EDUMATCHER · ORDER ENTRY

The ALF Protocol

ALmost FIX — a small, readable order-entry protocol, and the gateway that speaks it

A beginner's guide



What this deck covers

ROADMAP

01

The gateway

Why exchanges put a translator in front of the matching engine

02

The protocol

What ALF is, what it is not, and how a line is built

03

Connecting

Session lifecycle, handshake, heartbeats and timeouts

04

Orders

Every order category you can send, and what comes back

05

Managing

Amending, cancelling, monitoring and error handling

06

Building

Writing your own client, plus the technical reference

PART 1

The Gateway

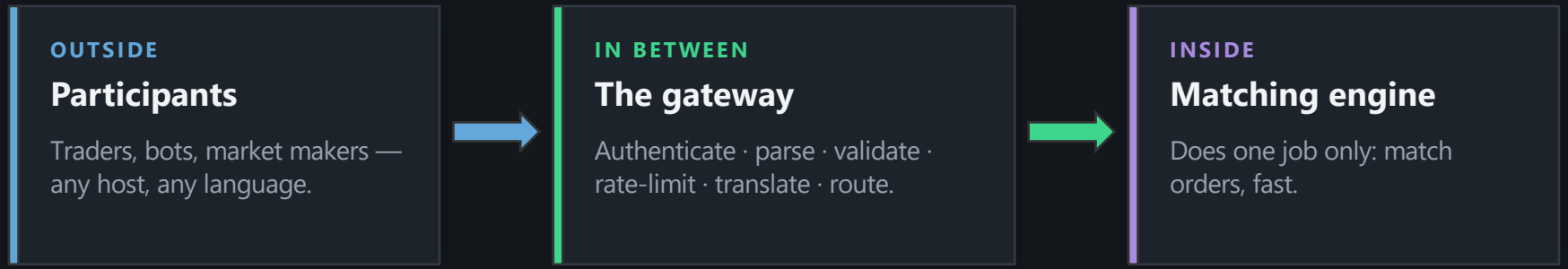
The front door of a trading system: who is allowed in, what language they must speak, and what gets forwarded to the engine.



What is a gateway?

CONCEPT

A gateway is the entry point through which outside participants send orders and receive answers. It translates their message format into the one the matching engine understands.



KEY IDEA

The engine never parses text and never manages sessions. Anything a client can do wrong is caught at the gateway, before it can slow matching down.

Why the gateway sits outside the engine

SEPARATION OF CONCERNS

Authenticate

Is this gateway ID on the allowlist? The engine is the authority; the gateway asks and enforces.

Parse & validate

Turn a text line into a structured request, or reject it defensively with a clear error code.

Protect

Rate limits, idle timeouts, outbound queue caps and error budgets keep one bad client from hurting others.

Translate & route

Convert commands into the engine's internal messages, then convert engine events back into protocol lines.

One more benefit: because format handling lives outside the engine, a venue can offer several protocols at once — an HFT firm on a binary feed and a retail broker on text can share the same engine.

Gateway protocols in the real world

CONTEXT

FIX 4.2 / 4.4 / 5.0	`Text (tag=value)`	Brokers, buy-side OMS	The lingua franca of institutional order routing
OUCH (Nasdaq)	`Binary`	HFT, prop traders	Ultra-low latency, fixed-length binary fields
ITCH (Nasdaq)	`Binary, market data`	Data consumers	One-way feed for order book reconstruction
FAST / SBE	`Binary`	Data consumers	Simple Binary Encoding — CME, Eurex market data
BOE (Cboe) · ETI (Eurex)	`Binary`	HFT, derivatives	Venue-specific binary order entry interfaces
REST / WebSocket	`Text / JSON`	Retail, algorithmic	Crypto venues and easy-to-integrate retail stacks

ALF borrows FIX's field=value idea — and deliberately stops there.

Where ALF fits in

INTERFACE FAMILY

EduMatcher exposes several interfaces. ALF is the order-entry one.

ALF	`pm-alf-console`	ZMQ direct, local only	Manual trading at the engine's own machine
ALF	`pm-alf-gwy`	TCP text :5565	External bots and remote scripts, any language
CALF	`pm-md-gwy`	TCP text	External market data: top of book, trades, state
RALF	`pm-ralf-gwy`	TCP text	Post-trade: clearing, drop copy, audit
REST / WS	`pm-api-gwy`	HTTP + WebSocket	Browsers and API-native applications
Internal	`ZMQ PUB / PULL`	ZMQ binary	Processes that import the package directly

Two ways to speak ALF

CLIENT CHOICE

Same command vocabulary, two very different front doors.

pm-alf-console

The interactive terminal

SITS ON THE ZMQ BUS

- Connects straight to the engine's ZMQ sockets (PUSH :5555, SUB :5556)

