixer user interface (with all Flash/AS3 source)
- the music browser (with all HTML5, CSS, Javascript source)
- the server side script for recording / broadcasting mixes (with all PHP source)
- the iwebdj music analyzer tool to add your music
- all technical instructions (this document)

Customization of the mixer

The source code is given, you are free to customize, rebrand (= change the logo), modify the source and the graphical interface of the mixer.

If you are not able to customize the mixer or if you decide it afterwards, just contact us. What we can do for you :

- customize the graphical user interface (gui)
- remove the iWebDJ logo and put yours (rebranding)
- give us a picture, we will design your player from it
- add new features or behaviors
- adapt the API for a better integration on your system

Installation of the iWebDJ package

The installation of the iWebDJ package is very easy, simply copy all the files on your web server, that's it!

The **index.php** is the landing page of the DJ mixer.

The music browser displays all files which are stored in the default music directory **./music**. This folder can be modified to anything else in the file **playlist_list.php** (line 5). Again you are free to modify the source as much as you want.

You can also develop your own music browser. To load a song in the DJ mixer, simply us the loadSong Javascript command of the iWebDJ API by providing the absolute url of the mp3 file (see chapter 3 - iWebDJ API - Sending commands).

To analyze your music files, if you are on Windows, simply drag and drop one or multiples files or browsers on the MUSIC ANALYZER/iwebdj_analyzer_win32.exe in the Windows Explorer. If you are on Linux (on your webserver for instance), use the command line syntax (see part 6). The final MP3 with the special iWebDJ tag are outputted in the ./output folder.

The DJ mixer interface is the file **./iwebdj_files/iwebdj_mixer.swf**, if you want to modify anything (change the graphical layout for instance) please open the source **iwebdj_mixer.fla** in Flash Professional CS5.5 or above or any compatible. This software is available with a 30 days free demo on the Adobe website (http://www.adobe.com/products/flash.html). Other options can be can modified in the **iWebDjMixer.as** files.

If you want to use the recording feature, you will have to set up a MongoDB (http://www.mongodb.org/) on your server because the DJ mixer will have to save the users' mixes on your server. Once installed, you will also need to put your MongoDB login information in the file **./iwebdj_files/iwebdj_api_record.php**. You are free to use another DB system than MongoDB but you will have to modify the source code by yourself.

Part 3 : iWebDJ API - Sending commands

We provide a versatile API for sending commands to iWebDJ.
You can access the API in Javascript or in ActionScript3 depending of your integration.

List of commands :

Function JS / AS3	`loadSong` Allow to load a MP3 with a URL.
Function JS / AS3	`sendCommand` Allow to send playback or configuration commands to iWebDJ.
Function JS / AS3	`loadSample` Allow to load / play / stop external MP3s in the sampler.

loadSong

Allow to load a MP3 into the iWebDJ mixer with a URL.

```
loadSong(_playerName, _absoluteUrl, _urlPostData, _infoGui, _bpmRangeMax, _beatOffset)
```

- `_playerName:String` can be "player1" or "player2".
- `_absoluteUrl:String` is the absolute path of the MP3 file (relative paths give an error).
- `_urlPostData:String -` some POST data to embed into the HTTP request. If not used, this value should be set to "".
- `_infoGui:String` is the string which will be passed to the GUI. Depending of the GUI implementation, this could be the artist, title or artwork picture url. Important : this string must be unique for each song.
- `_bpmRangeMax:Number` is the upper limit of the bpm range for this song. By default it is good to set _bpmRangeMax to 145, the bpm range for the song will be 72.5-145 bpm. There is also a special mode optimized for dubstep, to enable it, set the value to 333.
- `beatOffset:Number` - a beat offset correction to apply for this song. This value is given by the beatgrid editor. By default, this value must be egal to 0.
- `intro:Number` (optional) - set the intro cue point manually for this song (in milliseconds). The intro cue point is used by the automix algorithm. During a transition from song A to song B, the intro cue point of the song B will be the end of the transition. If not given or set to 0, the algorithm will detect the intro automatically (default).

- `outro:Number` (optional) - set the outro cue point manually for this song (in milliseconds). The outro cue point is used by the automix algorithm. During a transition from song A to song B, the outro cue point of the song A will be the beginning of the transition. If not given or set to 0, the algorithm will detect the outro automatically (default).

