BUILDING AN EFFECTIVE OBSERVABILITY FRAMEWORK FOR UPI STACKS IN BANKS

Enhancing Performance, Partner Collaboration, and Future Innovations, all while guaranteeing a seamless user experience
Section 1:

The Rise of Digital Payments and the UPI Stack's Complexity

The meteoric rise of digital payments in India has been fuelled in large part by the exponential growth in the number of UPI (Unified Payments Interface) transactions. From a modest start in August 2016 with a recorded value of 3.09 Cr worth of transactions, UPI recorded its highest-ever number of 8.7 billion transactions valued at 14.10 lakh crore in March 2023, according to National Payments Corporation of India (NPCI).

UPI has disrupted the traditional banking model, with customers using the platform to transact for amounts ranging from a few rupees to a few thousands. It has been at the forefront of financial inclusion in India, and has propelled our country to the leading spot in digital payments. However, a system that sees the sheer volume and velocity that UPI is subject to, would obviously need to have a robust infrastructure and architecture to support it as well.

The UPI system involves multiple parties in a single transaction flow. These parties include the Payer PSP (Payment Service Provider), Payee PSP, Remitter Bank, Beneficiary Bank, NPCI, Bank Account holders, and Merchants.

The complexity of the UPI stack stems from surging volumes and the involvement of multiple organizations, interconnected applications, multiple interfacing APIs (Application Programming Interfaces), and the underlying infrastructure devices navigated in a single transaction. The success of a payment journey depends on the performance of each one of these touch-points, applications, APIs, infrastructure elements etc. Unexpected failures lead to longer resolution cycles.

*Fig 1: The complexity of a single UPI transaction. 6 entities and 12 micro-legs per transaction translate to billions of records that need to be monitored daily.*
With the volume of UPI transactions witnessing an unprecedented growth, the need to make them failure-proof is growing by leaps and bounds. Not only is there a responsibility to maintain customer expectations, but also the added pressure of upholding SLAs in partner interactions and holding up to regulatory scrutiny. This is why a new paradigm of Observability is needed for UPI Stacks.

In this whitepaper we describe this paradigm in depth and examine the steps that banks need to take to implement it for their UPI stacks.

Section 2:

OTT and UPI – A Match Made in Heaven

Challenges to Monitoring UPI Transactions in Banks

As the number of UPI journey events per second rapidly increases each day, real-time performance monitoring of UPI transactions becomes more and more challenging. Some factors that contribute to this include:

- **Multiple APIs, asynchronous responses:** A single UPI transaction involves multiple API calls that need to travel across multiple parties for debiting and crediting funds from the remitter to the beneficiary. Tracing a transaction end-to-end is difficult since API calls between parties occur asynchronously (i.e., there is a time delay or gap between initiating an action and receiving a response or completing the transaction, which means the request and response are not sequential.)

- **Enormous volumes:** UPI transactions involve massive volumes, which continue to grow. This necessitates a constantly evolving infrastructure to handle the increasing load.

- **Failures and sub-optimal performance:** Errors, timeouts, latency, and high Turnaround Times (TATs) are inevitable in UPI transactions. These issues lead to a poor user experience and can have significant consequences, including denting users' trust in the ecosystem, impacting banks' revenues and reputation, and potentially triggering regulatory action if predefined thresholds are crossed.

- **Responsibility attribution challenges:** Assigning responsibility for transaction failures or latency is complex due to the involvement of multiple parties. PSPs and payment aggregators often place blame on the bank, making it challenging to determine the exact source of the problem.
Goals of a UPI Observability Framework

In a multi-party system like UPI, the essential requirement from an Observability framework transcends simply monitoring the system, to ensuring that customer experience goals are met and partner interactions with other stakeholders in the API ecosystem are adhering to the basic tenets of trust and transparency. With that in mind, here are some high-level requirements from an Observability framework for a system as complex as UPI:

**Query capabilities:** In-house IT teams should be able to query transaction or system state at any given time. Detailed drill-down options should be available for effective troubleshooting. This requires well-indexed logs, metrics, and traces.