Has no TCP listener — it must run on the same machine as the engine

Reads stdin, prints to stdout: tab completion, history, live event display

Keeps local caches, so it can answer STATUS, POS and QLEGS by itself

pm-alf-gwy

The ALF TCP gateway

SPEAKS TCP/IP :5565

- Binds a TCP port and accepts many simultaneous connections

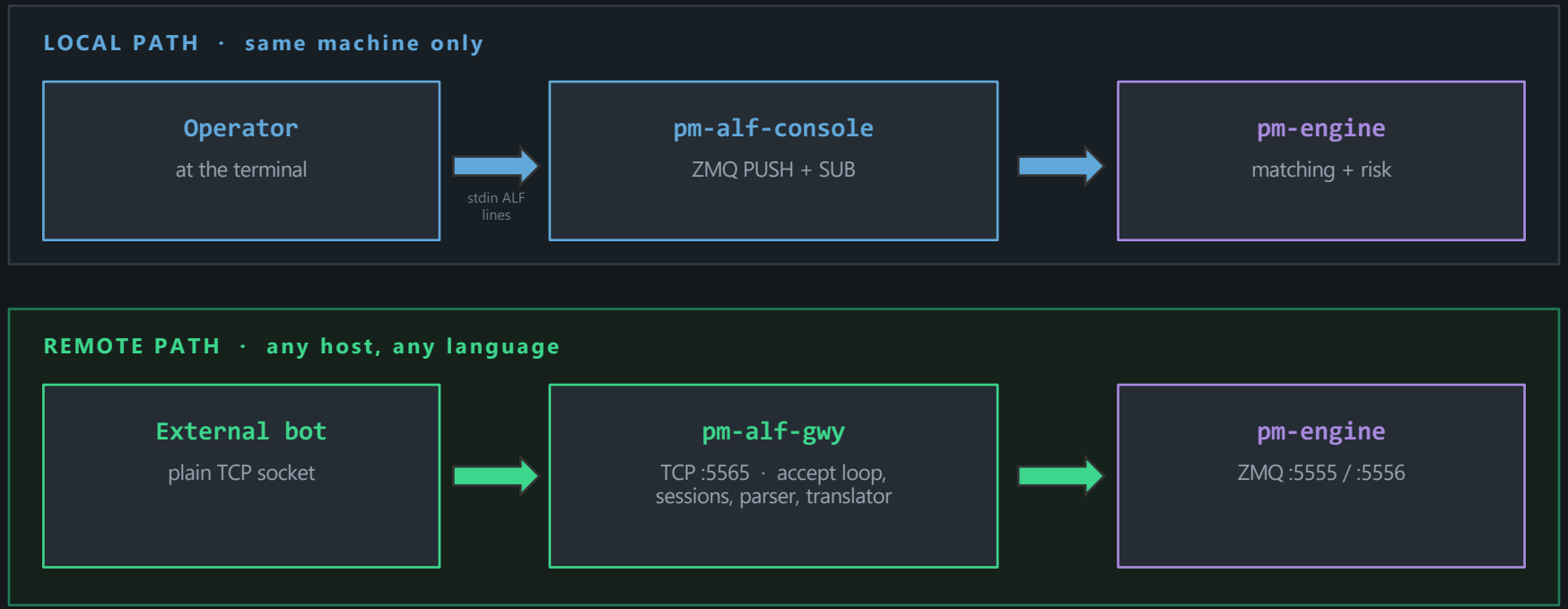
Reachable from any host, in any language — a plain socket is enough

Validates every line defensively, then translates it to the same engine messages

Built for bots and remote scripts, not for humans at a keyboard

The same engine, two paths in

ARCHITECTURE



The gateway simply adds a TCP accept loop in front of the same ZMQ bridge — so the client needs nothing but a socket.

Which one should I use?

DECISION GUIDE

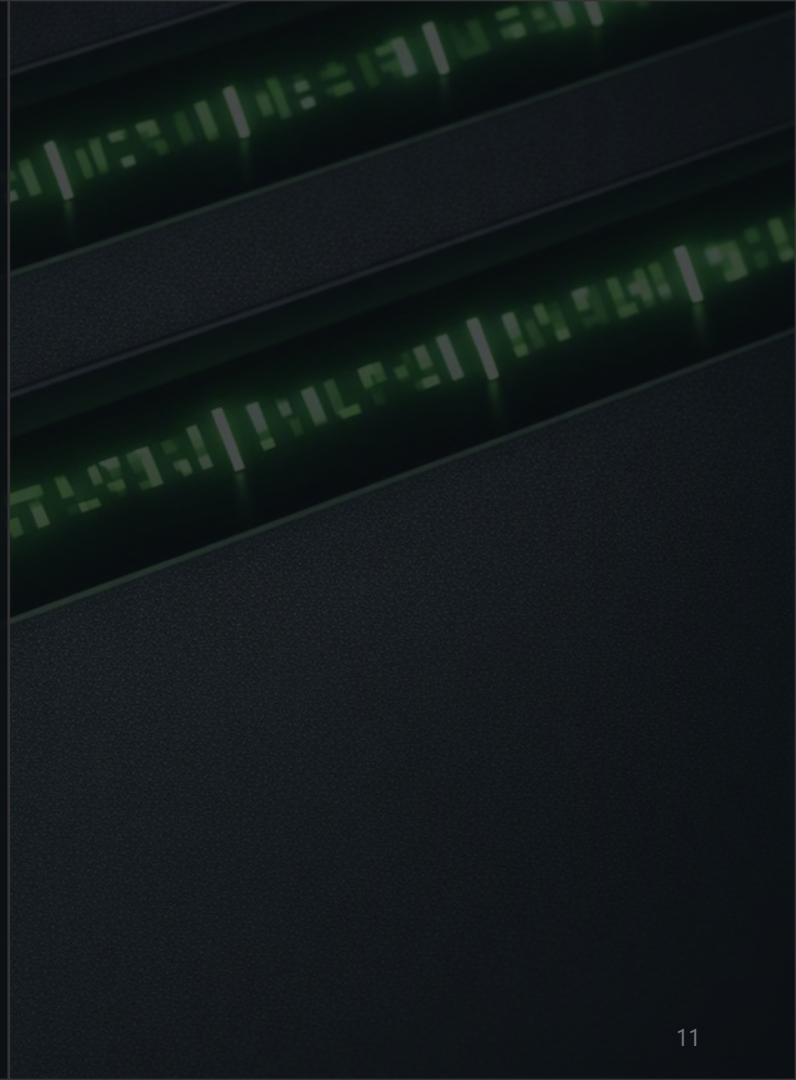
Human operator on the same machine	`pm-alf-console`	Tab completion, history, positions and P&L
A bot or script — same host or remote	`pm-alf-gwy`	Plain TCP, any language, many clients at once
Browser UI or REST-native stack	`pm-api-gwy`	HTTP and WebSocket
Read-only market data	`pm-md-gwy (CALF)`	One-way price and trade feed
Post-trade, clearing or audit	`pm-ralf-gwy (RALF)`	Drop copy and clearing records

Remember: the console cannot accept connections from other hosts. If the client is not on the engine's machine, you need the TCP gateway.