Examples :

Loading a MP3 via absolute url.

```
iWebDjPlayer.loadSong("player1", "http://you.com/song.mp3", "", "artist#title", 145, 0);
```

sendCommand

Allow to send playback or configuration commands to iWebDJ. You are free to add commands.

`sendCommand(_key:String, _value:Number)`

- `_key:String` : the command name
- `_value:Number` : the command number value

List of commands :

command name	value	description
"automix"	0 (off) or 1 (on)	Turn on or off the automix
"automix_next"	-	The automix will request a new song and will crossfade it quickly. Use this command to load a song on the automix player (the compact-sized player). The event LOAD_SONG_REQUEST will trigger.
"automix_mode"	1,2 or 3	You can select the automix behaviour at any time allowing more complex algorithm in charge of selecting the songs to automix. Mode 1 (default) : long transition (up to one minute) + beatsync + reset pitch at the end / Mode 2 : long transition (up to one minute) + beatsync + NO reset pitch at the end / Mode 3 : short transition (3-5 seconds) + NO beatsync
"unload"	1 or 2	Unload player1 or player2
"main_volume"	0 to 200 %	Adjust the iWebDJ main volume. Be careful for value higher than 100%.
"keylock"	0 (off) or 1 (on)	Turn on or off the keylock algorithm in iWebDJ. The keylock allows to change the playback speed without altering the tone of the song.
"headphone"	0 (off) or 1 (on)	Turn on or off the headphone cueing.

"midi"	0 (off) or 1 (on)	Turn on the midi (beta)
"latency"	2048 to 8192	Adjust the iWebDJ buffer size which is directly linked to the latency and the audio reliability. 2048 value is reactive but less robust. 4096 is a good balance (default). 8192 will give extra robust audio.
"playerX_playstop"	0 (stop) or 1 (play)	Play or stop playerX (X is 1 or 2) To stop everything, call both player1_playstop and player2_playstop with 0 as parameter.
"playerX_seek"	seek position (ms)	Seek playerX (X is 1 or 2)
"radio_playpause"	not used	Automix player - play / pause
"radio_seek"	seek position (%)	Automix player - seek position (value is 0 to 100)
…	…	any playback command can be sent…

Examples :

Loading a MP3 via absolute url.

```
iWebDjPlayer.sendCommand ("main_volume", 50);
```

loadSample

Allow to load / play / stop external MP3s in the sampler.

```
loadSample(_sampleUrl:String)
```

To play a sample, just call loadSample with your sample MP3 file (absolute url).

If it the first time this sample is used, it will be first loaded by the iWebDJ mixer.

To stop the sample, just loadSample again while the sample is playing.

Examples :

Loading and playing a sample :

```
iWebDjPlayer.loadSample("http://you.com/sample.mp3");
```

Part 4 : iWebDJ API - Retrieving info

We provide a versatile API for retrieving information from iWebDJ.
You can access the API in Javascript or in ActionScript3 depending of your integration.

List of events :

Event JS / AS3	`IWEBDJ_READY` Event triggered when iWebDJ is up and running after loading.
Event JS / AS3	`LOAD_SONG_REQUEST` Event triggered when iWebDJ requests for a new song to load (automix mode).
Event JS / AS3	`GET_INFORMATION` Allow to retrieve playback states or various info from iWebDJ.

IWEBDJ_READY

Event triggered when iWebDJ is up and running after loading. All commands sent before the ready event are ignored.

```
IWEBDJ_READY(_iWebDjVersion)
```

- `_iWebDjVersion:String` : the current iWebDJ audio engine version

LOAD_SONG_REQUEST

Event triggered when iWebDJ requests for a new song to load (automix mode). When the event is called, you must load the new song as soon as possible (< 1 second).

```
LOAD_SONG_REQUEST(_askingPlayerName)
```

- `_askingPlayerName:String` : egal to “player1” or “player2“.

