EDUMATCHER · RALF1

RALF

The post-trade protocol

Reconciliation ALF — how clearing, risk and audit systems get the trade data they need, in plain text over TCP.

A beginner's guide · pm-ralf-gwy · TCP 5580



Eight questions this deck answers

01 **What is a gateway?**
Why an exchange never lets you talk to the engine directly

02 **What is RALF for?**
The job the protocol exists to do

03 **How is it used?**
The session, from connect to disconnect

04 **What can be sent?**
Three channels, two event families

05 **How is data shaped?**
The line format and its fields

06 **How do errors work?**
Sequence gaps, replay and error codes

07 **How do I build a client?**
A checklist and working code

08 **Where do I read more?**
The reference map

RALF in one picture

Trades happen inside the engine. RALF is how they leave the building.



Read-only: clients subscribe and receive. Nobody sends an order over RALF.

SECTION 01

What is a gateway?

Every exchange has a hard border. A gateway is the controlled doorway through it — and the reason external systems can be trusted with internal data.

THE PROBLEM

The engine is not something you plug into

Internal event buses are fast, fragile and unauthenticated. They are not built for outsiders.

Wrong transport

The engine publishes on an internal ZeroMQ socket. Outside networks cannot and should not reach it.

Wrong format

Internal events are raw engine objects — they change whenever the engine changes.

No identity

The bus has no idea who you are, and no way to limit what you see.

No recovery

Miss a message and it is simply gone. There is no journal to ask again.

A gateway solves all four at once — and that is the whole idea.

CONCEPT

What a gateway actually does

Four jobs, performed on every message, for every client.

1	TRANSLATE	Turns internal engine events into a stable, documented wire format that will not break when the engine changes.
2	AUTHENTICATE	Identifies the client, checks its declared role, and limits it to the data that role is entitled to see.
3	SEQUENCE	Numbers every message so a client can prove nothing was lost, and keeps a journal so gaps can be refilled.
4	PROTECT	Buffers slow consumers, heartbeats idle links, and disconnects anything that endangers the feed.

RALF is the wire format. pm-ralf-gwy is the process that performs all four jobs.

ORIENTATION

EduMatcher has four external protocols

They are siblings — same naming family, very different jobs.

ALF

Almost FIX

Human-readable order entry

`pm-alf-console`
`pm-alf-gwy`

BALF

Binary ALF

Low-latency order entry

`pm-balf-gwy`

CALF

Channel ALF

Market data (pre-trade)

`pm-md-gwy`

RALF

Reconciliation ALF

Post-trade dissemination

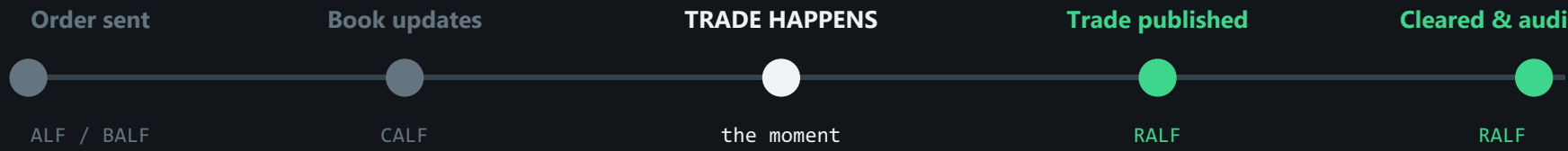
`pm-ralf-gwy`

They all share one style: uppercase message type, then FIELD=VALUE pairs separated by a pipe. Learn one, and the rest read easily.

WHERE RALF SITS

Before the trade, and after it

RALF only speaks about things that have already happened.



If you need the current best bid and offer, RALF is the wrong protocol — use CALF. If you need to know what was actually executed, RALF is exactly right.

SECTION 02

The purpose of RALF

Post-trade systems do not care how a price was discovered. They care what was executed, by whom, and whether their copy of the record is complete.

DEFINITION

RALF in one sentence

RALF streams **executed-trade and end-of-day events** over **plain-text TCP** to external back-office consumers.

READ-ONLY

Clients subscribe and receive. They never submit orders.

DERIVED

Every line comes from an engine event — never invented by the gateway.

RECOVERABLE

Numbered and journalled, so a client can always prove it saw everything.

`Version token: RALF1 · Transport: TCP · Encoding: UTF-8 lines ending in a newline`

AUDIENCE

Three kinds of system listen to RALF

Each has a different question, and each gets its own channel.

CLEARING

"Do our books match yours?"

- Trade reconciliation
- Position keeping
- Clearing obligations

DROP_COPY

"What is our exposure right now?"