PART 2

The ALF Protocol

One line, one command. A pipe-delimited text language you can read, type by hand, and generate from any programming language.



ALF = ALmost FIX

DEFINITION

ALmost Fix

EduMatcher's compact text command protocol for entering orders. It borrows FIX's field=value idea — and nothing else.

ONE ORDER, ONE LINE

```
NEW|SYM=AAPL|SIDE=BUY
   |TYPE=LIMIT|QTY=100|PRICE=150.25
```

Small enough to type by hand. Simple enough to generate from a shell script.

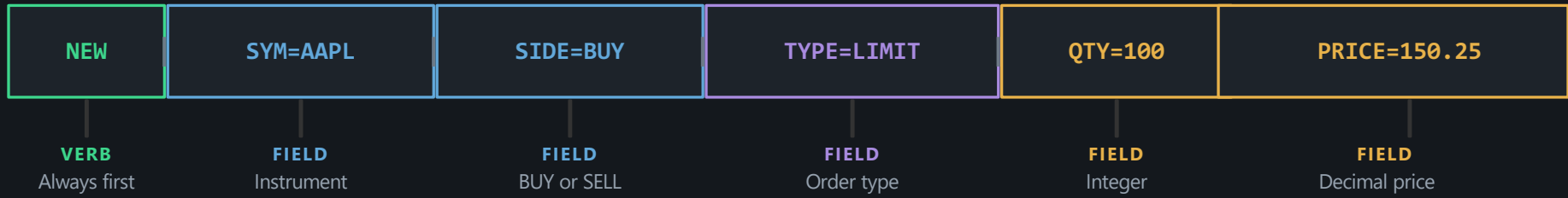
What ALF deliberately leaves out

- NO Numeric tags
- NO A FIX session layer
- NO Sequence numbers
- NO Checksums and body length
- NO Repeating-group rules
- NO Standard FIX message types

Anatomy of an ALF line

WIRE FORMAT

```
<COMMAND>|<FIELD>=<VALUE>|<FIELD>=<VALUE>| ... \n
```



The verb comes first

Everything after it is a pipe-delimited list of key=value segments.

The first = splits the pair

Anything after that first equals sign is the raw value text.

There is no escaping

Never put a pipe character inside a value — there is no way to quote it.

Rules the parser actually follows

WIRE FORMAT

Case is folded to UPPERCASE

Verbs, field names, enum values and symbols are all normalised. So are your own labels: OCO_ID=exit-aapl becomes EXIT-AAPL.

Field order is free

After the verb, order does not matter — the gateway builds a map, then validates what it finds. Only combo leg numbering must stay consistent.

Last duplicate wins

SYM=AAPL|SYM=MSFT parses as MSFT. A repeated key is not an error.

Unknown fields are ignored

Extra fields pass through harmlessly unless the command actually requires them.

Whitespace is only trimmed at the ends

The line is trimmed, individual fields are not. Never write spaces around your pipes.

Prices are decimals, quantities are integers

PRICE=150.25 and QTY=100. The gateway converts prices to internal ticks for you.

Three mistakes beginners make

GOTCHAS

Spaces around the pipes

Field-level whitespace is not trimmed, so the value becomes " AAPL" and validation fails.

```
NO  NEW | SYM=AAPL | SIDE=BUY
```

```
YES NEW|SYM=AAPL|SIDE=BUY
```

Assuming one read = one line

TCP is a byte stream. One read may hold half a line, or three lines. Always buffer.

```
NO  data = sock.recv(4096)
```

```
YES buffer, then split on \n
```

Using a shortened order ID

Displays abbreviate IDs for readability. AMEND and CANCEL need the full identifier.

```
NO  AMEND|ID=abc123|PRICE=151
```

```
YES AMEND|ID=<full id>|PRICE=151
```

The five command families

VOCABULARY

Trading commands Forwarded to the engine	`NEW` · `AMEND` · `CANCEL` · `QUOTE` · `QUOTE_CANCEL` · `KILL` · `SYMBOLS` · `ORDERS` · `QBOOT` · `SESSION`
Client-local information Answered by the client itself	`STATUS` · `POS` · `HELP`
Divergent between clients Local cache or engine round trip	`QLEGS`
Drop-copy relay control Handled inside the gateway process	`DC`
Session control Ends the connection	`EXIT` · `QUIT`

If you are writing a client that submits orders, the trading commands are the ones that matter.

Roles decide what you may send

ENTITLEMENTS

Every gateway ID is configured with a role. Sending a command your role is not entitled to returns ERR|CODE=ROLE_DENIED.

TRADER

The default

CAN DO

- NEW, AMEND, CANCEL

OCO and COMBO orders

All monitoring commands

KILL switch

CANNOT DO

- QUOTE

QUOTE_CANCEL

MARKET_MAKER

Everything a trader can do, plus quoting

CAN DO

- Two-sided QUOTE

QUOTE_CANCEL

QBOOT bootstrap state

QLEGS leg inspection

CANNOT DO

- Nothing within its configured symbols and limits

ADMIN

Operational, per configuration and policy

CAN DO

- Operational commands as configured by policy

CANNOT DO

- Business flow outside policy

PART 3

Connecting

Before you can trade, you must say hello. The session lifecycle is short, strict, and identical for every client.

The session lifecycle

FIVE STEPS

1 Connect

Open a plain TCP socket to the gateway's port — 5565 by default.

2 HELLO

The very first line must be a HELLO. Anything else is refused and the connection closes.

3 Authenticate

The gateway asks the engine whether your ID is allowed. The engine is the authority.

4 WELCOME

You get the protocol version, gateway name, heartbeat interval and idle timeout — then the symbol list.

5 Trade

Send any supported command. Responses and broadcast events arrive asynchronously.

