Open Sound Control (OSC) for silentjack

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

The program **silentjack** can be used to listen to **JACK** ports and detect if the audio signal is below or above a defined level (dB) for a defined time interval. A fork of silentjack was made in order to sense and control the program via the UDP protocol **OSC**. The library **liblo** is used to implement OSC in silentjack.

silentjack OSC is a derivate of "silentjack - Silence / Dead Air Detector" http://www.aelius.com/njh/silentjack/

1 Prerequisites

1.1 JACK Audio Connection Kit

http://www.jackaudio.org

1.2 liblo - Lightweight OSC Implementation

```
git clone git://liblo.git.sourceforge.net/gitroot/liblo/liblo
```

See example programs `oscsend` and `oscdump`

A small modification will make the oscdump tool ready for shell hacks. https://gist.github.com/7890/8d83f887b9dc771dcb00

Many applications are possilbe without using the OSC features of silentjack simply by using the *run command on silence* feature.

However with the OSC modification, silentjack becomes even more powerful. Once started, it is a fully controllable network node that does inter-process communication (IPC) via OSC.

It provides a way of reconfiguring running instances of silentjack. As an example, the trigger level or silence period of a running silentjack instance could be configured by a scheduler. This allows to taylor sensing of a source to specific needs based on time (i.e. on a per-radio-show basis).

```
$ silentjack -h

silentjack version 0.3 OSC

Usage: silentjack [options] [COMMAND [ARG]...]
Options:  -c <port>   Connect to this port
          -n <name>   Name of this client (default 'silentjack')
          -l <db>     Trigger level (default -40 decibels)
          -p <secs>   Period of silence required (default 1 second)
          -d <db>     No-dynamic trigger level (default disabled)
          -P <secs>   No-dynamic period (default 10 seconds)
          -g <secs>   Grace period (default 0 seconds)
          -v          Enable verbose mode
          -q          Enable quiet mode
          -o <port>   Set OSC port for listening (default 7777, ? for random)
          -H <port>   Set OSC host to send to (default 127.0.0.1)
          -O <port>   Set OSC port to send to (default 7778)
          -X          Disable all OSC functions
          -h          Show help
```

2 OSC Settings

The following three settings define how OSC messages are processed.

- Port to start OSC Server and listen for incoming messages

- Host and port to send outgoing messages to (report to this address)

The default settings are

- Start OSC Server on port 7777
- Report to 127.0.0.1, port 7778

Example command line call with alternative OSC settings

```
$ silentjack -o 9999 -H 10.10.10.255 -O 10000
```

Note
The OSC *telling and answering* mechanism is different from other implementations

- Messages are not sent back to the host and port of the requester by default
- There is no pub/sub mechanism and therefore no subscribers list to serve

This allows the following scenarios

- A requesting process can be decoupled from the node that receives the answer
- Address of report host can be a broadcast address to serve all on subnet

There is a seperate `-V` (verbose) command line switch for OSC.

Most of the settings like silence/grace period, trigger level, verboseness can be read and set while the program is running via OSC.

3 Messages sent by silentjack

Important
for all patterns, a client name `silentjack` is used. The patterns will differ if silentjack is started for instance with `-n sj1`. An answer would look like `/sj1/...` in that case.

3.1 /silentjack/started

```
Message Pattern:
        /silentjack/started ssiifi

Parameters:
        s: jack_client_name
        s: osc_server_port
        i: silence_period       [sec]
        i: grace_period         [sec]
        f: trigger_level        [dB]
        i: is_verbose

Sent on event:
        silentjack ready for service after startup.

Sent if:
        not matter if verbose
```

```
Example:
        /silentjack/started ssiifi "silentjack" "7777" 5 0 -40.000000 1
```

3.2 /silentjack/not_connected

```
Message Pattern:
        /silentjack/not_connected ss

Parameters:
        s: jack_client_name
        s: osc_server_port

Sent on event:
        silentjack is not yet or not any longer connected to any jack source port.
        Message is repeated every second.

Sent if:
        verbose (`-V`)
Example:
        /silentjack/not_connected ss "silentjack" "7777"
```

3.3 /silentjack/connected

```
Message Pattern:
        /silentjack/connected ss

Parameters:
        s: jack_client_name
        s: osc_server_port

Sent on event:
        silentjack was just connected to a jack source port.
        Message is sent once on connect and is followed by `/silentjack/level` messages.

Sent if:
        verbose (`-V`)

Example:
        /silentjack/connected ss "silentjack" "7777"
```

3.4 /silentjack/level

```
Message Pattern:
        /silentjack/level ssiif

Parameters:
        s: jack_client_name
        s: osc_server_port
        i: is_above_threshold
        i: seconds_in_period    [sec]
        f: level

Sent on event:
        silentjack just made another evaluation of the audio signal.
        This happens once per second and only if connected and
```

```
        not in grace period.

Sent if:
        verbose ('-V')

Example:
        /silentjack/level ssiif "silentjack" "7777" 0 4 -8.267532
```

3.5 /silentjack/silent

```
Message Pattern:
        /silentjack/silent ssf

Parameters:
        s: jack_client_name
        s: osc_server_port
        f: level                [dB]

Sent on event:
        silentjack detected silence (max silence was reached).

Sent if:
        verbose ('-V')

Example:
        /silentjack/silent ssf "silentjack" "7777" -45.267532
```

