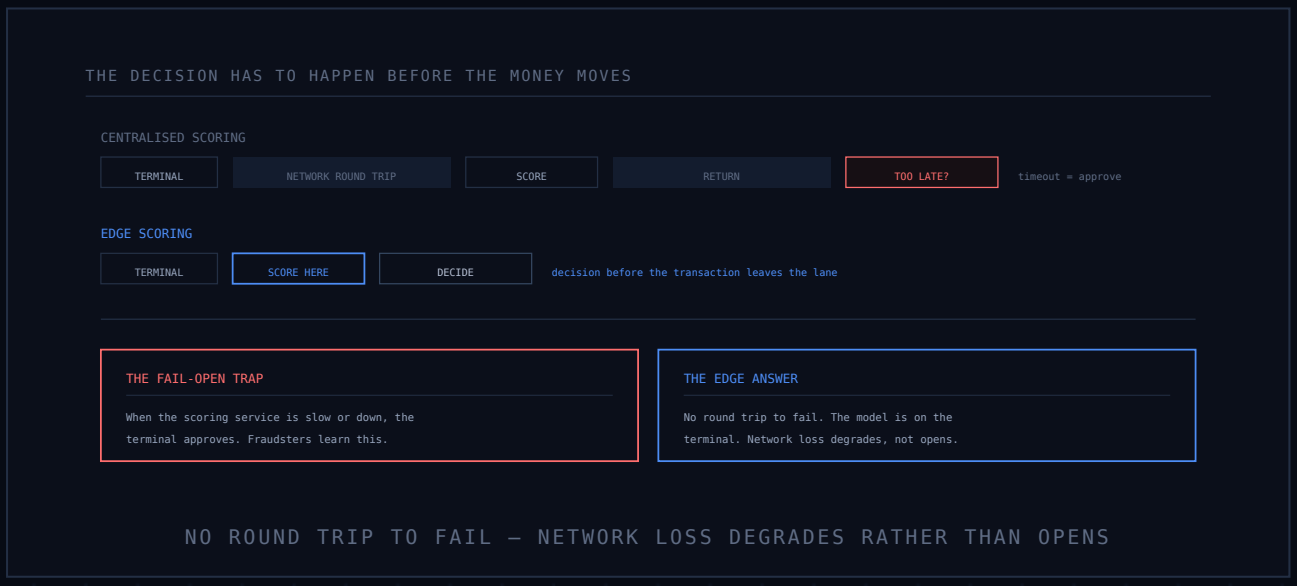
By the Time the Model Answered, the Money Had Already Left.

Fraud scoring in most institutions happens somewhere else. The terminal takes the transaction, sends it to a scoring service, and waits. The model is excellent. The features are rich. The problem is the waiting, because the transaction has a latency budget measured in milliseconds and the round trip consumes most of it before the model has seen anything.

So the budget gets managed. Timeouts get set. And a timeout in a payments path has exactly one sensible default, which is to approve — because a terminal that declines every transaction when the network hiccups is a terminal that gets removed from the lane by lunchtime. **Which means the institution's fraud control has a documented failure mode of "let it through", and the people attacking it are not unaware of this.**

We built fraud detection that runs on the terminal itself. The model scores the transaction at the point of sale, in the lane, before the transaction leaves the device. There is no round trip to be slow, and no round trip to fail.

WHERE THE DECISION HAPPENS



WHAT WE BUILT

01

Inference on the terminal

The model runs on the point-of-sale device. Scoring completes within the transaction's own latency budget rather than competing with it.

02

Fail-degraded, not fail-open

Losing the network costs the terminal its freshest intelligence. It does not cost the terminal its ability to decide.

03

Local data, local decision

Transaction data is scored where it is created. Sensitive detail does not traverse a network to be evaluated and then traverse it back.

04

Fleet model management

A model on ten thousand terminals is ten thousand deployments. Versioning, rollout and rollback were engineering problems in their own right.

05

Constrained-device engineering

A POS terminal is not a server. The model had to fit the silicon that was already in the lane, not the silicon we would have chosen.

+

Central intelligence, local action

The edge decides; the centre learns. Cross-terminal patterns still require a view the terminal does not have.

WHY LATENCY IS A SECURITY CONTROL

Fraud detection is usually discussed as an accuracy problem. It is at least as much a `deadline` problem, and the deadline is not negotiable: the decision must be made before the transaction completes, or it is not a decision. It is a report.