- Real-time position risk
- Pre-trade limit monitoring
- Compliance surveillance

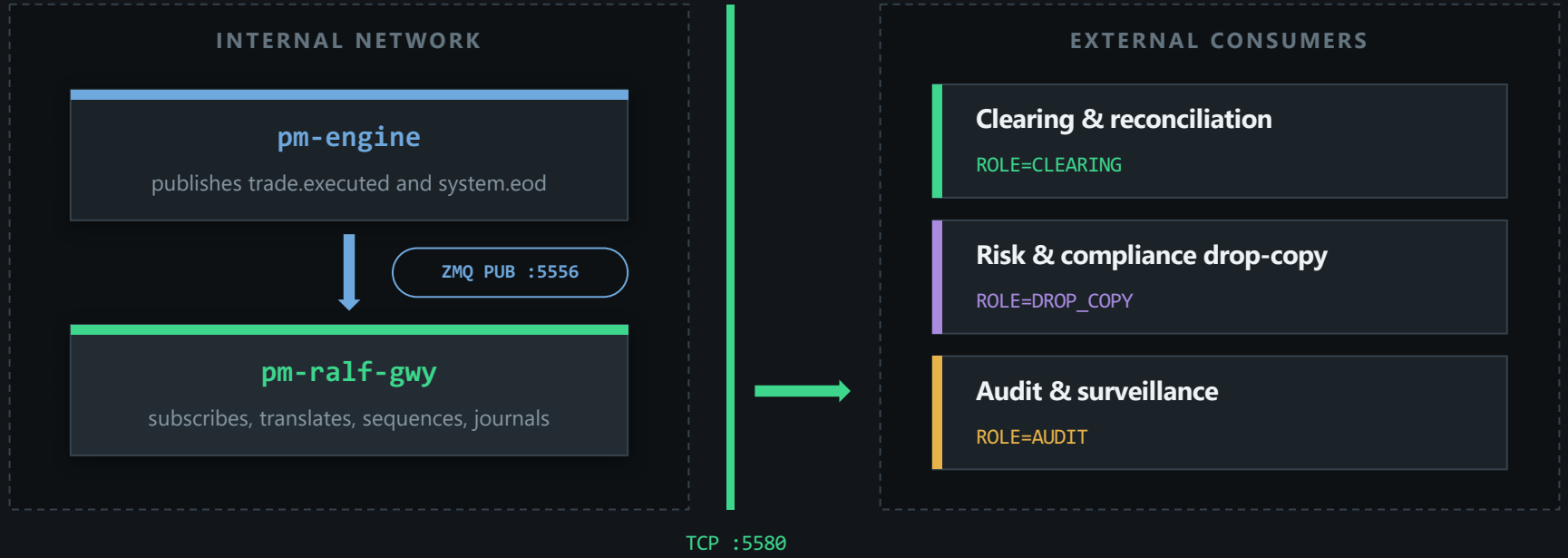
AUDIT

"Can we reconstruct the day?"

- Regulatory reporting
- Trade reconstruction
- End-of-day audit files

Where the gateway sits

The engine's internal bus stays internal. Only the gateway crosses the border.



What the gateway does for you

Five guarantees you get for free by using RALF instead of the raw bus.

1	**Translation**	Engine events become RALF lines, one per subscribed channel.
2	**Sequencing**	A monotonic SEQ is assigned per channel, so gaps are detectable.
3	**Replay journal**	A time-bounded journal lets reconnecting clients recover what they missed.
4	**Baseline snapshot**	A SNAP is sent automatically on subscribe, giving you a starting sequence.
5	**Access control**	Roles are enforced per channel, and slow clients are dropped to protect the feed.

When RALF is the right answer — and when it is not

APPROACH	TRANSPORT	BEST FOR	NOT SUITABLE FOR
RALF (pm-ralf-gwy)	TCP text	External clearing, drop-copy and audit consumers	Pre-trade market data; top-of-book
CALF (pm-md-gwy)	TCP text	External market data: TOP, TRADE, STATE, INDEX	Post-trade events
Internal ZMQ PUB	ZMQ binary	Internal Python processes inside the engine host	External clients
REST / WebSocket	HTTP/JSON	Web dashboards and one-shot queries	High-frequency streaming

> Rule of thumb: if the question starts with "what was executed", it is RALF. If it starts with "what is the price", it is CALF.

