International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(17s) 2026 pp. 918-929
https://doi.org/10.70917/ijcisim-2026-4810
Article

Engineering a Reverse Logistics Ecosystem for Mobile Device Commerce

V. L. Sowjanya Gajjala

Independent researcher, India, Email:sowjanya.mvl@gmail.com

Abstract: Reverse logistics in mobile device commerce — encompassing service-cancellation returns, device exchange returns, device upgrade returns, and trade-ins — is frequently engineered as an afterthought to forward fulfillment, resulting in fragmented, channel-specific workflows with limited operational visibility. This gap is acute in telecom mobile commerce, where returned and traded-in devices must be authorized, physically recovered through multiple channels, assessed for condition, credited or exchanged through billing systems, and routed to disposition outcomes such as resale, refurbishment, or recycling. This paper presents a Business Process Management (BPM)-driven architecture for an enterprise reverse logistics ecosystem, implemented as an extension of an existing BPM-driven order management platform. The proposed architecture introduces a unified return lifecycle state machine spanning in-store and ship-from-home return channels, a trade-in valuation and condition assessment workflow that reconciles quoted value against physically assessed condition, and a configurable disposition routing engine that directs recovered devices to resale, refurbishment, recycling, or forward inventory based on assessed condition. An event-driven integration layer using Apache Kafka coordinates asynchronous communication with logistics providers, carrier return-label services, and billing systems for credit and refund issuance. The architecture and the operational results reported here reflect a production extension of an existing order management platform within a major telecommunications carrier's mobile commerce business. Results demonstrate full return-state visibility where none previously existed, systematic condition-mismatch handling, and the ability to extend disposition logic without modifying the core orchestration layer. The architecture provides a practical reference for reverse logistics engineering in telecommunications and other device-centric commerce domains.

Keywords: Reverse Logistics, Returns Management, Trade-In, Closed-Loop Supply Chain, Business Process Management, Order Management System, Mobile Device Commerce, Workflow Orchestration

1. INTRODUCTION

Forward fulfillment — the process of getting a product from inventory to a customer — has been the dominant focus of commerce engineering literature and practice. The reverse journey — returns, trade-ins, warranty exchanges, and device recovery — receives comparatively little engineering attention despite representing a substantial and growing share of commerce operations. Rogers and Tibben-Lembke [1] define reverse logistics as the process of planning, implementing, and controlling the efficient flow of goods from the point of consumption back to the point of origin, and document that organizations consistently treat it as a secondary concern relative to forward distribution.

This asymmetry is particularly costly in telecom mobile commerce. Mobile devices have high unit value, a well-established secondary market, and a regulatory and competitive environment that drives frequent trade-in promotions tied to new device launches. Each of these — a customer returning a defective device, trading in a device from any manufacturer or carrier toward a new purchase promotion, or exchanging a device under warranty — triggers a distinct reverse journey — one that must be authorized, physically recovered, assessed for condition, reconciled against any quoted value, credited or refunded through billing systems, and routed to a disposition outcome. Unlike forward fulfillment, where the system has full knowledge of the item being shipped, reverse logistics carries inherent uncertainty: the condition of a returned or traded-in device is not definitively known until it is physically received and inspected, creating a structural mismatch between commitments made at the point of return initiation and the value that can ultimately be recovered.

Organizations that treat reverse logistics as a secondary system — built independently of the forward order management system, with channel-specific logic for in-store versus mailed returns, and without a formal state model — accumulate the same structural problems documented in forward order management: limited operational visibility, reactive exception handling, and high engineering cost when extending the system to new return types or disposition outcomes.

This paper presents a Business Process Management (BPM)-driven architecture for an enterprise reverse logistics ecosystem in telecom mobile device commerce. Rather than treating reverse logistics as an independent system, the proposed architecture extends an existing BPM-driven order management platform with child process definitions dedicated to return and trade-in workflows, sharing the same orchestration engine, integration layer, and operational visibility model used for forward fulfillment.

This paper makes the following contributions:

(1) A unified return lifecycle state machine that manages in-store and ship-from-home return channels through a single orchestration engine, despite their structurally divergent condition-assessment timing.

(2) A trade-in valuation and condition assessment workflow that reconciles a pre-return quoted value against physically assessed condition, with structured handling for value mismatches.

(3) A configurable disposition routing engine that directs recovered devices to resale, refurbishment, recycling, or forward inventory based on assessed condition, without requiring changes to the core orchestration layer.

(4) Implementation experience from extending a production BPM-driven order management platform with reverse logistics capability at a major U.S. telecommunications carrier.

The paper is organized as follows. Section II reviews related work. Section III defines the problem. Section IV presents the proposed architecture. Section V describes implementation. Section VI evaluates results. Section VII concludes.