This is why the fail-open default exists, and why it is so hard to remove. The people who set it were not careless. They were balancing a real trade-off between fraud loss and a checkout lane that stops working, and in that trade-off availability wins, because a declined legitimate customer is a visible, immediate, measurable cost while fraud is a number that appears in a report next quarter.

Moving the model to the terminal dissolves the trade-off rather than resolving it. If the scoring happens locally, there is no round trip whose failure mode needs a default. The network carries updated intelligence and receives telemetry, and both of those are things the terminal can survive briefly without.

THE TRADE-OFF

Fraud loss versus checkout availability. Availability wins, so the timeout approves, so the control has a known bypass.

THE DISSOLUTION

Score locally. There is no round trip, so there is no timeout, so there is no default to exploit.

THE COST

You now operate a model fleet across constrained devices. This is a real burden and it is the honest price of the property.

WORKSTREAM 01

Making the Model Fit the Lane

A POS terminal is a constrained device with a fixed hardware lifecycle, chosen for reasons that had nothing to do with running inference. The model does not get to specify its host.

THE CHALLENGES WE ENCOUNTERED

- **The hardware was already deployed.** Terminals in the field cannot be replaced to suit a model. Whatever the model needed, it had to need less than what was already sitting in the lane.
- **Feature availability is asymmetric.** A central model sees the customer's full history across every channel. The terminal sees this transaction and whatever it has been given. The features that make the central model accurate are precisely the ones the edge lacks.
- **Class imbalance is extreme.** Genuine fraud is a vanishing fraction of transactions. A model that predicts "legitimate" unconditionally achieves outstanding accuracy and is worthless, and naive training converges toward exactly that.
- **False positives are expensive in a way false negatives are not.** A declined legitimate customer at a checkout is a visible failure with a human standing in front of it. The asymmetry is real and it constrains the operating point.
- **Thermal and power envelopes.** Sustained inference on a device designed for transaction processing has physical consequences that do not appear in a benchmark.

HOW WE RESOLVED THEM

- The model was designed against the deployed hardware's actual envelope from the outset rather than trained first and compressed afterwards. Quantisation applied to a model that was never going to fit produces a small model that is bad, rather than a good model that is small.
- Feature engineering was split by locality. The terminal computes what the terminal can observe — transaction shape, velocity, local behavioural signals — and receives distilled intelligence about broader patterns rather than trying to reconstruct them.
- Imbalance was handled through the training regime and, more importantly, through evaluating the model on fraud recall at a fixed false-positive rate rather than on accuracy. Accuracy on a 1-in-10,000 problem is a number that means nothing and reassures everyone.
- The operating point was set with the business rather than by the data science. The acceptable false-positive rate at a checkout lane is a commercial decision, and a model tuned without that input is tuned to the wrong objective.
- Sustained-load behaviour was tested at temperature, over duty cycles matching real lane throughput, rather than in bursts on a bench.

HOW WE PROVED IT WORKED

The model was evaluated on fraud recall at the agreed false-positive rate, on the target hardware, under representative load — not on a workstation with unbounded time. Latency was measured as the terminal experiences it, end to end within the transaction path, because a model that scores in two milliseconds inside a framework that takes forty to reach it has not solved the problem it was built for. **Degraded operation was tested by removing the network entirely and confirming that the terminal continued to decide rather than continuing to approve.**

> **RISK AVOIDED**
>
> A fraud control with a documented, reproducible bypass: induce latency, trip the timeout, and the system approves. This is not a theoretical weakness — it is an operational default that attackers probe for deliberately.

> **WHAT A US BANK SHOULD TAKE FROM THIS**
>
> Ask what your terminals do when the scoring service does not answer in time. If the answer is approve, you should know how often that path is taken, and whether the frequency is random or correlated with anything.

WORKSTREAM 02

Operating a Model Fleet

One model in one place is a deployment. The same model on every terminal in the estate is an operational discipline, and it is the part that determines whether edge inference survives contact with production.

THE CHALLENGES WE ENCOUNTERED

- **Fraud patterns change faster than fleets update.** A model that is current at the centre and three versions old in the field is not one system. It is a distribution of systems with a distribution of behaviours.
- **Drift is invisible from the terminal.** A device cannot know that its model has become stale relative to a threat landscape it cannot see.
- **Rollback across a fleet is not a button.** A bad model reaching every terminal simultaneously is a worse outcome than the fraud it was meant to stop, and a partial rollout means multiple model versions live at once.
- **The terminal is physically accessible.** A model on a device in a shop is in an environment the institution does not control, which raises questions about tampering and extraction that a server-side model never faces.