**Real-time lead-indicators:** Real-time performance metrics such as Turnaround Times (TATs) and error counts should be provided. These metrics should ideally be computed separately for transactions associated with different partners (e.g., PSPs or payment aggregators), so the required drilldowns and filtering mechanisms are enabled. There should also be drill-downs on other dimensions like merchant name (e.g., Amazon/Zomato) or geo-codes. This not only makes for intuitive and rich operational dashboards, but also facilitates targeted remediation and enables capacity planning, infra-resizing, and smart load-balancing to handle volume surges from specific channels.

**Performance metric exposure:** The bank should be able to share its performance metrics with relevant partners. This constitutes a core tenet of transparency. For instance, if large payment aggregators hold accounts with multiple banks, transparent performance information allows them to dynamically route transactions to the bank with the lowest current TATs or avoid banks experiencing unusually high error counts. This benefits the entire ecosystem and all stakeholders. With stakeholders independently monitoring each other's system performance, it becomes easier to assign responsibility in case of service degradation.

Another instance of transparency is a scenario where a payment platform A facilitates transactions using Virtual Payment Addresses (VPAs) associated with different banks such as Bank X, Y, or Z. When a user initiates a transaction using platform A with Bank X’s VPA, the transaction request directly reaches Bank X. For each successful transaction, A pays a certain amount to Bank X as a transaction fee. However, if a transaction fails due to technical issues on the bank’s end, Bank X may be liable to pay a penalty for each failed transaction.
In such cases, maintaining transparency becomes crucial. It ensures that all parties involved, including the payment platform, bank, and the users, have clear visibility into the transaction process and any potential failures.

While these features form the core requirements, having a framework with these capabilities also allows the bank to build additional use-cases on top with minimal extra effort.

**OTT Observability Framework for Multi-API Systems**

Keeping in mind the above basic requirements for observability in multi-API systems like UPI, we propose a novel framework to approach the issues of visibility into all components of the ecosystem, transparency and trust among all partners involved and traceability of all transactions across touchpoints. We name this framework **OTT – Deep Observability Inside, Transparency Outside and Traceability Across.**

The following infographic explains the concept of OTT.
The 3 pillars of Observability – logs, metrics, and traces – are collected across the breadth of the business journey spanning a complex heterogeneous stack, and multiple services. Internal and external touchpoints across infrastructure and applications are monitored via signals collected using common standards like Open Logging and Open Telemetry.

By establishing a clearly defined contract based on mutual trust and well-documented method signatures at every internal and external API interface, the enterprise can promote Transparency Outside in interactions between integration partners.

All business flows in multi-party systems are not always observable, which means that there might be some steps in a transaction we are unable to trace. This leads to the question: When there is a dispute or a customer complaint, how can the ID of a specific transaction in question be traced? How can better interconnection be facilitated across disparate entities to identify failure points? How can every micro-leg of a transaction be traced without compromising on privacy considerations? This is why Traceability Across is required.

**OTT Applied to the UPI Observability Stack**

UPI is a prime example of a complex API system on Rails, working at a 9-billion-transactions-a-month scale. It thus presents an ideal case study for the OTT framework to provide trust and transparency, so parties across silos can have a real-time view of journey lead indicators, as well as effectively troubleshoot business and IT errors as they occur.

---

**Fig 4:** The overlay of business data along with monitoring data on each micro-leg, infrastructure element and API of a typical UPI transaction.
Enriching and contextualizing data for the UPI business flow entails associating each micro-leg of the journey (for instance, the call from the Payer PSP to UPI) with lead indicators like turnaround time, latency, and IT/business error codes. This provides an end-to-end journey-centric view of the system earlier solutions lacked.

Any observability solution for UPI should be **flexible** and hyper-configurable to work with data adapters on a multitude of sources, especially logs and business traces. It should provide a **unified view** of data spread across multiple disparate silos. Most importantly, it must be able to handle data at **scale**, with big data infrastructure for AI0ps and MLOps, since manually analysing data at the volume and velocity expected in UPI is virtually impossible. Operational views and actionable insights based on journey lead indicators ensure **smarter event correlation, more accurate anomaly detection, and faster issue resolution**.

