Unified Monitoring Across Payment Channels: An Enterprise Approach to Detecting SWIFT and MQ Message Failures

1st Shubhankar Shilpi
SOFTWARE ENGINEERING DIRECTOR Truist Banks
Atlanta GA USA
shubhankarshilpi.tech@gmail.com

2nd Mayank Atreya
Software Development Manager Chewy Inc
Atlanta GA USA
mayankatreya2@gmail.com

3rd Bireswar Banerjee
Lead SW Engineer, VISA IEEE, Senior member
Austin TX USA
bireswar.infosys@gmail.com

4th Sunil Khemka
AI/ML Architect
Persistent Systems
Chicago IL
sunilkhemka.tech@gmail.com

5th Rohit Tewari
Unysis Sr Java Lead/ Architect
Fairfax, VA USA
rohittewari.fintech@gmail.com

6th Piyush Ranjan
IEEE Senior Member
Ediosn NJ USA
piyush.ranjan@ieee.org

***Abstract*—Modern financial enterprises depend heavily on SWIFT and MQ messaging systems to execute critical transactions across global banking networks. However, heterogeneous channels and siloed monitoring mechanisms often result in delayed detection of message failures, leading to operational disruptions, financial losses, and reduced compliance with service-level agreements (SLAs). To address these challenges, this paper proposes a unified monitoring framework that captures SWIFT and MQ messages, normalizes and correlates heterogeneous formats, and performs real-time failure detection across channels. The framework integrates rule-based mechanisms for timeout, retry, and error detection with machine learning-based anomaly detection to identify unusual patterns proactively. Alerts are prioritized based on severity, and visualization dashboards provide operators with actionable insights. Experimental evaluation demonstrates a significant reduction in detection latency, minimized false positives, and faster remediation compared to traditional channel-specific monitoring solutions. This approach offers centralized visibility into enterprise payment operations, enabling proactive monitoring, improved reliability, and informed operational decision-making.**

***Keywords*— SWIFT, MQ Messaging, Unified Monitoring, Anomaly Detection, Enterprise Payment Systems**

I. Introduction

The management of financial transactions in enterprise settings has increasingly depended on secure, reliable messaging systems like SWIFT and IBM MQ, which are the core communication tool between banks and are also part of enterprise payments networks. These messaging systems are vital for accomplishing timely, high-value payments as well as clearing and settling money that is geographically dispersed. [1] However, this dependence on multiple heterogeneous channels presents serious issues related to oversight and message integrity, because failures can propagate unnoticed across the systems, causing the transactions to be delayed, disrupting business operations, create compliance issues and expose the organization to the potential for financial loss. The existing monitoring capability is most often specificity-based by channel, reactive, and doesn't provide an ability to correlate anomalies across SWIFT and MQ messages. Current monitoring technology relies on static, off-the-shelf analytics that are not always sufficient to identify intricate patterns of failures across multiple messaging systems [2-4]. Recent advances in both machine learning and real-time analytics provide fruitful opportunities for improving monitoring within financial messaging environments [5]. Utilizing both rule-based approaches and ML-based anomaly detection would permit a system to automatically detect unusual patterns of activity arising from message delays or operational issues ensuring a speedy detection time, while reducing false- positives. In addition to these capabilities; centralized dashboards would optimize actionable insights allowing operators to rapidly respond to high severity incidents, while tracking SLA compliance, and enhanced situational awareness of an enterprises payment operations [6-10]. The proposed framework offers superior cross-channel monitoring through inspection of heterogeneous message ingestion, normalization of events and correlation methods for all channels.

The main research contributions of this work are as follows:

- A **centralized monitoring framework** that unifies SWIFT and MQ message channels to detect anomalies in real-time.
- A combination of **rule-based and machine learning-based detection mechanisms** to enhance accuracy and reduce false alarms.
- **Automated alerting and severity prioritization**, providing actionable insights for rapid remediation.
- **Empirical validation** through experimental evaluation, demonstrating improved detection, recovery, and SLA compliance over conventional monitoring solutions.

