OBSERVABILITY · AI · PAYMENTS INFRASTRUCTURE

Sense and EWS: an agentic observability platform for India's payments infrastructure

How we are moving UPI operations from detecting incidents after impact to predicting, reasoning and acting on emerging failures — and what we have measured so far.

SUMMARY

UPI processed 23.66 billion transactions in July 2026 — an average of roughly 530,000 every minute. At that run-rate, the interval between a failure beginning and an operations team understanding it is not an engineering inconvenience. It is a measurable quantity of payments that were not completed.

This paper describes the two platforms our technology teams have built to shorten that interval: **Sense**, a unified signal collection layer across infrastructure, applications and operations, and the **Early Warning System (EWS)**, an agentic intelligence layer that predicts emerging failures, assembles evidence into a probable root cause, and recommends remediation under human-approved controls. We set out the architecture, the predictive pipeline, the first production prediction module at the web application firewall layer, and a worked case in which the platform issued a HIGH-severity warning four minutes before the first customer-visible error.

1. Why observability at UPI scale is a public-interest problem

UPI is the largest real-time retail payment system in operation anywhere. In July 2026 it carried 23.66 billion transactions worth ₹29.88 lakh crore, its highest monthly volume in a decade of operation and a rise of about 22% year on year. On a typical day that is 763 million transactions, roughly half of all real-time payments recorded globally.

The number that matters operationally is none of those. It is the per-minute rate. At around 530,000 transactions a minute, a single minute of degradation covers salaries, rent, school fees, fuel and a merchant's day's takings. Industry medians put mean time to detect at 42 minutes and mean time to restore at 58 minutes for financial services and insurance.[1] Applied to our run-rate, a median resolution window spans roughly 30 million transactions. That arithmetic is the reason we treat observability of this platform as public infrastructure work rather than as an IT function.

23.66 bn	₹29.88 lakh cr	763 mn	~530 k
Transactions, July 2026	Value, same month	Transactions per day	Transactions per minute — the unit every degradation window is measured in

Volume and value: NPCI monthly statistics, July 2026. Share of global real-time payments: Ministry of Finance / IMF, 2025. Per-minute figure derived from the July 2026 run-rate.

2. What a reactive posture costs

2.1 THE SEQUENCE IS LINEAR AND HUMAN-PACED

Our current methods of failure identification, remediation, and root cause analysis are reactive, siloed, manual, and time-consuming. An incident moves through six steps: the issue occurs, an alert fires, an engineer investigates, a root cause is identified, a fix is applied, and the fix is validated. Customer impact begins at the first step and persists until the last. Understanding arrives four steps after the impact does.

There is a second, blunter way to describe the same sequence, which is who finds out in what order. In a reactive posture, it is the customer, then the merchant, then social media and the press, and then operations. By the time the bridge calls convene, the transactions are already lost.

2.2 EVERY LAYER FAILS DIFFERENTLY, AND EACH IS INSTRUMENTED SEPARATELY

A UPI transaction crosses network, firewall, application, queue, cache and database layers. Each has its own telemetry, its own tooling and its own owning team, so correlation across them is performed by people, on a call, under time pressure. The resolution times below are drawn from our own operational assessment of these layers.

LAYER	CHARACTERISTIC FAILURE MODES	OBSERVED MTTR
Network	Packet loss, latency, jitter, routing changes; ISP or cloud path problems; congestion, link failure, BGP changes	60–90 min
WAF	Sudden spikes in blocked or allowed requests; misconfigured rules and false positives; attack patterns not detected early	45–75 min
Application	API errors, slow responses, timeouts; dependency failures and misconfiguration; code issues found only after impact	60–120 min
Queue	Backlog build-up and consumer lag; message delivery failures and retries; throughput limits	45–90 min
Cache	Cache-miss spikes and low hit ratio; evictions under memory pressure; stale data and invalidation issues	30–60 min
Database	Slow queries, locks and deadlocks; connection-pool exhaustion; disk I/O, replication and failover issues	60–120 min

Reading across the table, the pain points repeat regardless of layer: siloed tools and data, manual correlation across teams, long investigation and resolution times, fragmented context with missing dependency information, and a consequent cost in risk and customer impact.