Example of application in Javascript:

```
function LOAD_SONG_REQUEST(_askingPlayerName)
{
   iWebDjPlayer.loadSong(_askingPlayerName, … );
}
```

GET_INFORMATION

Allow to retrieve playback states or various info from iWebDJ. Absolutely any kind of info can be retrieve through the API. Depends of the project, not all events are triggered.

```
GET_INFORMATION(_key:String, _value:String)
```

- `_key:String` : the information name
- `_value:String` : the information string value

List of information :

information name	value	description
"iwebdj_error”	-	Event triggered on engine error
"load_song_start”	song url	Event triggered when iWebDJ load a song
"automix_enable”	-	Event triggered when automix is on
"automix_disable”	-	Event triggered when automix is off
"player_play”	player id (1 or 2)	Event triggered when the player play
"player_stop"	player id (1 or 2)	Event triggered when the player stop (pause)
"player_end"	player id (1 or 2)	Event triggered when the player reach the end of the song.
"beatmatch_yes"	-	Event triggered when both players are synced
"beatmatch_no"	-	Event triggered when both players are not synced
"beat_editor"	correction values	Event triggered when new correction values has been validated in the beat editor (see chapter 7).
"sample_play"	sample url	Event triggered when a sampler is played
"sample_stop"	sample url	Event triggered when a sampler is stop
"radio_info"	position (ms)	Automix player - Event triggered when the automix player position has been modified. Gives info such as the player state (play/stop/load), the song playback position (in seconds), the total song duration (in seconds).
"radio_playing" "radio_mixing"	song info (the _infoGui string you passed)	Automix player - Event triggered when a new song get mixed. The radio_playing song is the main song being played, the radio_mixing song is the next song being mixed.
…	…	any information can be retrieved…

Example of application in Javascript:

```
function GET_INFORMATION(_key:String, _value:String)
{
   console.log(_key, _value);
   switch(_key) {
     …
   }
}
```

Part 5 : Anti-theft technology (optional)

Problem

Without an effective protection, it is easy to grab/steal an online music stream. The hackers will be then able to download your music on their hard-drives (e.g. as an MP3 file). This can be devastating for your business model !

Our solution

Many forms of hacking expose your music to be stolen and absolutely all of them must be considered for adopting an anti-theft protection. After one year of intense R&D and thanks to our deep expertise in audio processing, web audio and hacking, we are able to offer an ultimate protection against all forms of attack, such as :

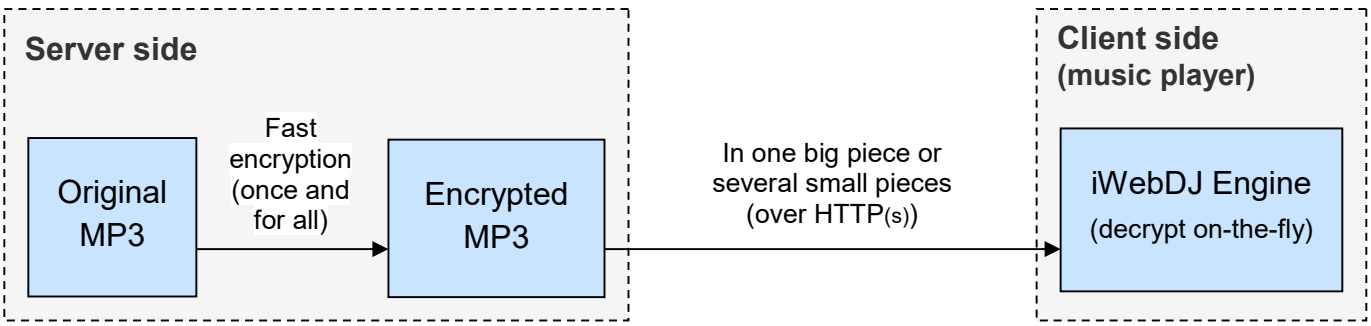
- Network packet grabber
- Browser cache grabber
- Memory scanner / grabber
- Decompilation of the source code
- Desobfuscation of the source code
- Kernel step-by-step debugger
- Music encryption bruteforce

Long-term commitment

The fight against hackers never really stops and reactivity is the key of success. We commit ourselves to provide music theft protection for a long-term period. If a new hacking technique emerges in the future, we guarantee that we will disable it !

Technical implementation

The MP3 files are first encrypted once and for all on the server side. Then, they are sent over HTTP(s) in one big piece or several small pieces and decrypted on-the-fly during playback on the client side. Several other layers of protection are also implemented in the iWebDJ engine.



iWebDJ mp3 encrypter tool

To encrypt the MP3 on the server side, we provide a fast command-line tool.