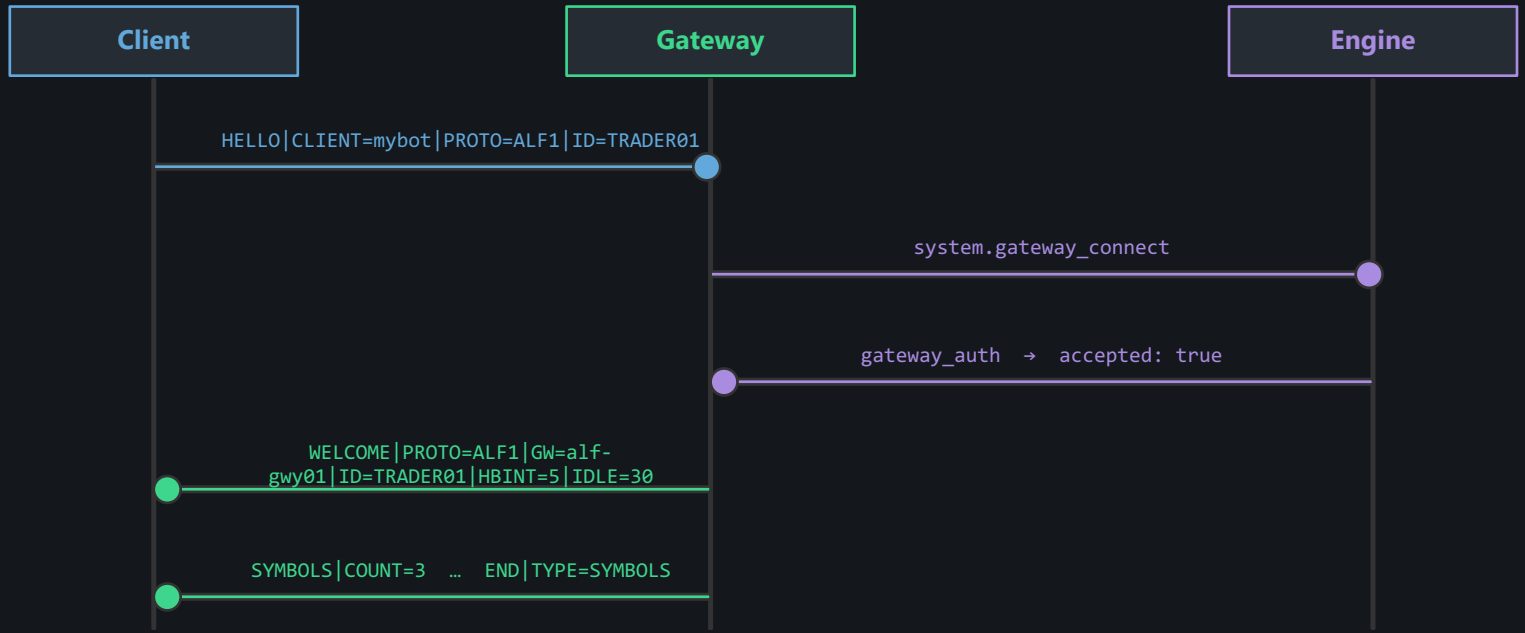
6 EXIT

A graceful close: the gateway tells the engine, flushes pending output and closes the socket.

Every ALF session follows this exact sequence — there are no optional stages and no negotiation.

The handshake, line by line

HELLO / WELCOME



HELLO needs exactly three fields: CLIENT (a free-text label), PROTO (must be ALF1) and ID (must be on the allowlist).

Any other first line gets ERR|CODE=AUTH_REQUIRED and the connection is closed immediately.

Staying connected

HEARTBEATS, LIMITS AND TIMEOUTS

The gateway defends itself with a handful of timers and counters. All of them are configurable; these are the defaults.

5s	10s	30s	64	100	50
Heartbeat interval	Handshake timeout	Idle timeout	Max connections	Commands / second	Errors before drop

A quiet link still ticks

When there is no other outbound traffic the gateway sends HB|TS=... every few seconds, so you can tell a live link from a dead one.

Silence has a deadline

If nothing arrives from you for the idle timeout, you are disconnected. Send PING periodically if your bot is naturally quiet.

Too fast is also a problem

A token bucket caps commands per second. Exceed it and you get ERR|CODE=RATE_LIMITED — the connection survives, the command does not.

Slow readers get dropped

Each session has an outbound queue. If you stop reading and it fills, the gateway closes you with SLOW_CLIENT.

Your first session, without writing code

TRY IT

Before building a client, prove the gateway is reachable end to end. A raw TCP tool is enough.

WHAT YOU TYPE

```
nc 127.0.0.1 5565

HELLO|CLIENT=test|PROTO=ALF1|ID=TRADER01
SYMBOLS
EXIT
```

WHAT COMES BACK

```
WELCOME|PROTO=ALF1|GW=alf-gwy01|ID=TRADER01
SYMBOLS|COUNT=3
SYMBOL|SYM=AAPL|TICK=0.01
END|TYPE=SYMBOLS
HB|TS=...
```

Nothing listening?

Connection refused means the gateway is not started, or is on a different port than you think.

Hangs with no output?

A firewall is in the way. Prove it on the loopback address first, then open the port.

Unreachable from another host?

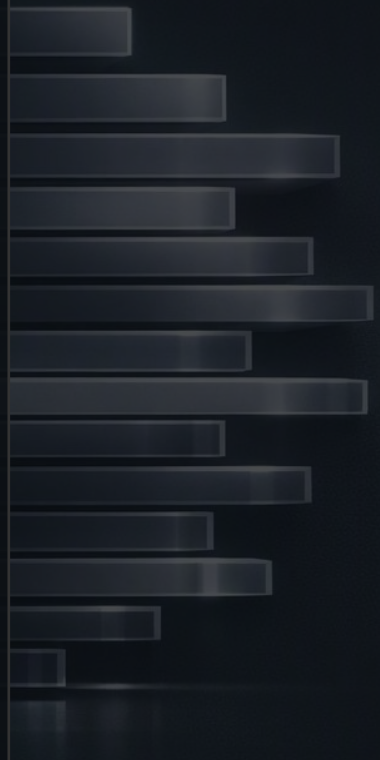
The gateway may be bound to the local interface only. Bind it to all interfaces instead.

