Socat Cheatsheet

Socat Cheatsheet

A practical quick-reference guide for `socat` (SOcket CAT) — a powerful bidirectional data relay tool. Socat establishes two bidirectional byte streams and transfers data between them. Each data channel can be a TCP/UDP socket, UNIX socket, file, PTY, device, SSL connection, or exec subprocess.

General Syntax

Socat takes two address specifications and connects them:

```
# Basic structure
socat [options] <address1> <address2>

# Common options
socat -d -d <address1> <address2>    # Verbose (up to -d -d -d -d)
socat -v <address1> <address2>       # Verbose data dump
socat -s <address1> <address2>       # Stay in foreground (stderr)
socat -t <timeout> <addr1> <addr2>   # Timeout for half-close
socat -T <timeout> <addr1> <addr2>   # Total activity timeout
socat -b <size> <addr1> <addr2>      # Set buffer size
socat -lf <file> <addr1> <addr2>     # Log to file
socat -lu <addr1> <addr2>            # Log with microseconds
socat -lm <addr1> <addr2>            # Log with milliseconds
```

Address Format

```
PROTOCOL:host:port
PROTOCOL-LISTEN:port
PROTOCOL:path
PROTOCOL:fd
```

TCP Connections

TCP Client

```
# Connect to a TCP server
socat - TCP:example.com:80

# Connect and send HTTP request manually
echo -e "GET / HTTP/1.1\r\nHost: example.com\r\n\r\n" | socat - TCP:example.com:80

# Connect with source port and interface
socat - TCP:example.com:80,bind=192.168.1.100,sourceport=20000

# Connect with connection timeout
socat - TCP:example.com:80,connect-timeout=5

# Connect with keepalive
socat - TCP:example.com:80,keepalive,keepidle=30,keepintvl=10,keepcnt=5

# Connect and fork for each connection (requires listen side)
socat TCP:example.com:80 -
```

TCP Listener (Server)

```
# Listen on a port (single connection)
socat TCP-LISTEN:8080 -

# Listen on a specific address
socat TCP-LISTEN:8080,bind=192.168.1.100 -

# Listen and accept multiple connections (fork)
socat TCP-LISTEN:8080,fork -

# Listen with backlog
socat TCP-LISTEN:8080,backlog=100 -

# Listen with a specific interface
socat TCP-LISTEN:8080,bind=0.0.0.0,reuseaddr -

# Listen with max children
socat TCP-LISTEN:8080,fork,max-children=50 -

# Listen and exec a script on each connection
socat TCP-LISTEN:8080,fork EXEC:./handler.sh

# Listen with TCP wrappers (hosts.allow/hosts.deny)
socat TCP-LISTEN:8080,fork,tcpwrap=service_name -
```

TCP Port Forwarding

```
# Forward local port 8080 to remote host:80
socat TCP-LISTEN:8080,fork TCP:remotehost:80

# Forward with specific source address
socat TCP-LISTEN:8080,fork TCP:remotehost:80,bind=192.168.1.100

# Forward a range of ports
socat TCP-LISTEN:5000,fork TCP:localhost:5000 &
socat TCP-LISTEN:5001,fork TCP:localhost:5001 &
```

```
# Reverse forwarding (remote → local)
# On remote: socat TCP-LISTEN:9090,fork TCP:localhost:8080
```

UDP Connections

UDP Client

```
# Send data via UDP
echo "hello" | socat - UDP:example.com:53

# UDP client with source port
socat - UDP:example.com:53,bind=:20000

# UDP multicast
socat - UDP:224.0.0.1:1234
```

UDP Listener

```
# Listen for UDP packets
socat UDP-LISTEN:5353 -

# Listen on specific address
socat UDP-LISTEN:5353,bind=192.168.1.100 -

# Listen with reuseaddr
socat UDP-RECVFROM:5353,fork -
```

UDP Port Forwarding

```
# Forward UDP port 5353 to remote
socat UDP-LISTEN:5353,fork UDP:remotehost:5353

# UDP broadcast relay
socat UDP-LISTEN:8888,fork UDP-DATAGRAM:255.255.255.255:9999,broadcast
```