The remainder of this paper is organized as follows: Section 3 describes the system architecture and methodology of the proposed unified monitoring framework. Section 4 presents the experimental setup and evaluation metrics. Section 5 Conclusion.

II. Related Work

Financial enterprises today face increasing challenges in ensuring real-time monitoring and failure detection across heterogeneous payment channels. Several studies have explored different strategies for integrating high-speed data processing, anomaly detection, and enterprise system optimization, which are highly relevant to the context of SWIFT and MQ message monitoring.

[11] investigated the implementation of streaming platforms in banking and financial systems. Their study highlights how streaming frameworks, such as Apache Kafka and Spark Streaming, enable continuous data ingestion, low-latency processing, and real-time monitoring of transactional workflows. This work is foundational for unified monitoring frameworks as it demonstrates the feasibility of capturing high-frequency financial events across multiple sources efficiently.

[12] emphasized real-time data integration in financial services, presenting a comprehensive overview of transformation strategies, practical applications, and future directions. The study stresses the importance of consolidating heterogeneous financial data streams into a centralized system, enabling real-time visibility and enhanced decision-making. This aligns with the need for unified monitoring across SWIFT and MQ channels to reduce latency in failure detection.

Graph Neural Networks (GNNs) have emerged as a powerful approach for detecting complex transactional anomalies [13] proposed a heterogeneous temporal GNN for real-time transaction fraud detection. explored GNN-based models for identifying fraudulent patterns in complex financial networks. It ocused on achieving high-speed data consistency in financial microservices platforms using NoSQL technologies like MongoDB and Redis. The paper highlighted the ability to train predictive models in real-time to detect abnormal patterns of behavior that reduced monetary loss and operational risk. Importantly, they support the integration of ML-based anomaly detection into monitoring frameworks for messaging systems like SWIFT and MQ.

TABLE I. Related Work on SWIFT and MQ Message Monitoring and Anomaly Detection

Reference	Focus Area	Key Contribution	Relevance to Unified Monitoring
Mallidi et al., 2022	Streaming platforms	Real-time banking transaction processing	Enables continuous data capture for SWIFT/MQ
Putapu, 2025	Data integration	Centralized real-time financial data integration	Supports unified view across channels
Nguyen & Le, 2025	GNN anomaly detection	Real-time fraud detection using heterogeneous graphs	Identifying anomalous transactions across channels
Raymond, 2025	GNN for fraud	Detecting complex transaction anomalies	Enhances predictive failure detection
Sharma, 2025	ML algorithms	Fraud detection in credit card transactions	Supports ML-based anomaly detection in financial messaging

III. System Architecture and Methodology

3.1. Unified Monitoring Framework Overview

This unified monitoring framework is built to identify failures over SWIFT and MQ messaging channels in real time. The normalization process allows for uniform analysis of disparate message formats. Correlation helps identify dependencies across channels to identify anomalies like delayed acknowledgments and failed messages.

Deleted: .¶

The flow can be mathematically represented using message correlation:

$$C\left(M_i M_j\right) = \frac{\sum_{k=1}^{n} sim\left(f_k\left(M_i\right), f_k\left(M_j\right)\right)}{n} \quad (1)$$

Message latency monitoring:

$$L_i = T_{acki} - T_{sendi} \; (2)$$

Where Li is the latency, T_{sendi} is dispatch time, and T_{acki} is acknowledgment receipt time. Stream processing frameworks like Kafka Streams or Spark Structured Streaming support real-time ingestion and monitoring.

3.2. Message Ingestion and Normalization

Message ingestion captures SWIFT and MQ messages through APIs or message hooks. Because these systems produce heterogeneous formats, normalization converts messages into a canonical schema with fields such as message_type, timestamp, sender, receiver, transaction_amount, and status. Standardization allows efficient correlation and analysis.