The gateway does not terminate TLS. For anything remote, put it behind a reverse proxy.

PART 4

Sending Orders

Everything the protocol can express: eight single-leg order types, linked pairs, multi-leg combos and two-sided quotes.



What kinds of order can I send?

THE MAP

Single-leg orders

One instruction, one book

- `MARKET`
- `LIMIT`
- `IOC`
- `FOK`
- `STOP`
- `STOP_LIMIT`
- `ICEBERG`
- `TRAILING_STOP`

Linked orders

Several orders that live and die together

- OCO — one cancels the other
- COMBO — 2 to 10 legs, all or none

Market-maker quotes

One command, two sides of the book

- QUOTE — bid and ask together
- QUOTE_CANCEL — pull the quote

All of them start with the same verb: NEW. The TYPE field decides which shape the rest of the line takes.

Single-leg types — the core four

ORDER TYPES 1 OF 2

MARKET

Needs no price.

Trade now, at whatever price is available.

```
NEW|SYM=AAPL|SIDE=BUY|TYPE=MARKET|QTY=50
```

LIMIT

PRICE required.

Trade at my price or better; otherwise rest in the book and wait.

```
NEW|SYM=AAPL|SIDE=BUY|TYPE=LIMIT|QTY=100|PRICE=150.00
```

IOC

PRICE required.

Immediate or cancel: take what is there right now, throw the rest away.

```
NEW|SYM=AAPL|SIDE=BUY|TYPE=IOC|QTY=100|PRICE=150.00
```

FOK

PRICE required.

Fill or kill: all of it immediately, or none of it at all.

```
NEW|SYM=AAPL|SIDE=BUY|TYPE=FOK|QTY=100|PRICE=150.00
```

Single-leg types — the advanced four

ORDER TYPES 2 OF 2

STOP

STOP required.

Sleeps until the market reaches your trigger, then becomes a market order.

```
NEW|...|TYPE=STOP|QTY=100|STOP=148.00
```

STOP_LIMIT

STOP + PRICE.

Same trigger, but it becomes a limit order so you keep price control.

```
NEW|...|TYPE=STOP_LIMIT|QTY=100|STOP=148.00|PRICE=147.50
```

ICEBERG

VISIBLE < QTY.

A large order that only shows a small peak to the market at a time.

```
NEW|...|TYPE=ICEBERG|QTY=1000|PRICE=150.00|VISIBLE=100
```

TRAILING_STOP

TRAIL required.

A stop that follows the market by a fixed offset, locking in gains.

```
NEW|...|TYPE=TRAILING_STOP|QTY=100|TRAIL=0.50
```

For a trailing stop the initial STOP is optional — leave it out and the engine derives it from the last trade price.

Two optional fields worth knowing

TIF AND SMP

Time in force

TIF — how long the order stays alive. Defaults to DAY.

DAY	Valid for the trading day only
GTC	Good till cancelled
ATO	At the open — opening auction only
ATC	At the close — closing auction only

Self-match prevention

SMP — what to do if your order would trade with your own resting order.

NONE	Allow the self-trade
CANCEL_AGGRESSOR	Cancel the incoming order
CANCEL_RESTING	Cancel the resting order, keep matching
CANCEL_BOTH	Cancel both sides

A subtle difference worth remembering

Omitting SMP entirely is not the same as sending SMP=NONE. If you omit it, the engine falls back to whatever default is configured for your gateway. Sending NONE explicitly is always honoured as-is.

OCO — one cancels the other

LINKED ORDERS

Two linked orders on the same symbol, with the same quantity. When one leg fills, its sibling is cancelled automatically. The classic use is a take-profit and a stop-loss guarding the same position.

LEG 1 · take profit

SELL LIMIT @ 155.00

If the market rises to your target, this one trades — and the stop below is pulled.

LINKED

LEG 2 · stop loss

SELL STOP @ 148.00

If the market falls instead, this one triggers — and the take-profit above is pulled.

ONE LINE, TWO LEGS

```
NEW|TYPE=OCO|OCO_ID=EXIT-AAPL|SYM=AAPL|QTY=100|TIF=GTC
   |LEG1_SIDE=SELL|LEG1_TYPE=LIMIT|LEG1_PRICE=155.00
   |LEG2_SIDE=SELL|LEG2_TYPE=STOP|LEG2_STOP=148.00
```

Both legs share the symbol, quantity and TIF. Each gets its own engine order ID; cancel the whole pair with CANCEL|OCO_ID=EXIT-AAPL.

COMBO — multi-leg orders

LINKED ORDERS

A parent order with 2 to 10 child legs across different symbols, executed all-or-none. Use it for spreads, pairs trades and arbitrage-style baskets.

COMBO parent

COMBO_ID=ARB-01

COMBO_TYPE=AON
LEG_COUNT=3
TIF=DAY



```
LEG0    BUY   AAPL   200 @ 210.00
LEG1    SELL  MSFT   100 @ 415.00
LEG2    SELL  GOOG    50 @ 170.00
```

Zero-based numbering

LEG0, LEG1, LEG2 … and LEG_COUNT must agree with them.

All or none

COMBO_TYPE defaults to AON. If one leg is cancelled or expires, the rest cascade-cancel.

Duplicate symbols rejected

Each leg must reference a different instrument.

What OCO and COMBO cannot express

KNOWN LIMITS

The text syntax exposes fewer per-leg fields than the engine supports. Knowing where the ceiling is saves you a lot of debugging.

