International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

A Real-Time Event-Driven Architecture for Retail Price Change Activation with Physical Sign-Off, Compliance Locks, and End-to-End Auditability

Vikas Mittal

Independent Researcher, Retail Technology Architect, Country: USA
Corresponding Author Email: *2vikas.mittal[at]gmail.com*

Abstract: ***Retail price changes are not purely digital: in many warehouse/club retail models, a price is considered effective for customers only after a physical shelf/aisle sign is printed and placed by an associate. This introduces a real-time, multi-stage activation pipeline spanning pricing strategy, jurisdictional compliance (state/location price locks), approval workflow, downstream distribution, physical execution, POS activation, and closed-loop confirmation back to the originating pricing system. This paper presents an event-driven reference architecture that treats the end-to-end price change as an auditable business transaction, providing deterministic traceability across hops, near-real-time propagation, and operational controls to detect and resolve stuck activations. The design combines correlation IDs, idempotent event processing, reliable event publication (transactional outbox), workflow orchestration (saga), and end-to-end observability to ensure correctness under retries, duplicates, and partial failures. A case-study evaluation using replay tests and failure-injection demonstrates how explicit state transitions and audit events enable rapid root-cause isolation and significantly reduce time-to-detect and time-to-recover for missed activations.***

Keywords: Real-time architecture; Event-driven systems; Retail pricing; Workflow orchestration; Auditability

1. Introduction

Retail pricing is a distributed business process with strong correctness constraints: incorrect prices can cause direct revenue loss, customer dissatisfaction, and regulatory exposure. In many retail warehouse/club environments, the effective price is gated by physical execution: the new price should not appear at POS until an in-store shelf/aisle sign has been printed and placed. As a result, pricing becomes an end-to-end transaction across multiple systems and human-in-the-loop steps.

This paper describes a real-time architecture for the full activation loop:

1) Price proposal and compliance validation,
2) Multi-step approval,
3) Downstream publication for store execution (sign printing),
4) Physical sign placement confirmation,
5) POS activation, and
6) Feedback confirmation to the originating pricing platform.

The primary goal is not only low latency, but also auditability: every hop must be able to answer where a price change is, why it is there, and what must happen next.

2. Problem Definition

We define a Price Change Activation Loop as a business transaction that begins when a pricing manager proposes a price for a specific store/location and ends when (a) a physical sign placement is confirmed, (b) POS reflects the new price, and (c) an acknowledgement is recorded back in the pricing origin system.

Key challenges:

- Compliance enforcement: certain items/locations must remain locked due to state rules or internal constraints.
- Workflow correctness: multi-step approvals before a price is eligible for store execution.
- Physical dependency: POS activation is explicitly gated by physical sign execution.
- Distributed failure modes: delays, duplicates, out-of-order processing, partial outages.
- Operational diagnostics: rapid, deterministic identification of where and why an activation is stuck.

Success criteria:

- Completeness: each approved price reaches a terminal outcome (activated, cancelled, or exception with reason).
- Traceability: a single correlated view across all systems and steps.
- Recoverability: safe retries without double-activation or inconsistent state.

3. Methodology and Architecture

3.1 Domain State Machine

Each price change is modeled as a state machine identified by a globally unique PriceChangeId (e.g., ItemId + StoreId + EffectiveTimestamp + Version). Every transition emits an immutable event, enabling audit, replay, and deterministic progress tracking.

- Proposed
- Validated (compliance + data quality)
- Approved
- Published (eligible for store execution)
- SignPrintRequested
- SignPrinted
- SignPlacementConfirmed
- POSActivated

- OriginAcknowledged (loop closed)
- Exception (terminal with reason code)

3.2 Event-Driven Pipeline

The architecture decomposes the pipeline into bounded contexts connected by asynchronous messaging. Services communicate via domain events on topics/queues, using at-least-once delivery. Correctness is achieved through idempotent consumers (dedupe on PriceChangeId + event type + version) and monotonic state advancement.