UNIX Sockets

UNIX Stream Client

```
# Connect to a UNIX stream socket
socat - UNIX-CONNECT:/var/run/docker.sock

# Query Docker API via UNIX socket
echo -e "GET /containers/json HTTP/1.1\r\nHost: localhost\r\n\r\n" | socat - UNIX-CONNECT:/va

# Connect with custom socket path
socat - UNIX-CONNECT:/tmp/mysocket.sock
```

UNIX Stream Listener

```
# Listen on a UNIX stream socket
socat UNIX-LISTEN:/tmp/mysocket.sock,fork -

# Listen and remove existing socket file first
socat UNIX-LISTEN:/tmp/mysocket.sock,unlink-early,fork -

# Listen with specific permissions
socat UNIX-LISTEN:/tmp/mysocket.sock,mode=666,fork -
```

UNIX Datagram

```
# UNIX datagram client
socat - UNIX-DATAGRAM:/tmp/mysocket.sock

# UNIX datagram listener
socat UNIX-RECVFROM:/tmp/mysocket.sock,fork -
```

UNIX Socket Forwarding

```
# Forward UNIX socket to TCP (expose UNIX socket over network)
socat TCP-LISTEN:8080,fork UNIX-CONNECT:/var/run/docker.sock

# Forward TCP to UNIX socket
socat UNIX-LISTEN:/tmp/proxy.sock,fork TCP:example.com:80
```

SSL/TLS

SSL Client

```
# Connect to an SSL/TLS server
socat - OPENSSL:example.com:443

# Connect with certificate and key
socat - OPENSSL:example.com:443,cert=client.pem,key=client.key

# Connect with CA certificate for verification
socat - OPENSSL:example.com:443,cafile=ca.pem

# Connect without certificate verification (testing only)
socat - OPENSSL:example.com:443,verify=0

# Connect with specific TLS version
socat - OPENSSL:example.com:443,method=TLS1.3
```

SSL Listener (Server)

```
# Create an SSL server
socat OPENSSL-LISTEN:8443,cert=server.pem,key=server.key,fork -
```

```
# SSL server with CA verification
socat OPENSSL-LISTEN:8443,cert=server.pem,key=server.key,cafile=ca.pem,verify=1,fork -

# SSL server with specific cipher
socat OPENSSL-LISTEN:8443,cert=server.pem,ciphers=HIGH:!aNULL:!MD5,fork -

# Self-signed certificate generation (one-liner)
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes
```

SSL Port Forwarding

```
# SSL termination proxy (expose TLS as plain TCP)
socat OPENSSL-LISTEN:8443,cert=server.pem,fork TCP:localhost:8080

# TLS to remote TLS (TLS passthrough)
socat OPENSSL-LISTEN:443,cert=server.pem,fork OPENSSL:remotehost:443,verify=0

# Plain TCP → remote TLS (TLS client wrapper)
socat TCP-LISTEN:8080,fork OPENSSL:remotehost:443,verify=0
```

File and Device Operations

File I/O

```
# Use a file as data source or sink
socat FILE:input.txt FILE:output.txt

# Read from file and send to TCP
socat FILE:data.bin TCP:example.com:80

# Receive data and write to file
socat TCP-LISTEN:8080 OPEN:received.txt,creat,append

# Read from file descriptor
socat FD:3 TCP:example.com:80
```

TTY and PTY

```
# Create a pseudo-terminal
socat - PTY

# PTY with specific link
socat - PTY,link=/tmp/virtual-tty

# Raw terminal mode
socat - PTY,raw,echo=0

# Connect serial device to TCP
socat TCP-LISTEN:8080,fork FILE:/dev/ttyUSB0,nonblock,b115200,raw

# Serial to TCP (with hardware flow control)
socat TCP-LISTEN:7001,fork,reuseaddr FILE:/dev/ttyAMA0,b115200,crtscts=1,raw
```

```
# Socat as virtual modem
socat PTY,link=/tmp/modem,raw,echo=0,waitslave TCP-LISTEN:7000,reuseaddr
```