Leg type	In an OCO pair	In a COMBO leg	Why
MARKET	YES	YES	
LIMIT	YES	YES	
IOC	YES	YES	
FOK	YES	YES	
STOP	YES	NO	no leg STOP
STOP_LIMIT	YES	NO	no leg STOP
TRAILING_STOP	YES	NO	no leg TRAIL
ICEBERG	NO	NO	no leg VISIBLE

A combo leg only carries SYM, SIDE, QTY, TYPE and PRICE — there is no per-leg stop, trail or visible size.

Market-maker quotes

QUOTE AND QUOTE_CANCEL

A market maker shows both sides of the book at once. One QUOTE command submits or replaces a bid and an ask together — internally it becomes two limit orders.

BID · you buy

149.90 × 500

BID · BID_QTY

ASK · you sell

150.10 × 500

ASK · ASK_QTY

SUBMIT, REPLACE, CANCEL

```
QUOTE|SYM=AAPL|BID=149.90|ASK=150.10
     |BID_QTY=500|ASK_QTY=500|QUOTE_ID=MM-AAPL-1
QUOTE_CANCEL|SYM=AAPL
```

- MARKET_MAKER role only
- BID must be strictly below ASK
- Both quantities must be positive
- A new quote replaces the previous one

Inspect your legs with QLEGS, and recover bootstrap state after a reconnect with QBOOT.

The KILL switch

RISK CONTROL

One command withdraws your exposure. It applies to your gateway only — it never touches anyone else's orders, and it never reverses fills that already happened.

EVERYTHING, OR ONE SYMBOL

```
KILL
KILL|SYM=AAPL
```

```
KILL_ACK|ACCEPTED=TRUE|ORDERS=3|QUOTES=2
```

QUOTES counts cancelled legs, not quote records — one two-sided quote usually contributes 2.

What it cancels

- Your active market-maker quote legs — bid and ask

Your ordinary resting limit-style exposure

Your other resting child orders sitting in a book

What it does not touch

- Other gateways' orders

Fills that already happened

Completed orders no longer resting in a book

Worked example

A market maker with quotes in AAPL and MSFT plus resting orders in AAPL and GOOG sends KILL|SYM=AAPL. The AAPL quote legs and AAPL resting orders go; the MSFT quote and the GOOG order stay untouched.

What comes back

RESPONSES AND EVENTS

Nothing is synchronous. You send a line; acknowledgements and lifecycle events arrive on the same connection whenever the engine produces them.



An order's life: acknowledged, then filled in one or more pieces, then finished or withdrawn.

Addressed to your gateway

ACK	ORDER_ID · ACCEPTED · REASON · SYMBOL · SIDE · TYPE
FILL	ORDER_ID · FILL_QTY · FILL_PRICE · REMAINING · STATUS
AMENDED	ORDER_ID · PRICE · QTY · REMAINING · PRIORITY_RESET
CANCELLED / EXPIRED	ORDER_ID
KILL_ACK	ACCEPTED · REASON · ORDERS · QUOTES

Group, quote and error events

QUOTE_ACK / QUOTE_STATUS	QUOTE_ID · ACCEPTED · BID_ID · ASK_ID · STATUS
OCO_ACK / OCO_CANCELLED	OCO_ID · LEG1_ID · LEG2_ID · CANCELLED_ID
COMBO_ACK / COMBO_STATUS	COMBO_ID · ACCEPTED · STATUS · REASON
DC_ACK / DC_FILL	STATE · SEQ · FILL_QTY · FILL_PRICE · LIQUIDITY
ERR	CODE · DETAIL

Broadcasts and drop copy

UNSOLICITED TRAFFIC

Your client must expect lines it did not ask for. There is nothing to subscribe to — broadcasts start flowing after WELCOME.

SESSION	The trading phase changed — pre-open, auction, continuous, closed
HALT	A circuit breaker halted a symbol
RESUME	That symbol is trading again
TRADE	Any matched trade, on any symbol
HB	Heartbeat — the link is alive and quiet

Drop copy

An optional second view of your own fills, relayed on the same connection.

OPT IN PER SESSION

```
DC|STATE=ON
DC_ACK|STATE=ON
DC_FILL|SEQ=42
  |LIQUIDITY=TAKER
```

Drop copy is in addition to, not instead of

With DC on you still receive the ordinary FILL messages. DC_FILL adds a sequence number and a maker/taker flag that FILL does not carry, but omits the remaining quantity and status that FILL does. Every session opts in independently — it is off by default.

PART 5

Amend, Cancel, Recover

Orders rarely stay as first written. How to change one, how to pull one, and what the gateway tells you when something is wrong.



AMEND — changing a resting order

ORDER MANAGEMENT

Identify the order by its full engine-generated ID, then send a new price, a new quantity, or both. At least one of them is required.

THREE FORMS

```
AMEND|ID=ORD-12345678|PRICE=151.00
AMEND|ID=ORD-12345678|QTY=200
AMEND|ID=ORD-12345678|PRICE=151.00|QTY=200
```

THE REPLY

```
AMENDED|ORDER_ID=…|PRICE=…|QTY=…
       |REMAINING=…|PRIORITY_RESET=TRUE
```

Only two types can be amended

Resting LIMIT and ICEBERG orders. Market, IOC, FOK, stop, stop-limit and trailing-stop orders are not amendable.

QTY is a new total, not a delta

Send the quantity you want the order to be, not the amount you want to add or remove.

The new total must exceed what already filled

You cannot shrink an order below the quantity that has already traded, and it must stay positive.

Use the full identifier

Displays abbreviate order IDs for readability. Take the full ID from your order table or the acknowledgement stream.

The rule that surprises people

QUEUE PRIORITY

Your place in the queue at a price level is valuable — it decides whether you trade first. Some amendments keep it; others send you to the back of the line.