The encryption is based on top-notch techniques specially made for MP3 files. It is super fast (200ms per song) and doesn't re-encode the mp3 (preserve sound quality)

Syntax :

```
./iwebdj_mp3_encrypter input.mp3 output.encrypted
```

Load encrypted files

Then to load an encrypted MP3 on iWebDJ, you simply need to use `loadSong` with `_encryptedMp3` set to true if you want to send the file in one big piece.

A note about RTMP protocol

It is super easy to rip a RTMP(e/s) stream (with tools such as rtmpdump) and you can't do anything about it since RTMP is a high-level closed protocol. That's why iWebDJ use HTTP(s) for secure music delivery.

Part 6 : iWebDJ song metadata

For each song, the iWebDJ mixer will need some special metadata attached to it. This song metadata is a character string containing unique information such as to the tempo, the detected loudness, the structure of the song, the waveform…

Generating the metadata

The metadata has to be pre-computed once and for all on the server side with the iWebDJ music analyzer tool (see below). The computed metadata is then stored in the files' ID3 tag.

Using the metadata into your service

You are free to extract and use the information stored into the iWebDJ metadata for enhancing your service. More precisely, from the metadata you can get : the song tempo (bpm), the super crisp waveform (which is not an average on the audio samples, it uses a much more complex method), the song total length, the volume normalization coefficient (loudness)…

The iWebDJ music analyzer

Analyzer features :

- fully automatic though command-line for Windows32 and Linux64
- also usable by drag-and-drop in the windows explorer
- multiple files or folders batch processing
- super fast (1-2 seconds per song), multi-threaded and 64 bits

Input :

Audio file(s) or folder(s) containing audio files (for folders, only one level is supported).
About the audio format : any MP3, any WAV, any sample rate.

Output :

For each file, the analyzer outputs a metadata string containing the result of the analysis (typical around 3kB character string). This string will be added in the ID3 tag of file. If you are using WAV files as input, the analyzer will generate a MP3 file. The output directory is ./output.

Syntax :

```
iwebdj_analyzer file1 file2 ... folder1 folder2 ... [ options ]
```

Example (linux) :

```
./iwebdj_analyzer_linux64 song1.mp3 song2.mp3 folder1
```

Options :

-b	--bitrate	output mp3 bitrate [between 64 - 320] (default = 128)
-i	--instance	to run multiple instances of the analyzer, specify a different instance number for each of them [between 0 – 65535] (default = 0)
-d	--deep	switch to deeper analysis mode for songs with error_flag = 010 [0 or 1] (default = 0) /!\ use this mode only for songs with problems, not by default

Use by drag and drop :

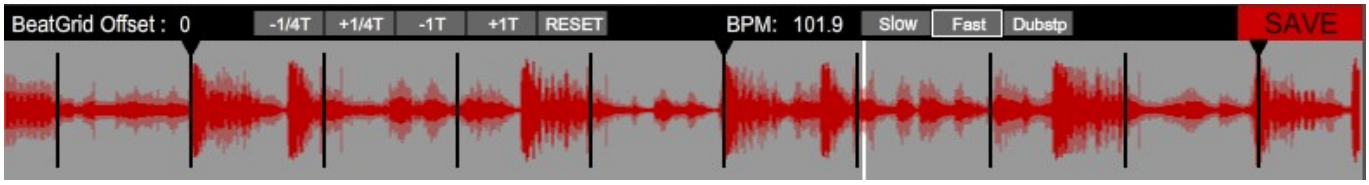
If you are on Windows, simply drag and drop one or multiples files or browsers on the program iwebdj_analyzer_win32.exe in the Windows Explorer.

Note about FTP :

If you transfer the analyzer on your server with FTP, please use the “Binary” mode instead of the “Ascii” mode otherwise the analyzer executable will be corrupted.

Part 7 : Beatgrid editor (optional)

Beatgrid Editor



The beat editor is an optional tool (not provided in the standard version) which allows to correct eventual errors made by the iWebDJ Music Analyzer without re-analyzing the songs. For most project, the beat editor is not needed since the iWebDJ Analyzer is extremely accurate and reliable, however if you really encounter some problems on complex songs, you have the ability to manually adjust the detected beatgrid.