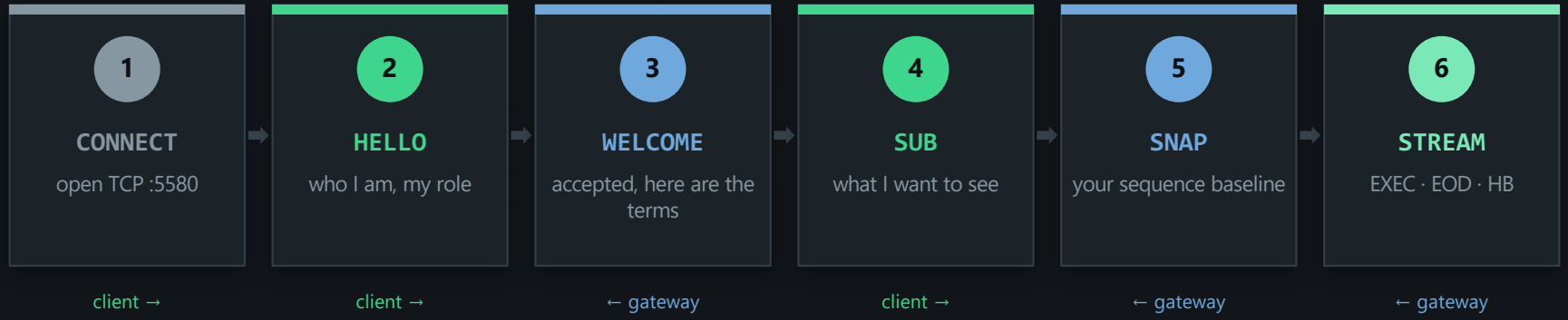
SECTION 03

How the protocol is used

A RALF session is a short conversation followed by a long silence-punctuated stream. Six steps, and you have seen all of them.

The whole conversation, at a glance

One authenticated session per TCP connection, held open for as long as you need it.



Client may also send

UNSUB · **PING** · **EXIT**

Gateway may also send

HB · **PONG** · **ERR** · **EXIT**

HELLO must come first. Any other opening message is rejected with AUTH_REQUIRED.

STEP 1 OF 6

HELLO — introduce yourself

Declares who you are, which protocol version you speak, and what you are entitled to see.

HELLO	CLIENT=ext01	PROTO=RALF1	ROLE=CLEARING	LASTSEQ=0
message type, always first	free-text id used in gateway logs	must be exactly RALF1	your access tier — checked on every SUB	0 = start live. Any other value replays

FIELD	REQUIRED	WHAT IT MEANS
CLIENT	yes	An identifier for you. Maximum 32 characters. It only appears in gateway logs.
PROTO	yes	The protocol version token. Anything other than RALF1 is refused.
ROLE	yes	One of CLEARING, DROP_COPY or AUDIT — and it must be in the gateway's allowed_roles.
LASTSEQ	optional	A replay checkpoint. Ask for everything after this sequence number.

> Missing CLIENT, PROTO or ROLE gives AUTH_REQUIRED. A role outside allowed_roles gives ENTITLEMENT_DENIED.

STEP 2 OF 6

WELCOME — the gateway states the terms

Do not hard-code these values. Read them from WELCOME and let the gateway tell you.

WELCOME	PROTO=RALF1	GW=ralf-gwy01	ROLE=CLEARING	REPLAY=86400	HBINT=1
session accepted	negotiated version	which gateway you reached	the role that was accepted	seconds of journal you can ask back for	heartbeat cadence in seconds

REPLAY — default 86400 s

How far back the journal reaches. Use it to decide whether a reconnect can be lossless — the default covers a full trading day.

HBINT — default 1 s

How often the gateway will send a heartbeat while the stream is quiet. Use it to set your own liveness alarm.

If the gateway refuses you, it does not send WELCOME — it sends an ERR line or simply closes the connection.

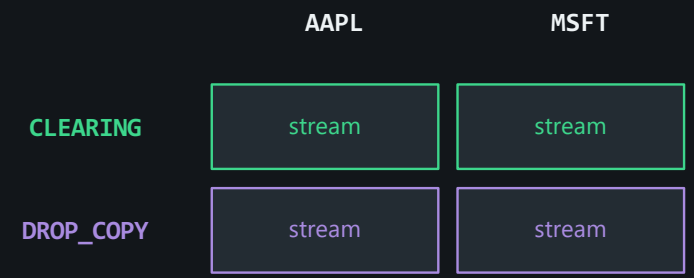
SUB — say what you want to see

A subscription is every channel you name, multiplied by every symbol you name.

```
SUB|CH=CLEARING|SYM=*
SUB|CH=CLEARING,DROP_COPY|SYM=AAPL,MSFT
```

SYM defaults to * (all symbols). Multiple SUB lines are cumulative — they add to what you already have.

CH × SYM = your streams



Two channels × two symbols = four independent streams, each with

Two rules to remember

- A role may only subscribe to its own channel — except AUDIT, which may subscribe to any.
- Every new (channel, symbol) pair triggers one SNAP.

UNSUB is the mirror image

UNSUB|CH=DROP_COPY|SYM=AAPL removes exactly the pairs it names and leaves everything else running. It is idempotent, so it is always safe to send.