2.3 THE EXPENSIVE PART IS TIME-TO-UNDERSTANDING

Alerting is not where the cost sits. Published medians for financial services and insurance put detection at 42 minutes and restoration at 58 minutes; around 37% of incidents are attributed to misconfiguration or human error, which produces wrong fixes and repeat incidents; and organizations in a reactive posture spend two to three times as much time on incident resolution, which crowds out the prevention work that would reduce the next incident.[1]

WHAT A 58-MINUTE WINDOW MEANS HERE

~30 mn	$2.2M	34%	30%+
Transactions inside a median resolution window, at our run-rate	Median industry cost per hour of a high-impact outage	Outages attributed to network failures	Customers likely to churn after one major outage

1 New Relic, State of Observability for Financial Services & Insurance, 2024. Transaction figure derived from the NPCI July 2026 run-rate.

3. Three postures, not three products

We frame the work as a change of operating posture rather than a tooling upgrade. A **reactive** posture detects after impact. A **predictive** posture forecasts degradation before impact. An **agentic** posture of reasons for the forecast and acts on it, under control a human has approved. The rest of this paper is how we get from the first to the third without giving up the governance a regulated payments operator has to keep.

TODAY	IN PRODUCTION NOW	NORTH STAR
Reactive	**Predictive**	**Agentic**
Threshold alerts fire once customer impact has begun	Failure probability scored every minute, 30 minutes ahead	Reason and act on emerging failure, under human-approved controls

4. Sense: one signal fabric

Prediction is not possible on signals you lose. Before any model, we needed reliable, high-volume, lossless collection of logs, metrics and traces from every source, in one schema, with the operational context that makes correlation meaningful. Sense is that layer.

4.1 SOURCES

Collection agents run as Docker containers on Kubernetes and cover three families of source. From **infrastructure**: node exporters for servers, VMs and containers; a JMX exporter for Java applications; an SNMP exporter for network devices; OpenTelemetry agents for applications and services; a FortiGate exporter for firewall and security telemetry; and Vector.dev as the log collector. From **engineering**: code repositories on Git and GitLab. From **operations**: service-desk tickets and incidents, BMC for monitoring and ITSM, and the CMDB for configuration state.

The operational sources are the ones that are easy to leave out and expensive to lack. Metrics tell you that something changed; tickets, CMDB entries and commits tell you what was changed, by whom, and what depends on it. Without them a model can detect an anomaly but cannot reason for it. Sense therefore carries six signal classes: logs, metrics, traces, events, transactions, and operational context.

4.2 PRODUCTION PIPELINES

Logs, traces and metrics run on separate, independently scalable paths into a single visualization layer, so that a surge in one signal class cannot starve another. Kafka is the shared streaming backbone; Kubernetes is the platform underneath every component rather than a stage in any pipeline; Grafana is the common surface for dashboards, exploration and correlation.

PIPELINE	PATH
Logs	OTel Gateway → Vector.dev → Kafka topics → VictoriaLogs
Traces	OTel Gateway → Kafka, with span-metrics transformation → ClickHouse / Victoria Traces
Metrics	VM Agent scrape → VM Insert → VM Storage (time-series DB) → VM Select query layer, powered end to end by VictoriaMetrics

4.3 THE FOUR PROPERTIES THAT MAKE IT AN AI SUBSTRATE

Sense was designed against four requirements, each of which is a precondition for prediction rather than a nice-to-have.

- **High throughput.** Millions of events per second, on Kafka as a distributed streaming platform.
- **Reliability.** Backpressure handling and at-least-once delivery, through Vector.dev as the collector and pipeline.
- **Scalability.** Horizontally scalable, containerized agents deployed as lightweight Docker images on Kubernetes.
- **Unified telemetry.** Logs, metrics and traces in one fabric, with OpenTelemetry standardizing collection and the Victoria stack providing storage.