Quantity decrease, same price

You are asking for less, so nobody behind you is disadvantaged.

```
AMEND|ID=…|QTY=80
```

PRIORITY PRESERVED

Price change

A new price is effectively a new order at a new level.

```
AMEND|ID=…|PRICE=150.10
```

PRIORITY RESET

Quantity increase

Claiming more size means queueing again behind everyone else.

```
AMEND|ID=…|QTY=120
```

PRIORITY RESET

`Starting point in every case: NEW|SYM=AAPL|SIDE=BUY|TYPE=LIMIT|QTY=100|PRICE=150.00`

CANCEL — one order, or a whole group

ORDER MANAGEMENT

Single order

Cancels one single-leg order.

```
CANCEL|ID=<order-id>
```

→

```
CANCELLED|ORDER_ID=...
```

An OCO pair

Cancels both legs of the pair at once.

```
CANCEL|OCO_ID=<label>
```

→

```
OCO_CANCELLED|OCO_ID=...|CANCELLED_ID=...|REASON=...
```

A combo

Cancels the parent and all still-resting child legs.

```
CANCEL|COMBO_ID=<label>
```

→

```
COMBO_STATUS|COMBO_ID=...|STATUS=CANCELLED
```

Cancelling never unwinds a trade. Fills that already happened stay done — cancel only removes what is still resting.

Knowing where you stand

MONITORING COMMANDS

SYMBOLS	Which instruments exist, and their tick size	`SYMBOLS|COUNT=3 → SYMBOL|… → END`
ORDERS	Your resting orders with full IDs, prices, remaining quantity and status	`ORDERS|COUNT=2|GW=… → ORDER|… → END`
SESSION	The current trading phase, without waiting for the next broadcast	`SESSION|STATE=CONTINUOUS|SESSIONS_ENABLED=TRUE`
QBOOT	Active quote bootstrap state — for market makers restarting	`QBOOT|COUNT=1 → QUOTE|… → END`
QLEGS	Quote legs with fill state; SHOW=ACTIVE, RECENT or ALL	`QLEGS|COUNT=2|RECENT_COUNT=1 → LEG|… → END`
STATUS · POS	Console-only: a session summary and locally computed positions and P&L	`Answered by the terminal, not the engine`

One command behaves differently in each client: the terminal answers QLEGS from its own cache, while the TCP gateway asks the engine and returns authoritative state.

How errors are reported

ERROR HANDLING

Every error is a single line with a machine-readable code and a human-readable detail. Parse the code; log the detail.

```
ERR|CODE=<CODE>|DETAIL=<message>
```

The only question that matters

CONNECTION KEPT

A single command was rejected. Fix it and carry on — the session is still good.

`BAD_MESSAGE` · `UNKNOWN_COMMAND` · `MISSING_FIELD` · `INVALID_VALUE` · `SYMBOL_NOT_CONFIGURED` · `ROLE_DENIED` · `RATE_LIMITED` · `INTERNAL_ERROR`

CONNECTION CLOSED

The session is over. Fix the cause, then reconnect and re-sync your state.

`AUTH_REQUIRED` · `AUTH_TIMEOUT` · `AUTH_FAILED` · `PROTO_MISMATCH` · `GATEWAY_ALREADY_CONNECTED` · `SLOW_CLIENT` · `IDLE_TIMEOUT` · `MAX_ERRORS`

Errors have a budget

Recoverable errors are counted in a sliding window. Accumulate too many of them — 50 in 60 seconds by default — and the gateway disconnects you anyway. This protects everyone else from a runaway client, so never retry a rejected command in a tight loop: fix the command instead.

PART 6

Technical Reference

The detail you will want open on a second screen while you write your first client.



Required fields by order type

REFERENCE

	SYM	SIDE	TYPE	QTY	PRICE	STOP	VISIBLE	TRAIL	TIF	SMP
MARKET	REQ	REQ	REQ	REQ					opt	opt
LIMIT	REQ	REQ	REQ	REQ	REQ				opt	opt
IOC	REQ	REQ	REQ	REQ	REQ				opt	opt
FOK	REQ	REQ	REQ	REQ	REQ				opt	opt
STOP	REQ	REQ	REQ	REQ		REQ			opt	opt
STOP_LIMIT	REQ	REQ	REQ	REQ	REQ	REQ			opt	opt
ICEBERG	REQ	REQ	REQ	REQ	REQ		REQ		opt	opt
TRAILING_STOP	REQ	REQ	REQ	REQ		opt		REQ	opt	opt

REQ = required · opt = optional · blank = not used by that type. TRAILING_STOP takes an optional STOP; leave it out and the engine derives the initial trigger from the last trade.

The formal grammar

REFERENCE

Written in ABNF-style notation: n*m element means "repeat at least n and at most m times".

ALF LINE GRAMMAR

```
alf-line = command *( "|" segment )
command  = 1*( ALPHA / "_" )
segment  = key "=" value
key      = 1*( ALPHA / DIGIT / "_" / "." )
value    = *VCHAR
```

SINGLE-LEG ORDER

```
new-single =
    "NEW" "|SYM=" symbol
    "|SIDE=" side  "|TYPE=" order-type
    "|QTY=" quantity
    *( "|" option )
```

1*X means one or more

*X means zero or more, and 2*4X means between two and four.

VCHAR means a visible character

In practice, read value as "everything after the first = up to the next pipe".

The parser is simpler than the grammar

It splits on pipes, splits each segment on the first equals sign, and takes the rest as raw text. There is no character-class check.

Error code catalogue

REFERENCE

CONNECTION IS CLOSED	
AUTH_REQUIRED	Any command before HELLO completes
AUTH_TIMEOUT	No HELLO within the handshake timeout
AUTH_FAILED	The engine rejected the gateway ID
PROTO_MISMATCH	HELLO carried the wrong PROTO value
GATEWAY_ALREADY_CONNECTED	That ID already has an active session
SLOW_CLIENT	Your outbound queue filled up
IDLE_TIMEOUT	No inbound traffic for the idle window
MAX_ERRORS	Too many errors in the sliding window

CONNECTION SURVIVES	
BAD_MESSAGE	Empty line, non-UTF-8, or over 4096 bytes
UNKNOWN_COMMAND	Unrecognised command verb
MISSING_FIELD	A required field was absent
INVALID_VALUE	A value failed validation
SYMBOL_NOT_CONFIGURED	Unknown symbol
ROLE_DENIED	Not allowed for this gateway's role
RATE_LIMITED	Commands arriving too fast
INTERNAL_ERROR	Unexpected gateway-internal exception

Design your client around this split: the left column needs a reconnect and a state re-sync, the right column needs a corrected command.