STEP 4 OF 6

SNAP — your baseline, not your book

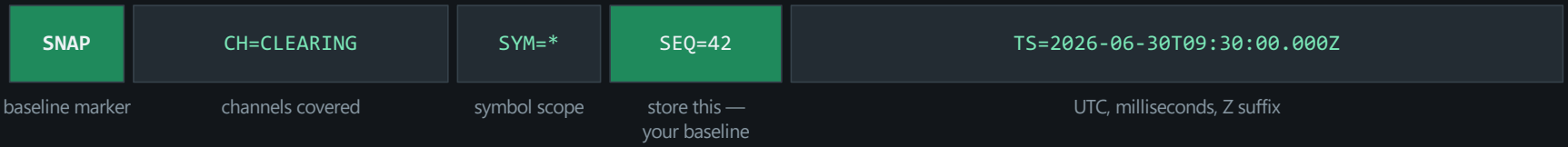
The single most misunderstood message in RALF.

✓ WHAT IT IS

A sequence baseline. It tells you the number your stream is currently at, so you know what to expect next.

✗ WHAT IT IS NOT

A state dump. RALF's SNAP carries no book, no positions and no open trades — only CH, SYM, SEQ and TS.



Store the SEQ per (channel, symbol). Everything you do later — gap detection, replay, reconnection — is measured from it.

STEP 5 OF 6

The live stream

After the snapshot, the gateway simply talks. Your job is to keep listening.

```
SNAP|CH=CLEARING|SYM=*|SEQ=0|TS=2026-06-30T09:30:00.000Z
EXEC|CH=CLEARING|SYM=AAPL|SEQ=1|TS=...|EXEC_ID=E001|SIDE=BUY|QTY=200|PX=150.12
EXEC|CH=CLEARING|SYM=AAPL|SEQ=2|TS=...|EXEC_ID=E002|SIDE=SELL|QTY=100|PX=150.14
HB|TS=2026-06-30T09:45:00.000Z
EOD|CH=CLEARING|SYM=AAPL|SEQ=3|TS=...|TRADE_COUNT=2|EXEC_COUNT=2
```

EXEC — One executed trade

Sent as each match occurs. This is the payload you came for.

EOD — End-of-day marker

One per symbol, summarising the session's trade counts.

HB — Heartbeat

Sent only when nothing else has been sent. Proof the link is alive.

STEP 6 OF 6

Staying alive, and leaving cleanly

The gateway will drop a client that goes quiet or falls behind. Both are avoidable.

UNSUB `UNSUB|CH=DROP_COPY|SYM=AAPL`

Removes matching pairs. Idempotent — sending it twice is harmless.

PING / PONG `PING → PONG|TS=...`

Your liveness probe. Also counts as inbound traffic, which resets the idle timer.

EXIT `EXIT|REASON=idle_timeout`

Either side can end the session. The gateway sends it with a reason.

THE THREE CLOCKS

1 s heartbeat interval

5 s idle timeout

10 000 max queued messages

Defaults — the first two are advertised in WELCOME.

Two ways to get disconnected: send nothing inbound for the idle timeout, or let your outbound queue overflow (SLOW_CLIENT). Read fast, and ping when idle.

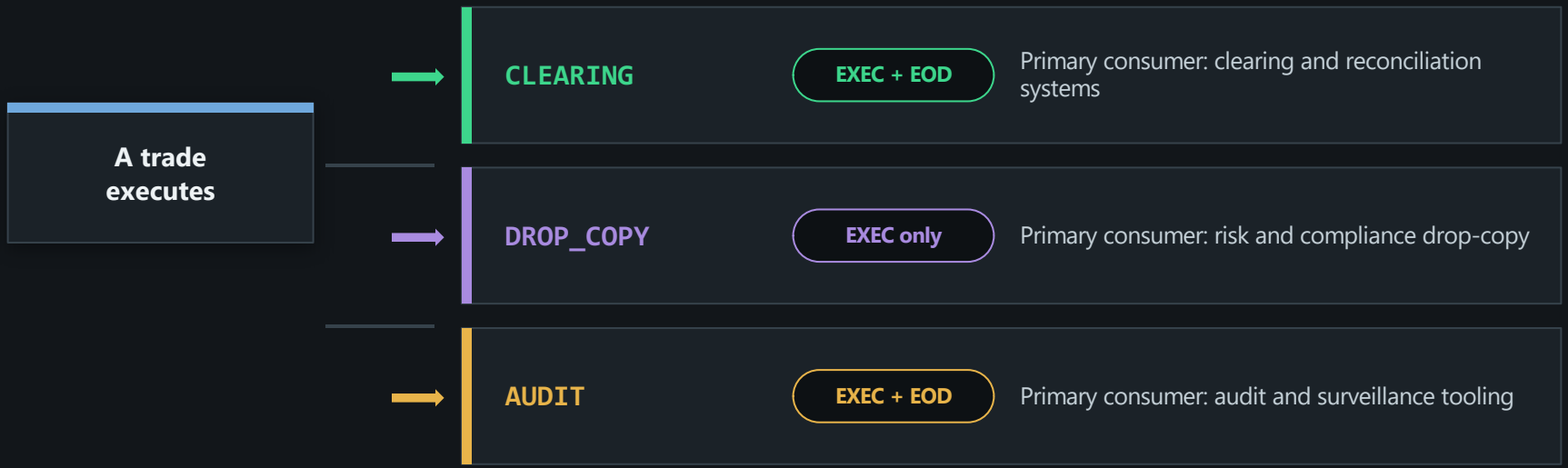
SECTION 04

What information can be sent?

