

Zusi-Soundthesizer Manual

by Andreas Karg

December 5, 2007

Kindly supported by:
Daniel Schuhmann
Jens Hauptert

Contents

1	Introduction	2
2	Syntax Stuff	2
2.1	A Tiny XML Primer	2
2.2	Serious Sounds	2
2.3	Dependencies	3
2.4	Operators	3
2.5	Triggers	4
3	Famous Last Words	5
A	Appendix: Zusi's Measurements	5

1 Introduction

The Zusi-Soundthesizer is intended to enhance Zusi's own sound system. One can create quite complex soundscapes using a special XML -based file format.

Back in them olden times, when the grass was green and the girls were pretty, there was one single minor flaw in the paradise city of Zusi:

All trains used the same rather generic sound system. Unfortunately, especially rotary current EMUs have a kind of musical world of their own which couldn't be reproduced in Zusi at the time. Thus I wrote the Soundthesizer. It uses Zusi's TCP interface by which almost every measurement in the simulation can be accessed. I've even created a super-special-mega cool programming language which is based on XML. It takes some time to get used to it, but once you've got the hang of it, you might even find that there's a bit of logic to it.

You can control four properties of each sound: **volume**, **pan**, **frequency** (i.e. pitch) and **trigger**. This allows for quite sophisticated soundscapes. Unfortunately, there are some drawbacks:

- You cannot create complex loops that start up or run out before/after the actual loop.
- Diesel engines are pretty hard to do because of their complex changes of volume and the limited measurements Zusi offers in that area.

2 Syntax Stuff

2.1 A Tiny XML Primer

XML is a simple file format based on a tree structure. Just like folders on your hard-drive, XML contains one root object that may have several sub-objects which again contain other sub-objects and so forth. Each object can have special properties, called "attributes".

Objects consist at least of an opening and a closing tag:

```
<sound>
...
</sound>
```

If an object doesn't have any sub-objects, you can write it in one single line:

```
<refpoint x="15" y="0.4" />
```

The above example also shows how attributes work. They're put inside the opening tag and consist of a name (x or y), an equal sign and their value enclosed in quotation marks.

As far as the Soundthesizer is concerned, numeric values are always decimal.

In the Soundthesizer format, the root object is called "soundset". It takes one attribute, **name**, that is supposed to be shown somewhere in the main program. I never coded this feature, though, so you might as well hide secret messages in there.

2.2 Serious Sounds

The root object contains any number of sound objects:

```
<sound name="Hintergrund" looped="False">
  <file name="\Sounds\Hintergrund.wav"/>
  <volume value="none"/>
  <pan value="none"/>
  <frequency value="none"/>
  <trigger value="none"/>
</sound>
```

Attributes

- | | |
|---------------|---|
| name | The new sound's name. (This time, the name shows up in the Soundthesizer, so the attribute actually does have some meaning.) |
| looped | This one is either True or False . It determines whether the sound will be played only once or in an infinite loop. You may omit this attribute. In that case, it defaults to True . |

Sub-objects

- | | |
|------------------|---|
| file | This is the only compulsory sub-object. Its only attribute name takes the relative path and wav -file name of this sound. |
| volume | |
| pan | |
| frequency | The four properties. |
| trigger | |

```
<volume value="Zugkraft_Gesamt">
  <refpoint x="0" y="0"/>
  <refpoint x="1" y="0.25"/>
  <refpoint x="3" y="1"/>
</volume>
```

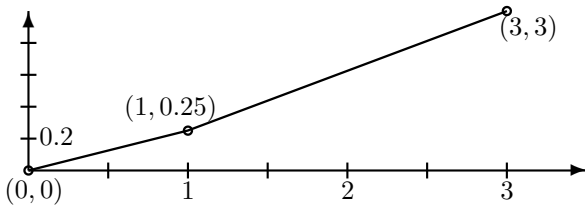


Figure 1: A simple example diagram

2.3 Dependencies

Each sound’s properties are dependent on one or more measurements. Think of an EMU’s transmission. You can usually hear the gears whirr when the train is running. The faster it goes, the higher the pitch. When the train is just jollily rolling along, the transmission is quiet. As soon as it gets some load to gnaw on it’ll start skreeching again.

So what we have here is a sound whose pitch depends on the train’s speed, and whose volume depends on the tractive effort. While the pitch will rise directly from zero to max between stop and full speed, the volume is only low when there is practically no tractive effort. As soon as there is, it will grow fast and already reach its peak at low effort.

Here, the Soundthesizer’s main advantage kicks in: For each property, you can define complex characteristic diagrams. Just name the measurement and a number of basing points.