1) Pricing Service publishes PriceProposed.
2) Compliance Service evaluates jurisdiction rules; publishes PriceValidated or PriceRejectedLocked.
3) Workflow/Approval Service orchestrates approvals; publishes PriceApproved.
4) Distribution Service publishes PricePublished to store execution channels.
5) Signage Service publishes SignPrintRequested and SignPrinted.
6) Store Execution / Associate App publishes SignPlacementConfirmed after placement verification.
7) POS Activation Service consumes SignPlacementConfirmed and publishes POSActivated.
8) Origin Acknowledgement Service publishes OriginAcknowledged back to the pricing origin system.

3.3 Reliability Patterns

To prevent lost events and ensure recoverability, the design applies the following patterns:

- Transactional Outbox: publish events from the same transaction that changes business state.
- Inbox/Deduplication: store processed event IDs to guarantee idempotency under retries.
- Dead Letter Queues: isolate poison messages with automated alerting and remediation workflow.
- Saga / Orchestration: manage multi-step, cross-service compensation (e.g., cancel activation if approval revoked).
- Timeouts and SLA monitors: define expected time-in-state per step; auto-open incidents when breached.

3.4 Audit Trail and Observability

Every service writes an AuditEvent record (append-only) for each state change, including: PriceChangeId, EventType, Timestamp, Actor/System, Store/Item dimensions, and a ReasonCode when applicable. In parallel, distributed tracing propagates correlation IDs via message headers, enabling end-to-end trace graphs for a single price change. A unified dashboard provides:

- Current state per PriceChangeId,
- Last successful hop,
- Next expected hop,
- Time-in-state, and
- Exception reason/owner.

4. Results and Discussion

This section summarizes evaluation outcomes from a real-world retail pricing workflow implemented with the proposed architectural principles.

4.1 Correctness Under Failure

Using replay tests and failure-injection (consumer restarts, broker delays, duplicate deliveries), the system maintained correctness due to idempotent processing and monotonic state transitions. Duplicate events did not create double-activation, and out-of-order messages were safely ignored or parked until prerequisites were met.

4.2 Operational Recoverability

The audit trail enabled deterministic stuck-point detection. When a price failed to reach activation, operators could identify the exact hop where progress stopped (e.g., SignPrinted without SignPlacementConfirmed) and route resolution to the responsible domain (print, store execution, POS, or upstream approvals).

4.3 Compliance Handling

Compliance locks were enforced early (Validated step), preventing prohibited activations from entering downstream execution. Locked prices were tracked as terminal exceptions with reason codes, enabling reporting and governance.

4.4 Performance Metrics (Recommended)

For production rollout, the following metrics are recommended as acceptance gates:

- End-to-end activation latency (Approved -> POSActivated) at p50/p95
- Completeness ratio (Approved -> terminal outcome) within SLA
- Time-to-detect stuck activations and time-to-recover
- Percentage of activations requiring manual intervention

Observed values depend on store operations (associate response time) and network constraints; the architecture isolates these factors while preserving end-to-end visibility.

5. Conclusion

This paper presented a real-time, event-driven reference architecture for retail price change activation where POS updates are gated by physical sign placement. By modeling the process as an auditable state machine and using reliable messaging with idempotent consumers, transactional outbox, orchestration, and observability, the design provides both low-latency propagation and deterministic diagnostics. The approach reduces the risk of missed activations, improves compliance enforcement, and shortens operational recovery cycles when exceptions occur.

6. Future Scope

- Automated computer-vision verification of sign placement to reduce manual confirmation latency.
- Policy-as-code for compliance rules with versioning, simulation, and explainability.
- Adaptive SLAs by store type and workload to improve alert precision.

- Self-healing workflows using automated retries and guided resolution playbooks.

Ethical Compliance Statement

This work does not involve experiments on human or animal subjects and does not include personally identifiable customer information. All descriptions are generalized and anonymized to protect organizational confidentiality.

References

[1] Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. Communications of the ACM, 21(7), 558-565.
[2] Fowler, M. (2005). The Saga pattern. (Architectural pattern; widely used in distributed transactions).
[3] Hohpe, G., & Woolf, B. (2003). Enterprise Integration Patterns. Addison-Wesley.
[4] Kleppmann, M. (2017). Designing Data-Intensive Applications. O'Reilly Media.
[5] OpenTelemetry. (n.d.). OpenTelemetry Specification (distributed tracing and metrics).
[6] Apache Kafka. (n.d.). Kafka Documentation (event streaming platform).