---

*Fig 5: A view of OTT as is applicable to UPI and other multiparty, multi-API systems*
Unified Operational Dashboards in UPI Stacks – thanks to OTT

Deep Observability into touchpoints means that hundreds of metrics across the UPI stack can be ingested, enriched, and contextualized, as well as distilled into operational indices like a User Experience Index, Operational Predictive Index or Capacity Planning Index, or into a natural language summary of the system state that CXOs can easily review and analyze.

Fig 6: CXO dashboards can show a system SWOT analysis in natural language, and ITOps dashboards can show operational indices derived from hundreds of metrics, thanks to deep observability into infrastructure elements, services, and APIs.
**Transparency** guarantees the availability of data to render operational dashboards that provide a unified real-time view of the ecosystem to the various partners, so they can all realize their common objective of meeting Service Level Agreements (SLAs), honouring partner contracts and optimizing customer journeys.

![Operational Dashboard Example](image)

*Fig 7: Transparency enables operational dashboards presenting a unified view of system performance, including API failures and error codes to IT and Business Teams, along with actionable recommendations for effective decision-making.*

Traceability and observability are paramount in mapping the entire customer journey across disparate touchpoints. By associating lead indicators with each micro-leg of the journey, such as turnaround time, latency, and error codes, organizations gain an unparalleled end-to-end journey-centric view. Leveraging deep observability and traceability, ITOps teams can access a unified transaction map, enabling efficient incident detection, root cause analysis, and ultimately enhancing system reliability.
Fig 9: Filtering on a specific merchant name is made possible due to traceability across touchpoints, with each micro-leg of the journey associated with lead indicators and business metadata.
Traceability also means that it is possible to search for erroneous business flows by parameters like transaction ID, merchant name, PSP name, or error code – which makes it possible to isolate failure points and underlying reasons, and extrapolate that data for compliance reporting (for instance, all transactions in a particular geo-location or for a particular merchant facing latency) or incident investigation (for instance, failure of high value transactions).

Fig 10: Filtering on a particular transaction ID narrows down the incident investigation to the exact journey that failed, and allows ITOps teams to zero in on the faulty infrastructure element, perform root cause analysis and expedite issue resolution.
A particularly important use case for transparency is NPCI i.e., the central entity in the case of UPI, interacting with its partners like banks, regulators, and PSPs. Business Journey Observability with OTT allows Site Reliability Engineering and IT Operations Management teams of NPCI to notify a partner if failures seem to be unusually high for incoming requests from that channel, or from a particular merchant further upstream.

Fig 11: Unusually high Technical Declines (TD) from a Payer App, or high Business Declines (BD) at a Payee PSP, can be flagged thanks to Unified Visibility and Transparency.
In Section 2, we saw how the OTT Framework is the ideal approach to monitoring UPI stacks and ensuring transparency, trust, and traceability. In this section, we see how banks can amalgamate OTT into their existing monitoring/observability setups.

**Best Practices for Logging**

Logs are continuous written records of the system state generated as the application runs. Traditionally, logs have been used for troubleshooting purposes, providing insights into what went wrong when failures occur. In these cases, logs focus on system-level information such as function names, variables, and error codes.

However, from a modern observability standpoint, logs can be designed to serve two important aims:

1. **Real-time tracking of individual transaction stages**: To achieve traceability across, unique transaction IDs need to be generated and logged by the application. This allows for monitoring the progress of each transaction as it moves through different stages.

2. **Actionable views into transactions**: Enriching log entries with additional dimensions such as MerchantName, PayeeBank, Geocode, and Amount provides more insightful information about each transaction, and allows for richer transaction dashboards with intelligent drilldowns to enable transparency.

To design logs that serve these purposes, the bank or UPI stack owner must make two decisions: (a) which stages of the transaction flow to track and (b) what additional dimensions to record for each transaction. Application developers should follow structured logging practices tailored to capture (a) and (b). This approach can be termed **trace-dimension logging**.