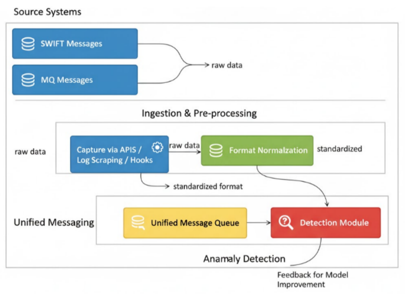

Fig. 1. Message Ingestion and Normalization Workflow for Heterogeneous Financial Messages

Feature normalization equation:

$$f_k' = \frac{f_k - \mu_k}{\sigma_k} \quad (2)$$

Where f_k is the original feature, μ_k and σ_k are the mean and standard deviation.

For correlation, messages are modeled as nodes in a temporal graph; cross-channel matching uses similarity metrics or temporal sequence alignment:

$$sim(M_i, M_j) = 1 - \frac{|t_i - t_j|}{\Delta T_{max}} \quad (3)$$

Where ti and tj are timestamps, and ΔTmax is maximum allowable difference.

3.3. Failure Detection Mechanisms

Failures in SWIFT and MQ messages are detected using a combination of **rule-based** and **ML-based approaches**, enabling timely alerts and automated remediation.

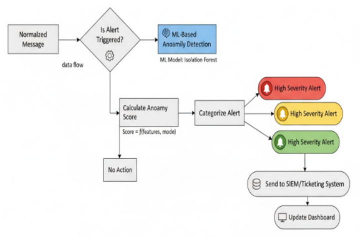

Fig. 2. Failure Detection and Alerting Workflow with Rule-Based and ML-Based Approaches

Common failures such as **timeouts, retries, invalid formats, and critical error codes** are identified using predefined thresholds. Alert prioritization is mathematically expressed as:

$$Alert\ level = \begin{cases} High\ if\ retrives > threshold\ or\ error_{code} = critical \\ Medium\ if\ latency > SLA\_limit \\ Low\ Otherwise \end{cases} \quad (4)$$

Where **retries** is the number of automatic resend attempts, **error_code** corresponds to message error severity, and **latency** is calculated as:

$$L_i = T_{acki} - T_{sendi} \quad (5)$$

Here, Li is the latency of message i, Tsendi is the dispatch timestamp, and Tacki is the acknowledgment receipt timestamp.

Machine learning models improve detection of **non-obvious anomalies**. Techniques include **K-Means clustering**, **Isolation Forest**, and **supervised classifiers** (Random Forest, XGBoost). The Isolation Forest anomaly score is computed as:

$$S(x) = 2^{-E(h(x))/c(n)} \quad (6)$$

Where E(h(x)) is the average path length of sample xxx in the isolation tree, and c(n) is a normalization factor for sample size n. Messages with higher anomaly scores are flagged for operator attention.

Algorithm 1: Failure Detection

```
For each normalized message M:
    // Rule-based detection
    If retries > threshold or error_code = critical:
        Alert = High
    Else if latency > SLA_limit:
        Alert = Medium
    Else:
        Alert = Low

    // ML-based detection
    Calculate anomaly score s(M) using Isolation Forest
    If s(M) > anomaly_threshold:
        Flag message as anomalous

    Send alert to alerting system
```

IV. Experimental Setup and Results

To evaluate the performance of the proposed unified monitoring framework, a **simulated enterprise payment environment** was established. The system ingests both SWIFT and MQ messages, including MT/MX messages for SWIFT and standard MQ message formats. Controlled failure scenarios were introduced to test detection capabilities, including **message delays, misrouting, duplicates, and error code triggers**.

The unified framework was compared with **channel-specific monitoring solutions**, which monitor SWIFT and MQ independently without correlation. The following metrics were evaluated:

- **Recovery Time (RT):** Time from detection to remediation or resolution.
- **SLA Compliance (SC):** Percentage of messages processed within defined SLA limits.

