HOW THE ENGINE DECIDES

Every rule between a dealer's price request and a delivered car: what is checked, in which order, with which setting, and what happens next.

1 QUOTE

Every price request is answered at once: priced between our cost and what this dealer accepts, or declined.

2 DISPATCH

A won order becomes a gig for freelance drivers: which cars, in which posting mode, with which windows.

3 CHAINS AND THE TRUST RUSH

Two cars for one driver where it fits, and a big dealer's car delivered today for its trust.

4 FEE CLOCK

What a gig pays: open low, hold, rise only near the deadline. A gig that is given back starts over.

5 EVENTS, REVIEW AND THE LIVE LOOP

What each message from the world changes, the review after every batch, and the loop that runs it.

the world sends it · a decision · a posting mode · learned from data · sent to the world · no order · kept on disk · goes on · way out · feeds a later decision · comes back

`name=value` the setting a rule reads, level 8

1 QUOTE

Every price request is answered at once: priced between our cost and what this dealer accepts, or declined.

the world sends it · a decision · a posting mode · learned from data · sent to the world · no order
kept on disk · goes on · way out · feeds a later decision · comes back
name=value the setting a rule reads, level 8

1.1 REQUEST AND FACTS

price_request EVENT
A dealer asks for a price: one car, or two in a trade-in. Up to three options (same day, next day, within 3 days), each with a list price and a deadline. The dealer chooses at respondBy.
Counted: requests in all and per dealer.

Read the dealer's note STEP
Free text in Finnish, Swedish or English becomes facts: not before, real pickup and delivery hours, an own deadline, manual gearbox, plates, summer tyres. When a fact repeats, the stricter value wins.
`notes=1`

Work out the real times STEP
- ready = the latest of respondBy, the pickup window's start and the note's not-before
- deadline = the option's, or the note's if earlier; it moves by whole days when the car is ready later
- electric car under 30 % charge on a route over 200 km: +40 min to charge

```
buffer = max(15 min, 15 % of the drive)
slack = last on-time pickup - buffer - earliest pickup
```
`bufferMin=15` `bufferShare=0.15`

Fair fee: the lane's fee, not the request's noise MODEL
A request's typical fee is the lane's fee × one factor per world, plus a noise of 13 % that no driver knows of. A table of 589 lanes gives the lane's fee; the world's factor is the median over the requests seen, trusted from the 21st request on.
```
fee = min(typical fee, cap × lane's fee)
```
The cap is 1.04: for the lateness model always, for the cost and the floor 0.95 on easy lanes, for the gig's fee clock 0.94 and 0.87 where drivers book at once (sheet 4). A typical fee under the cap is left alone.
`refCap=1.04` `refCapPrice=1` `refCapRisk=1` `easyPrice=0.95`

State, saved every 2 seconds KEPT
Beliefs, pending quotes, counts per dealer, the joined-late flag. Trust and request counts are rebuilt from the world's history after a restart.

1.2 DECLINE RULES, IN THIS ORDER

IF the note asks for summer tyres IF → yes
no ↓

IF the note shortens a site's hours IF → yes
and it is a trade-in or a private address, or no day of the next 8 fits pickup + drive + buffer inside the real hours with 45 min to spare (the pickup ends 20 min before closing).
`noteHours=1`
no ↓

IF a car is premium IF → yes
`premium=0`
no ↓

IF two cars (a trade-in) IF → no
`tradeIns=0`
yes ↓

DECLINE EXIT
Sent as a quote that cannot win: the first option at 4 × its list price. Nothing is learned from the dealer's answer to it.
Reason logged: summer tyres, shorter hours, premium car, trade-in, too little time, too risky, no price pays.
`declineMult=4.0`

Switched off at level 8 NOTE
Rules the engine has and does not use here: a second quote after a loss, the price-war watch, outage and snow rules, premiums for speed and same-day cars, the share cap, a cost per dealer, direct offers to drivers, a guard grace, a Monday risk factor, a reprice in place of a new post.

