

# linkEngine Manual

<b>Installation</b>	<b>1</b>
Compilation	1
Prerequisites	1
Building from source	2
<b>Running linkEngine</b>	<b>2</b>
<b>Creating CHAIN specifications</b>	<b>2</b>
<LINK> elements	3
<CHAIN> elements	3
<PROCESS> elements	3
<PARAM> elements	3
<BREAKPOINT> elements	4
<b>Adding New Processing</b>	<b>4</b>
<b>linkEngine Reference</b>	<b>5</b>

## Installation

*linkEngine* is a single executable file that is supplied as an exe for Windows 32-bit and 64-bit:

- Software/1-linkEngine/Win32
- Software/1-linkEngine/Win64

These folders can be placed anywhere on a Windows computer, depending on the platform.

## Compilation

The source code for linkEngine is supplied in the folder:

- Software/1-linkEngine/linkEngine

Should a recompilation be required, for example, on a different platform (Mac, Linux), follow the steps outlined below:

## Prerequisites

- Cmake

- C++ compiler (C++11)
- libsndfile

## Building from source

- 1) Inside the source folder, make a folder called 'build'
- 2) Inside the build folder, run:  
`cmake ..`
- 3) On Linux or Mac, where 'make' is utilised, run:  
`make`

## Running linkEngine

*linkEngine* is a command line program. As such, it must be run from relevant command prompt for the operating system.

From within the executable folder, run the following, providing a path to a CHAIN XML file as a command-line argument:

```
./linkEngine.exe demo-chain.xml
```

## Creating CHAIN specifications

*linkEngine* utilises CHAIN specification files to outline the desired processing. An example CHAIN file is contained within the Win32 and Win64 folders alongside the executable. To run correctly a sound file named 'in.wav' should be placed inside the folder to be processed. Alternatively, the path specified in the `CHAIN` element can be modified, as outlined in the next section.

A simple example is shown below:

```
<LINK>
  <CHAIN inFile="in.wav" outFile="out.wav">
    <PROCESS name="Gain">
      <PARAM type="breakpoint" name="gain">
        <BREAKPOINT time="0" value="1" />
        <BREAKPOINT time="15" value="0.5" />
      </PARAM>
    </PROCESS>
  </CHAIN>
</LINK>
```

---

## <LINK> elements

The root XML element must always be `LINK` and the next element should be a `CHAIN` as shown above.

## <CHAIN> elements

Each `CHAIN` must have an `inFile` and `outFile` specified, using either a relative path based on the location of the executable, or a full system path. There can be multiple `CHAIN` elements within a single `LINK` element, with separate `inFile` and `outFiles`.

## <PROCESS> elements

Each `CHAIN` can have as many `PROCESS` elements as desired. A `PROCESS` must be given a name that corresponds to an available process within *linkEngine*, and each name begins with an upper-case letter. A list of `PROCESS`s and `PARAM`eters are given in the reference section below.

- Gain
- Delay
- Highpass
- Lowpass
- Normalise
- Split

## <PARAM> elements

Each `PROCESS` has addressable parameters. Each has default values, but specific values can be given using `PARAM` elements.

`PARAM` elements relate to the individual parameters within the `PROCESS`. The list of parameters for each `PROCESS` is given in the reference section below.

Each parameter has a `type` that corresponds to how it behaves:

- `static`
- `breakpoint`

A static parameter has a fixed value, whereas a breakpoint parameter can change over time. By default, all parameters are fixed at their default values. The following shows how to set a specific fixed value for a `PARAM` element:

```
<PARAM type="static" name="gain">0.5</PARAM>
```

## <BREAKPOINT> elements

`BREAKPOINT` elements are used to specify time-varying values for parameters. To take effect, the parameter must be set to `type="breakpoint"`.

When a parameter has the breakpoint type, it expects to contain a set of `BREAKPOINT` elements, each with a time and value attribute, as shown below:

```
<PARAM type="breakpoint" name="gain">
  <BREAKPOINT time="0" value="1" />
  <BREAKPOINT time="15" value="0.5" />
</PARAM>
```

A `PARAM` can have as many `BREAKPOINT` elements as desired. It is important to note that **time is specified in seconds**.

## Adding New Processing

A `PROCESS` in *linkEngine* corresponds to an *Effect* of the same name within *linkEngine*. It is possible to quickly develop new processes, following the API as outlined in the accompanying thesis. The API is based on that of FSOM (Free Sound Object Mixer), written by Stephen Pearse, David Devaney and Dave Moore (<https://github.com/spearse/fsom>), in the hope that they can easily share code. Once they are created, they must be added to the *EffectFactory* so that *linkEngine* can automatically create them when asked. The new code must also be registered with the Cmake build system in order to compile.

# linkEngine Reference

PROCESS name	PARAM name	PARAM limits	PARAM unit	Notes
Gain	gain	-100–100	gain	
Delay	delayTime	1–441,000	samples	delayTime in samples
	feedback	0–0.99	gain	
Normalise	max	0–1	gain	
Lowpass	freq	20–20,000	Hz	Fixed rolloff
Highpass	freq	20–20,000	Hz	Fixed rolloff
Split				Creates a new output file for each channel, using the naming scheme: <outFileName>-<channelNumber>.wav
Duration	time	0-26,460,000	seconds	Used to specify longer output file durations if required