Three channels carry two event families. That is the entire surface area of RALF1 — and it is deliberately small.

Three channels, one source of truth

The same trade appears on more than one channel — each copy with its own sequence number.



Every channel sends a SNAP on subscribe, and every channel accepts SYM=* for all symbols.

CHANNEL 1 OF 3

CLEARING — the reconciliation feed

One EXEC line per matched trade, plus an EOD summary per symbol at end of day.

TYPICAL USES

- Trade reconciliation
- Position keeping
- Clearing obligation calculation

HOW TO SUBSCRIBE

`SUB|CH=CLEARING|SYM=*`

The gateway replies immediately with a SNAP baseline, then streams events as they occur.

ON THE WIRE

```
SNAP|CH=CLEARING|SYM=*|SEQ=0|TS=2026-06-30T09:30:00.000Z
EXEC|CH=CLEARING|SYM=AAPL|SEQ=1|EXEC_ID=E001|BUY_GW=gw01|SELL_GW=gw01|SIDE=BUY|QTY=200|PX=150.12
EXEC|CH=CLEARING|SYM=AAPL|SEQ=2|EXEC_ID=E002|BUY_GW=gw01|SELL_GW=gw02|SIDE=SELL|QTY=100|PX=150.14
EOD |CH=CLEARING|SYM=AAPL|SEQ=3|TRADE_COUNT=2|EXEC_COUNT=2
```

This is the channel to start with. If you are building a back-office integration, CLEARING is almost certainly what you want.

CHANNEL 2 OF 3

DROP_COPY — the real-time risk feed

The same EXEC events, aimed at systems that must react to fills as they happen.

TYPICAL USES

- Real-time position risk
- Pre-trade limit monitoring
- Compliance surveillance

HOW TO SUBSCRIBE

```
SUB|CH=DROP_COPY|SYM=AAPL,MSFT
```

The gateway replies immediately with a SNAP baseline, then streams events as they occur.

ON THE WIRE

```
SNAP|CH=DROP_COPY|SYM=AAPL|SEQ=0|TS=2026-06-30T09:30:00.000Z
EXEC|CH=DROP_COPY|SYM=AAPL|SEQ=1|EXEC_ID=E001|MATCH_ID=E001|BUY_ORDER_ID=O100
    |SELL_ORDER_ID=O101|BUY_GW=gw01|SELL_GW=gw01|SIDE=BUY|QTY=200|PX=150.12
```

DROP_COPY carries EXEC only — no EOD. If you need the end-of-day marker as well, use CLEARING or AUDIT.

CHANNEL 3 OF 3

AUDIT — the complete trail

All post-trade events, for consumers who must be able to reconstruct the whole day.

TYPICAL USES

- Regulatory reporting
- Trade reconstruction
- End-of-day audit file generation

HOW TO SUBSCRIBE

SUB|CH=AUDIT|SYM=*

The gateway replies immediately with a SNAP baseline, then streams events as they occur.

ON THE WIRE

```
SNAP|CH=AUDIT|SYM=*|SEQ=0|TS=2026-06-30T09:30:00.000Z
EXEC|CH=AUDIT|SYM=AAPL|SEQ=1|EXEC_ID=E001|SIDE=BUY|QTY=200|PX=150.12
EOD |CH=AUDIT|SYM=AAPL|SEQ=2|TRADE_COUNT=1|EXEC_COUNT=1
```

AUDIT is also the only privileged role: a client authenticated as AUDIT may subscribe to any channel, not just its own.

ACCESS CONTROL

Roles decide what you may subscribe to

Roles and channels share the same three names — but they are not the same thing.

ROLE ↓ \ CHANNEL →	CLEARING	DROP_COPY	AUDIT
CLEARING	allowed	denied	denied
DROP_COPY	denied	allowed	denied
AUDIT	allowed	allowed	allowed

AUDIT is the privileged role — it may subscribe to any channel. Everyone else is confined to their own. Subscribing outside your entitlement returns ENTITLEMENT_DENIED. Only roles listed in the gateway's allowed_roles may authenticate at all.

RALF1 has exactly two event families

Every data line you will ever receive is derived from one of two engine broadcasts.

engine event

`trade.executed`

becomes

EXEC

One line per executed trade, emitted once per subscribed channel.

engine event

`system.eod`

becomes

EOD

One per-symbol end-of-day marker, emitted on all three channels.