Log files can consist of two types of lines: (1) unstructured system–state dumps primarily used for troubleshooting, and (2) transaction records containing information about individual transactions as they progress through the system, including interactions with external entities.
For transaction-related log lines, it is essential to include a unique identifier, a transaction ID, which appears on all log lines pertaining to that transaction or at least on the lines corresponding to the transaction stages of interest. Additionally, when initiating a child process or an external API call, a new child-process ID can be generated and logged. This ensures traceability of each transaction throughout its lifecycle.

Likewise, the dimensions that are deemed important must appear on the transaction log lines as easily extractable key-value pairs. For example, here are some xml tagged dimensions:

```
<txnid="201da00f4d528dda943406ee9a9b"/> Message for outbound request: https://192.168.XXX.XXX:/upi/ReqPay/2.0/

<txnid="201da00c875f4d528dda943406ee9a9b"/> Acknowledgement Received for: https://192.168.XXX.XXX:/upi/ReqPay/2.0/
```

Fig 12a. Unique identifiers in log lines indicate that both these events pertain to the same transaction.

Fig 12b. XML snippet for extracting dimensions in the form of key-value pairs

The key-value pairs emergent thereof would include {"amount": "90.00", "merchantGenre": "ONLINE"...} and so forth.

No matter which formats the logs follow – XML, JSON, Plaintext, CSV etc., - application developers must be required to systematically incorporate transaction IDs and extractable dimensions into the structure of the logs. While logs are primarily being discussed so far, the above principles are equally applicable to other data sources such as Kafka streams.
When UPI transactions fail for a bank's customers, the bank often faces responsibility not only from the affected customers but also from regulatory bodies. However, with multiple participants involved in a single UPI journey, failures can originate from any leg of the transaction. Failures within internal components, such as a delay in completing a database query, can lead to a buildup of queued transactions, causing a cascading effect of subsequent failures due to breached queue thresholds or timeouts. Swiftly identifying the source of failure is crucial for faster remediation and minimizing cascading failures.

To achieve this, the bank requires two essential components. Firstly, leg-wise information such as Turnaround Times (TATs) and errors for internal application components allows pinpointing issues. For instance, if the Core Banking System (CBS) is functioning properly but the Database Error Logs indicate multiple query timeouts, it suggests a problem with the database. Secondly, the ability to compute TATs for external API workflows, from the initial request to the final response, is vital in understanding if a sudden surge in Technical Declines is due to failures at an external participant's end.

It is worth noting that while Application Performance Monitoring (APM) tools provide detailed information on individual function or API call performance, they have limitations in terms of actionable monitoring. For example, APMs typically track TATs for external API calls only from the initial response to the acknowledgment receipt, indicating the health of the external API gateway. However, transaction failures may still occur if the downstream CBS (Core Banking System) of the external bank is down, which is a critical piece of information not captured by APMs. In addition, APM tools can typically track requests and responses which are synchronous i.e., in sequence, whereas most multi-API systems like UPI follow asynchronous mechanisms. This limitation restricts the usefulness of APM tools in a modern UPI ecosystem.

Section 4:
A “UPI”verse of Possibilities – OTT for Diverse Use Cases

1. Optimizing MTTD (Mean Time to Detect) / MTTR (Mean Time to Resolve)
Tracking the transaction journey through both internal components and external participants can be effectively accomplished using the trace-dimension logging approach described in section 3.

Minimizing MTTR: Once the issue has been identified, it becomes crucial to have transaction-level information in order to resolve it effectively. This includes details such as which session IDs are frozen in the database. Fortunately, these specific details can be found within the logs, marked with unique IDs. With this valuable information at hand, targeted remediation becomes possible. For instance, it becomes feasible to terminate only the frozen process IDs while leaving other sessions unaffected. By taking this approach, the number of collateral failures can be minimized, and there is no need for wasteful technical declines caused by a broad and indiscriminate approach like server rebooting.

2. Patch releases and new feature rollouts

Application code is a dynamic entity that undergoes regular changes such as code updates, patches, and the introduction of new features. In some cases, entirely new applications need to be released, which often interact with existing applications within the stack and externally. While traditional APM monitoring tools provide insights into application performance and identify bottlenecks during testing of these enhancements, they lack the ability to track end-to-end transaction flow. This limitation hampers the understanding of crucial details, such as whether the application logic contributed to an unusual number of declines for larger transaction amounts or if specific categories of billers were disproportionately affected by additional authorization steps. To enable such nuanced testing, observability frameworks incorporating trace-dimension logging offer enriched views and comprehensive insights.

