Eclipse Audio

Loudspeaker Processing Interchange Format (LPIF)

Version 2.3, 20 June 2023

1 Introduction

This document describes a JSON format for transferring loudspeaker EQ settings from FIR Designer to DSP amplifier and processor control programs. The format can be passed as a file or over a network.

Any questions, email info@eclipseaudio.com.

Revisions

- V2.3, 20 June 2023: Added RAM Audio.
- V2.2, 13 August 2022: Added Biamp. Added Marani NXF filters.
- V2.1, 9 December 2021: New filter types for Marani.
- V2.0, 21 January 2021: Renamed LPIF. New filter types for BSS.
- V1.1, 6 November 2019: Minor edits and new filter types for Linea Research
- V1.0, 23 September 2019: Initial release as EA Transfer JSON spec

2 Scope

The document describes only the JSON structure. Specific range limits for each parameter may be different for different amplifiers, processors and control programs. Where ranges are supplied by the manufacturer, FIR Designer can apply validation checks and alert the user before exporting the JSON file or transmitting the JSON data.

The format:

- caters to both single channel processing chains and multi-way processing for multi-way loudspeakers,
- handles processing where crossovers and/or EQ maybe be split both pre and post limiter,
- provides for different sample rates in different parts of the processing, and
- provides for both parameterized EQ and raw filter coefficients.

The JSON structures and examples in this document show the layout present inside JSON text files. JSON code libraries automatically handle de-formatting of binary or text files.

3 Processing Block

A processing block consists of data for IIR filters, FIR filters, delay, gain and polarity and running at a common sample rate. Not all data need be present. For example, a processing block could consist of only IIR, or only IIR and FIR filter coefficients.

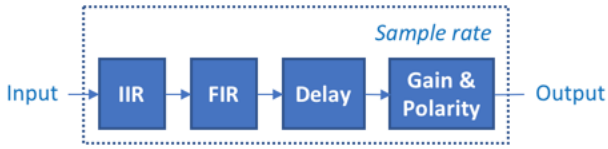

Figure 1: Processing Block

The order of processing – e.g. IIR, then FIR, then Delay… - is not implied nor specified, and the receiver that loads the JSON data may choose to process audio in any order.

Note: JSON code libraries usually don't maintain the write-order when creating JSON documents and typically use alphabetical order when displaying JSON data.

{			
"type":	string	enum	default ""
"channel":	number	(integer)	no default
"channel-name":	string		default ""
"sample-rate":	number	Hz	no default, required for FIR and biquads
"iir": [	array	See section 3.2	
]			
..or..			
"biquads": [	array		
]			
"fir": {		See section 3.3	
}			
"delay":	number	milliseconds	default 0
"gain":	number	dB	default 0
"invert":	bool	true or false	default false
}			

3.1 "type" and "channel"

The "type" enumerated field denotes the location of the Processing Block in a Loudspeaker Preset. Valid types are "input", "output-a" and "output-b".

The "channel" number is the input or output channel index in a loudspeaker Preset. If only one channel is present, this field can be omitted. This number starts at 1.

See Section 4 for more details.

3.2 IIR filters

An IIR filter can be specified parametrically (type, frequency, gain, Q or bandwidth), as raw coefficients (b0, b1, b2, a0, a1 & a2) calculated for the indicated sample-rate, or both. Since different processors often have slightly different interpretations of Q or bandwidth, raw

coefficients are preferred for specifying the IIR filtering. Where both are included, the receiver can choose to use either.

```
{
      “enabled”:          bool            true or false
      "platform":         string
      “type”:             string          enum
      “sub-type”:         string          enum
      “frequency”:        number          Hz
      “order”:            number          (integer)
      “gain”:             number          dB
      “ripple”:           number          dB
      “stop”:             number          dB
      “lobe-height”:      number          dB
      “q”:                number
      “bandwidth”:        number
      "slope":            number
      "bulk-gain":        number          dB (default 0)
      “biquads”: [        array
            {
                  “b0”:   number
                  “b1”:   number
                  “b2”:   number
                  “a0”:   number
                  “a1”:   number
                  “a2”:   number
            }
      ]
}
```

An array of IIR filters represents the IIR processing within a Processing Block. Each IIR filter can also include biquads.

