EXAMPLE 02

Investigate a wrong decision

The GBP 120 order is incorrectly accepted against an unchanged GBP 100 limit. The fault is deliberate: a comparison in the application has been reversed. Follow the recorded values to the rule, then check the correction against the same inputs.

1 rule corrected · 7 wrong decisions to 7 correct decisions



Illustration of the demonstrated workflow.

Observed result

One comparison was corrected. Against the same seven inputs, the result changes from seven wrong decisions to seven correct decisions. The GBP 120 order is now rejected against its GBP 100 limit, while equality and below-limit orders remain accepted.

Case or component	Recorded result
GBP 120 / GBP 100 limit	Faulty ACCEPTED / corrected REJECTED
GBP 120 / GBP 150 limit	Faulty REJECTED / corrected ACCEPTED
GBP 120 / GBP 120 limit	Faulty REJECTED / corrected ACCEPTED
Required outcomes matched	Faulty 0 of 7 / corrected 7 of 7

Controlled seeded-fault demonstration using an earlier three-node fixture, not Example 01's new four-node application. The fault and correction were known in advance; this is not blind AI discovery.

See the recorded fault: two-step HTML investigation reel

Download the matching investigation bundle

To reopen the investigation in the Analyser, save the HTML reel and bundle in the same folder.

FOLLOW THE DEMONSTRATION

4-stage walkthrough

1. What you see

A GBP 120 order is ACCEPTED against a GBP 100 limit. The recorded value and limit agree with the input; the decision contradicts the rule.

2. What it means

The supplied LimitCheck source uses greater-than. It marks an above-limit order as within the limit, explaining the recorded contradiction.

