

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

verified Fiverr earnings

54 / 54

campaign points completed

843

trades reconciled exactly

NINJATRADER 8 - RECONCILIATION AGAINST NATIVE RESULTS

Same 843 trades, both engines, reconciled to the cent.

A harness you cannot check is a harness you cannot trust. Every run this thing produces is reconciled against NinjaTrader's own Strategy Analyzer on the same bars — including, as here, when the honest answer is that the strategy loses money.

METRIC	BRIDGE	STRATEGY ANALYZER	MATCH
Total trades	843	843	exact
Net profit	-\$5,030	-\$5,030.00	exact
Max drawdown	-\$9,170	-\$9,170.00	exact
Profit factor	0.9509	0.95	to display rounding
Sharpe ratio	-0.8695	-0.87	to display rounding
Winners / losers	-	326 / 517	38.67% profitable

RUN CONFIGURATION

instrument NQ 09-26

bar type 1 minute

window 2026-06-28 .. 07-28

strategy AgentSampleStrategy

parameters 10 / 40 / 40 / 60

break at EOD OFF

RECONCILED

Exact match on trades, net profit and drawdown; profit factor and Sharpe agree to the precision the Analyzer displays. **The strategy is a deliberately ordinary sample and it loses money** — that is the number being reproduced, and reproducing it faithfully is the whole deliverable.

Verified 2026-07-29 · evidence written to run/reconcile_verified.json

No account, order or live-trading calls in the bridge

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.

NINJATRADER 8 - AUTONOMOUS RESEARCH LOOP

The agent writes NinjaScript, compiles it, reads its own compiler errors, and fixes them.

Headless compile against NinjaTrader's shipped assemblies. No NinjaScript editor, no F5, no screen scraping. Errors come back as structured JSON with file, line and column, which is what makes them repairable by a machine instead of readable by a person.

LIVE RUN — 2026-07-29

```
$ python nt8_compile.py AgentSampleStrategy.cs

X compile failed  exit 1 · 2 errors · 0.350 s

CS0103 line 77, col 28
The name 'RsiThreshold' does not
exist in the current context

CS1501 line 80, col 8
No overload for method 'CrossAbove'
takes 2 arguments

... diagnostics returned to the coding agent ...

$ python nt8_compile.py AgentSampleStrategy.cs

✓ compile ok  exit 0 · 0 errors · 0.394 s
```

WHAT THE LOOP DOES PER ITERATION

- 1 write / mutate the strategy
- ↓
- 2 headless compile
- ↓
- 3 errors? → structured JSON
→ back to step 1
- ↓
- 4 backtest via NT8's own engine
- ↓
- 5 acceptance gate → keep or bin
- ↓
- 6 version, log, next candidate

Safety is structural, not a setting. The bridge holds a **whitelist** and contains **no order or account calls at all**, so the research side cannot reach a live account through it.

Compile timings are warm-cache medians over repeated live runs on the same machine.

NinjaScript · C# · Python

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.

Joshua Hernandez | jblaz6335@gmail.com | Palm Desert, California

2

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.

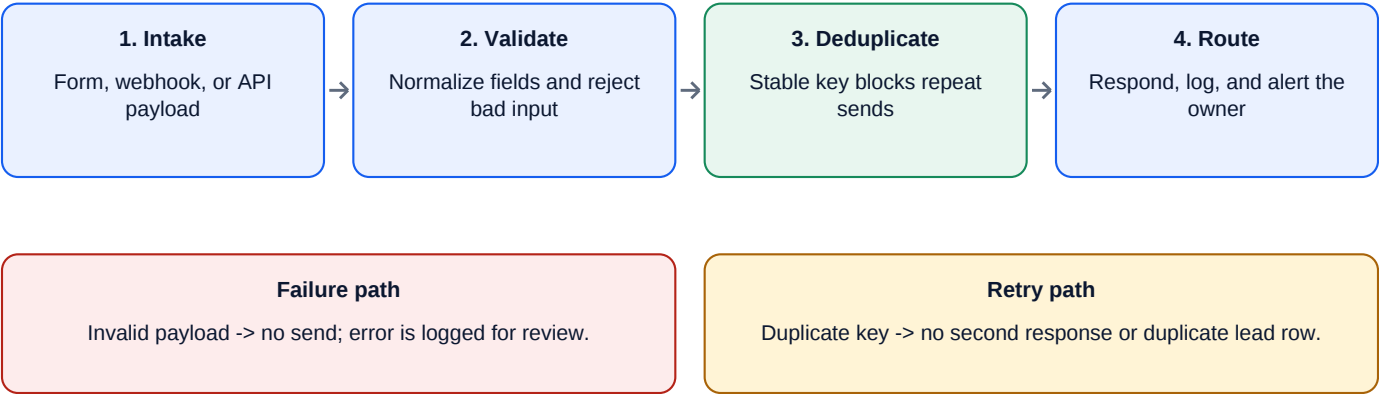


54 / 54 COMPLETED

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.



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.