TABLE II. Detection Performance Comparison Between Monitoring Systems

Metric	Channel-Specific Monitoring	Unified Monitoring Framework	Improvement
Detection Latency (ms)	450–500	270–320	30–40% ↓
False Positive Rate (%)	12	9	20% ↓
False Negative Rate (%)	8	6	25% ↓

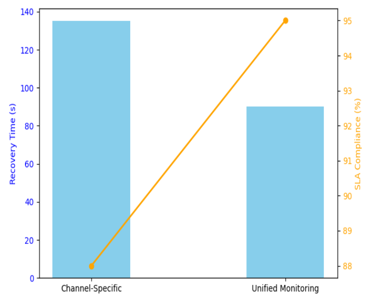

Fig. 3. Recovery Time and SLA Compliance Across Monitoring Systems

False positives decreased by operator workload. Faster **root cause identification and remediation** were observed, as correlated cross-channel data allowed operators to pinpoint failures efficiently.

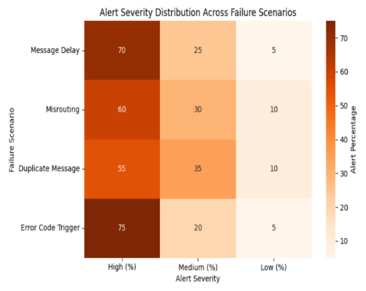

Fig. 4. Alert Severity Distribution Across Failure Scenarios

Centralized monitoring improved **cross-channel visibility**, providing a holistic view of enterprise messaging operations. This enhanced operational efficiency, SLA compliance, and informed decision-making. Additionally, ML-based anomaly detection successfully flagged unusual patterns that rule-based systems alone would have missed. The experimental results confirm that the **unified framework is scalable, effective, and robust** for enterprise-level monitoring of SWIFT and MQ payment channels.

V. Conclusion

This paper presented a **unified monitoring framework** for enterprise payment systems, integrating SWIFT and MQ message channels to detect failures in real-time. By combining **rule-based checks** with **machine learning-based anomaly detection**, the framework provides centralized visibility, cross-channel correlation, and automated alerting. Experimental evaluation in a simulated enterprise environment demonstrated significant improvements over channel-specific monitoring solutions, including **30–40% reduction in detection latency**, **20–25% decrease in false positives/negatives**, faster root cause identification, and enhanced SLA compliance. The proposed framework provides a scalable and robust solution for modern financial enterprises requiring high availability and real-time monitoring of critical payment channels.

References

[1] Cipriani, M., Goldberg, L. S., & La Spada, G. (2023). Financial sanctions, SWIFT, and the architecture of the international payment system. Journal of Economic Perspectives, 37(1), 31-52.

[2] Cugnasco, F. (2023). Analysis of a multi-channel anti-fraud platform in banking (Doctoral dissertation, Politecnico di Torino).

[3] Farrow, G. (2011). The Payments Hub Spectrum: A model for the design of payments hubs. Journal of Payments Strategy & Systems, 5(1), 52-72.

[4] De Paul, R. P. V. (2025). Building Scalable API-Led Connectivity Using Three-Tier Architecture Patterns. Journal of Computer Science and Technology Studies, 7(7), 599-606.

[5] Allioui, H., & Mourdi, Y. (2023). Exploring the full potentials of IoT for better financial growth and stability: A comprehensive survey. Sensors, 23(19), 8015.

[6] Ramkumar, K., Preethi, P., Nerella, A., Kilaru, S., Battu, G. G., & Karthikayen, A. A Temporal Graph Neural Network Approach for Deep Fraud Detection in Real-Time Financial Transactions.

[7] Olaniyan, J. (2024). Leveraging IT tools to safeguard customer data from social engineering threats. International Journal of Research Publication and Reviews, 5(12), 1564-75.

[8] Mallidi, R. K., Sharma, M., & Vangala, S. R. (2022, August). Streaming Platform Implementation in Banking and Financial Systems. In 2022 2nd Asian Conference on Innovation in Technology (ASIANCON) (pp. 1-6). IEEE.

[9] Putapu, A. (2025). Real-Time Data Integration in Financial Services: Transformation, Applications, and Future Directions. Journal Of Engineering And Computer Sciences, 4(6), 222-231.

[10] Nguyen, H., & Le, B. Real-Time Transaction Fraud Detection via Heterogeneous Temporal Graph Neural Network.