NOT IN RALF1 — reserved for future revisions

CORRECT · BUST · ALLOC · SETTLE

These families will appear only when the corresponding engine events exist. Write your parser so it ignores message types it does not know — that is how you stay forward-compatible.

SECTION 05

The structure of the information

One message per line. A type, then key-value pairs. If you can split a string, you can parse RALF.

Anatomy of a RALF line

Every message — control or data — follows the same shape.

```
MSGTYPE|KEY=VALUE|KEY=VALUE|...\n
```

The type

Uppercase ASCII only — letters, digits and underscore. Always the first token on the line.

The separator

Fields are separated by a pipe. Values never contain a pipe or a newline.

The split

Each field splits into key and value on the FIRST equals sign — later ones belong to the value.

The terminator

A single newline ends the message. Nothing else terminates a line.

Unknown fields must be ignored, not rejected. That single rule is what lets the protocol grow without breaking your client.

The envelope: four keys on almost every message

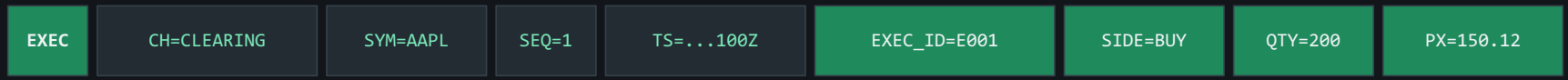
Learn these once and every RALF message becomes readable.

CH	**Channel**	CLEARING, DROP_COPY or AUDIT. On control messages it is what you asked for; on data messages it is where the event came from.
SYM	**Symbol**	The instrument. On control messages it is a filter; on data messages it is the actual symbol. * means all.
SEQ	**Sequence**	A channel-local counter that increases by exactly one per emitted line. Your gap-detection anchor.
TS	**Timestamp**	UTC ISO-8601 with milliseconds and a Z suffix — for example 2026-06-30T09:30:01.100Z.

SEQ is per channel — never global. Two channels can both be at SEQ=42 and mean completely different things.

Inside an EXEC — one executed trade

The envelope, then everything you need to book the trade.



FIELD	MEANING
EXEC_ID	The execution identifier — the engine's trade id.
MATCH_ID	The match identifier. In RALF1 it is always identical to EXEC_ID.
BUY_ORDER_ID / SELL_ORDER_ID	The resting and aggressing order identifiers on each side.
BUY_GW / SELL_GW	Which gateway each side came through — how you attribute the trade to a participant.
SIDE	The aggressor side: BUY or SELL.
QTY / PX	Executed quantity and execution price, in display units.

> A single trade is emitted once per channel — and each copy consumes its own sequence number on that channel.

EOD, and the small messages around it

Everything that is not an EXEC is short — and most of it carries only an envelope.

EOD — end of day, per symbol

```
EOD|CH=CLEARING|SYM=AAPL|SEQ=3
   |TRADE_COUNT=2|EXEC_COUNT=2
```

TRADE_COUNT trades seen for that symbol this session.
EXEC_COUNT is identical to it in RALF1.

Reading an EOD line

It is a marker, not a report. It tells you the symbol's trading session is closed and how many executions you should have counted — which makes it an excellent self-check on your own tally.

SNAP	Baseline. Envelope only — no book, no positions.
HB	HB|TS=… — sent only when nothing else was sent.
PING / PONG	Your probe, and the gateway's reply.
EXIT	EXIT|REASON=idle_timeout|TS=… — session over.

EOD is emitted once per symbol on all three channels.

Five parsing rules that will save you

Most first-attempt RALF clients fail on one of these.

1	**TCP is a byte stream**	One recv() is not one message. Buffer bytes and split on newline — always.
2	**Split on the FIRST equals**	A value may legitimately contain an equals sign. Only the first one separates key from value.
3	**Ignore unknown fields**	New fields will appear. A parser that rejects them will break on the next release.
4	**Ignore unknown message types**	Future event families are coming. Skip what you do not recognise instead of failing.
5	**Respect the line limit**	A line longer than 4096 bytes is refused with BAD_MESSAGE. Frame properly and this never happens.

SECTION 06



How errors are handled

RALF assumes things will go wrong: messages get missed, clients fall behind, connections drop. Every one of those has a defined recovery.

Sequence numbers and gap detection

SEQ is how you prove — not hope — that your copy of the day is complete.



SEQ=4 never arrived — that is a gap

```
gap when:
received_seq != last_seq + 1
```

Per channel

Each channel keeps its own counter, strictly increasing by one per emitted line.

Not global

Sequence numbers from different channels are independent — never compare or merge them.

Track per stream

Keep last_seq keyed on (CH, SYM) and check every single message you receive.

Detecting a gap is not an emergency — it is the system working. The next slide is what you do about it.