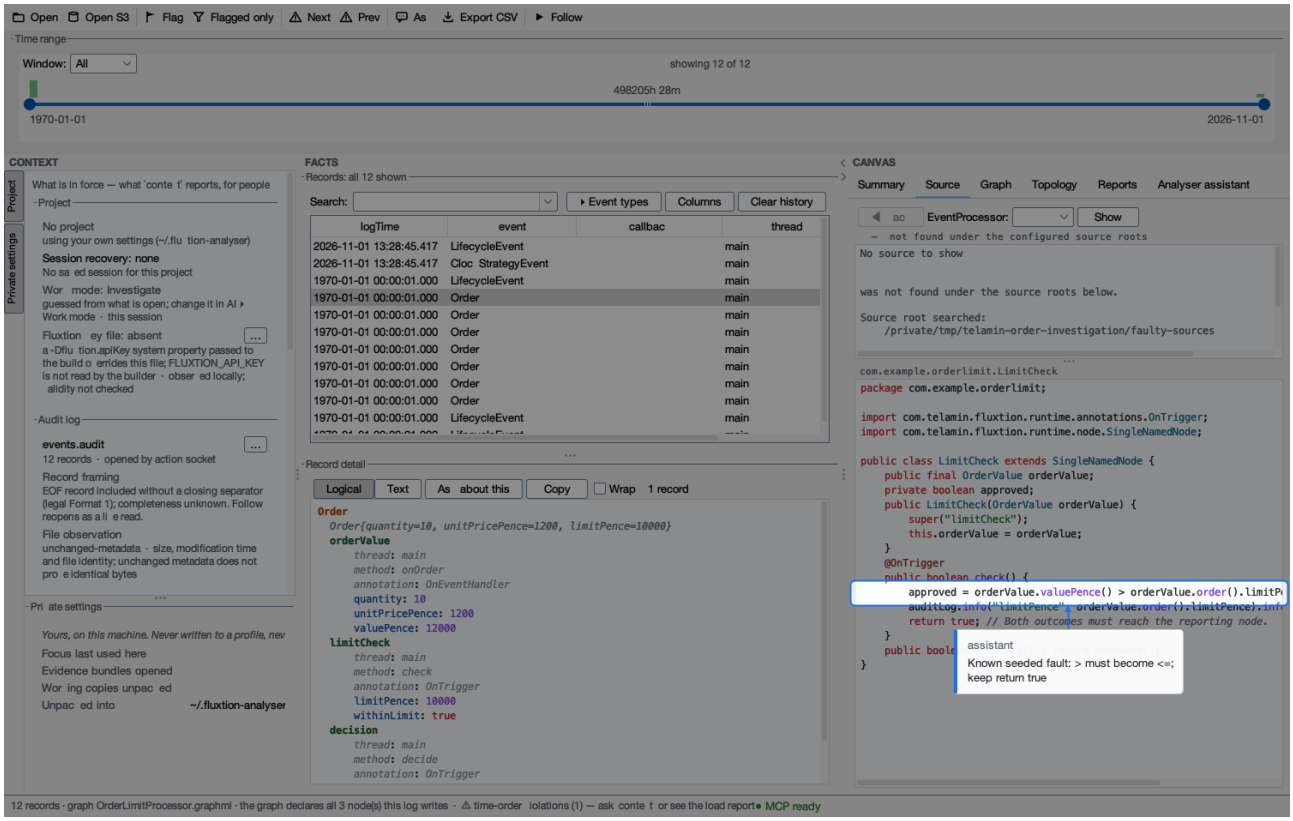
3. What to change

Restore less-than-or-equal in the limit check. Keep downstream notification so both accepted and rejected outcomes still reach the decision.

4. How to verify

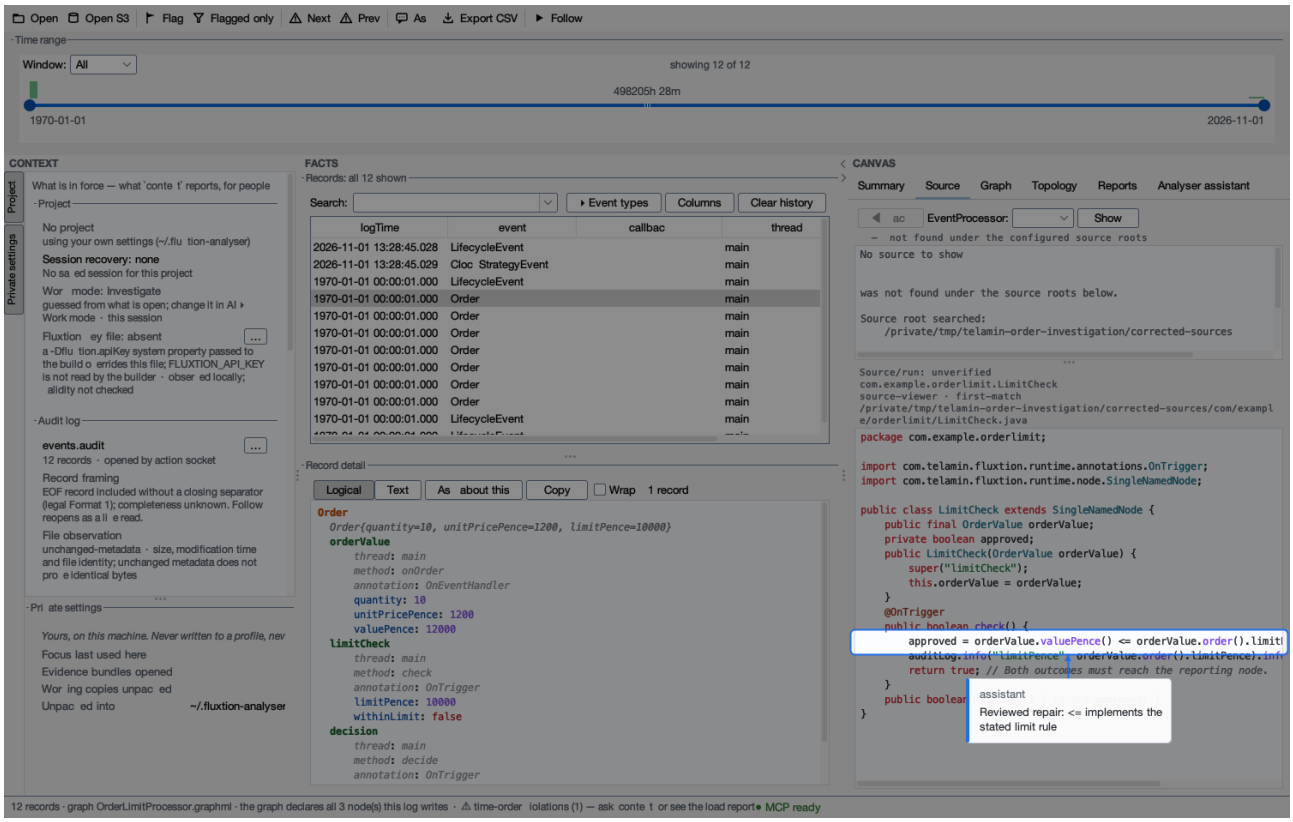
Run the same seven inputs. The faulty version matches none of the required outcomes; the corrected version matches all seven, including equality and both sides of the boundary.