The technology foundation is deliberately open-standard, open-source and containerized: OpenTelemetry, Vector.dev, Kafka, VictoriaMetrics, VictoriaLogs, ClickHouse / Victoria Traces, Grafana, Docker, Kubernetes, MinIO, PostgreSQL, Neo4j, MLflow, Kubeflow, Airflow, and Git / GitLab. Sense turns fragmented telemetry into an intelligence-ready signal of fabric.

Known challenges this layer set out to address: tool sprawl and complexity, siloed data across systems, manual correlation and high MTTR, the operational overhead of running many agents, and reactive detection after impact.

5. EWS: the intelligence architecture

The Early Warning System reads Sense and produces decisions. It runs as five stages over one shared context — sense, detect, correlate, predict, act — with a feedback path that returns validated outcomes to the models. It is a loop rather than a pipeline: every action produces an outcome, and every outcome is training data, which is what makes the second occurrence of a failure cheaper to handle than the first.

Structurally the platform has four macro layers. The technologies named against each are implementation choices; the layer boundaries are the architecture.

LAYER	RESPONSIBILITY	IMPLEMENTATION
01 Signal fabric	Collection, streaming, storage, visualization	Sense — OpenTelemetry, Vector.dev, Kafka, VictoriaMetrics, VictoriaLogs, ClickHouse / Victoria Traces, Grafana
02 Data & feature platform	Data pull, validation, feature engineering, scheduling	MinIO data lake with raw, processed and feature zones; PostgreSQL for metadata, feature store, predictions and model registry; Airflow for orchestration
03 AI / ML & knowledge	Anomaly models, reasoning, summarization, knowledge retrieval	AI/ML layer with an LLM; MLflow and Kubeflow for experiments, registry, versioning and serving; Neo4j for graph knowledge and MinIO for document knowledge
04 Decision & action	Orchestration, alerting, remediation, validation	AI-based orchestrator over specialized agents; alert and notification engine with a 30-minute warning window; remediation and execution with an approver gate

5.1 THE AGENTIC CORE

The part of the design that is genuinely new is not a single model. It is a set of specialized agents reasoning over the same operational context, coordinated by an orchestrator. Anomaly details are published to Kafka so that the agents relevant to a given anomaly pick it up.

- **Anomaly agent** — machine-learning and time-series detection on live signals.
- **RCA agent** — assembles evidence into a probable cause.
- **Dependency agent** — service and topology relationships.
- **Source code agent** — links commits and changes to observed symptoms.
- **Incident agent** — history, tickets and runbooks.
- **Communication agent** — notification across email, WhatsApp, SMS and Teams.
- **Remediation agent** — proposes and executes approved actions.

The **AI-based orchestrator** routes signals to agents, sequences their reasoning, resolves conflicting evidence and produces one decision with a confidence score. Four supporting stores give that reasoning something to reason over: an LLM for the natural-language interface, hypothesis generation and explainability; the AI/ML layer holding anomaly and prediction models per layer; graph knowledge in Neo4j for dependencies and blast radius; document knowledge in MinIO for runbooks and historical RCA; and operational context from CMDB, BMC, tickets, configuration and change records.

Downstream, the platform is consumed as real-time alerts and notifications, dashboards and visualizations, APIs and webhooks into ITSM, NOC and SOC systems, and an audit and compliance trail. The last of these is not an afterthought: in a regulated environment, an automated decision that cannot be reconstructed afterwards is not usable.

6. The predictive analytics pipeline

Prediction runs on two clocks. Learning happens offline on a schedule; inference runs every minute and looks thirty minutes ahead. Keeping them separate is what lets us retrain aggressively without touching the latency budget of the live path.

OFFLINE — LEARNING

Scheduled by Airflow, per layer and per error code

Data fetcher (every minute from Sense endpoints) → feature engineering across 1m / 5m / 15m / 30m / 1h windows → training job producing multiple models per layer with artifacts in MLflow → evaluation on ROC-AUC, PR-AUC, F1 and recall → promotion of the best model to Production in the MLflow registry.