Replay — asking for what you missed

Reconnect with a checkpoint and the gateway resends from the journal.

```
HELLO|CLIENT=ext01|PROTO=RALF1|ROLE=CLEARING|LASTSEQ=1200
```

"Send me everything after 1200."

✓ REPLAY HIT

Your checkpoint is inside the retention window.

- Journalled events with SEQ > LASTSEQ are resent
- The stream continues live, seamlessly
- Nothing is lost

✗ REPLAY MISS

Your checkpoint is older than the journal.

- ERR|CODE=REPLAY_MISS arrives first
- A fresh SNAP follows it
- You MUST reset local state to that baseline

The default journal covers 86 400 seconds — one full trading day — which makes an intraday reconnect lossless in normal operation.

Every error code, and what to do about it

ERR carries a CODE and a human-readable DETAIL. The code is what you branch on.

```
ERR|CODE=REPLAY_MISS|DETAIL=requested seq outside retention
```

CODE	TYPICAL CAUSE	WHAT YOU DO
AUTH_REQUIRED	SUB sent before a successful HELLO	Send HELLO first — it must be the opening message
PROTO_MISMATCH	Wrong or missing PROTO value	Use PROTO=RALF1 exactly
ENTITLEMENT_DENIED	Role not permitted for that channel	Subscribe to the channel matching your role; AUDIT may access all
INVALID_CHANNEL	Unknown or empty CH value	Use CLEARING, DROP_COPY or AUDIT
REPLAY_MISS	LASTSEQ is outside the journal window	Accept the recovery SNAP and reset your local baseline
SLOW_CLIENT	You cannot drain the stream fast enough	Terminal — reconnect and process faster
BAD_MESSAGE	Malformed or oversized line (over 4096 bytes)	Fix your line syntax and framing

Two are terminal — SLOW_CLIENT and a failed handshake end the session. The rest you can recover from in place.

SECTION 07

Using RALF as a client



You can test the protocol by hand in under a minute, then build a robust subscriber in about forty lines.

The client checklist

Do these eight things and your client is production-shaped.

1. Open one long-lived TCP connection to port 5580
2. Send HELLO first — CLIENT, PROTO=RALF1, ROLE, and LASTSEQ
3. Wait for WELCOME; read REPLAY and HBINT from it rather than assuming
4. Subscribe only to channels your role is entitled to
5. Store the SNAP sequence per (CH, SYM) as your baseline
6. Buffer bytes and split on newline — never treat a read as a message
7. Check every SEQ for gaps and remember the last one you saw
8. On reconnect, send your last good SEQ as LASTSEQ and handle REPLAY_MISS

Try it in sixty seconds — no code required

Validate the line protocol by hand before you write a single line of client.

Option A — by hand

```
nc 127.0.0.1 5580
HELLO|CLIENT=ops01|PROTO=RALF1|
      ROLE=CLEARING|LASTSEQ=0
SUB|CH=CLEARING|SYM=*
```

Expect WELCOME, then SNAP, then live EXEC and EOD lines, with HB while idle.

Option B — ready-made

```
pm-ralf-spy --role CLEARING
```

The protocol spy performs the handshake, subscribes, and pretty-prints every line for you — no hand-typed messages at all. It is read-only, so it is safe to point at a live gateway.

Before either works: pm-engine must be running and publishing, pm-ralf-gwy must be running, and your symbols must exist in engine_config.yaml.

A minimal client, with no dependencies

Enough to prove the connection works. Note the buffering — that part is not optional.

```
import socket

sock = socket.create_connection(("127.0.0.1", 5580))
sock.sendall(b"HELLO|CLIENT=bot01|PROTO=RALF1|"
             b"ROLE=CLEARING|LASTSEQ=0\n")
sock.sendall(b"SUB|CH=CLEARING|SYM=*\n")

buf = bytearray()
while True:
    chunk = sock.recv(4096)
    if not chunk:
        break
    buf.extend(chunk)
    while b"\n" in buf:
        idx = buf.index(b"\n")
        line = buf[:idx].decode().strip()
        del buf[:idx + 1]
        if line:
            print(line)
```

The one mistake everyone makes

Never assume one recv() returns exactly one message. TCP is a byte stream — it may hand you half a line, or three lines at once. Always buffer and split on newline.

