Radhika Kande[1], Chaithanya Kotla[2], Krishna kanth Thottempudi[3]
[1]Premier Inc, USA
[2]Devops and Cloud lead, State of Maryland, USA
[3]Infosys Limited, USA

Received: 29-06-2023; Revised: 03-08-2023; Accepted: 09-10-2023

Data Quality Firewall for Real-Time Schema Validation and Automated Quarantine in Analytics Pipelines

Abstract
Streaming analytics pipelines increasingly depend on immediate and reliable data intake, yet malformed records can still propagate into downstream dashboards, models, and decision systems before delayed validation catches them. Existing work on data quality management, schema enforcement, and observability highlights the importance of early validation, but practical streaming architectures that combine real-time schema checking with automated quarantine remain limited. This article presents a metadata-driven data quality firewall for analytics pipelines that performs inline schema validation, classifies failures during ingestion, routes invalid records into structured quarantine streams, and supports replay-based recovery after correction. The results show that the proposed framework maintains high schema validation accuracy and quarantine precision while adding only moderate ingestion latency, and it also improves downstream error reduction, replay recovery success, and pipeline continuity under multiple data quality failure scenarios. The study shows that a streaming data quality firewall can provide a practical foundation for protecting analytics pipelines from live data contamination.

Keywords: data quality firewall, schema validation, automated quarantine, streaming analytics, replay recovery, pipeline continuity.

1. Introduction

Streaming analytics pipelines now drive dashboards, alerts, models, and operational decisions in near real time, which means that malformed records can contaminate downstream systems before human intervention is possible. In traditional batch-oriented quality control, invalid data can often be discovered and corrected after ingestion. In streaming environments, however, the same delay can propagate broken schemas, null-sensitive failures, aggregation distortion, and misleading business metrics across multiple consumers. Recent work on data quality management in large-scale data systems has shown that quality assurance is no longer only a governance task; it has become an architectural requirement for dependable analytics operation [1]. This makes early validation a first-class concern in modern pipeline design.

A second challenge is freshness. Many organizations now optimize aggressively for low-latency delivery, but speed increases the risk that poor-quality events enter production flows before validation catches them. Studies on item freshness in data streams have shown that recency and correctness must be treated together, because highly fresh but invalid data is still analytically harmful [2]. In practice, teams often respond by adding ad hoc checks inside consumers or transformation jobs, which fragments quality control across the pipeline. This increases maintenance cost and makes validation behavior inconsistent. A more robust design must therefore separate quality enforcement from individual downstream applications.

Metadata-driven ingestion patterns offer an important foundation for solving this problem because validation can be attached to schema definitions, field rules, source contracts, and runtime handling policies rather than embedded repeatedly in custom code [3]. This allows source-specific quality requirements to be governed centrally while still being executed at stream speed. Observability research for data quality has similarly shown that real-time visibility into failing records, drift patterns, and validation anomalies improves responsible operation of production data systems [4]. Together, these directions suggest that streaming data quality should be managed as an active control layer. The architecture should be able to inspect, decide, isolate, and report in one continuous workflow.

Another practical limitation of many existing approaches is that invalid records are either dropped silently or allowed to proceed with warnings. Both choices are dangerous in analytical pipelines. Silent dropping hides data loss, while permissive forwarding spreads corruption into storage layers, dashboards, machine learning features, and business logic. Validation research based on explicit structural constraints has shown that rule-driven acceptance boundaries are far more reliable than weak downstream checking or informal assumptions about schema stability [5]. What is missing is an architectural model that enforces these boundaries before contamination occurs. A firewall-style pattern is therefore more appropriate than a passive monitoring pattern.

This study proposes a data quality firewall for analytics pipelines, implemented as a metadata-driven streaming architecture for real-time schema validation and automated quarantine. The framework validates records against schema contracts at ingestion time, assigns handling actions according to failure type, routes rejected events into structured quarantine streams, and preserves observability for replay and remediation. The evaluation focuses on schema validation accuracy, quarantine precision, ingestion latency, downstream error reduction, replay recovery success, and pipeline continuity under variable event volume and failure scenarios. The goal is to show that data quality can be enforced as an active streaming boundary rather than as a delayed corrective process.

2. Methodology

The proposed methodology organizes the firewall as an inline streaming control layer positioned between upstream event producers and downstream analytics consumers. Rather than waiting for transformations, warehouses, or dashboards to detect malformed records, the architecture intercepts each event at ingestion time and subjects it to contract-aware validation before it is allowed into the trusted analytical path. Enterprise schema-registry and quality-engine studies have shown that real-time validation becomes more effective when schema governance and execution logic are integrated directly into ingestion architecture [6]. The firewall therefore acts as both a gate and a classifier. Its main purpose is to preserve downstream trust without forcing all consumers to implement their own quality logic.