To load the beat editor, load the standalone iwebdj_beateditor.swf. Then to retrieve the modification by the beat editor, you must use the `GET_INFORMATION` “beat_editor” event which gives two values bpmRangeMax and beatOffsetCorrection each time you hit the “save” button in the beat editor. You must store these two values somewhere and then pass them to the `loadSong` method each time you load a song which has been corrected.

Part 8 : Recording API

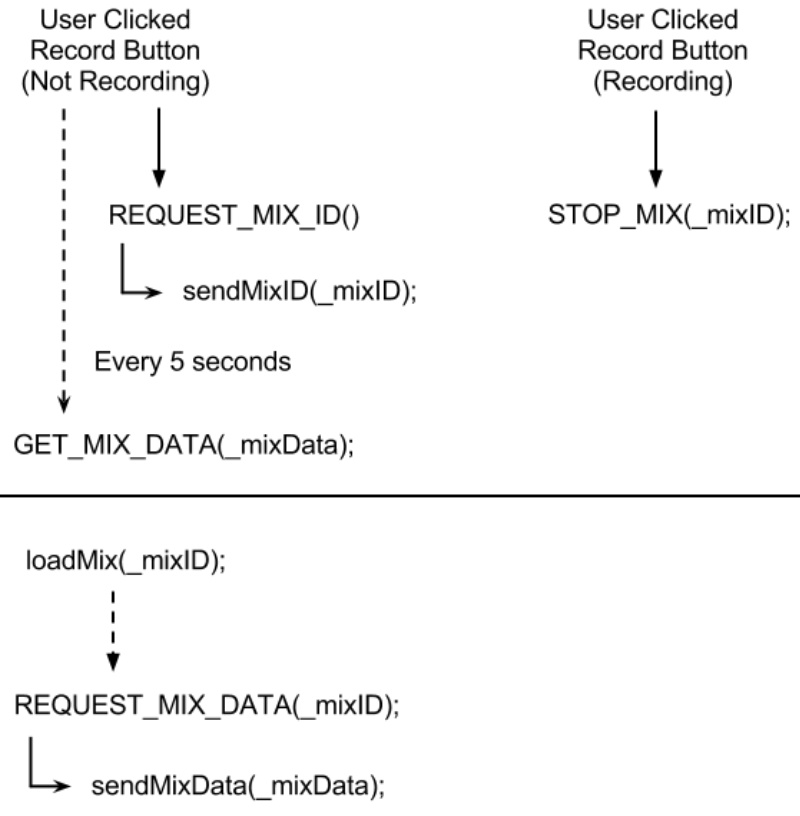
/!\ This information below are correct but are not easy to understand. The record / broadcast feature is now integrated into a system with a plug-and-play module (see chapter 2).

Once you have fully integrated iWebDJ into your system, you will be able to use the innovative recording and replaying features with the Recording Javascript API.

Here are the JavaScript commands :

iWebDJ	> HTML	`REQUEST_MIX_ID();`
HTML	> iWebDJ	`sendMixID(_mixID:String);`
iWebDJ	> HTML	`GET_MIX_DATA(_mixData:Object);`
iWebDJ	> HTML	`STOP_MIX(_mixID:String);`
HTML	> iWebDJ	`loadMix(_mixID:String);`
iWebDJ	> HTML	`REQUEST_MIX_DATA(_mixID:String);`
HTML	> iWebDJ	`sendMixData(_mixData:Object);`

Recording system overview :



Summary :

You'll have to be able to save a `_mixData` object and to retrieve it. The best way for this is to use a database. Any kind of database will do the job, but for better performance, a NO-SQL database is far superior (such as MongoDB…).

Because iWebDJ record the actions of DJ instead of the audio stream, recorded mixes don't weight much. A typical size for an one hour mix is 100-200kB.

Chart Explanation :

1. When receiving the `REQUEST_MIX_ID()` event, you'll have to call the `sendMixID(_mixID)` method on the iWebDJ Player and provide your `_mixID`.

   If `sendMixID (_mixID)` hasn't been called within 5 seconds, `_mixData.id` will be set to its default value, which is **"empty"**.

2. Every 5 seconds, you'll receive the `GET_MIX_DATA(_mixData)` event, and you'll have to save into your database the `_mixData` object (you may its content below).