REAL TIME — INFERENCE

Every minute, horizon of the next 30 minutes

Prediction job scores live features with the current production model → result store writes probability, model, layer and error code to PostgreSQL → LLM summarization produces a human-readable anomaly summary → watcher service compares against threshold and suppresses duplicate alerts → alert engine raises an early warning for the next 30 minutes where probability exceeds threshold.

Model governance sits across both clocks. Promotion runs through an approval workflow; every promotion, prediction and action is logged; model performance is tracked continuously; and rollback to a prior version is a first-class operation. Nothing reaches production inference without evaluation, and nothing that reaches it is beyond recall.

7. First production module: prediction at the WAF layer

We began with the web application firewall because it sits early in the transaction path, because its telemetry is unusually clean, and because the failure it produces most often — rate limiting — is directly felt by payment service providers. The module runs one model per HTTP response code across six horizons, aggregated into a single calibrated risk score.

1. **WAF telemetry.** Sense logs, metrics and distributed traces, arriving through the Kafka ingestion pipeline.
2. **Feature generation.** HTTP status code, latency, bank IP, WAF IPs, site, five-minute request volume, PSP name and transaction ID, held in the PostgreSQL feature store.
3. **Multi-model prediction.** An XGBoost multi-class classifier per response code — 400, 401, 403, 404, 408, 429, 500, 502, 503, 504 — evaluated at horizons of 5, 10, 15, 20, 25 and 30 minutes.
4. **Ensemble risk.** An ensemble layer with probability calibration produces a confidence and risk score and a cumulative prediction; the aggregation engine assigns final severity.
5. **Early warning.** Severity of Low, Medium, High or Critical; alerts are raised at Medium and above, to Teams, email, dashboards, incident management, and pager or SMS.

The same pipeline shape is being extended, with per-layer models, to the network, API gateway, Kafka, KeyDB and Cassandra layers.

8. Evidence: 14 August 2026

On 14 August 2026 the WAF module issued a HIGH-severity early warning for HTTP 429 on a single WAF path serving two payment service providers. The warning was generated at 10:56 with an overall risk of 69.3% and a thirty-minute window. The first observable spike — 152 HTTP 429 responses — appeared at 11:00:03. The prediction preceded the customer-visible error by four minutes.

PREDICTION ALERT, AS ISSUED

WAF IP	10.x.x.x
Data centre	Masked
PSPs	2 affected
Predicted code	429 (rate limit)
Overall risk	69.3%
Horizon	30 minutes
Severity	**HIGH**
Spike window	10:56 – 11:26

4 min

of warning before the first customer-visible error, on a path serving two PSPs

10:56 Prediction generated — HTTP 429, risk 69.3%, severity HIGH

11:00 Observed spike — 152 × HTTP 429 at 11:00:03

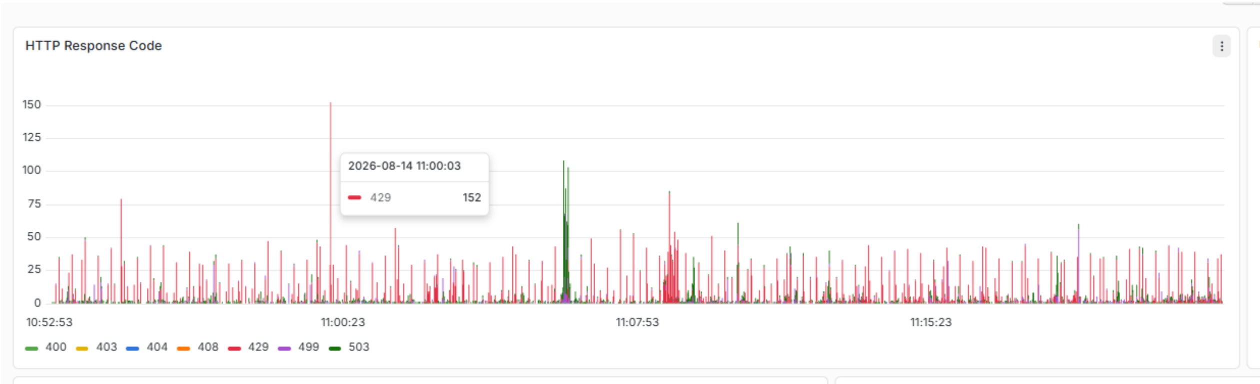

Figure 1 — Supporting evidence: Grafana HTTP response-code panel for the WAF path, 14 August 2026. The 429 spike is visible at 11:00, four minutes after the warning was issued. Address, site and institution identifiers are masked.