Alternatively, the IIR processing can simply consist of a “biquads” array, without any IIR filter type specifications. The JSON array structure ensures the biquad order is maintained.

The "platform" string is optional and can be used for a processor-specific or brand-specific label or code. Since different platforms have different interpretations of parameters like Q or BW, the "platform" code can help maintain accurate, platform specific interpretation of these parameters. FIR Designer will set this according to the FIR Designer IIR filter mode. Examples include "general", "powersoft", "linearesearch", "marani", "qscqsys", "symetrix", "bss", "ramaudio" and "biamp".

3.2.1 “type” enumeration

Types	Required fields
“parametric”	“frequency”, “gain”, “Q” or “bandwidth”
"band-pass"	"frequency", "Q" or "bandwidth"
"band-stop"	"frequency", "Q" or "bandwidth"
"notch"	"frequency", "Q" or "bandwidth"
“lowpass-butterworth”	“frequency”, “order”
“highpass-butterworth”	“frequency”, “order”
“lowpass-bessel”	“frequency”, “order”
“highpass-bessel”	“frequency”, “order”

“lowpass-bessel-m3db”	“frequency”, “order”
“highpass-bessel-m3db”	“frequency”, “order”
“lowpass-lr”	“frequency”, “order” (order must be even numbered)
“highpass-lr”	“frequency”, “order” (order must be even numbered)
“lowpass-chebyshev1”	“frequency”, “order”, “ripple”
“highpass-chebyshev1”	“frequency”, “order”, “ripple”
“lowpass-chebyshev2”	“frequency”, “order”, “stop”
“highpass-chebyshev2”	“frequency”, “order”, “stop”
“lowpass-elliptic”	“frequency”, “order”, “ripple”, “stop”
“highpass-elliptic”	“frequency”, “order”, “ripple”, “stop”
"lowpass-variable-q"	"frequency", "bandwidth" or "q"
"highpass-variable-q"	"frequency", "bandwidth" or "q"
“lowpass-ntnc”	“frequency”, “order”, “lobe-height”
“highpass-ntnc”	“frequency”, “order”, “lobe-height”
"lowpass-ntm-36"	"frequency"
"highpass-ntm-36"	"frequency"
"lowpass-ntm-52"	"frequency"
"highpass-ntm-52"	"frequency"
"lowpass-hardman"	"frequency", "order"
"highpass-hardman"	"frequency", "order"
"lowpass-nxf"	"frequency", "order", “lobe-height”
"highpass-nxf"	"frequency", "order", “lobe-height”
“low-shelf”	“frequency”, “gain”, “q” or “bandwidth” or "slope"
“high-shelf”	“frequency”, “gain”, “q” or “bandwidth” or "slope"
"dual-shelf"	“frequency”, “low-gain”, “high-gain”, “q” or “bandwidth” or "slope"
“allpass”	“frequency”, “order”, “q” or “bandwidth”
“custom”	“biquads” array

Note:

“frequency”	> 0.0 Hz and < sample-rate/2 Hz
“ripple”	> 0.0 dB
“stop”	< 0.0 dB
“...-ntnc”	Refers to the Neville Thiele notched crossover type. Valid “order” values are 4, 6 and 8.
“lobe-height”	< 0.0 dB. The height of the NTNC or NXF response lobe, beyond the notch. • For NTNC filters, the value is a continuous decimal number. Typical NTNC values are -36 dB for 4th order and -52 dB for 8th order. • For NXF filters, the value has discrete values of -40, -45, -50, -55, -60, -65 and -70.
"slope"	dB/oct

3.2.2 "sub-type" enumeration

This is used by some platforms to denote differences within a specific type.

