Welcome to FWSoundMixer 1.1

FlashyWrappers SoundMixer allows you to capture dynamically playing audio in Flash Player by replacing Sound class with FWSound class. You can connect FWSoundMixer to FWVideoEncoder for recording video and audio from Flash / AIR at the same time.

You can use FlashyWrappers on desktop (AIR Mac / Windows), web (Flash Player) or mobile (AIR iOS / AIR Android), even the native extensions for the desktop AIR platforms are not available in this version. ANE's for iOS and Android are available together with SWF / SWC files for the rest of the platforms.

Warning: Currently there are issues with microphone recording on Android (this will hopefully be fixed soon as Android is our focus currently).

Quickstart guide

This describes the most basic usage

1) Include the SWC / ANE for your platform into your project. **Desktop / Flash only:** Copy FWSoundMixer_SWFBridge.swf to your applications root so its together with your “main” swf for simplicity.

2) Make sure to target the latest the latest Flash Player runtime. You need to target **at minimum** Flash Player 11.5 (SWF version 18)**.**

3) Start coding!

Let's assume you want to capture playing track and microphone at the same time:

1) **Import FWSoundMixer:**

```
import com.rainbowcreatures.*;
import com.rainbowcreatures.swf.*;
import flash.events.StatusEvent;    // in case you're not importing this already
```

2) Get FWSoundMixer class instance and setup event listener:

```
var mySoundMixer:FWSoundMixer = FWSoundMixer.getInstance();
mySoundMixer.addEventListener(StatusEvent.STATUS, onStatus);
mySoundMixer.load();
```

3) Implement onStatus event handler & initialize FWSoundMixer.

```
private function onStatus(e:StatusEvent):void {
        if (e.code == "ready") {
                // If you want to hear the buffer you're capturing
                mySoundMixer.playSounds = true;
                mySoundMixer.init();
                var track:FWSound = new FWSound(null, null, new Track(),
                                                        mySoundMixer, false);
                // get ready for sounds playing
                mySoundMixer.startCapture(true);
                // just like Sound, start at position 0, play 100 times
                track.play(0, 100);
        }
}
```

4) Whenever you need to, stop capturing like this – this will also stop playing any sounds in case playSounds was set to true:

```
mySoundMixer.stopCapture();
```

5) If you need access to raw audio data, set a "sndData" method in "init". This method will be called right after the raw data is filled in FWSoundMixer.

```
mySoundMixer.init(processRawData);
function processRawData():void {
        var rawData:ByteArray = mySoundMixer.rawData;
        // you can now access the current rawData - you could fill your
           own ByteArray and later compress / send the audio to server etc.
        rawData.position = 0;
        myByteArray.writeBytes(rawData);
}
```

6) If you need to connect FWSoundMixer to FWVideoEncoder, init:

// This will automatically call addAudioFrame in myVideoEncoder whenever rawData is
// available in FWSoundMixer.

*mySoundMixer.init(..., **myVideoEncoder**);*

For complete example on how to connect FWVideoEncoder with FWSoundMixer, take a look at "exampleStarling" in FlashyWrappers SDK. It shows how to record video and audio from AIR game using FlashyWrappers and FWSoundMixer.

Getting rid of SWFBridge

You can optionally include the "standalone" SWC into your project to get rid of the SWF dependency. Do this at your own risk, as you might have issues with autocomplete, long compilation times and/or Undefined Reference errors in Adobe IDE's. That's why we are distributing this library in SWC / SWF format by default. Just use the "*lib/FlashPlayer/FWSoundMixer.swc*" library. From your code you will also need to erase the "load()" call, so that after getInstance() your code is immediatelly followed by code you would have in the onStatus '*ready*' handler (as no swf loading is performed).

A little explanation

Why would we need a custom sound mixer in the first place? In case we need to capture dynamically playing audio, our first guess might be trying to find a way to expose currently playing audio "raw data" of something like the SoundMixer class. Surprisingly, we are not allowed to do that. The only thing we can get from that class is limited sample data to display waveforms. Even if we could get data from SoundMixer, it wouldn't solve the issue of mixing microphone, as that's kind of separate entity from all the Flash Sound instances.

On any platform which is NOT Flash we might turn to native audio recording (which we are going to do in the near future), but in Flash Player which is a tight sandbox, we have no such way of natively recording audio.

That's why our solution was to recreate the basic parts of SoundMixer and virtually "play" Flash Sounds through that as well. By doing that (and thanks to extending the Flash Sound class), we're still able to play Flash sounds by our app as usual, but also mix the sounds into our own sound mixer and expose the raw audio data.

As for microphone, it's just a special type of sound in our sound mixer.