1.3 OPTIONS, LATENESS RISK, COST

FOR each option, fastest first: IF its slack is under 120 min IF → yes
the option is dropped: too little time.
`minSlackMin=120`
no ↓

Lateness risk of the option MODEL
The chance that this order arrives late: a logistic model of 38 features, fitted on our own practice weeks.
- drivers living within 30 and 80 km of the pickup and of the delivery
- slack, ready today, private stops, drive hours, distance, lead time
- our own unbooked orders within 80 km
- weekday, hour, the option, the fee against the list price
- orders won today, never booked, won in the last 2 hours

In a world with fewer than 720 drivers the risk counts 1.25 times: every weak week of the benchmark is in such a world.
`riskModel=4 (lane)` `thinRoster=720` `thinRisk=1.25`

Risk cap at this moment STEP
```
cap = 0.12 × 1.25 on a Friday × on-time guard
```
`riskMax=0.12` `riskFriday=1.25`

IF the risk is above the cap IF → next
the option is dropped: too risky.
after the last option ↓

IF no option is left IF → yes
no ↓

Cost of the order STEP
```
cost = fee × (1.2 - 0.25 × chain chance + risk) × 0.95 on a Friday + risk × (0.5 × list price + trust at stake)
```
fee: the fair fee of column 1.1. Chain chance: that the car shares a driver with a gig already open (sheet 3). In a world joined after day 1 with no order of ours: 1.1 for 1.2, and no Friday factor.
`costMult=1.2` `joinCost=1.1` `chainCostGain=0.25` `riskSlope=1` `fridayCost=0.95` `riskCost=0.5` `trustGuard=1`

1.4 PRICE

Price floor STEP
```
floor = 1.10 × fee + max(3 euros, 5 %)
```
`floorMult=1.10` `minMarginCents=300` `minMarginShare=0.05`

Market level for this request MODEL
The log of the price ratio the dealer accepts, as a sum of five beliefs, each a mean with a variance: the world (starts at 1.32 × list), today, this dealer, the pickup place, the delivery place.
```
level = sum of the means + ln(1 + 0.01 × trust)
```
`marketModel=2 (beliefs)` `beliefWorld=0.08` `beliefDay=0.07` `beliefDealer=0.04` `beliefPlace=0.04` `trustBoost=1`

Choose the price ratio STEP
```
win chance(r) = sigmoid((level - ln r) / width)
```
49 ratios from 4 widths under the level to 3 widths over it; those under the floor are skipped; kept is the one with the largest (r − cost ÷ list price) × win chance.
`beliefScale=0.07`

IF the win chance at that price is under 2 % IF → yes
`minWinChance=0.02`
no ↓

quote, before any other action SEND
Every option that is left is priced max(floor, list price × ratio): one ratio for all. The quote is kept as pending: dealer, places, day, level, ratio, width.

1.5 LEARN AND GUARD

order_won / order_lost EVENT
At respondBy the dealer chooses between our quote and the competitor's. Won: the order goes to dispatch (sheet 2).

Update the beliefs MODEL
For a pending quote only: a declined or unanswered request teaches nothing.
```
surprise = outcome - the win chance it had
```
The world, the dealer, both places and today each move by their own variance × surprise, and their variance shrinks: no fixed learning rate. Each morning today starts again at zero and dealers and places get a little variance back.
`beliefDrift=0.02`

Trust per dealer, 0 to 25 MODEL
+1 when an order's revenue arrives; halved by a late refund or an undelivered penalty. Each point lets the dealer accept a price 1 % higher.

On-time outlook MODEL
```
outlook = 1 - late / orders
```
orders = 30 + ours; late = 1.5 + refunded + the late chance of every open order. Without a driver, by the time to its last pickup: 1.0 at zero, 0.9 within 1 h, 0.6 within 3 h, 0.42 within 6 h, 0.32 within 12 h, 0.22 within 24 h, else 0.14. With a driver: 0.015.

On-time guard STEP
```
guard = clamp((outlook - 0.93) / 0.025, 0.15, 1)
```
The full risk cap at an outlook of 95.5 % or more, 15 % of it at 93.4 % or less. It acts before a late car shows in the account.
`guardFull=0.955` `guardFloor=0.93` `guardMin=0.15`

2 DISPATCH

A won order becomes a gig for freelance drivers: which cars, in which posting mode, with which windows.

the world sends it · a decision · a posting mode · learned from data · sent to the world · no order
kept on disk · goes on · way out · feeds a later decision · comes back
`name=value` the setting a rule reads, level 8

2.1 ORDER WON

order_won — EVENT
The dealer took our quote: one option, its deadline, its price. Handled in the same tick.