3.2.2.1 Marani IIR mode

Type	Sub Types
"low-shelf"	“v1”, “v2”
"high-shelf"	“v1”, “v2”

3.2.2.2 RAM Audio IIR mode

Type	Sub Types
"parametric"	“a”, “c”
"low-shelf"	“6”, “12”, “12q”
"high-shelf"	“6”, “12”, “12q”

3.2.3 “Q” and “bandwidth”

Only one of these should be present. If the JSON receiver only uses one of these, and receives the other, it can optionally indicate a receive error, or convert from one to the other.

3.2.4 “biquads”

A biquad array consists of raw transfer function coefficients in numerator denominator form. The JSON array structure ensures the biquad order is maintained. The receiving program can use the coefficients directly or convert them into other forms; e.g. state-space.

Notes:

- Biquad coefficients for Linea Research Hardman filters are not export.
- Biquad coefficients for Marani NXF filters are not exported, nor supported in LPIF, due to the implementation topology that isn't simply a concatenation of biquads.

3.3 FIR filter

The FIR filter data includes and “enable” flag, and optional "latency" sample number and an array of raw coefficients as decimals. A value of 1.0 followed by zeros represents unity or 0 dB, with no delay. The "latency" is informative only and represents the sample location of the time=0 peak in the FIR filter. It is set from the "FIR Filter Delay" setting in FIR Designer.

```
“fir”: {
        “enable”:           bool            true or false
        "latency":          number          number of samples
        “coefs”: [          array
                number,
                number,
                …
                number
        ]
}
```

4 Loudspeaker Preset

A Loudspeaker Preset consists of one or more Processing Blocks that provide all EQ processing necessary for single and multi-way loudspeakers.

The preset consists of one optional “Input” Processing Block, and one or more output Processing Blocks, where different output blocks denote settings for different drivers in a multi-way loudspeaker.

There are two “type” indications for output processing blocks; “output-a” and “output-b”. Where one Processing Block is adequate for all output EQ settings, the block should be labelled “output-b”. For situations where output EQ is spread either side of other processing – such as when splitting crossovers either side of output limiters or separating speaker EQ from output EQ – both “output-a” and “output-b” blocks can be specified.

Each Processing Block can have a different sample rate. (For example, the LAKE FIR 3-way module runs input processing at 96 kHz and output processing at either 12 kHz, 24 kHz or 48 kHz.)

