

**NAME**

`jack_audio_send` – JACK client to transmit audio data in a LAN

**SYNOPSIS**

**jack\_audio\_send** [OPTIONS] target\_host target\_port

**DESCRIPTION**

*jack\_audio\_send* & *jack\_audio\_receive* are JACK clients allowing to transmit uncompressed native JACK 32 bit float audio data on the network using UDP OSC messages.

All involved JACK servers must share the same sampling rate but can run at different period sizes. NO resampling is involved. Diverging clocks of distributed audio interfaces can pose an issue.

- The main purpose is to transmit audio in one direction from host a to host(s) b(, c, ...)
- Sender and receiver are separate programs
- There is no such thing as master / slave
- The programs run as a regular JACK clients
- There is no resampling done at either the sender or receiver side

The term "mc" period refers to multichannel period which contains the period of each channel for a given JACK cycle.

`audio_rxtx` uses `liblo` >= 0.27

**THIS PROGRAM COMES WITHOUT ANY WARRANTY**

`jack_audio_send` version is 0.71

**OPTIONS**

- help** (w/o argument)  
Display help and quit
- version** (w/o argument)  
Display version and quit
- loinfo** (w/o argument)  
Display `liblo` props and quit
- lport** (integer)  
Local port. Default: 9990
- in** (integer)  
Number of input/capture channels. Up to 256 channels are possible. Use small JACK period sizes with high channel counts. Also Gigabit networking might be required. Default: 2
- connect** (w/o argument)  
Autoconnect ports. This will connect all physical input/capture ports to `jack_audio_send`. Default: off
- name** (string)  
Jack client name. Default: `jack_audio_send`
- sname** (string)  
Jack server name. Default: `default //:`
- update** (integer)  
Update info displayed in terminal every nth JACK cycle. Default: 99
- limit** (integer)  
Limit totally sent messages. This is useful for tests. Send an exact amount of messages, then quit program. Default: off

**--nopause** (w/o argument)  
 Immediate send, ignore /pause, /deny. Use with multiple receivers. Default: off

**--drop** (integer)  
 Drop (don't send) every nth message (for test purposes)

**target\_host** (string)  
 A valid target (receiver) hostname or IP address. Broadcast IP addresses should work too (i.e. 10.10.10.255)

**target\_port** (integer)  
 Port to send audio to. If target\_host is a broadcast address, target\_port could be interpreted as target\_bus.

## EXAMPLES

Send 4 channels to port 1234 on host 10.10.10.3:

```
$ jack_audio_send --in 4 10.10.10.110 1234
```

Example output of jack\_audio\_send:

```
sending from osc port: 9990
target host:port: 10.10.10.110:1234
sample rate: 44100
bytes per sample: 4
period size: 128 samples (2.902 ms, 512 bytes)
channels (capture): 4
immediate send, no pause or shutdown: no
multi-channel period size: 2048 bytes
message rate: 344.5 messages/s
message length: 2112 bytes
transfer length: 2188 bytes (6.4 % overhead)
expected network data rate: 6030.7 kbit/s (0.75 mb/s)
```

```
# 65142 (00:03:09) xruns: 0 tx: 142530696 bytes (142.53 mb) p: 0.0
```

Legend:

- #: sequential message number since start of program
- (HH:MM:SS): elapsed time corresponding to message number
- xruns: local xrun counter
- tx: calculated network traffic sum
- p: how much of the available process cycle time was used to do the work (1=100%)

Send 16 channels to subnet broadcast address 10.10.10.255, "bus" 1234:

```
$ jack_audio_send --in 16 --nopause 10.10.10.255 1234
```

jack\_audio\_send has no buffer. A message with all channels as blobs is sent in every JACK cycle.

## ERROR MESSAGES

jack\_audio\_send does not automatically start a JACK default server if there is none running. This will lead to the following message:

```
Cannot connect to server socket err = No such file or directory Cannot connect to server request channel
jack server is not running or cannot be started jack_client_open() failed, status = 0x11 Unable to connect to
```

JACK server

Simply start JACK before using jack\_audio\_send

## PROGRAM STATUSES

jack\_audio\_send statuses:

- 0) initializing, starting up with given parameters
- 1) offering audio to given host
- 2) received **/deny** transmission (if offered audio was incompatible)
- quit

OR

- 3) received **/accept** transmission (if offered audio was compatible)
- 4) sending **/audio** to receiver (one message = one multi-channel period)
- 5) received **/pause** transmission
- offering again

jack\_audio\_send statuses with option **--nopause**:

- 0) initializing, starting up with given parameters
- 1) sending **/audio** to receiver (one message = one multi-channel period)

## OSC FORMAT VERSION 1.0

The OSC messages that are sent by jack\_audio\_send are defined as follows:

**/offer fiiiih**

- 1) f: audio rx/tx format version
- 2) i: sampling rate
- 3) i: bytes per sample
- 4) i: period size
- 5) i: channel count
- 6) f: expected network data rate
- 7) h: send / request counter

**/audio hhtib\***

- 1) h: message number
- 2) h: xrun counter
- 3) t: timetag (seconds since Jan 1st 1900 in the UTC, fraction  $1/2^{32}$ nds of a second)
- 4) i: sampling rate
- 5) b: blob of channel 1 (period size \* bytes per sample) bytes long
- ...
- 68) b: up to 64 channels

All properties refer to the sending host.

The OSC messages that are understood by `jack_audio_send` are defined as follows:

- `/accept`
- `/deny fi`
- `/pause`

Please also see manpage of `jack_audio_receive`. The liblo tool programs *oscdump* and *oscsend* should also be mentioned here.

## RESOURCES

Github: [https://github.com/7890/jack\\_tools](https://github.com/7890/jack_tools)

## BUGS

Please report any bugs as issues to the github repository. Patches and pull requests are welcome.

## SEE ALSO

**`jack_audio_receive(1)`** **`jackd(1)`** **`jack_netsource(1)`** **`jacktrip(1)`** **`zita-njbridge(1)`**

## AUTHORS

Thomas Brand <tom@trellis.ch>

## COPYING

Copyright (C) 2013 – 2014 Thomas Brand. Free use of this software is granted under the terms of the GNU General Public License (GPL).