   Important : `info.duration` and `info.live` both need to be updated (updated). All others have to be appended (added).

3. When receiving the `REQUEST_MIX_DATA(_mixID)` event, you'll have to retrieve the `_mixData` object, based on the `_mixID` argument from your database and call the `sendMixData(_mixData)` method on the iWebDJ Player.

   If you're replaying a mix in live, the `REQUEST_MIX_DATA(_mixID)` event is triggered every 5 seconds.

Mix Data Object format :

- `id:String` is the `_mixID` sent through `sendMixID(_mixID)` function (default : **null**).
- `info:Object` is an object storing the mix duration and live status.
  - `duration:Int`
  - `live:Boolean`
- `packets:String` is an encoded string of the DJ's mix actions.
- `beatMap:Array` is the waveform of the mix.
- `songs:Object` is an object with the list of songs in the mix for each player.
  - `player1:Array`
  - `player2:Array`
  - `started:Array`

Implementation Example :

```
function REQUEST_MIX_ID() {
  $.get('get_id.php').success(function(_response) {
    iWebDjPlayer.sendMixID(_response.id);
  }
}
```

```
function GET_MIX_DATA(_mixData) {
  $.post('save_mix.php', {
    data: JSON.stringify(_mixData)
  });
}
```

```
function STOP_MIX(_mixID) {

}
```

```
function REQUEST_MIX_DATA(_mixID) {
  $.get('get_mix.php', {
    id:_mixID
  }).success(function(_response) {
    iWebDjPlayer.sendMixData(_response);
  });
}
```

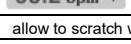
Additional Information :

- You may implement User Authentication inside the `REQUEST_MIX_ID()` function.
- You may display information / redirect your users when `STOP_MIX()` is triggered.
- You may remove/clean mixes under some circumstances in any PHP Script.
- If a user doesn't re-click the record button (Computer Reboot, Page Closing), it will stay flagged as "live" and you'll have to manually fix it.

Part 9 : DJ Mixer GUI specification

List of all graphical elements

Here are all the GUI elements of the iWebDJ DJ mixer. This list is only for customers who wants to you design their own GUI by themselves.