2. RELATED WORK

A. Reverse Logistics and Closed-Loop Supply Chains

Reverse logistics has been studied extensively within operations and supply chain management. Rogers and Tibben-Lembke [1] provide the foundational definition of reverse logistics as a distinct managerial discipline and document, through industry survey data, the organizational barriers that lead firms to underinvest in returns infrastructure relative to forward distribution. Guide and Van Wassenhove [2] trace the evolution of closed-loop supply chain research from a narrow technical focus on individual recovery activities toward a holistic, business-process view of value recovery from returned products, and argue that closed-loop supply chains poorly linked to the forward business risk destroying rather than recovering value.

This body of work establishes the strategic and economic importance of reverse logistics but is concentrated in operations research and supply chain management, addressing questions of network design, inventory policy, and acquisition pricing. The engineering architecture required to operationalize a reverse logistics process — the workflow orchestration, state management, and system integration layer — is not its focus.

B. Returns in Omnichannel and E-Commerce

Hua, Hou, and Bian [3] examine optimal shipping strategy and return service charge policy under no-reason return policies in online retailing, reflecting the growing research attention on returns as e-commerce has scaled. Madhu [11] reviews order management system capabilities broadly, including return and exchange handling among other fulfillment functions, but does not address the workflow engineering required to operationalize reverse logistics as a first-class process alongside forward fulfillment. Omnichannel commerce compounds this complexity: a return or exchange may be initiated in a physical store, by mail, or through a hybrid flow where authorization occurs digitally

but physical handoff occurs in-store. Each entry point carries different timing characteristics — an in-store return allows immediate condition assessment at the point of return, while a mailed return leaves the device condition unknown until it physically arrives at a processing facility. Telecom mobile commerce introduces additional complexity not present in standard retail returns: trade-in devices may originate from any manufacturer or carrier, requiring the platform to assess and route devices that were never part of its own forward commerce ecosystem. Existing literature addresses the commercial and policy dimensions of this variation but does not address the engineering architecture required to manage these divergent return types and channels through a single operational system.

C. BPM and Workflow Orchestration in Commerce Platforms

Business Process Management has been established as an effective architecture for forward order orchestration in enterprise omnichannel commerce, externalizing workflow logic from application code into configurable, traceable process definitions. Engines such as Flowable [4] implement this through the Business Process Model and Notation (BPMN) 2.0 standard [5], with Van der Aalst et al. [7] providing the foundational workflow management theory underlying modern BPM engines. This hybrid pattern — a BPM orchestration layer coordinating with downstream systems through an event-driven integration layer, consistent with orchestration-choreography designs shown to balance coupling, visibility, and scalability trade-offs [10] — has been applied to forward fulfillment across ship-to-home, in-store, store pickup, on-demand delivery, and pre-order channels. Apache Kafka [6] is established as the event-streaming backbone for this style of asynchronous, fault-tolerant integration at enterprise scale. The TM Forum's Business Process Framework (eTOM) [13] covers fulfillment, assurance, and billing as its core telecom process domains — under which device returns fall as part of the fulfillment lifecycle — but, consistent with its function as a business architecture framework, does not provide technical implementation guidance for reverse logistics orchestration. While BPM-driven orchestration is well established for forward order management, its extension to reverse logistics — with its distinct state model, condition-assessment uncertainty, and disposition routing requirements — has not been documented.

D. Research Gap

The intersection of BPM-driven workflow orchestration, reverse logistics engineering, and telecom mobile device commerce is not addressed in existing literature. This paper addresses that gap with a reference architecture that extends a production BPM-driven order management platform to manage the full reverse logistics lifecycle — returns, trade-ins, and exchanges — through the same orchestration and integration patterns used for forward fulfillment.

3. PROBLEM STATEMENT

A. Return Channel Complexity

A telecom mobile commerce platform must support multiple return types — service-cancellation returns, device exchange returns, and device upgrade returns — alongside trade-in transactions, each initiated through multiple channels with fundamentally different operational characteristics. In-store returns allow immediate physical inspection: a store associate can assess device condition, confirm completeness, and resolve discrepancies with the customer present. Ship-from-home returns carry inherent uncertainty — the customer self-reports device condition at the point of return initiation, but the actual condition is not known until the device physically arrives at a processing facility, often days later. For service-cancellation, device exchange, and device upgrade returns, the refund amount is determined automatically by the device grade: a full or unopened device receives a full refund, a device in acceptable but used condition receives a partial refund with a condition-based deduction applied, and a device in unacceptable condition receives no refund. A platform that encodes each channel and grade outcome through separate, independently maintained logic creates duplicated state handling, inconsistent refund outcomes, and fragmented condition-assessment timing across what should be a single underlying return process.