The first architectural component is the schema registry and rule layer. Each source topic, event family, or ingestion channel is registered with versioned structural definitions, mandatory-field conditions, datatype expectations, range constraints, enumerated values, timestamp policies, and optional business-rule extensions. Incoming records are checked against the currently active schema contract before any analytical routing takes place. Research on automatically generating data-quality constraints for recurring pipelines has shown that repeatable validation becomes more reliable when rule application is formalized across execution cycles [7]. This allows the firewall to handle both strict structural failures and more nuanced semantic mismatches within one governed control surface.

The second component is the record inspection engine. As each event arrives, the engine parses the payload, resolves the applicable schema version, evaluates rule conditions, and assigns a validation outcome. Records can be marked as accepted, corrected-with-tag, suspicious, or rejected depending on the failure severity and configured action policy. Adaptive systems research has shown that runtime control becomes more dependable when decision behavior is informed by changing operational context rather than fixed assumptions alone [8]. In the present framework, this principle is reflected in how the inspection engine uses source context, schema state, and validation history to classify incoming records. The result is a deterministic but context-aware streaming decision process.

A third component is the quarantine router, which is responsible for isolating events that fail critical checks. Instead of discarding these records or forwarding them downstream with warning flags, the firewall writes them into structured quarantine streams together with failure metadata, schema version references, source identifiers, event timestamps, and validation error codes. This preserves traceability and supports later replay after correction. The quarantine design therefore treats bad records as controlled exceptions rather than invisible losses. This is important because reliable analytics systems need both protection from contamination and accountability for rejected data.

The main validation and quarantine flow is summarized in Table 1. The table is placed here because it directly supports the architectural explanation of how records move through the firewall and how rule outcomes are translated into routing decisions.

Table 1. Validation Stages, Quarantine Actions, and Record Handling Roles in the Data Quality Firewall

Validation Stage	Primary Check	Handling Action	Operational Role
Source registration	Source identity and topic binding	Accept source into monitored stream set	Establishes governed ingress path
Schema resolution	Version and contract lookup	Bind event to active validation profile	Ensures correct rule context
Structural validation	Required fields, type checks, field presence	Accept or flag	Prevents malformed structure propagation
Semantic validation	Range, enum, business-rule checks	Accept, suspicious tag, or reject	Filters logically inconsistent data
Quarantine routing	Failure classification and severity assignment	Route to quarantine stream	Isolates contaminated records
Metadata tagging	Error code, schema ID, source ID, timestamp	Append lineage and audit context	Supports replay and remediation
Replay admission	Corrected-record revalidation	Return to trusted stream	Restores data after correction
Monitoring and audit	Quality metrics and anomaly visibility	Emit observability events	Preserves operational transparency

To support operational handling, the firewall maintains a metadata envelope around each rejected record. This envelope contains validation context, failure stage, source lineage, and replay eligibility status so that quarantine is not merely a dead-end storage location. The same metadata is exposed to monitoring systems and remediation workflows, which allows operations teams to identify recurring schema drift, source-specific failures, or rapidly growing quarantine volumes. Observability work for production data quality has emphasized that visibility into failure patterns is as important as failure detection itself [9]. The firewall therefore combines enforcement with explanation. This makes it possible to improve upstream sources instead of endlessly reacting downstream.

Replay and remediation are handled through a controlled recovery path. Once quarantined records are corrected manually or programmatically, they are revalidated against the applicable schema before re-entry into the trusted stream. This prevents quarantine from becoming a bypass around the firewall and

ensures that recovered records satisfy the same acceptance logic as first-pass events. In practice, this supports both source-side remediation and downstream correction workflows. Streaming and serverless pipeline studies have shown that recovery-aware pipeline design becomes more important as data movement grows more distributed and event-driven [10]. The replay mechanism therefore closes the loop between rejection and restoration.

Because the firewall operates inline with live ingestion, monitoring of latency, throughput, and validation overhead is essential. Monitoring research for cloud services has shown that production reliability depends heavily on continuous visibility into pipeline behavior rather than only into final outcomes [11]. The firewall therefore emits metrics for validation pass rate, quarantine rate, per-source anomaly concentration, replay success, inspection latency, and stream continuity. These metrics are used to assess both protection quality and operational sustainability. In this way, monitoring becomes part of the control plane rather than a passive reporting layer.

The evaluation environment compares the proposed architecture against a baseline ingestion pipeline with delayed downstream validation. The main metrics are schema validation accuracy, quarantine precision, ingestion latency, downstream error reduction, replay recovery success, and pipeline continuity under varied event volume and failure types. The framework is assessed as a streaming reliability mechanism rather than simply a rule engine. Its purpose is to determine whether real-time quarantine can protect analytical systems without causing unacceptable disruption to ingestion performance.