Book the order, apply the note's facts — STEP
Not-before, real hours, the note's deadline, manual gearbox, plates. An order never seen as a request (after a restart) gets a typical fee of ¾ of its price.

Which cars need a gig — STEP
Every undelivered car without a live gig, not yet collected, and past its retry time. Nothing is posted for a cancelled order, or while the history of a joined world is replayed.

IF exactly one car is free and it is not a trust-rush car — IF
a chain is tried first (sheet 3). Posted as a chain: done.

no, or no chain fits

Base fee and timing of the gig — STEP

```
base = fair fee × drive now ÷ drive as quoted
```

The fair fee is the typical fee under the cap of its lane (sheets 1 and 4). A dealer can send a car somewhere else after the order is won. At least 1 euro.

- first = the earliest pickup from now: site hours, window, not-before
- lastPickup = the world's last on-time pickup, or the engine's own when the note has a deadline or shorter hours
- buffer = max(15 min, 15 % of the drive)

`refCap=1.04` `bufferMin=15` `bufferShare=0.15`

2.2 POSTING MODE: FIRST THAT FITS

IF the note shortened the hours, one car, and a day fits — IF
a day on which pickup and delivery are both open and leave a pickup window of 45 min or more.

yes → DAY
no ↓

IF an earlier post of this car was refused — IF
it goes with the order's own windows only: TIGHT or LATE.

yes → TIGHT or LATE
no ↓

IF lastPickup − buffer is not before first, and more than 45 min away — IF
`finalMin=45`

no → TIGHT or LATE
yes ↓

IF it is a trust-rush car — IF
a big dealer's car that can still arrive today (sheet 3).
`trustRush=0.08`

yes → RUSH
no → BUFFERED

IF lastPickup is not before first and still ahead of now — IF

yes → TIGHT
no → LATE

In every mode — STEP

- pickup not before the retry time after a failed visit
- shorter hours but no day window: the pickup still gets a window inside the real hours of the next open day
- audience: everyone; manual licence, plates and charging ride on the stops

2.3 WHAT EACH MODE POSTS

DAY — MODE
One day's windows inside the real hours. Pickup: from the latest of ready, the pickup site opening and the delivery site opening − drive, to the earlier of closing − 20 min and arrive-by − drive. Arrive by closing − buffer, and by the deadline − buffer while that day is on time. Bookable until the pickup window ends; then the next day. A first day after the deadline: the late fee plan.

RUSH — MODE
The delivery window ends today: the site's last open minute − buffer. Bookable until the last pickup that still arrives today.

BUFFERED — MODE
The usual mode. The delivery window ends a buffer before the last minute the site takes the car by the deadline (no window of our own at a private address). Bookable until lastPickup − buffer.

TIGHT — MODE
The order's own windows: on time only without the buffer. Bookable until lastPickup. If the note's deadline is earlier than the order's, the delivery window ends at it.

LATE — MODE
No on-time pickup is left: the refund is owed. The world's windows run to its end.

2.4 THE WORLD'S ANSWER

post_gig — SEND
Stops (a pickup and a delivery per car), the audience, and the fee plan of its mode: up to 48 steps (sheet 4).

IF the post is accepted — IF
the gig is open in the drivers' app; events follow (sheet 5). If its car found another carrier while the post was on its way, it is withdrawn at once.

no ↓

IF refused because a driver took the car's earlier gig in the same minute — IF
nothing was wrong with the post: the car is not marked.
`raceSafe=1`

no ↓

Refused for another reason — STEP
The car is marked plain and posted again at once with the order's own windows only: the world always takes that.

```
2nd refusal on: wait 20 min × min(refusals - 1, 6)
```

The review brings the car back: it is never dropped.

3 CHAINS AND THE TRUST RUSH

Two cars for one driver where it fits, and a big dealer's car delivered today for its trust.

the world sends it · a decision · a posting mode · learned from data · sent to the world · no order
kept on disk · goes on · way out · feeds a later decision · comes back
`name=value` the setting a rule reads, level 8

3.1 MAY THE ORDER CHAIN

From dispatch: one free car, not a rush car — STEP

A chain puts the cars of two orders in one gig, each collected near where the one before is delivered. One driver moves two cars: more cars per driver, which is what the shortage is short of.