We are deliberately precise about what this case does and does not show. It shows that a calibrated forecast on one layer preceded the observable error on that layer, on that path, by four minutes. It does not by itself establish an end-to-end reduction in customer-visible failures; that requires the module set to cover the layers where degradation is gradual rather than abrupt, which is the work now under way.

9. AI-assisted root cause analysis: a worked example

ALL FIGURES IN THIS EXAMPLE ARE SIMULATED

The following illustrates how the RCA agent turns a throughput dip into a named, evidenced probable cause. The anomaly: the FIN service showed observed throughput of 24,791 TPS against an expected baseline of 26,436 TPS — a 6.22% dip, classified tps_dip.

Evidence was supplied for two layers only, and the summary is scoped accordingly. **Network telemetry** showed seven peers down across two sites, with no peer restart or network flap detected. **WAF telemetry** showed a modest volume of HTTP 503 upstream errors and a larger volume of HTTP 429 rate-limit responses, with upstream response times reaching 75 ms, against approximately 17.6 million successful HTTP 200 responses in the same window.

AI-GENERATED CONCLUSION

Probable root cause: network capacity reduction

Relative to the successful response volume, the WAF-level error counts are small and likely account for only a fraction of the TPS shortfall. The down network peers represent a more probable direct capacity-reduction event. Presented with a confidence score, not as certainty.

Two properties of that output matter more to us than the conclusion itself. The first is proportional reasoning: the agent weighed the error counts against the successful volume rather than treating any error as causal. The second is scope discipline: it stated which layers it had evidence for and did not speculate beyond them. An RCA that overstates its own certainty is worse than no RCA, because it produces confident wrong fixes — the failure mode that accounts for a large share of repeat incidents.

10. Coverage: what EWS watches, and what it does with it

A UPI transaction crosses eight components: PSP applications, DNS and network, the WAF, the API gateway, application services, the KeyDB cache, Kafka, and Cassandra. EWS observes all of them from one layer, and applies to each the kind of intelligence that layer earns — detection where symptoms are loud, prediction where degradation is gradual, correlation where business impact is the signal.

LAYER	SCENARIOS MONITORED	PLATFORM RESPONSE
WAF	High blocked requests, rule misconfiguration, policy tuning issues, SSL handshake failures, DDoS and bot traffic spikes	**Detects → early warning**
Application	High 5xx error rate, slow API response, thread and connection-pool exhaustion, memory and CPU pressure, exception and crash loops	**Detects → pinpoints root cause**
Cache (KeyDB)	High cache-miss rate, eviction spikes, high memory usage, slow queries, primary/replica sync issues	**Predicts → avoids failure**
Network	High latency and packet loss, DNS resolution failures, bandwidth saturation, TCP resets and retransmits, device errors	**Detects → isolates impact**
Database (Cassandra)	High read/write latency, timeouts and failures, compaction and repair issues, disk and IOPS saturation, node instability	**Predicts → ensures stability**
Processing unit	High transaction failure rate, slow settlement or response, switch connectivity issues, queue backlogs, business rule failures	**Correlates → business impact**

11. Closed-loop remediation, and the controls around it

End to end, the operational journey has five steps. **Collect** takes multi-layer signals — logs, metrics, traces, topology — with thousands of metrics ingested in real time. **Correlate** builds cross-layer context and reduces alert noise. **Predict** uses the AI/ML layer to detect anomalies and forecast impact before users feel it. **Diagnose** produces an AI-driven RCA with probable root cause and blast radius. **Recommend and act** has agentic AI propose, and where approved trigger, automated remediation.