Or use the bundled library

```
from ralf_parser import (
  parse_ralf_line,
  build_ralf_line)
```

The shape of a real subscriber

Condensed from the bundled example — the four patterns that matter.

```
with socket.create_connection((HOST, 5580), timeout=5) as sock:
    reader = LineReader(sock)          # buffers, yields lines

    send(sock, "HELLO", {"CLIENT": "mybot", "PROTO": "RALF1",
                         "ROLE": "CLEARING", "LASTSEQ": "0"})
    welcome = parse_ralf_line(reader.recv_line())
    assert welcome.msg_type == "WELCOME"
    replay_sec = welcome.fields.get("REPLAY")

    send(sock, "SUB", {"CH": "CLEARING", "SYM": "*"})
    last_seq: dict[tuple[str, str], int] = {}

    while True:
        msg = parse_ralf_line(reader.recv_line())
        if msg.msg_type in ("EXEC", "EOD", "SNAP"):
            key = (msg.fields["CH"], msg.fields["SYM"])
            seq = int(msg.fields["SEQ"])
            prev = last_seq.get(key)
            if prev is not None and seq != prev + 1:
                handle_gap(key, prev, seq)   # reconnect w/ LASTSEQ
            last_seq[key] = seq
        elif msg.msg_type == "ERR":
            if msg.fields["CODE"] == "SLOW_CLIENT": break
```

1 Line buffering

A reader that turns bytes into whole lines.

2 Assert the handshake

Fail loudly if WELCOME does not arrive.

3 Per-stream sequence map

Keyed on (CH, SYM), checked on every message.

4 Terminal error branch

SLOW_CLIENT means reconnect — do not retry in place.

REFERENCE

Gateway configuration you should know about

The post_trade_gateway block in engine_config.yaml — read only by pm-ralf-gwy.

KEY	DEFAULT	PURPOSE
name	ralf-gwy01	Gateway id, echoed in WELCOME|GW=
bind_address	0.0.0.0	TCP listen address
port	5580	TCP listen port
replay_retention_sec	86400	Journal window; advertised as REPLAY
heartbeat_interval_sec	1	HB cadence; advertised as HBINT
idle_timeout_sec	5	Inbound-idle disconnect
max_client_queue	10000	Backlog before SLOW_CLIENT
allowed_roles	all three	Roles permitted to authenticate

For a single-host lab, bind to 127.0.0.1. For multi-host, use a controlled interface and restrict access at the network boundary.

```yaml
post_trade_gateway:
  name: "ralf-gwy01"
  bind_address: "0.0.0.0"
  port: 5580
  replay_retention_sec: 86400
  heartbeat_interval_sec: 1
  idle_timeout_sec: 5
  max_client_queue: 10000
  allowed_roles:
    - CLEARING
    - DROP_COPY
    - AUDIT
```

Generate it instead

pm-config-gen --post-trade-gateway writes this block alongside the main engine config, so you never hand-edit it.

Conformance, and one deviation to know

What a conforming RALF1 implementation must do — and where the current gateway differs.

A CONFORMING RALF1 IMPLEMENTATION

- Derives events solely from engine broadcasts — trade.executed becomes EXEC, system.eod becomes EOD
- Sends HELLO and WELCOME before any data, and rejects pre-auth data with AUTH_REQUIRED
- Enforces the role entitlement matrix on every SUB
- Applies per-channel SEQ semantics — there is no global sequence stream
- Advertises REPLAY and HBINT in WELCOME matching its own configuration

! KNOWN DEVIATION

Replay is entitlement-filtered, but not subscription-filtered.

Replay happens during HELLO, before any SUB is processed — so it cannot know your subscriptions yet. You may receive replayed events for symbols and channels you did not ask for. Filter them yourself on the client side.

> The normative reference is the RALF protocol appendix. Where the tutorial and the appendix disagree, the appendix wins.

Where to find more information

Four documents, each answering a different kind of question.

Post-Trade Dissemination (RALF)

`250-ralf-gateway.md`

The operational guide: quick start, configuration, channel descriptions, client examples and the operator runbook.

"How do I run and use it?"

Appendix — RALF Protocol

`930-app-ralf-protocol.md`

The normative wire reference: framing, message set, field tables, entitlements, sequencing and error codes.

"What is the exact rule?"

External Protocols Overview

`210-protocols-overview.md`

The map of ALF, BALF, CALF and RALF, with a quick selection guide for choosing between them.

"Which protocol do I want?"

RALF Protocol Spy

`251-ralf-spy-cli.md`

A read-only CLI that watches the raw feed in human-readable or JSON form — the fastest way to see it working.

"Can I just look at it?"

IN SUMMARY

Five things to remember

1	**A gateway is a controlled doorway**	It translates, authenticates, sequences and protects — so the engine never has to talk to the outside world.
2	**RALF is post-trade, and read-only**	It answers "what was executed", never "what is the price", and no client ever sends an order over it.
3	**Three channels, two event families**	CLEARING, DROP_COPY and AUDIT carry EXEC and EOD. Your role decides which you may see.
4	**One line, one message**	Type, then FIELD=VALUE pairs, terminated by a newline. Buffer your bytes and split on the first equals.
5	**SEQ is the safety net**	Track it per channel and symbol, detect gaps, and reconnect with LASTSEQ to recover from the journal.

Next step: run pm-ralf-spy --role CLEARING against a live gateway and watch the lines arrive.