IF the order may chain — IF

- fewer than 2 chain attempts so far
- one car, not collected, not cancelled
- no private address at either end
- hours not shortened by the note, not a refused post, not a premium car
- not urgent: no deadline_close for it yet

`chains=1` `chainTries=2`

yes → 3.2 FIND A PARTNER

no → **The car is posted alone** — STEP

in the posting mode that fits (sheet 2).

Chain chance, used in the quote's cost (sheet 1) — MODEL

The chance that a driver books a chain within its attempt: a logistic model of 3,545 attempts. Hours at the wheel, road hours from the last delivery back to the first pickup, hop hours, first car ready today, hours the older order has waited, Monday, Thursday or Friday, drivers living within 30 km of the last delivery.

`chainCostGain=0.25`

3.2 FIND A PARTNER

Candidates — STEP

Every open single-car BUFFERED gig whose own order may chain too. The new car is tried before it and after it.

IF a pairing fits — IF

- every hop between two cars is 70 road minutes or less
- driving and hops together are 12 hours or less
- 180 min or more between the earliest start (07:00 when it is night) and the last start that keeps every car on time, buffer included
- the first pickup site opens by that last start

`chainHopMin=70` `chainSlackMin=180` `chainMaxCars=2`

no → 3.1 THE CAR IS POSTED ALONE

yes

The pairing with the shortest hop wins — STEP

→ 3.3 withdraw the partner's gig, post the chain

3.3 THE CHAIN, AND WHAT ENDS IT

withdraw the partner's gig, post the chain — SEND

One gig: car 1 collected and delivered, then car 2, each delivery with its buffer.

`base = 0.9 × the sum of both base fees`

The attempt lasts 90 min and never runs past the last start. Made between 22:00 and 07:00: the attempt and its fee clock count from 07:00, and it gets 60 min more.

`chainFeeShare=0.9` `chainTryMin=90` `chainNight=1`

IF the attempt ends unbooked, or the post is refused — IF

the chain is withdrawn and each car is dispatched again: into one more chain if it has an attempt left, else alone. Both cars still have time.

yes → 3.2 FIND A PARTNER

booked

IF a driver gives the booked chain back — IF

and it still fits and no car is urgent: withdrawn and posted again as a chain for 90 min, its fee clock from the opening fee. 73 % of these are booked again, at 0.81 of the typical fees. Else it is taken apart in the same round.

`chainRecoverMin=90`

yes → 3.3 withdraw the partner's gig, post the chain

kept

Booked and driven — STEP

Two cars delivered by one driver, on one fee.

3.4 TRUST RUSH

Why — NOTE

A dealer accepts prices 1 % higher for each order delivered to it on time, up to 25. The count moves at the delivery, so a big dealer's car that is ready today is posted to arrive today.

IF all of this holds — IF

- Monday to Thursday: on Friday no request is left to sell the trust into
- a one-car order, not rushed before, not a refused post
- no private address, no shortened hours, no deadline in the note
- the dealer's trust is under 25
- 20 requests or more seen, 8 % or more of them from this dealer
- the deadline is after today 20:00, and the delivery site is still open today
- 120 min or more from the first pickup to the last one that arrives today

`trustRush=0.08` `trustRushMin=120`

yes

RUSH — MODE

Posted to arrive today. The fee opens at 0.80, is 1.00 after an hour and creeps to 1.25 by the last pickup that arrives today: no deadline pressure. Never chained.

IF no driver booked it by then — IF

the review withdraws it and the car is posted again with its own deadline (BUFFERED). It has lost nothing but the hours. An order is rushed once.

4 FEE CLOCK

What a gig pays: open low, hold, rise only near the deadline. A gig that is given back starts over.

the world sends it · a decision · a posting mode · learned from data · sent to the world · no order
kept on disk · goes on · way out · feeds a later decision · comes back
`name=value` the setting a rule reads, level 8

4.1 THE BASE FEE AND THREE CLOCKS

The base fee follows the lane — MODEL

```
base = min(typical fee, cap × lane's fee) × drive now ÷ as quoted
```

- cap 1.04 on any lane
- 0.94 on middle lanes: 22 % or more of the lane's orders were booked within ten minutes
- 0.87 on easy lanes: 34 % or more

A table of 110 lanes. There a clock that opened lower was booked at a fee as much lower, and no later. The clock also needs the minute it starts and bookableUntil: the last minute a driver can still collect on time.