The loop then closes: every action is validated, and the validated outcome returns to the models as training signal. Two controls govern what may execute automatically. Policy guardrails define the class of action an agent may take at all, and an approver gate stands between a recommendation and its execution for anything outside that class. Post-action validation is mandatory rather than optional, and the full chain — prediction, evidence, recommendation, approval, execution, outcome — is written to the audit trail.

We regard this as the load-bearing part of the design rather than a compliance wrapper around it. Automation that operates a national payment system has to be re-constructible after the fact, and an agent whose actions cannot be explained is an agent that cannot be given scope.

12. What changes in the operating model

DIMENSION	REACTIVE OBSERVABILITY TODAY	WITH EWS
Detection	Threshold alerts fire after customer impact has begun	Failure probability scored every minute, 30 minutes ahead
Correlation	Manual, on a bridge call, tool by tool and team by team	Automated cross-layer correlation over service topology
Root cause	Reconstructed by hand after the fact; around 37% erroneous	Probable cause with supporting evidence and a confidence score
Action	Human-paced runbook lookup and manual execution	Recommended runbook, approval gate, automated execution and validation
Learning	Post-incident review, rarely fed back into detection	Outcomes retrain the models; the next incident is cheaper
Team posture	Firefighting; two to three times as much time on incident resolution	Prevention and engineering, with automation carrying the routine

Reactive-column figures: New Relic, 2024.

13. Where this goes next

Six outcomes describe what we are building toward: proactive issue detection ahead of user impact; lower MTTR through faster RCA and automated remediation; higher reliability from a stable, resilient, self-healing platform; a better experience for PSPs and end users through fewer failures and lower latency; operational efficiency from less manual effort; and business impact in the form of a higher success rate and more transactions completed.

TARGET OUTCOMES — NOT MEASURED RESULTS

50%+	40%+	99.99%+	Better UX
Reduction in downtime	Faster MTTR	Platform reliability	Higher success rate, more trust

These are targets for the platform, stated as targets. The measured result we can point to today is the one in section 8. The near-term programme is to extend prediction modules across the network, API gateway, Kafka, KeyDB and Cassandra layers on the same pipeline shape, and to widen the set of remediations an agent may execute under guardrail as the evidence for each accumulates.

The arc is a simple one. Sense makes the platform legible. EWS makes it predictable. The agentic core is what makes it self-correct — so that a UPI failure is prevented, rather than announced by the people it failed.

Notes on data, method and redaction

- **UPI volumes.** NPCI monthly statistics for July 2026. The share of global real-time payments is from the Ministry of Finance / IMF material, 2025. Per-minute and per-window transaction counts are arithmetic derivations of the July 2026 run-rate and are labelled as such wherever they appear.
- **Industry medians.** MTTD, MTTR, erroneous-RCA share, cost per hour of outage, network-attributed outage share and post-outage churn are from New Relic, State of Observability for Financial Services & Insurance, 2024. They are industry figures, not NPCI measurements.
- **Layer MTTR ranges.** From our own internal operational assessment of the layers concerned.
- **The 14 August 2026 case.** Drawn from the platform's own prediction record and the corresponding Grafana panel. The WAF address, the data centre and the affected institutions are masked; the address appears only as 10.x.x.x. No customer or transaction-level data appears in this paper.
- **The RCA worked example.** All figures in section 9 are simulated. It is included to show the agent's reasoning shape, not to report an incident.
- **Target outcomes.** The figures in section 13 are platform targets, not measured results.

AUTHORS

Rugraj Devraj · Ajay Mahto · Nalla Tharun · Barun Kumar Yadav · Tipirishetty Praveen Kumar · Ajay Joshi · Somprasad Theegala · Satwik Sakamuri · Sindhu Priya Shanigarapu · Sharath Boyini · Yash Mishra · Pujitha Pendli · Priyanka Rodda · Duleshwari Sahu · Vivek Upadhyay · Sai Basani · Mohammed Abdulrahman · Ashok Surampudi · Shamanth MH

Technology, National Payments Corporation of India