

Resistance-Timer Exit: Simulation Findings

Research-only exit concept, tested against the 2026-09-18 replay via `simulate.py`. Not wired into the live bot. Prepared for review.

Question tested

Proposed idea: once price stops making new highs, check whether trade imbalance is still positive (buyers still active). If so, treat that as “at resistance” and start a fixed real-time recovery window. If price hasn't made a new high by the time that window elapses, exit. This is a deliberately simpler alternative to the existing multi-stage PRF (Price Response Failure) state machine (`exit_prf_experimental.py`) built the previous session around the same GLOO trade (2026-09-18, entered \$4.93, peaked \$5.58 on repeated +90-100% buy imbalance that never produced a new high, then broke down — the real bot's exit layers stayed silent until 18:17).

Implementation

New module: `exit_resistance_timer_experimental.py`. Wired into `simulate.py` as its own toggle (`--resistance-timer-exit`), independent of the existing giveback / fast-slope / PRF experimental layers, so its effect can be isolated. Two tunables exposed via CLI for this test:

- `--resistance-timer-window` — `recovery_window_seconds` (default 60)
- `--resistance-timer-imbalance` — imbalance threshold required to count as “still positive” (default 0.0, i.e. any positive read; PRF's own “strong buying” bar is 0.70)

Mechanics: price stalling at its own peak (within a 0.15% band) while imbalance clears the threshold starts a wall-clock timer (measured against real poll timestamps, not poll counts, since cadence isn't perfectly constant). A fresh new high at any point cancels the timer outright — same reversible convention as PRF. If the timer expires with no new high, the position exits.

Method

Full-day replay of 2026-09-18 through the real entry/exit code paths (`smart_engine.py`, `exit.py`, `stream.py`'s `SymbolBuffer`) driven off the actual logged poll timestamps from that session (`data/decisions/2026-09-18.jsonl`, 4606 real polls), same fidelity-tested harness used for all prior exit experiments. `max_positions=3` throughout, matching the day's real configuration. Baseline is the real `exit.py` alone, no experimental layer.

Results

Configuration	Trades	W / L	Total P/L
Baseline (no resistance-timer exit)	10	5 / 5	\$559.56
60s window, imbalance > 0	18	11 / 7	\$470.09
120s window, imbalance > 0	17	11 / 6	\$502.29
60s window, imbalance ≥ 0.70	17	8 / 9	\$247.57

120s window, imbalance ≥ 0.70	17	11 / 6	\$521.06
------------------------------------	----	--------	----------

Every tested configuration finished the day below the \$559.56 no-extra-exit baseline. 60s + ≥ 0.70 was the worst result tested.

Why the loose version (imbalance > 0) underperforms

The trigger condition is too easy to satisfy: any positive imbalance tick plus “no literal new print above the exact prior peak” is common during ordinary consolidation on a 5-second poll cadence, not just during real reversals. It fired on nearly every profitable position in the session, not only GLOO:

Position	Baseline result	60s-timer result
FWDI (13:37 entry)	held to EOD: +12.79% / \$260.13	cut 13:49: +2.35% / \$47.84
SBET (13:42 entry)	held to EOD: +6.99% / \$142.13	cut 14:25: +3.25% / \$66.13
SECZ	+9.17% held to EOD	cut similarly

Direct trace evidence: GLOO's 60s-timer exit fired at **0.00% giveback** with imbalance still at **+86%** — buyers still clearly in control, not a reversal, just a stock sitting flat at its peak for under a minute. Widening the window to 120s reduced (but did not eliminate) this: FWDI's clip moved from +2.35% to +2.65%, still nowhere near its +12.79% baseline outcome, and one FWDI exit at 120s still fired with imbalance at +47%.

Why raising the imbalance bar to ≥ 0.70 doesn't fix it either

Requiring a stronger buying read before the clock can even start makes the timer arm less often, not just less falsely. At 60s + ≥ 0.70 , this delayed detection of GLOO's second position that day (14:54 entry) enough that the timer never armed before the real bot's own deterioration layer took over — the same slow-reacting exit path the whole resistance/PRF line of work exists to improve on — and the position rode out to a real **-4.92% / -\$100.05** loss at 18:17. Tightening the trigger to avoid false positives cost the one catch this design is meant to make. At 120s + ≥ 0.70 the timer did arm in time (exit at 14:39, +4.33%) and the day recovered to \$521.06 — still below baseline, and now dependent on two tuned constants (window length and imbalance floor) both fit to this single day.

Conclusion

This is the same open tension already flagged by the PRF work and by Bug #3's original memory entry: no combination of a flat time window and a flat imbalance threshold cleanly separates “real reversal” from “normal pause in an intact breakout” in this bot's small-cap universe. Every lever that makes the exit fast enough to catch a genuine reversal (like GLOO) also clips genuine winners that pause before continuing (FWDI, SBET, SECZ); every lever loose enough to let winners run arrives too late for the reversal it's meant to catch. Across five configurations tested (the original PRF state machine plus these four resistance-timer variants), none has beaten the no-extra-exit baseline on the one day available.

Recommendation

- Do not wire this exit into the live bot in its current form.

- Both configurable knobs (recovery_window_seconds, resistance_imbalance_threshold) are single-day fits and should be treated as unvalidated, same convention already used for exit_prf_experimental.py's giveback gate.
- If this direction gets revisited: scale the recovery window or imbalance requirement by something that reflects the stock's own volatility (e.g. ATR-relative, per the original Bug #3 proposal) rather than a flat constant, and validate against more than one trading day before treating any threshold as settled.

Code: exit_resistance_timer_experimental.py (new). Wired into simulate.py via --resistance-timer-exit, --resistance-timer-window, --resistance-timer-imbalance. Not imported by monitor.py or any live code path.