`refCap=1.04` `midCap=0.94` `midFrom=0.22` `easyCap=0.87` `easyFrom=0.34`

IF the car is urgent, TIGHT or LATE, or a tenth of the week's orders never found a driver — IF

the 1.04 cap again: a car under pressure cannot wait for a cheaper driver, and a week short of drivers is no week to try.

`rescueFrom=0.10`

1 · Discovery: look for the cheapest driver — STEP

```
0.80 × base at the post, +0.05 every 15 min, 1.00 after 60 min
```

`feeStart=0.8` `feeStep=0.05` `stepMin=15` `feeMid=1.0`

2 · Hold: a slow rise while the gig waits — STEP

```
1.00 → 1.25 in 6 steps, reached 240 min before bookableUntil
```

Only the hours from 07:00 to 22:00 count: at night hardly a driver looks.

`holdMax=1.25` `holdSteps=6` `holdAwake=1` `leadMin=240`

3 · Deadline pressure — STEP

```
1.00 → 2.00 from 240 to 45 min before bookableUntil, in 6 steps
```

Nothing before that; 2.00 at once inside the last 45 min.

`feeMax=2.0` `leadMin=240` `finalMin=45` `urgentSteps=6`

4.2 THE PLAN, BY POSTING MODE

The fee plan — STEP

```
fee(t) = base × max(discovery, hold, pressure)
```

Steps on a 5-minute grid, at most 48, each later and higher than the one before; never under 1 euro. Capped at 3 × the price of the orders the gig carries, a late plan at 2 ×.

`onTimePriceCap=3` `latePriceCap=2`

BUFFERED — MODE

All three clocks, toward lastPickup − buffer.

DAY — MODE

All three clocks, toward the end of the day's pickup window. The late plan when that day is after the deadline.

CHAIN — MODE

All three clocks on 0.9 × the two typical fees, toward the last start that keeps both cars on time. Bookable only until the attempt ends.

RUSH — MODE

Discovery and hold only: 1.25 at the last pickup that arrives today. After the rush the car still has its own deadline.

TIGHT — MODE

```
2.00 × base at once, one step
```

LATE — MODE

```
1.00 × base, flat for 480 min
```

A higher fee did not bring a late car in sooner. Then 8 steps up to the lower of 4 × base and 2 × the order's price, reached 600 min before the world ends: an undelivered car costs far more than any fee.

`lateStart=1` `lateMax=1` `lateRampMin=480` `lastMult=4` `lastPriceShare=2` `lastLeadMin=600`

4.3 CLOCKS THAT START OVER

IF a BUFFERED gig is posted between 22:00 and 07:00 with 360 min or more of room after 07:00 — IF

it shows 0.80 all night and its clock starts at 07:00.

`nightClock=1`

yes → BUFFERED

gig_dropped / no_show — EVENT

A driver gives a booked gig back. Its fee went on creeping while the driver held it.

IF it is a chain that still fits — IF

a new chain attempt of 90 min, from the opening fee (sheet 3).

`chainRecoverMin=90`

yes → CHAIN

no ↓

IF it is a BUFFERED gig of one order, not urgent, not collected, with 360 min or more left to book — IF

withdrawn and dispatched anew: the fee opens at 0.80 again, and the car may go into a chain. Given back, such a gig used to return at 1.09 × the typical fee against 0.93 at its first booking.

`dropFreshMin=360`

yes → BUFFERED

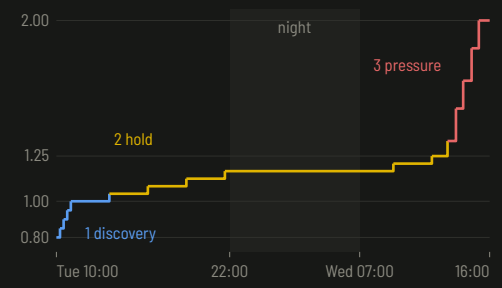
no ↓

ELSE: plan again from now — STEP

The plan is rebuilt for the time that is left. The fee never falls below what the app shows now. The same on deadline_close, when the last pickup moves.