3. Results and Discussion

The data quality firewall was evaluated under streaming workloads that combined valid events, malformed schema instances, null-heavy payloads, datatype mismatches, field-order drift, and semantically inconsistent records. The baseline pipeline accepted all events into the main stream and relied on delayed validation inside downstream processing stages. Across the full workload range, the firewall reduced the spread of invalid data into analytical consumers while preserving stable ingestion behavior. The strongest gains appeared when malformed events arrived in bursts rather than as isolated outliers, because the quarantine path prevented batch contamination of downstream windows and state stores. This indicates that real-time isolation is especially valuable under clustered data-quality failure conditions.

Figure 1 presents schema validation accuracy, quarantine precision, and ingestion latency under increasing streaming event volume. The results show that validation accuracy remains high across the workload range, which suggests that the contract-aware inspection engine is preserving rule correctness even as throughput pressure grows. Quarantine precision also remains strong, meaning that rejected events are largely true failure cases rather than excessive false positives. Ingestion latency rises

moderately with volume, but the increase remains bounded, which indicates that the firewall adds manageable overhead relative to the protection it provides. These results show that the architecture can enforce quality control at stream speed without becoming prohibitively intrusive.

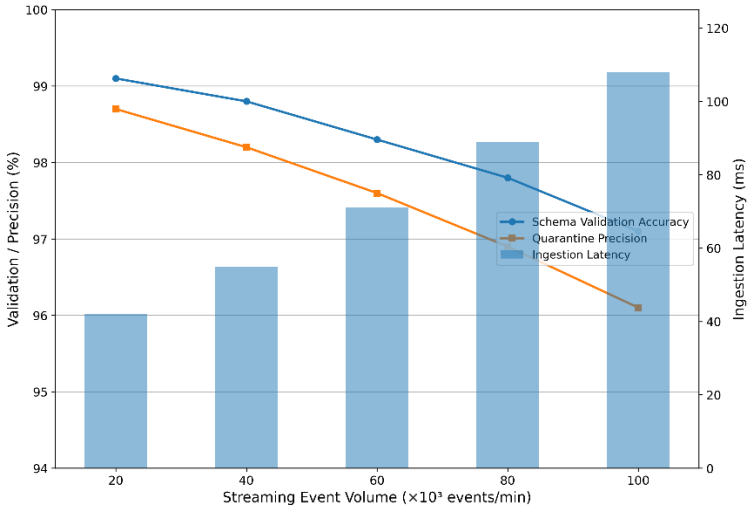

Figure 1. Schema Validation Accuracy, Quarantine Precision, and Ingestion Latency Under Increasing Streaming Event Volume

The practical significance of these results lies in the tradeoff between protection and throughput. A quality layer that blocks bad data but cripples ingestion speed is not operationally useful, while a fast pipeline that forwards corrupted records is analytically unsafe. In the proposed architecture, the firewall retains strong validation performance with only moderate latency growth, which means the inspection and routing logic remains compatible with real-time pipeline operation. This is especially important for streaming dashboards, operational alerts, and online feature generation, where malformed records can have immediate effects if not intercepted. The firewall therefore improves trustworthiness without forcing a return to slow post-ingestion cleansing.

Figure 2 shows downstream error reduction, replay recovery success, and pipeline continuity across data quality failure scenarios. The results indicate that downstream error rates fall sharply when malformed events are isolated before analytical consumption, particularly under schema drift and mixed-failure cases where conventional delayed validation often struggles. Replay recovery success remains high because quarantined records are preserved with sufficient metadata for structured correction and re-entry. Pipeline continuity also remains strong, which means that quarantining invalid records does not destabilize the trusted event path for valid data. This is an important property because the architecture is intended to isolate contamination, not fragment the streaming pipeline itself.

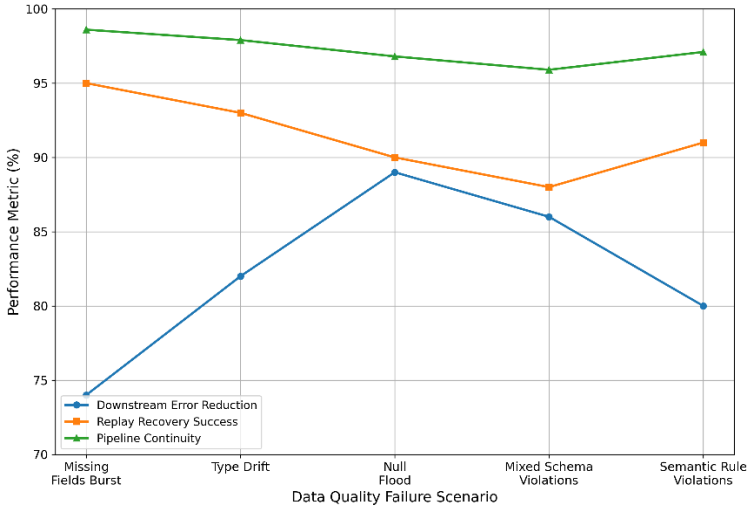

Figure 2. Downstream Error Reduction, Replay Recovery Success, and Pipeline Continuity Across Data Quality Failure Scenarios

