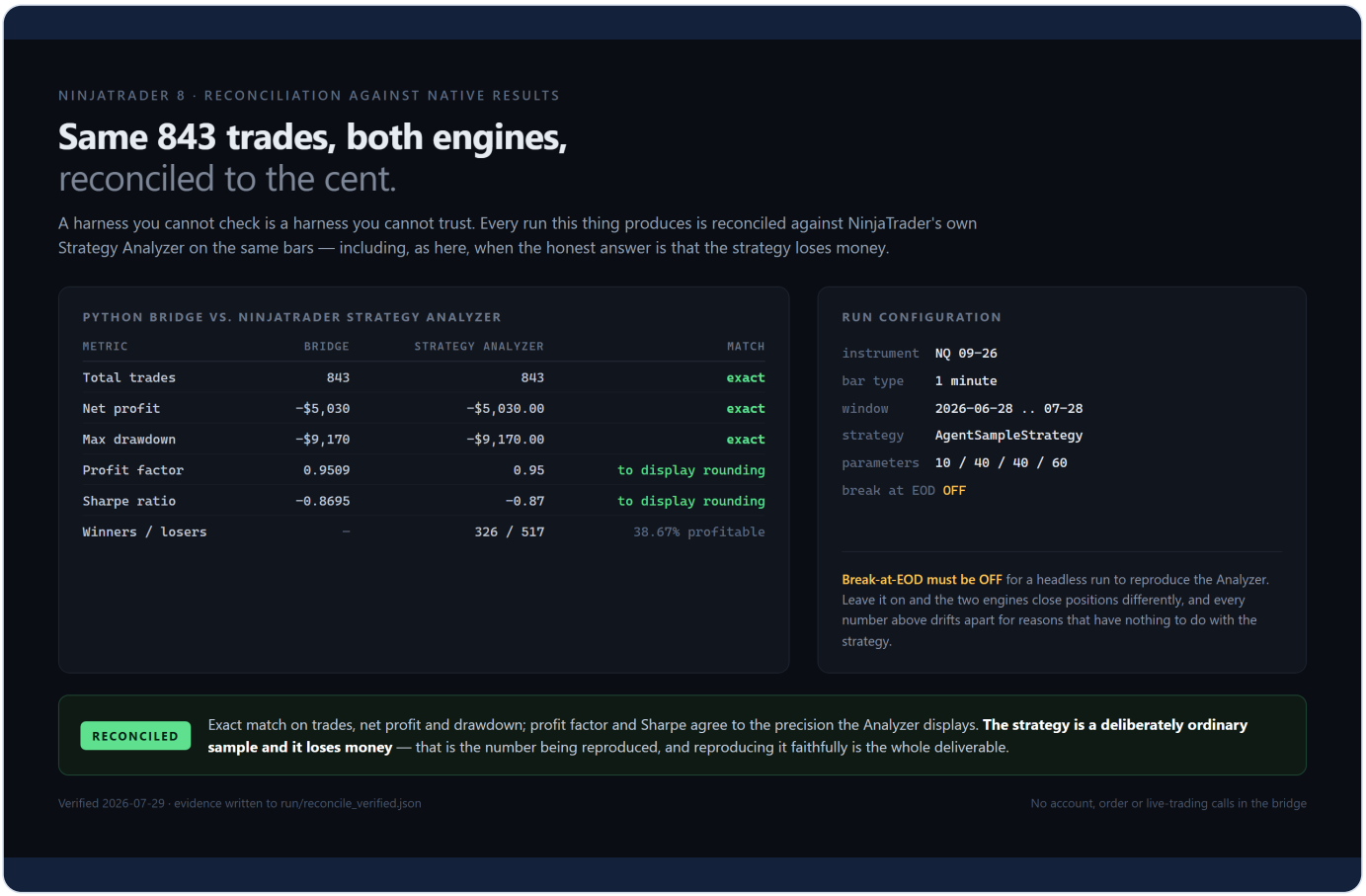
SANITIZED APPLICATION PACKET

Automation work samples

Reliable workflows, observable failure paths, and verified outputs.

I build and troubleshoot Python, C#, API, webhook, and AI-assisted systems around explicit acceptance cases. This packet uses only non-client-specific test artifacts and contains no credentials, customer data, account identifiers, proprietary client logic, or live trading access.

$1,220	54 / 54	843
verified Fiverr earnings	campaign points completed	trades reconciled exactly



EXACT RECONCILIATION

Two independent result paths matched across 843 trades, including total trades, net result, drawdown, and display-level ratios. The sample intentionally loses money; the deliverable is faithful reproduction, not a performance claim.

SAMPLE 01

Compiler failures became structured feedback.

A controlled C#/Python loop writes or mutates a strategy, invokes a headless compile, returns file/line/column diagnostics as JSON, and only continues when the result satisfies the defined gate.

2 ERRORS -> 0 | 0.394 S CLEAN COMPILE | NO ORDER OR ACCOUNT CALLS

What this proves

Machine-readable failures: errors are returned to the coding loop as structured data rather than hidden in a GUI.
Bounded execution: the bridge exposes a whitelist and contains no order or account calls.
Separated acceptance: a clean compile is necessary, but promotion still depends on downstream test and validation gates.
Operator handoff: each iteration can be versioned, logged, and reproduced.

Transferable pattern: the same loop applies to APIs, CRM automations, and AI workflows: capture a failure in a stable schema, route it to a bounded repair step, retest, and require evidence before release.

SAMPLES 02 AND 03

Fast execution is useful only with honest gates.

The campaign runner completed every configured point, then rejected all 54 because none met the sample-size floor. The workflow treats a plausible-looking result as insufficient evidence until the predeclared gate passes.

54 / 54 COMPLETED

0 / 54 PROMOTED

Lead-response automation - transferable implementation pattern

A separate Python/Flask kit applies the same reliability discipline to inbound business leads. The browser demo is illustrative; production endpoints and credentials are configured only for the buyer's actual systems.

1. Intake
Form, webhook, or API payload

→

2. Validate
Normalize fields and reject bad input

→

3. Deduplicate
Stable key blocks repeat sends

→

4. Route
Respond, log, and alert the owner

Failure path
Invalid payload -> no send; error is logged for review.

Retry path
Duplicate key -> no second response or duplicate lead row.

Implementation: Python, Flask, CSV-based durable logging, SMTP/email, configurable templates, and business-hour controls. **Security boundary:** secrets stay outside the artifact and are supplied through scoped environment configuration.