Command to response map

REFERENCE

NEW (single)	ACK|ORDER_ID=…|ACCEPTED=TRUE|FALSE	then FILL, CANCELLED or EXPIRED
NEW|TYPE=OCO	OCO_ACK|OCO_ID=…|LEG1_ID=…|LEG2_ID=…	then an ACK per leg, later OCO_CANCELLED
NEW|TYPE=COMBO	COMBO_ACK|COMBO_ID=…|ACCEPTED=…	then COMBO_STATUS and per-leg events
AMEND	AMENDED|…|REMAINING=…|PRIORITY_RESET=…	tells you whether you kept queue position
CANCEL	CANCELLED / OCO_CANCELLED / COMBO_STATUS	depends on which identifier you cancelled by
QUOTE	QUOTE_ACK|QUOTE_ID=…|BID_ID=…|ASK_ID=…	later QUOTE_STATUS as legs fill or expire
KILL	KILL_ACK|ACCEPTED=…|ORDERS=N|QUOTES=N	quote count is legs, not quote records
DC	DC_ACK|STATE=ON|OFF	then unsolicited DC_FILL lines while on
SYMBOLS / ORDERS / QBOOT / QLEGS	header line, detail lines, END|TYPE=…	multi-line: always read to the END marker
PING	PONG|TS=…	cheap way to reset the idle timer

Configuration reference

REFERENCE

Gateway limits

enabled	true
name	alf-gwy01
bind_address	0.0.0.0
port	5565
heartbeat_interval_sec	5
handshake_timeout_sec	10
idle_timeout_sec	30
max_connections	64
max_client_queue	10000
max_commands_per_second	100
max_errors_before_disconnect	50
error_window_sec	60

Per-gateway identity

id	Gateway identity — required, and checked against the engine's allowlist
description	Human-readable label
role	TRADER (default), MARKET_MAKER or ADMIN
disconnect_behaviour	CANCEL_QUOTES_ONLY (default), CANCEL_ALL or LEAVE_ALL
quote_refresh_policy	INACTIVATE_ON_ANY_FILL, ON_FULL_FILL or NEVER
enforce_mm_obligation	Per-gateway market-maker obligation checks — off by default
mm_max_spread_ticks	Widest quote allowed under obligation checks — default 10
mm_min_qty	Smallest quote size allowed — default 100

Anything listed under the ALF gateway allowlist may connect — the gateway needs no separate credentials block. The engine remains the authority on who is allowed in.

A working client in about thirty lines

BUILD IT YOURSELF

Standard library only — no message-bus dependency, no framework. Connect, handshake, send, read.

MINIMAL ALF CLIENT — PYTHON

```
import socket

sock = socket.create_connection((host, 5565), timeout=5)
buf  = bytearray()

def send(line):
    sock.sendall((line + "\n").encode("utf-8"))

def recv_line():
    while True:
        nl = buf.find(b"\n")
        if nl >= 0:
            line = bytes(buf[:nl]).decode("utf-8")
            del buf[:nl + 1]
            return line
        chunk = sock.recv(4096)
        if not chunk: raise RuntimeError("closed")
        buf.extend(chunk)

send("HELLO|CLIENT=bot|PROTO=ALF1|ID=TRADER01")
while not (line := recv_line()).startswith("WELCOME"):
    if line.startswith("ERR"): raise RuntimeError(line)

send("NEW|SYM=AAPL|SIDE=BUY|TYPE=LIMIT|QTY=100|PRICE=150.00")
send("EXIT")
```

Buffer, always

One read is never one line. Accumulate bytes and split on the newline yourself.

Wait for WELCOME

Do not send anything else until the handshake completes.

Expect surprises

Heartbeats, session changes and trades can arrive before your acknowledgement does.

Keep the link warm

Send PING if your bot goes quiet, or the idle timer will close you.

Reconnect and re-sync playbook

OPERATIONS

After any disconnect, never resume trading blind. Rebuild your view of the world in this order.

1	SYMBOLS	Confirm which books are active and their tick sizes.
2	ORDERS	Recover your resting single-leg orders — and their full IDs, which you need for any follow-up amend or cancel.
3	`QBOOT, then QLEGS|SHOW=ALL`	Market makers only: discover quote slots seeded before you connected, then inspect live and recently closed legs. This is what stops you double-quoting.
4	POS	Rebuild your position view and check it against what you expected.
5	Resume	Only once all of the above agree with each other.

Your gateway's configured disconnect behaviour decides what the engine did to your exposure while you were away — cancel quotes only, cancel everything, or leave it all resting.

Where to go from here

RECAP

Ready-made libraries

Python and C libraries ship with the examples: a parser that turns lines into structures and back, plus a session helper that handles connect and handshake for you.

Interactive clients

Both languages also ship a full interactive client with tab completion, command history and a background thread that displays events while you type.

The five things to remember

- A gateway authenticates, validates and translates so the engine only has to match.

ALF is one line per command — a verb, then pipe-separated fields.

Use the terminal locally; use the TCP gateway for anything remote or automated.

Every error tells you both what went wrong and whether your session survived.

Buffer your reads, wait for WELCOME, and re-sync after every reconnect.

Start here

Open a raw TCP connection, say HELLO, ask for SYMBOLS, and send one small limit order. Once you have seen an ACK come back on a socket you opened yourself, everything else in this deck is just more fields.