Figure 1 shows a small example of how characteristic diagrams work in the Soundthesizer language. The `value` attribute contains the measurement on which our property depends. In our case, it’s total tractive effort¹. Each `refpoint` defines one point on the diagram.

As seen in figure 1, the Soundthesizer interpolates linearly between each pair of reference points. When Zusi sends a value, the Soundthesizer calculates the y-coordinate at the specified x-value. This coordinate is then used as a parameter for the prop-

¹For a complete list of measurements and their translations, see appendix A.

	-1	0	1	2
<code>volume</code>	–	silent	full	–
<code>pan</code>	left	center	right	–
<code>frequency</code>	–	min	normal	double
<code>trigger</code>	on	off	on	on

Table 1: The properties’ codomains

erty. The codomain for each property is listed in table 1.

2.4 Operators

Sometimes you might need not only one but several measurements as a dependency. In that case, you have to use operators. Operators are objects that contain several dependencies and/or other operators. Each sub-object is first evaluated and then linked to the other ones. For example, the `multiply` operator in figure 2 first calculates the result of the first dependency (“Zugkraft Gesamt”²), then does the same to the next dependency (“V1st” = Current speed) and then multiplies them.

You can combine operators in any order and number. They are evaluated hierarchically and in the order they appear.

There is one more way of using an operator. Look at the first line of the example. Instead of a `value` attribute, the `frequency` property has an `operator`. This is probably the most common form of an operator in the Soundthesizer. Instead of writing something like in figure 3, just put the operator in its appropriate attribute. In fact, I’m not even sure if it would work the other way around. Just don’t do it and everything will be fine. The following operations are supported:

- `add`
- `subtract`
- `multiply`
- `divide`
- `modulo`

²Total tractive effort, again

```

<frequency operator="add">
  <multiply>
    <dependency value="Zugkraft_Gesamt">
      <refpoint x="-10" y="0"/>
      <refpoint x="0" y="0" />
      <refpoint x="0.05" y="1" />
      <refpoint x="10" y="1" />
    </dependency>
    <dependency value="V1st">
      <refpoint x="0" y="0.6"/>
      <refpoint x="10" y="0.6"/>
      <refpoint x="10" y="1"/>
      <refpoint x="20" y="0.8"/>
      <refpoint x="160" y="0.8"/>
    </dependency>
  </multiply>
  <multiply>
    <dependency value="Zugkraft_Gesamt">
      <refpoint x="-10" y="1" />
      <refpoint x="-0.05" y="1" />
      <refpoint x="0" y="0" />
      <refpoint x="10" y="0" />
    </dependency>
    <dependency value="V1st">
      <refpoint x="-160" y="0.8"/>
      <refpoint x="160" y="0.8"/>
    </dependency>
  </multiply>
</frequency>

```

Figure 2: A rather complex example.

```

<frequency value="None">
  <add>
    ...
  </add>
</frequency>

```

Figure 3: Don't try this at home!

```

<trigger value="Zugkraft_Gesamt"
  direction="both">
  <refpoint x="-10" y="1"/>
  <refpoint x="-0.1" y="1"/>
  <refpoint x="0" y="0"/>
  <refpoint x="0.1" y="1"/>
  <refpoint x="10" y="1"/>
</trigger>

```

Figure 4: Trigger Happy Example

2.5 Triggers

While the properties `volume`, `pan` and `frequency` are more or less self-explanatory, `trigger`, alas, is not. A Soundthesizer trigger checks at every program cycle if its rounded value has changed from zero (`=false`) to any other number (e.g. 5 or `-1`, `=true`) or vice versa. Depending on its parameters, it starts or stops the sound accordingly. Figure 4 shows a little example of how a trigger might look. The trigger object has its unique attribute `direction`. It can take on the following values:

- up
- down
- both

An `up` trigger starts its sound it only when its value changes from `false` to `true`. A `down` trigger does the same on the change from `true` to `false`. And, finally, a `both` trigger starts the sound on any change. If the sound is set to non-loop, it will automatically stop playing after finish. If the sound loops, the trigger will activate and deactivate it according to its parameters. Thus, it is nonsense to trigger a sound loop with the `both` parameter, because the trigger will try to start the sound on any change.

An useful application of a trigger is the main circuit breaker of an electrical loco. If it is off, Zusi sends the measure as 0. If it's on, the measure is 1. Now, usually these breakers sound different depending on whether you switch them on or off. You would then code two sounds: One for switching on and one for switching off. Both would have a trigger depending on the `LM_Hauptschalter`³ mea-

³Indicator Lamp Main Circuit Breaker

surement. The *on* sound's trigger would be set to up, so it plays when the measure changes from 0 to 1. The other one gets a down trigger and thus plays on switching off.