Devices

```
# Read from a device
socat - /dev/input/mouse0

# Write to device
socat FILE:input.txt /dev/usb/lp0

# Capture audio from ALSA
socat - ALSA:default
```

Exec and Program Integration

Running Commands

```
# Connect a socket to a command's stdin/stdout
socat TCP-LISTEN:8080,fork EXEC:/bin/bash

# Execute command with PTY
socat TCP-LISTEN:8080,fork EXEC:/bin/bash,pty,stderr

# Execute with environment variable
socat TCP-LISTEN:8080,fork EXEC:"env TERM=xterm /bin/bash",pty,stderr

# Execute a script per connection
socat TCP-LISTEN:8080,fork EXEC:./handle_connection.sh

# Execute with setsid (new session)
socat TCP-LISTEN:8080,fork EXEC:/bin/bash,setsid

# Execute and close stderr
socat TCP-LISTEN:8080,fork EXEC:/bin/bash,nofork 2>/dev/null
```

Systemd Socket Activation

```
# Accept socket passed by systemd
socat FD:3 TCP:example.com:80

# Socat as systemd service helper
# [Service]
# ExecStart=/usr/bin/socat FD:3 TCP:remote:80
```

Proxy Support

SOCKS4/5 Proxy

```
# Connect through a SOCKS4 proxy
socat - SOCKS4:socks-proxy:example.com:80,socksport=1080

# Connect through a SOCKS5 proxy
socat - SOCKS5:socks-proxy:example.com:80,socksport=1080

# SOCKS5 with username/password
socat - SOCKS5:socks-proxy:example.com:80,socksport=1080,socksuser=user,sockspass=pass

# HTTPS via SOCKS proxy
socat - SOCKS5:localhost:example.com:443,socksport=1080
```

HTTP Proxy (CONNECT)

```
# Connect through an HTTP CONNECT proxy
socat - PROXY:proxy.example.com:example.com:80,proxyport=8080

# HTTP proxy with credentials
socat - PROXY:proxy.example.com:example.com:80,proxyport=8080,proxyauth=user:pass

# HTTPS via HTTP proxy
socat - PROXY:proxy.example.com:example.com:443,proxyport=8080
```

Practical Patterns

Port Knocking

```
# Simple port knock sequence
echo "" | socat - TCP:server:7000,connect-timeout=1
echo "" | socat - TCP:server:7001,connect-timeout=1
echo "" | socat - TCP:server:7002,connect-timeout=1
# Port 22 should now be open
socat - TCP:server:22
```

HTTP Proxy Server (Minimal)

```
# Minimal HTTP forward proxy
socat TCP-LISTEN:3128,fork,reuseaddr TCP-LISTEN:3128,reuseaddr
# (This is a raw TCP relay, not a full HTTP proxy)
```

SSH via Socat (ProxyCommand)

```
# SSH through an HTTP proxy
ssh -o ProxyCommand="socat - PROXY:proxy:host:22,proxyport=8080" user@host

# SSH through SOCKS5
ssh -o ProxyCommand="socat - SOCKS5:localhost:%h:%2,socksport=1080" user@host

# SSH reverse shell via socat (server-side backdoor alert)
```

```
# Server: socat TCP-LISTEN:8080 EXEC:/bin/bash
# Client: socat TCP:server:8080 -
```

Network Stress Test

```
# Generate arbitrary TCP traffic
dd if=/dev/urandom bs=1M count=100 | socat - TCP:target:9999

# Measure throughput
time dd if=/dev/zero bs=1M count=1000 | socat - TCP:target:9999
```

Broadcast Relay

```
# Forward broadcast packets from eth0 to wlan0
socat UDP-RECVFROM:8888,broadcast,fork UDP-DATAGRAM:255.255.255.255:9999,broadcast,interface=
```