Recorded acceptance beside the supplied limit-check source.



VERIFY THE CHANGE

The corrected run rejects the same £120 order.



The same input now records a rejected outcome. The native fixture checks all seven required decisions, including equality and both sides of the limit. The genuine exported bundle preserves the recorded fault and saved walk; it does not execute the correction.

An evidence bundle is a portable investigation created by the Analyser from selected records, the graph and a saved walkthrough. A person and their AI assistant guide the investigation. The bundle can be reopened; the reel and PDF are for reading and sharing.

What is an evidence bundle?

Why this matters

The before-and-after result is measurable: seven wrong decisions become seven correct decisions for the agreed cases. A reviewer can follow the recorded contradiction, inspect the supplied source and check the correction without relying on a confident narrative.

Apply it to your process

Choose a real failure question and agree the required result before investigation. A useful pilot should establish which inputs and state can be captured, which observations are missing, and whether a second reviewer can reproduce and assess the proposed correction.

Discuss a five-stage pilot

TECHNICAL APPENDIX

Reproduce this result

Unzip the source package and run from its order-check directory with Python 3, JDK 21 and the separately supplied reviewed runtime.

```
python3 run.py \
  --runtime /path/to/fluxtion-runtime-1.1.3.jar \
  --out results
```

The runner verifies the runtime hash, builds the included source and checks the captured outcomes. The output directory must be new or empty. No model or generation request is made.

Source and reproduction package for this example

Released Analyser 1.40.0 opened the final audit, linked the supplied graph and source context, and exported a genuine .fexp investigation. A separate reviewer reran the native fixtures on the same machine.

Runtime prerequisite: com.telamin.fluxtion:fluxtion-runtime:1.1.3. Download the JAR without an account; the runner checks its SHA-256. The source ZIP includes RUNTIME-PREREQUISITES.txt with the complete download command.

Download runtime JAR | Published POM: declares GNU AGPL version 3, separate from Apache 2.0 demo code.

SHA-256: 2bd45eb62409403422d5a676a7ad177f1984aa80dc399c20ceb6b467eab4b775

Detailed evidence

faulty/decisions.csv

corrected/decisions.csv

manifest.json

faulty/events.audit

corrected/events.audit

Open the genuine Analyser evidence bundle

Actual tool requests and responses

Capture method, reopen instructions and limits

All exported PDFs, screenshots and verification receipts

Scope and interpretation

This is a known seeded application defect. The corrected source existed before the investigation; Codex reviewed it and a separate reviewer reran the fixture, but no human repair approval or blind-diagnosis claim is made. No novice-usability, time-saving or token-saving result was measured.

The tool transcript records actual local REST actions; it is not an MCP-transport or local-model-inference demonstration. Source navigation uses supplied context and does not prove deployed-binary identity.

The bundle reopened on the same machine in a disposable workspace, not a second machine. Sources and flags do not travel in it; the source ZIP and separate exported PDF supply those materials.

Audit timestamps jump from initialization wall time to a fixed fixture clock. Analyser reports out-of-order records and unknown stream completeness. This investigation uses record indexes and values, not a time-window or production replay claim.

Ordinary unsigned member hashes detect changed bytes, not sender authenticity, universal correctness or certification.

The native runner builds a reversed greater-than comparison and the original less-than-or-equal correction. Each revision receives the same seven inputs in a fresh processor with a fixed application clock after initialization. This is a fixture-driven regression comparison, not replay of a recorded production incident. Screenshots may clip long source lines; the package contains the complete supplied class.

This seeded fault uses the earlier released three-node order fixture. It is separate from the freshly authored four-node implementation in Example 01; the business rule and checked boundary cases are the same.

Demo code is Apache 2.0 under owner authorization dated 1 November 2026. Runtime, third-party and general customer-output terms remain separate. Source and artifact hashes compare bytes; they do not authenticate a sender or certify correctness.

Deployment and data flows | Applicable product rights