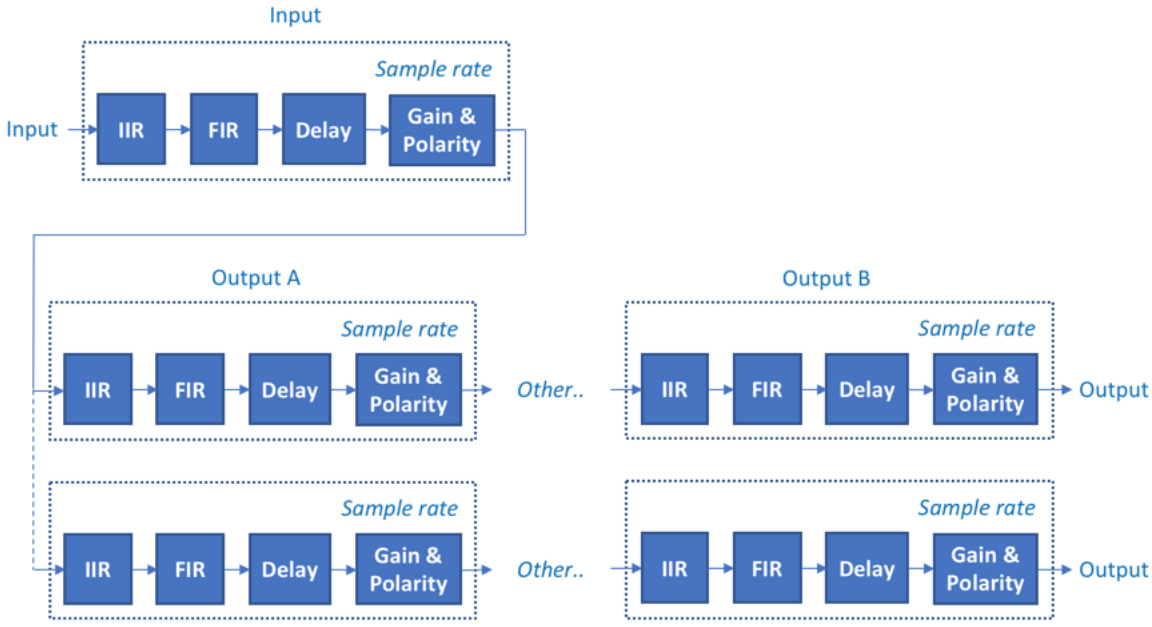

Figure 2: Loudspeaker Preset

If a channel's FIR filter has a different sample rate to the IIR and Delay blocks, the FIR filter should be placed in a separate Processing Block.

4.1 Format

The loudspeaker Preset file begins with some informative fields, followed by an array of Processing Blocks.

“preset”: {		
“title”:	string	e.g. Loudspeaker make/model
“program”:	string	Program used to make the file

```
        “program-version”:      string          Program version
        “date-time”:            string          File creation date
        “processing-blocks”: [  array           See Section 3
        ]
}
```

4.2 Loudspeaker Preset Examples

4.2.1 One output processing block


```
{
  "preset": {
    "date-time": "Sun Apr 15 21:02:06 2018",
    "processing-blocks": [
      {
        "delay": 0,
        "fir": {
          "coefs": [
            0,
            0,
            0.9999999999999998,
            0,
            0,
            0,
            0,
            0
          ],
          "enable": true
          "latency": 2,
        },
        "gain": 0,
        "iirs": [
          {
            "biquads": [
              {
                "a0": 1,
                "a1": -1.9814885091445689,
                "a2": 0.9816582826171297,
                "b0": 0.9907866979404248,
                "b1": -1.9815733958808497,
                "b2": 0.9907866979404248
              }
            ],
            "enabled": 0,
            "frequency": 100,
            "order": 2,
            "type": "highpass-butterworth"
          },
          {
            "biquads": [
              {
                "a0": 1,
                "a1": -1.99473479382958,
                "a2": 0.9949057022407183,
                "b0": 0.9974101240175746,
                "b1": -1.9948202480351491,
                "b2": 0.9974101240175746
              },
              {
                "a0": 1,
                "a1": -1.9853905187431602,
```

```
                    "a2": 0.985560626538992,
                    "b0": 0.9927377863205383,
                    "b1": -1.9854755726410767,
                    "b2": 0.9927377863205383
                  },
                  {
                    "a0": 1,
                    "a1": -1.978297652923569,
                    "a2": 0.9784671530043143,
                    "b0": 0.9891912014819707,
                    "b1": -1.9783824029639414,
                    "b2": 0.9891912014819707
                  },
                  {
                    "a0": 1,
                    "a1": -1.974480116200167,
                    "a2": 0.9746492891952577,
                    "b0": 0.987282351348856,
                    "b1": -1.974564702697712,
                    "b2": 0.987282351348856
                  }
                ],
                "enabled": 0,
                "frequency": 100,
                "order": 8,
                "type": "highpass-butterworth"
              }
            ],
            "invert": false,
            "sample-rate": 48000,
            "type": "output-b"
          }
        ],
        "program": "FIR Designer",
        "program-version": "2.4.2",
        "title": "Test loudspeaker"
    }
}
```


4.2.2 One output processing block with only a FIR filter


```
{
    "preset": {
        "date-time": "Sun Apr 15 21:02:06 2018",
        "processing-blocks": [
            {
                "delay": 0,
                "fir": {
                    "coefs": [
                        0,
                        0,
                        0.9999999999999998,
                        0,
                        0,
                        0,
                        0,
                        0
                    ],
                    "enable": true
                    "latency": 2,
                },
                "sample-rate": 48000,
```

```
            "type": "output-b"
          }
        ],
        "program": "FIR Designer",
        "program-version": "2.4.2",
        "title": "Test loudspeaker"
    }
}
```