Function	Items name	Items type	Items description
player playback x 2	play / pause	clickable button	play or pause the current song
	beatmatch	clickable button	beatmatch the current song with the other player
player progress bar x 2	progress bar	clickable bar	display the current playing position of the song but also allows to seek the song by clicking on it. (this element use to be semi-transparent)
	loading bar	non-clickable bar	display the current loading progress of the song. (this element use to be semi-transparent)
	waveform area	non-clickable display area	display the song structure, available in 2 modes ("simple" like the current iwebdj demo, or "mirror" like Traktor or MixVibes Cross)
player beatjump x 2	beatjump forward	clickable button	while playing, jump 8 beats forward by staying in sync. while pausing jump 1 beat forward (helping to set the cue)
	beatjump backward	clickable button	while playing, jump 8 beats fackward by staying in sync. while pausing jump 1 beat forward (helping to set the cue)
player cue x 2 (config 1*)	set cue	clickable button	while playing or pausing, set the cue
	go to cue	clickable button	while playing or pausing, go to the cue
player cue x 2 (config 2*)	cue	clickable button	while playing, go to the cue. while plausing, set the cue (cue behavior used on lot of dj hardware)
player loop x 2 (config 1*)	loop on/off	clickable button	do looping using the current looping length
	loop change forward	clickable button	change the loop length forward, this order : 2, 4, 8, 2, 4, 8…
	loop change backward	clickable button	change the loop length backward, this order : 8, 4, 2, 8, 4, 2… this button can be easily omitted !
	loop length indicator	non-clickable textarea	indicate the current loop length (2, 4 or 8) or can also be (small, medium, big)
player loop x 2 (config 2*)	loop 2 beats (small loop) on/off	clickable button	do 2 beats looping (small loop)
	loop 4 beats (medium loop) on/off	clickable button	do 4 beats looping (medium loop)
	loop 8 beats (big loop) on/off	clickable button	do 8 beats looping (big loop)
player effects x 2 (config 1*)	sfx on/off	clickable button	do sfx
	sfx change forward	clickable button	change the current sfx forward, this order : 1, 2, 3, 1, 2, 3…
	sfx change backward	clickable button	change the current sfx backward, this order : 3, 2, 1, 3, 2, 1… this button can be easily omitted !
	sfx type indicator	non-clickable textarea	indicate the current sfx type (1, 2 or 3) or can also be (juggling, crazycut, backspin)
player effects x 2 (config 2*)	sfx 1 (juggling) on/off	clickable button	do sfx 1 (juggling)
	sfx 2 (crazycut) on/off	clickable button	do sfx 2 (crazycut)
	sfx 3 (backspin) on/off	clickable button	do sfx 3 (backspin)
player pitch control x2 (config 1*)	pitch slider	clickable slider	control pitch (song playback speed)
player pitch control x2 (config 2*)	small vertical arrows near the bpm value	clickable area	control pitch (song playback speed) with small verticals arrows near bpm value : 
player scratch control x 2	mouse scratch	clickable area	allow to scratch with the mouse. Usually is rotating shape/image.
player text areas x 2	playing position	non-clickable textarea	song position
	song total length	non-clickable textarea	song total length
	song name	non-clickable textarea	song name
	song bpm	non-clickable textarea	current song bpm, ajusted with the pitch control
	song pitch	non-clickable textarea	current pitch applied (in %)
player animations x 2	current sound intensity	raw data	useful to make a VU metter
> any kind of animation using :	current beat intensity	raw data	useful to make something blink with the beat
	current beat position	raw data	useful to make something rotate with the beat
mixer volume control	volume faders x 2	clickable sliders	control the volume of each players
	cross fader x 1	clickable slider	control the volume of both players
mixer visualization	vumeter x 2	non-clickable display area	indicate the volume and intensity of each players
mixer knobs x 2 ***	bass frequency	rotary knob	ajust bass frequency (can be called also "low")
	vocal frequency	rotary knob	ajust vocal frequency ("vocal" means a mix of mid and high frequency, can be called also "mid/high" or "mid")
	hipass/lowpass filter	rotary knob	ajust hipass/lowpass resonant filter (Traktor style) - great effect!
	gain (see note...)	rotary knob	gain ajustement (absolutely not necessary if you have volume faders!), the gain ajustement in iWebDJ is automatic!
mixer kill switch x 2 (config 1*)	kill bass frequency on/off	clickable button	kill bass frequency (can be called also "low")
	kill vocal frequency on/off	clickable button	kill vocal frequency (can be called also "mid/high" or "mid")
mixer kill switch x 2 (config 2*)	kill low frequency on/off	clickable button	kill low frequency
	kill mid frequency on/off	clickable button	kill mid frequency
	kill high frequency on/off	clickable button	kill high frequency
mixer headphone cueing x 2 **	headphone cueing on/off	clickable button	activate the headphone cueing for each deck
mixer automix	automix on/off	clickable button	turn on/off the automix
	automix next song	clickable button	automix the next song
beat visualization (config 1*)	unified visualisation (big window)	clickable display area	display the beats of both players on the same big window, usually on the top (can also be a spectrum analyzer or an oscilloscope by clicking on it)
beat visualization (config 2*)	separated visualisation (2x small window)	non-clickable display area	display the beats of both players on 2 separated small windows, usually one on each player (clicking will have no effect)
recording	recording button	clickable button	recording function / play, stop, pause the recoding + display the recording time
preloading animation	preloading animation	non-clickable display area	if you want, you can design the preloading animation the way you want. Try to make something very light in term of complexity because the preloader must weight only a few kB.

Remarks / caption

- Any elements can be omitted (you don't need to use all of them)

- All buttons must have 4 states : normal (unpressed), rollover (the mouse over it), pressed 1 (with player1 color), pressed 2 (with player2 color). Each states of the buttons must be on seperated layers, please look at the example PSD to see exactly the image format we are expecting.

- * there is 2 ways to do this function : config1 OR config2

- ** The headphone cueing will be activated or not through a config menu, so the headphone preview buttons will only be displayed if the option is turned on (this implies to have a Y-splitter so only a small proportion of the users will be able to do it)!

- *** For knobs which are more complex than a rotating bitmap/vector, we will need to create an image with 47 frames, such as :

Additional options

options	default value	description
_automixOnStart	false	activate automix on start
_requestLoadSongOnStart	true	load a song on start
_playSongOnStart	true	play the loaded song on start