HOW WE RESOLVED THEM

- Model versioning was treated as first-class state. Every scoring decision records the version that produced it, so that post-hoc analysis of a fraud event can determine what the terminal actually knew rather than what the current model would have concluded.
- Rollout was staged rather than simultaneous, with a defined cohort, an observation window, and explicit criteria for proceeding. A model reaches the fleet the way a change reaches production, not the way a signature update does.
- Rollback was tested rather than documented, on the assumption that the first time it is exercised should not be the first time it is attempted.
- The centre retained the cross-terminal view. Patterns that only exist across devices — the same card at three lanes in an hour — are structurally invisible at the edge, and pretending otherwise would have been the fashionable mistake. The edge decides; the centre learns and pushes back what it has learned.

HOW WE PROVED IT WORKED

Fleet behaviour was verified by deliberately running mixed versions and confirming that decisions remained attributable and analysable, because mixed versions are not an anomaly to be eliminated — they are the steady state of any real fleet. Rollback was exercised end to end under load. Drift was monitored from the centre against the terminals' own reported outcomes rather than inferred from central data alone.

RISK AVOIDED

A fraud estate that nobody can reason about: many model versions, no record of which decided what, and an investigation after a loss event that cannot establish whether the model failed or was simply old.

WHAT A US BANK SHOULD TAKE FROM THIS

Your model risk function will ask which model made a given decision. On a fleet, that question has an answer only if you designed for it. Retrofitting version attribution after a loss event is not possible.

WHY THIS MATTERS MORE ON THE NEW RAILS

Card and ACH have a safety net that nobody thinks about because it has always been there: the transaction can be reversed. Detection after the fact still recovers the money. **Instant payment rails remove the net, and in doing so they convert fraud detection from an analytics function into a real-time control.**



On FedNow and RTP, settlement is final. There is no chargeback, no reversal, and no recovery window. A fraud model that produces an excellent answer four hundred milliseconds after the payment settled has produced a report about money that is gone. The entire value of the model collapses into whether it answered before the deadline.

Which is precisely the property that edge inference was built for. The engineering problem at a point-of-sale terminal — decide locally, decide within the budget, do not depend on a round trip that might not return — is the same engineering problem an instant payment rail presents, with a larger number attached to getting it wrong.

YOUR REQUIREMENT	WHAT THIS ENGINEERING PROVIDES	MAPS TO
Irrevocable rails	Scoring inside the transaction's own latency budget, so the decision precedes settlement rather than describing it.	FEDNOW / RTP
Fraud sophistication	Behavioural scoring at the point of transaction rather than pattern matching after it.	OCC SRP SPRING 2026
No fail-open path	Network loss degrades intelligence rather than removing the control. There is no timeout to induce.	OP RESILIENCE
Model risk on a fleet	Version attribution on every decision, staged rollout, tested rollback.	SR 11-7
Data locality	Transaction detail scored where it is created rather than shipped to be evaluated.	PCI / RESIDENCY
False positive cost	Operating point set with the business, evaluated on recall at a fixed FPR rather than on accuracy.	CUSTOMER IMPACT

THE HONEST LIMITATIONS

- **The edge cannot see what the edge cannot see.** Cross-terminal, cross-channel and cross-institution patterns require a central view. Edge inference complements that; it does not replace it, and any pitch claiming otherwise is selling.
- **Fleet operation is a real cost.** You are trading a network dependency for an operational one. That is usually the right trade for an irrevocable rail and often the wrong one elsewhere.
- **The device is in someone else's building.** Physical access raises tampering and model extraction questions that a server-side deployment never has to answer.
- **This was POS.** The engineering transfers to payment rails; the deployment does not. We would expect to establish the latency and accuracy profile in your environment rather than ask you to accept figures from a different one.

Decide at the lane

The model runs where the transaction is, inside the budget the transaction has, with no round trip whose failure mode is approval. On card rails that is a good property. On rails where settlement is final and there is no chargeback, it stops being a property and becomes the requirement.

WHERE THIS GOES NEXT

Instant payments were sold to your institution as a product feature. They arrived as a fraud architecture problem, because every control that assumed a reversal window quietly stopped being a control.

If you are standing up FedNow or RTP volume, the question worth asking early is where your fraud decision physically happens, and what it does when the answer does not arrive in time.

TECH4BIZ SOLUTIONS | CODENUK | AUDITTRAX LABS

tech4bizsolutions.com codenuk.com auditraxlabs.com

contact@tech4biz.io