3.6 /silentjack/run_cmd

```
Message Pattern:
        /silentjack/run_cmd sss(s*)

Parameters:
        s: jack_client_name
        s: osc_server_port
        s: command
        (s: param 1)
        (s: param n)

Sent on event:
        silentjack runs a command (after silence detected) if silentjack
        was started with a command argument (last option).

Sent if:
        no matter if verbose

Example:
        /silentjack/run_cmd sssss "silentjack" "7777" "/usr/local/bin/myscript.sh"
```

3.7 /silentjack/grace

```
Message Pattern:
        /silentjack/grace ssi

Parameters:
        s: jack_client_name
```

```
        s: osc_server_port
        i: seconds_in_grace_period      [sec]

Sent on event:
        silentjack is in grace period.
        The message is repeated until the period is over.

Sent if:
        verbose (‘-V‘)

Example:
        /silentjack/grace ssi "silentjack" "7777" 9
```

3.8 /silentjack/settings

```
Message Pattern:
        /silentjack/settings ssiifi

Parameters:
        s: jack_client_name
        s: osc_server_port
        i: silence_period       [sec]
        i: grace_period         [sec]
        f: trigger_level        [dB]
        i: is_verbose

Sent on event:
        silentjack was reconfigured or requested to tell the current configuration.

Sent if:
        no matter if verbose

Example:
        /silentjack/settings ssiifi "silentjack" "7777" 5 10 -40.000000 1
```

3.9 /silentjack/quit

```
Message Pattern:
        /silentjack/quit ss

Parameters:
        s: jack_client_name
        s: osc_server_port

Sent on event:
        silentjack is about to quit.
        This is normally caused by CTRL+C or sending the message ‘/silentjack/quit‘.

Sent if:
        no matter if verbose

Example:
        /silentjack/quit ss "silentjack" "7777"
```

4 Messages received by silentjack

Important
all get/set methods have a free-defined string as an additional argument (`req_id`). The string will be repeated in the requested answer from silentjack.

4.1 /silentjack/get_settings

```
Message Pattern:
        /silentjack/get_settings

Parmeters:
        s: req_id

Action:
        request silentjack to tell current settings.

Reply:
        /silentjack/settings ssiifis

        s: jack_client_name
        s: osc_server_port
        i: silence_period       [sec]
        i: grace_period         [sec]
        f: trigger_level        [dB]
        i: is_verbose
        s: req_id
```

4.2 /silentjack/set_trigger_level

```
Message Pattern:
        /silentjack/set_trigger_level f

Parameters:
        f: trigger_level        [db]
        s: req_id

Action:
        request silentjack to reconfigure trigger level.

Reply:
        /silentjack/settings ssiifis
```

4.3 /silentjack/set_silence_period

```
Message Pattern:
        /silentjack/set_silence_period i

Parameters:
        i: silence_period       [sec]
        s: req_id

Action:
```

```
        request silentjack to reconfigure silence period.

Reply:
        /silentjack/settings ssiifis
```

4.4 /silentjack/set_grace_period

```
Message Pattern:
        /silentjack/set_grace_period i

Parameters:
        i: grace_period         [sec]
        s: req_id

Action:
        request silentjack to reconfigure grace period.

Reply:
        /silentjack/settings ssiifis
```

4.5 /silentjack/set_verbose

```
Message Pattern:
        /silentjack/verbose i

Parameters:
        i: is_verbose
        s: req_id

Action:
        request silentjack to be or not to be verbose OSC-wise (0/1).

Reply:
        /silentjack/settings ssiifis
```

4.6 /silentjack/quit

```
Message Pattern:
        /silentjack/quit

No Parameters

Action:
        request silentjack to quit.

Reply:
        /silentjack/quit ss

        s: jack_client_name
        s: osc_server_port
```

5 Test it

```
$ silentjack -n "stream_silence_listener" -l -50 -p 30 -g 60 -V -c "mpg123-6153:left"
```

This will start an instance of silentjack in OSC verbose mode (`-V`) named `stream_silence_listener` (`-n`) and connect it to output port of mpg123 (`-c`) or any other JACK source of your choice. Since no specific arguments for OSC are given, the standard values are used: silentjack will receive messages on port 7777, and will report any events or answers to requests to localhost, port 7778.

If the audio level falls below `-50` dB (`-l`) for at least `30` seconds (`-p`). When silence is detected, evaluation will start again after a grace period of `60` seconds (`-g`).

The `oscdump` program is an example client of liblo and can be found there. It will dump all messages sent by silentjack.

```
$ oscdump 7778
```

Messages to control a running silentjack can be sent for instance with `oscsend`.

```
$ oscsend localhost 7777 /stream_silence_listener/set_trigger_level fs -10 "foo"
```

5.1 Application Notes

In order to detect a faulty audio signal in means of a missing left or right channel, use two instances of silentjack, one per channel.

silentjack can be used for any audio routing logic that depends on available / not available signals in an automatic way, for instance to drive fade-in / fade-out by precedence rules over an array of inputs.