# socat Bidirectional byte relay

**START HERE**

`socat -` **TCP:**_<host>_**:**_<port>_
Talk to a TCP port

`socat` **UNIX-LISTEN:**_<path>_**,fork -**
Listen on a Unix socket

`socat` **TCP-LISTEN:**_<port>_ **-**
Accept one connection

`socat - UNIX-` **CONNECT:**_<path>_
Connect to a Unix socket

`socat` **TCP-LISTEN:**_<port>_**,fork,reuseaddr -**
Serve many clients

`socat -u TCP-` **LISTEN:**_<port>_ **CREATE:**_<file>_
Receive a file

`socat` **TCP-LISTEN:**_<lport>_**,fork TCP:**_<host>_**:**_<port>_
Forward a port

`socat -d -d` _<addr1>_ _<addr2>_
See what it does

**Address 1** TYPE:args,options → both ways → **socat** relay loop → both ways → **Address 2** TYPE:args,options

## CONCEPTS

socat <addr1> <addr2> opens two endpoints and copies bytes both ways between them. Everything else is choosing the two addresses.

An address is TYPE:arg:arg,option,option=value. Options follow the commas; the type name is case-insensitive.

The first address that fails or hits EOF ends the relay. Add fork on a listener to keep serving, and shut-down or -t if the peer needs EOF.

Quote addresses that contain commas, spaces or shell characters. Use socat -V to see which address types your build supports.

## BASICS

`socat` _<addr1>_ _<addr2>_
Relay between two addresses

`socat -` _<addr>_
Connect stdin/stdout to an address

`socat -u` _<addr1>_ _<addr2>_
One way only: first read, second write

`socat -d -d` _<addr1>_ _<addr2>_
Log more: -d notices, -d -d info

`socat -v` _<addr1>_ _<addr2>_
Dump transferred text to stderr

`socat -x` _<addr1>_ _<addr2>_
Dump transferred data as hex to stderr

`socat -T` _<secs>_ _<addr1>_ _<addr2>_
Quit after this much inactivity

`socat -V`
Print version and built-in features

## TCP AND UDP

`socat -` **TCP:**_<host>_**:**_<port>_
Connect to a TCP port

`socat` **TCP-LISTEN:**_<port>_ **-**
Listen for one TCP connection

`socat` **TCP-LISTEN:**_<port>_**,fork,reuseaddr -**
Serve each client in a child process

## TCP AND UDP (cont.)

`socat` **TCP-LISTEN:**_<port>_**,bind=**_<ip>_ **-**
Listen on one local address only

`socat` **TCP-LISTEN:**_<port>_**,range=**_<cidr>_ **-**
Accept only peers in a network

`socat - TCP4:`_<host>_**:**_<port>_
Force IPv4 (TCP6 forces IPv6)

`socat` **TCP:**_<host>_**:**_<port>_**,connect-timeout=**_<s>_ **-**
Give up connecting after a while

`socat - UDP:`_<host>_**:**_<port>_
Send and receive UDP datagrams

`socat` **UDP-LISTEN:**_<port>_**,fork -**
Receive UDP datagrams, one peer per child

## UNIX SOCKETS AND FILES

`socat` **UNIX-LISTEN:**_<path>_**,fork -**
Listen on a Unix socket

`socat - UNIX-` **CONNECT:**_<path>_
Connect to a Unix socket

`socat` **UNIX-LISTEN:**_<path>_**,mode=**_<mode>_ **-**
Set permissions of the socket file

`socat` **UNIX-LISTEN:**_<path>_**,unlink-early -**
Remove a stale socket file first

`socat - GOPEN:`_<file>_
Open a file or socket, whichever it is

`socat - OPEN:`_<file>_
Read an existing file

`socat -u - CREATE:`_<file>_
Write stdin to a new file

`socat -u -` **OPEN:**_<file>_**,creat,append**
Append stdin to a file

## FORWARD AND PROXY

`socat` **TCP-LISTEN:**_<lport>_**,fork,reuseaddr TCP:**_<host>_**:**_<port>_
Forward a local port to a remote one