Named Pipe (FIFO)

```
# Create a named pipe listener over TCP
socat TCP-LISTEN:8080,fork PIPE:/tmp/mypipe

# TCP ↔ FIFO bidirectional
socat PIPE:/tmp/mypipe,rdonly TCP-LISTEN:8080
socat TCP-LISTEN:8081 PIPE:/tmp/mypipe,wronly
```

socat as a Daemon

```
# Run socat in background with pid file
socat -d -d -lf /var/log/socat.log TCP-LISTEN:8080,fork TCP:remote:80 &
echo $! > /var/run/socat.pid

# Using systemd service:
# [Unit]
# Description=Socat TCP Forwarder
# After=network.target
#
# [Service]
# Type=simple
# ExecStart=/usr/bin/socat -d -d TCP-LISTEN:8080,fork TCP:remote:80
# Restart=always
# RestartSec=5
#
# [Install]
# WantedBy=multi-user.target
```

Advanced Address Options

Address Option Reference

```
# --- TCP options ---
# connect-timeout=<s>    Connection timeout
# sourceport=<port>      Source port
# bind=<addr>           Bind to address
# keepalive             Enable TCP keepalive
# keepidle=<s>          Keepalive idle time
# keepintvl=<s>         Keepalive interval
# keepcnt=<n>           Keepalive count
# nodelay               Disable Nagle's algorithm
# mss=<bytes>           Maximum segment size
# backlog=<n>           Listen backlog
# reuseaddr             Allow reuse of address

# --- General options ---
# fork                  Fork for each connection
# max-children=<n>      Maximum child processes
# retry=<n>             Retry connection attempts
# intervall=<s>         Interval between retries
# forever               Keep retrying forever
# pf=<family>           Protocol family (ip4, ip6)

# --- SSL options ---
# cert=<file>           Certificate file
# key=<file>            Key file
# cafile=<file>         CA certificate
# capath=<dir>          CA directory
# verify=<0-4>          Verification level
# cipher=<list>         Cipher list
# method=<name>         TLS method

# --- UNIX socket options ---
# unlink-early          Remove existing socket before bind
# unlink-late           Remove socket after close
# mode=<perm>           Socket file permissions
# user=<name>           Socket file owner
# group=<name>          Socket file group
```

Fork vs No Fork

```
# Without fork: single connection, exits after client disconnects
socat TCP-LISTEN:8080 TCP:example.com:80

# With fork: handles multiple connections
socat TCP-LISTEN:8080,fork TCP:example.com:80

# No fork but wait for next connection after close
socat TCP-LISTEN:8080,reuseaddr TCP:example.com:80
```

Retry and Reconnect

```
# Keep trying to connect every 5 seconds
socat - TCP:server:80,retry=10,intervall=5

# Keep trying forever
socat - TCP:server:80,forever,intervall=10
```

```
# Keep retrying on the listen side
socat TCP-LISTEN:8080,forever,fork -
```

socat vs nc (netcat)

Feature	socat	nc (netcat)
SSL/TLS	Built-in (`OPENSSL`)	Separate (`ncat --ssl`)
UNIX sockets	Yes (`UNIX-CONNECT` , `UNIX-LISTEN`)	Basic (`-U`)
SOCKS proxy	Built-in	No
HTTP CONNECT proxy	Built-in (`PROXY:`)	No
PTY allocation	Yes (`PTY` , `pty`)	No
Fork per connection	Address option (`fork`)	Flag (`-k`)
Retry logic	Built-in (`retry=` , `forever`)	No
Complex address chaining	Yes (two addresses with options)	Limited

Troubleshooting

Debugging Connections

```
# Verbose mode with data dump
socat -d -d -v TCP-LISTEN:8080 -

# Even more verbose (debug level)
socat -d -d -d -d TCP-LISTEN:8080 -

# Check if socat is listening
ss -tlnp | grep socat

# Test with a simple echo
socat TCP-LISTEN:8080,fork EXEC:"cat"
# Then: echo "hello" | nc localhost 8080
```