Another important observation is that the firewall changes the operating model of data quality from reactive cleanup to proactive containment. Instead of treating quality issues as downstream repair tasks, the architecture creates an upstream decision boundary that classifies and isolates risk before it can propagate. This makes analytical systems more resilient under schema instability, evolving sources, and bursty ingestion patterns. The results therefore show that a streaming data quality firewall is not simply a validation utility. It is a protective control layer that preserves both data trust and pipeline continuity under live operational conditions.

4. Conclusion

This study presented a data quality firewall as a metadata-driven streaming architecture for real-time schema validation and automated quarantine in analytics pipelines. The proposed framework combines schema-aware inspection, deterministic rule evaluation, quarantine routing, metadata-rich rejection handling, and replay-based recovery within one inline control layer. By placing this logic at ingestion time rather than in downstream consumers, the architecture prevents malformed events from contaminating analytical state before corrective action is possible. The work therefore reframes data quality enforcement as a streaming protection problem rather than a delayed cleansing problem.

The results showed that the firewall maintains strong validation accuracy and quarantine precision while adding only moderate ingestion overhead under increasing event volume. It also significantly reduces downstream error propagation and preserves high replay recovery success across multiple failure scenarios. A key contribution of the study is that it demonstrates how automated isolation can improve trust in live analytics systems without breaking the operational continuity of the valid event path. This

makes the architecture suitable for environments where data quality failures can quickly propagate into dashboards, automated decisions, or machine learning features.

A broader implication of this work is that future analytics platforms will increasingly require active data quality enforcement at ingestion time as streaming systems become more central to enterprise decision-making. Delayed validation and ad hoc downstream checks are less effective when malformed events move at production speed across many consumers. A firewall-style streaming boundary offers a practical route toward safer and more accountable analytics pipelines because it combines prevention, traceability, and controlled recovery in one architecture. The framework developed here provides a useful basis for more resilient streaming data platforms in which quality failures are contained before they become analytical failures.

References

1. Taleb, I., Serhani, M. A., Bouhaddioui, C., & Dssouli, R. (2021). Big data quality framework: a holistic approach to continuous quality management. *Journal of Big Data*, *8*(1), 76.
2. Liu, Z., Jiang, Z., Zhang, A., Shi, Z., Tian, Y., & Yang, T. (2025, August). Measuring Item Freshness in Data Streams. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2* (pp. 1963-1974).
3. Rucco, C., Longo, A., & Saad, M. (2025). MIND: A Metadata-driven INgestion Design pattern for efficient data ingestion. *Big Data Research*, 100574.
4. Truong, H. L., & Nguyen, N. N. T. (2024). TENSAI-Practical and Responsible Observability for Data Quality-aware Large-scale Analytics. *ACM Journal of Data and Information Quality*, *16*(4), 1-43.
5. Bachinger, F., Ehrlinger, L., Kronberger, G., & Wöß, W. (2024). Data validation utilizing expert knowledge and shape constraints. *ACM Journal of Data and Information Quality*, *16*(2), 1-27.
6. Zhao, Z., & Wang, X. (2021, September). Design and Implementation of Enterprise Public Data Management Platform Based on Artificial Intelligence. In *International Conference on Cognitive based Information Processing and Applications (CIPA 2021) Volume 1* (pp. 702-710). Singapore: Springer Singapore.
7. Tu, D., He, Y., Cui, W., Ge, S., Zhang, H., Han, S., ... & Chaudhuri, S. (2023, August). Auto-validate by-history: Auto-program data quality constraints to validate recurring data pipelines. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 4991-5003).
8. Weerasinghe, S., Zaslavsky, A., Loke, S. W., Medvedev, A., Abken, A., Hassani, A., & Huang, G. L. (2024). Reinforcement learning based approaches to adaptive context caching in distributed context management systems. *ACM Transactions on Internet of Things*, *5*(2), 1-32.

9. Srinivas, P., Husain, F., Parayil, A., Choure, A., Bansal, C., & Rajmohan, S. (2024, April). Intelligent monitoring framework for cloud services: A data-driven approach. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice* (pp. 381-391).
10. Valiaiev, D. (2025). *Implementing Dataops: A Scalable Framework for Modern Data Warehousing* (Doctoral dissertation, University of Arkansas at Little Rock).
11. Gavhale, A., Joshi, N., Jahagirdar, R., & Varade, P. S. (2024, December). Data Pipeline Approaches in Serverless Computing. In *International Conference on Information and Communication Technology for Competitive Strategies* (pp. 205-216). Singapore: Springer Nature Singapore.