## FORWARD AND PROXY (cont.)

`socat` **TCP-LISTEN:**_<lport>_**,fork,bind=**_<ip>_ **TCP:**_<host>_**:**_<port>_
Forward, listening on one address only

`socat` **UNIX-LISTEN:**_<path>_**,fork TCP:**_<host>_**:**_<port>_
Expose a TCP service as a Unix socket

`socat` **TCP-LISTEN:**_<port>_**,fork UNIX-CONNECT:**_<path>_
Expose a Unix socket over TCP

`socat` **TCP-LISTEN:**_<lport>_**, fork PROXY:**_<proxy>_**:**_<host>_**:**_<port>_
Connect through an HTTP proxy

`socat` **TCP-LISTEN:**_<lport>_**, fork SOCKS4A:**_<socks>_**:**_<host>_**:**_<port>_
Connect through a SOCKS4A proxy

`socat -v TCP-` **LISTEN:**_<lport>_**,fork TCP:**_<host>_**:**_<port>_
Forward and print the traffic

## TLS

`socat -` **OPENSSL:**_<host>_**:**_<port>_
Connect over TLS (verifies the server)

`socat -` **OPENSSL:**_<host>_**:**_<port>_**, verify=0**
Connect over TLS, skip verification

`socat -` **OPENSSL:**_<host>_**:**_<p>_**,cafile=**_<c>_
Verify against a given CA file

`socat OPENSSL-LISTEN:`_<p>_**, cert=**_<f>_**,verify=0 -**
TLS server, no client cert needed

*Without verify=0 the server demands a client certificate. <f> must hold key and cert, or add key=<key>.*

`socat OPENSSL-LISTEN:`_<p>_**, cert=**_<f>_**,cafile=**_<c>_ **-**
TLS server that requires client certs

`socat TCP-` **LISTEN:**_<lport>_**,fork OPENSSL:**_<host>_**:**_<port>_
Add TLS in front of a plain client

## TLS (cont.)

`socat OPENSSL-LISTEN:`_<p>_**, cert=**_<f>_**,verify=0 TCP:**_<host>_**:**_<port>_
Terminate TLS for a plain server

## PROGRAMS AND TERMINALS

`socat TCP-` **LISTEN:**_<port>_**,fork EXEC:**_<prog>_
Run a program per connection

`socat TCP-` **LISTEN:**_<port>_**,fork SYSTEM:'**_<shell cmd>_**'**
Run a shell command per connection

`socat -` **EXEC:**_<prog>_**,pty,stderr**
Run a program on a pty with stderr merged

`socat -,raw,echo=0 EXEC:'bash -i',pty,stderr,setsid`
Full interactive shell over a pty

`socat -,raw,echo=0` _<addr>_
Raw local terminal, no echo

`socat PTY,link=`_<path>_`,raw,echo=0` _<addr>_
Create a pty device linked at a path

## RECIPES

### Send a file between hosts

```
# receiver
socat -u TCP-LISTEN:5000
CREATE:in.tar
# sender
socat -u OPEN:in.tar
TCP:<host>:5000
```

### Watch traffic through a forward

```
socat -v TCP-LISTEN:8080,fork \
  TCP:example.com:80
# other shell
curl localhost:8080/
```

### Spy on a Unix socket

```
socat -v UNIX-LISTEN:s.sock,\
  mode=600,fork \
  UNIX-CONNECT:/run/docker.sock
# then, in another shell
curl --unix-socket s.sock \
  http://localhost/version
```

### Quick TLS test server

```
openssl req -x509 -newkey \
  rsa:2048 -nodes -subj \
  /CN=localhost \
  -keyout k.pem -out c.pem
cat k.pem c.pem > both.pem
socat OPENSSL-LISTEN:4433,\
  cert=both.pem,verify=0,fork -
# other shell
socat - OPENSSL:localhost:4433,
\
  verify=0
```