Common Issues

```
# "Address already in use"
# -> Use reuseaddr option
socat TCP-LISTEN:8080,reuseaddr -

# "Connection refused"
# -> Check server is running and reachable
# -> Test with: nc -zv server port

# "Socket operation on non-socket"
# -> Check address format (TCP: vs TCP-LISTEN:)

# "Address family not supported"
# -> Force IPv4 or IPv6
```

```
socat - TCP4:example.com:80
socat - TCP6:example.com:80

# "Certificate verification failed"
# -> Check certificate path, set cafile, or use verify=0
socat - OPENSSL:example.com:443,verify=0

# Process hanging
# -> Set activity timeout
socat -T 30 TCP-LISTEN:8080,fork TCP:remote:80

# Too many open files
# -> Increase ulimit or limit max-children
socat TCP-LISTEN:8080,fork,max-children=100 TCP:remote:80
```

Test socat Is Working

```
# Echo server test
socat TCP-LISTEN:9999,fork EXEC:cat &
echo "hello world" | socat - TCP:localhost:9999
# Expected output: hello world
kill %1

# TCP proxy test
socat TCP-LISTEN:8080,fork TCP:example.com:80 &
curl -s http://localhost:8080 | head -5
kill %1
```

Best Practices

Production Forwarding

```
# Use systemd for supervision (auto-restart on failure)
# Use -d -d for minimal logging, -lf for log file
# Set -T timeout to prevent hung connections
# Limit max-children to prevent resource exhaustion
# Use reuseaddr for clean restarts

# Recommended production forwarder
socat -d -d -T 3600 -lf /var/log/socat-forward.log \
  TCP-LISTEN:8080,reuseaddr,fork,max-children=200,backlog=100 \
  TCP:upstream:80
```

Security Considerations

```
# Never expose socat's EXEC:/bin/bash without authentication
# Use tcpwrap for host-based access control
socat TCP-LISTEN:8080,fork,tcpwrap=socat-service EXEC:/bin/bash

# Restrict bind address (don't use 0.0.0.0 unless needed)
socat TCP-LISTEN:8080,bind=127.0.0.1,fork TCP:remote:80

# Use SSL for sensitive data
```

```
socat OPENSSL-LISTEN:8443,cert=server.pem,verify=1,fork OPENSSL:remote:443,verify=1

# Drop privileges after bind (run as non-root)
# Requires socat 1.8+ or use systemd with DynamicUser=yes

# Monitor open socat processes
watch -n 5 "ss -tlnp | grep socat"
```

Quick Reference

Address Type	Syntax	Purpose
TCP client	`TCP:host:port`	Connect to TCP server
TCP server	`TCP-LISTEN:port`	Listen for TCP connections
UDP client	`UDP:host:port`	Send/receive UDP datagrams
UDP server	`UDP-LISTEN:port`	Listen for UDP datagrams
UDP recv	`UDP-RECVFROM:port`	Receive UDP with reply
UDP datagram	`UDP-DATAGRAM:addr:port`	Raw UDP datagram I/O
UNIX stream client	`UNIX-CONNECT:path`	Connect to UNIX socket
UNIX stream server	`UNIX-LISTEN:path`	Listen on UNIX socket
UNIX datagram	`UNIX-DATAGRAM:path`	UNIX datagram I/O
SSL client	`OPENSSL:host:port`	Connect to TLS server
SSL server	`OPENSSL-LISTEN:port`	Listen for TLS connections
SOCKS4/5	`SOCKS4:proxy:host:port`	Connect via SOCKS proxy
HTTP proxy	`PROXY:proxy:host:port`	Connect via HTTP CONNECT
Execute	`EXEC:command`	Run a program
File	`FILE:path`	Read/write a file
PTY	`PTY`	Pseudo-terminal
PIPE	`PIPE:path`	Named pipe
FD	`FD:num`	File descriptor
STDIO	`-` or `STDIO`	Standard input/output

See Also

- Network Tools Cheatsheet — ip, tcpdump, dig, curl, ss, nc, iptables
- SSH Cheatsheet — SSH tunneling, ProxyCommand with socat
- Bash CLI Tools Cheatsheet — Text processing and CLI utilities