4.4 ONE GIG, HOUR BY HOUR

Base fee 100 euros, posted Tuesday 10:00, bookable until Wednesday 16:00 — NOTE



1 · DISCOVERY: +0.05 EVERY 15 MIN	
Tue 10:00	80
10:15 · 10:30 · 10:45	85 · 90 · 95
11:00	100
2 · HOLD: 6 STEPS, AWAKE HOURS ONLY	
Tue 13:40 · 16:20 · 19:00 · 21:40	104 · 108 · 113 · 117
Wed 09:20 · 12:00	121 · 125
3 · PRESSURE: THE LAST 240 MIN	
Wed 13:05 · 13:40 · 14:10	133 · 151 · 167
14:45 · 15:15	185 · 200

The fee in euros. Most bookings come in the first hours, at 0.8 to 1.05 × the typical fee.

5 EVENTS, REVIEW AND THE LIVE LOOP

What each message from the world changes, the review after every batch, and the loop that runs it.

the world sends it · a decision · a posting mode · learned from data · sent to the world · no order
kept on disk · goes on · way out · feeds a later decision · comes back
`name=value` the setting a rule reads, level 8

5.1 GIG EVENTS

gig_viewed — EVENT
Counted per gig and per driver.

gig_booked — EVENT
The gig has a driver and a fee. If the car was posted again meanwhile (the booking raced a repost), the newer open gig is withdrawn.

gig_dropped / no_show — EVENT
The gig is open again. A chain gets a new attempt; a single gig with room a fresh fee clock; else its plan is rebuilt from now (sheets 3 and 4).

gig_expired — EVENT
Its cars are dispatched again at once.

visit_failed — EVENT
The car comes off its gig and is dispatched at once, without the cause:

- not_ready, closed: not before now + 120 min
- licence: a manual licence is required
- no_plates: the driver brings plates

`retryAfterMin=120`

gig_started / picked_up / delivered — EVENT
A collected car is never reposted or chained. A delivered car is done, late if after the deadline. A gig ends with its last car.

5.2 ORDER EVENTS

deadline_close — EVENT
The order is urgent: its last on-time pickup is updated and its open gigs are planned again, which starts the pressure. No chain and no fresh clock for it from now.

order_changed — EVENT
The dealer changed the plan. The facts are applied again; open gigs are withdrawn and posted again so that windows, buffer and fee fit. Booked gigs are left alone.

order_cancelled — EVENT
Its cars leave their gigs. A gig the world withdrew or released: its other cars are dispatched again. A gig that goes on, goes on with them.

ledger line — EVENT
- revenue: the dealer's trust +1, up to 25
- late_refund, undelivered_penalty: trust halved; the order counts as late in the outlook
- driver_fee, failed_visit, release_fee, waiting, fine: booked to the order

Read, not acted on — NOTE
forecast (stored), driver_late (logged), release.

5.3 REVIEW, AFTER EVERY BATCH

IF an open gig is past its bookableUntil — IF
it is withdrawn and its cars are dispatched in the mode that fits now:

- BUFFERED becomes TIGHT, then LATE
- a chain comes apart: each car one more chain if it has an attempt left, else alone
- RUSH goes on with its own deadline
- DAY moves to the next day

Every open order without a gig is dispatched — STEP
A car that waited for its retry time comes back here: no car is ever left without a gig.

Dispatch — STEP
One free car: a chain is tried, else it is posted alone in its mode (sheets 2 and 3).

the outbox — SEND
quote, post_gig, withdraw_gig, reprice_gig. Quotes leave first.

5.4 THE LIVE LOOP

Wait for a final — STEP
Asked every 10 seconds.

Join the world — STEP
The full history and a snapshot. The history is replayed without sending anything; orders and gigs come from the snapshot; the saved state is loaded. Then every open request is quoted and every car without a gig is posted.

Joined after day 1 with no order of ours: the world is cold, cost × 1.1.

`joinCost=1.1`

Tick, every 150 ms — STEP
`clock → events → handle in order → review → send`

The clock is read every 500 ms; events with a 3 s timeout.

actions, in batches of 50 — SEND
5 s timeout. A refused batch is sent again one action at a time. The state is saved every 2 s.

IF reads fail for 12 s, or any error or panic — IF
pause 0.5 s, doubling up to 5 s, and join again: the engine rebuilds itself from the world.

yes → Join the world

IF the clock is at the world's end and a snapshot confirms it — IF
stop, and write the result.

no → Tick, every 150 ms