3. Tracking large customer accounts and Relationship management

Banks often manage large customer accounts that generate substantial transaction volumes within their UPI stacks. These accounts can include PSPs, prominent online merchants, or payment aggregators serving numerous small-scale merchants. To cater to these accounts effectively, the observability framework within the UPI stack should empower the bank’s relationship managers with both technical and business insights specific to their accounts. Here are some examples:

a. Relationship managers should have the ability to promptly notify their customers if technical failures are occurring from a particular PSP, ensuring immediate attention and resolution – an example of a business insight being put into action.
When a major merchant plans to launch a festival sale or campaign that is expected to generate significant transaction volumes, the relationship manager can collaborate with the bank’s technical team to implement capacity planning via smart load-balancing or temporary infrastructure resizing. This proactive approach mitigates the risk of failures caused by overwhelming spikes in volume.

Tracking the performance of transactions originating from specific channels enables the bank to showcase its efficiency, attracting more customers. This is particularly relevant for payment aggregators who prefer to maintain accounts with banks offering a superior transaction experience. Additionally, small merchants associated with aggregators also have a choice and prioritize aggregators that provide a hassle-free collection experience with lower transaction processing times and fewer overall failures.

4. Regulatory compliance and reporting

Banks have the responsibility of maintaining technical decline rates below specified threshold percentages as mandated by the regulator. As the financial landscape continues to evolve with new products and services, the regulator establishes threshold expectations for different transaction categories within the UPI stack. To ensure compliance and accurate reporting, it becomes crucial to analyze the real-time performance data of the UPI stack across these specific categories. While overall aggregates are already being reported, diving into more examples can provide valuable insights. Here are a few illustrative possibilities:

a. What are the technical and business decline percentages for AePS (Aadhaar-enabled Payment System) transactions originating from Tier-2 geographies? This analysis helps understand the performance of AePS transactions in specific geographical areas and identify any patterns or issues that may be affecting transaction success rates, like poor networks, or inability to authenticate due to language barriers.

b. What is the bill-fetch failure percentage for electricity utility companies in the BBPS (Bharat Bill Payment System)? This examination sheds light on the effectiveness of the BBPS platform in retrieving bills from utility companies within the electricity sector. It helps identify potential areas of improvement and ensures smoother bill payment experiences for customers.
When it comes to transaction information, successful transactions are stored in the database, while failures are only recorded in the logs. In the process of dispute resolution, it becomes crucial to retrieve the trace of a specific transaction ID and determine the exact step at which the failure occurred. This is particularly important when the automatic settlement process fails to reverse a payment. The trace-dimension logging approach ensures that a particular transaction ID can be easily queried and tracked.

By leveraging trace-dimension logging, banks can streamline their dispute resolution processes, enhance customer support capabilities, ensure compliance with regulatory requirements, and conduct effective incident investigations. By searching for a customer ID or account number associated with the problematic transaction in question, they can analyze the relevant journey attributes. This allows them to isolate the journeys involved in the issue and extract data for compliance reporting purposes, such as identifying all transactions in a specific geographical location experiencing latency. It also facilitates incident investigation, such as identifying payments to a specific API that are failing or experiencing slowdowns.

The ability to trace and analyze transactions at a granular level empowers teams to address issues promptly and maintain the integrity and reliability of their payment systems.

5. Dispute resolution and settlements

When it comes to transaction information, successful transactions are stored in the database, while failures are only recorded in the logs. In the process of dispute resolution, it becomes crucial to retrieve the trace of a specific transaction ID and determine the exact step at which the failure occurred. This is particularly important when the automatic settlement process fails to reverse a payment. The trace-dimension logging approach ensures that a particular transaction ID can be easily queried and tracked.