3 Famous Last Words

I hope that this manual is understandable as well as reasonably well-written. Any proposals for improvement and corrections are welcome. Either put them on the Zusi-Forum or send a mail to AKarg@fahrpult.de.

That's about it, then....

A Appendix: Zusi's Measurements

Measurement	Translation
KeineFunktion	No function
V1st	Current speed
Druck_Hauptluftleitung	Main brake pipe pressure
Druck_Bremszylinder	Brake cylinder pressure
Druck_Hauptluftbehälter	Main air reservoir pressure
Druck_Hilfsluftbehälter	Auxiliary reservoir pressure
Zugkraft_Gesamt	Total tractive effort
Zugkraft_Achse	Tractive effort per axle
Strom	Current
Spannung	Voltage
Motordrehzahl	Engine rpm
Fahrstufe	Running notch
Uhrzeit_Stunde	Time: hour
Uhrzeit_Minute	Time: minute
Uhrzeit_Sekunde	Time: second
LZB_VZiel	CTCS speed approach
LZB_Sollgeschwindigkeit	CTCS set point speed
LZB_Sollgeschwindigkeit_alt	CTCS set point speed (old)
LZB_Zielweg	CTCS approach distance
AFB_Sollgeschwindigkeit	Cruise control set point speed
LM_PZB_Zugart_0	IL: IATC train type O
LM_PZB_Zugart_M	IL: IATC train type M
LM_PZB_Zugart_U	IL: IATC train type U
LM_PZB_500Hz	IL: IATC inducement 500Hz
LM_PZB_2000Hz	IL: IATC inducement 2000Hz
LM_PZB_1000Hz	IL: IATC inducement 1000Hz
LM_LZB_H	IL: CTCS H
LM_LZB_G	IL: CTCS G
LM_LZB_E40	IL: CTCS E40
LM_LZB_EL	IL: CTCS EL
LM_LZB_Ende	IL: CTCS termination
LM_LZB_V40	IL: CTCS V40
LM_LZB_B	IL: CTCS B
LM_LZB_S	IL: CTCS S
LM_LZB_Ü	IL: CTCS Ü
LM_LZB_Prüfen	IL: CTCS test
LM_Sifa	IL: AWS
LM_Hauptschalter	IL: Main circuit breaker
LM_Getriebe	IL: Transmission
LM_Schleudern	IL: Wheel skid
LM_Gleiten	IL: Wheel slip
LM_Mg-Bremse	IL: Magnetic brake
LM_H-Bremse	IL: Hydraulic brake

Measurement	Translation
LM_R-Bremse	IL: Rapid brake
LM_Hochabbremung	IL: High deceleration
LM_Schnellbremung	IL: Emergency brake
LM_Notbremung	IL: Emergency brake
LM_Tueren	IL: Doors
LM_Drehzahlverstellung	IL: Engine rpm change
LM_Fahrtrichtung_vor	IL: Reverser forward
LM_Fahrtrichtung_zurück	IL: Reverser reverse
LM_LZB_Zielweg	IL: CTCS approach
LM_GNT_G	IL: GNT G
LM_GNT_Ü	IL: GNT Ü
LM_GNT_B	IL: GNT B
LM_GNT_S	IL: GNT S
Blockname_Frei	Next unoccupied block
Tfz_Nummer	Vehicle number
Tfz_VMax	Vehicle max speed
Uhrzeit_digital	digital time
Schalter_Fahrstufen	Switch: Throttle
Schalter_Führerbremsventil	Switch: Main brake
Schalter_Dyn-Bremse	Switch: Dynamic brake
Schalter_Zusatzbremse	Switch: Shunting brake
Schalter_AFB-Geschwindigkeit	Switch: Cr. Cont. speed
Schalter_AFB	Switch: Cruise Control
Schalter_Mg-Bremse	Switch: Magnetic brake
Schalter_PZB_Wachsam	Switch: IATC confirm
Schalter_PZB_Frei	Switch: IATC cancel
Schalter_PZB_Befehl	Switch: IATC override
Schalter_Sifa	Switch: AWS
Schalter_Hauptschalter	Switch: Main circ. breaker
Schalter_Motor	Switch: Engine start
Schalter_Fahrtrichtung	Switch: Reverser
Schalter_Pfeife	Switch: Horn
Schalter_Sanden	Switch: Sand
Schalter_Türen	Switch: Doors
Schalter_Glocke	Switch: Bell
Schalter_Lokbremse_entlüften	Switch: Release loco brake
Schalter_Schleuderschutzbremse	Switch: Slip control brake
Schalter_Lüfter	Switch: Blower