By leveraging trace-dimension logging, banks can streamline their dispute resolution processes, enhance customer support capabilities, ensure compliance with regulatory requirements, and conduct effective incident investigations. By searching for a customer ID or account number associated with the problematic transaction in question, they can analyze the relevant journey attributes. This allows them to isolate the journeys involved in the issue and extract data for compliance reporting purposes, such as identifying all transactions in a specific geographical location experiencing latency. It also facilitates incident investigation, such as identifying payments to a specific API that are failing or experiencing slowdowns.

The ability to trace and analyze transactions at a granular level empowers teams to address issues promptly and maintain the integrity and reliability of their payment systems.

6. Tracking user experience in specific verticals or geographies

Let's consider a scenario where a bank introduces a new financial product specifically aimed at non-urban customers or a specific region within the country. In order to ensure a seamless user experience and monitor the performance of this product, tracking the geo-code dimension within transaction logs becomes essential. By analyzing transaction logs in real-time, the bank can closely monitor the user experience of customers from that particular region and promptly address any issues or concerns that may arise.
Similarly, by leveraging the trace-dimension logging paradigm, the bank can effectively track and analyze the turnaround times (TATs) and failures associated with specific verticals, such as Bill Pay or Autopay mandate executions. This granular analysis enables the bank to gain insights into the performance of these verticals, identify any bottlenecks or areas for improvement, and take appropriate actions to optimize the overall customer experience.

By utilizing transaction logs and employing a trace-dimension logging approach, the bank can obtain valuable data on the user experience and performance metrics related to their new financial product. This allows them to make informed decisions, address issues promptly, and enhance the product's effectiveness in serving the targeted customer segment or region.

Section 5:

Conclusion - Transacting our Way into the Future

With continuous innovation in the digital financial products space, the landscape of financial transactions is rapidly evolving from traditional face-to-face interactions involving physical cash and documents to an electronic future characterized by application stacks and API workflows. In order to establish user trust, drive adoption, and inspire regulatory confidence in this digital framework, certain foundational elements need to be in place.

Firstly, it is crucial that every individual transaction journey, regardless of its outcome, can be traced across the entire ecosystem. This traceability ensures transparency and accountability, allowing for effective dispute resolution and error identification. By enabling stakeholders to follow the path of a transaction from start to finish, trust is built, and issues can be promptly addressed.

Secondly, real-time system performance data of all participants within the ecosystem should be accessible to all other stakeholders. This means that each entity involved, whether a bank, payment service provider, or other financial institution, should have visibility into the performance metrics of their counterparts. This shared information promotes a collaborative environment and facilitates the timely detection and resolution of any issues or bottlenecks that may arise.

Furthermore, each participant should have the ability to monitor their own application performance and take immediate, targeted corrective actions when failures or transaction declines increase.
By having visibility into their own system's performance metrics, stakeholders can proactively address issues, mitigate risks, and maintain a high level of service quality. This empowers organizations to swiftly identify and resolve any potential problems before they impact the overall user experience or regulatory compliance.

To achieve these goals, an observability framework based on the trace-dimension logging paradigm can be implemented. This framework leverages various technologies such as logs, Kafka, or similar data sources to capture and analyse transaction-related information. By adopting this approach, financial institutions can establish a robust system of observability, enabling them to monitor, analyse, and act upon critical data in real-time. This not only enhances the overall reliability and efficiency of the ecosystem but also strengthens user trust, fosters regulatory confidence, and paves the way for a seamless and secure electronic future of financial transactions.
About VuNet Systems

VuNet’s flagship product vuSmartMaps™ introduces a domain context to observability and augments business journey to transaction metrics traceability across the stack of distributed systems in a digital-first world. vuSmartMaps™ is a next-generation full-stack observability solution built using big data and machine learning in innovative ways to improve user experience. vuOTT™ is our strategic framework for driving Deep Observability Inside, Transparency Outside and Traceability Across. Together with vuSmartMaps™, it is a key pillar for digital transformation, as it enables teams across the board to have a unified and real-time view of interactions with multiple partners and stakeholders.

VuNet works with more than 20+ large financial institutions and monitors at the scale of 10 billion transactions monthly, helping to integrate all disparate touchpoints to intelligent journey views and insights, cutting across instant payments to remittances, retail to corporate channels and from middleware to core banking systems.