B. Trade-In Valuation and Condition Assessment Challenge

Trade-in transactions introduce a structural timing mismatch absent from simple returns: a promotion is offered to the customer at the point of sale conditional on returning a device — from any manufacturer or carrier — in a qualifying condition, before that device has been physically received or graded. The actual grade, determined either by a store agent at the point of in-store submission or by a third-party processing facility for mail-in trade-ins, produces one of three outcomes: a full grade match confirms the full promotion; a lower-than-quoted grade results in an adjusted, reduced promotion offered to the customer; and an unacceptable grade results in the device being returned to the customer with no promotion applied. Without a formal workflow governing this three-path reconciliation, grade outcomes are handled inconsistently and create downstream billing and customer experience issues.

C. Multi-Party Integration Complexity

A reverse logistics ecosystem coordinates with the same categories of external systems as forward fulfillment — but for different purposes. Third-party logistics providers issue return shipping labels and provide pickup or drop-off tracking; carrier APIs deactivate or reassign devices; billing platforms issue credits, refunds, or adjusted invoices; warehouse and disposition systems route physical devices to resale, refurbishment, recycling, or back into forward inventory. Each integration point introduces a potential point of failure, and without structured workflow management, a return can reach an indeterminate state — authorized but never physically received, received but never assessed, or assessed but never credited — with no systematic way to detect or resolve the gap.

D. Forward-Reverse Coupling and Inventory Visibility

Reverse logistics does not operate in isolation from forward fulfillment. A device recovered through a return or trade-in and assessed as suitable for resale may re-enter forward-facing inventory, directly affecting fulfillment availability for other customers. A reverse logistics system engineered as an entirely separate platform from forward fulfillment loses this coupling, requiring manual or batch reconciliation between recovered inventory and forward order availability — introducing delay and reducing the commercial value of recovered devices.

4. PROPOSED ARCHITECTURE

A. Architecture Overview

The proposed architecture extends an existing BPM-driven order management platform rather than introducing a separate system. The platform retains the same three-layer structure used for forward fulfillment: a frontend API and order capture layer that receives return and trade-in initiation requests from all channels; a BPM workflow service built on Flowable that manages the reverse logistics lifecycle through dedicated child BPMN process definitions; and a service layer that encapsulates all external integrations — third-party logistics providers, carrier APIs, billing platforms, and disposition systems — using Apache Kafka for asynchronous integration, as illustrated in Fig. 1.

Reverse logistics processes are implemented as child BPMN process definitions within the same Fulfillment Workflow Service used for forward order orchestration, sharing the parent process's customer and order context rather than duplicating it. This shared-platform design allows a returned or traded-in device to be linked directly to its originating order and, where applicable, to the new order it is being traded toward, without a separate cross-system reconciliation step.

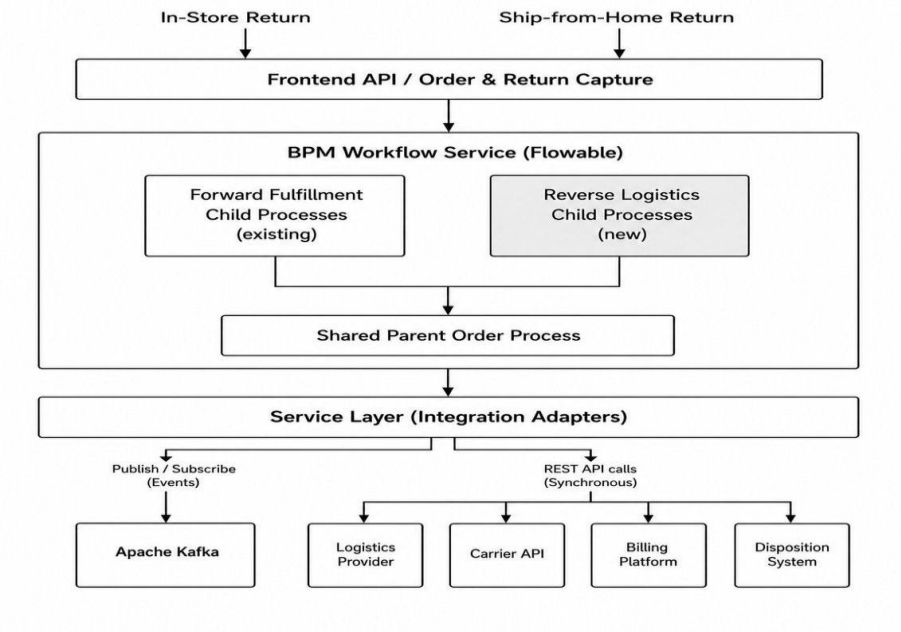

Fig. 1. Reverse Logistics Architecture as an Extension of the Fulfillment Workflow Service

B. Return Lifecycle State Machine

A unified return state machine defines valid states and transitions across both return and trade-in process types. Core states include: RETURN_INITIATED, RETURN_AUTHORIZED, AWAITING_PHYSICAL_RETURN, RECEIVED_AT_FACILITY, CONDITION_ASSESSED, CREDIT_ISSUED, DISPOSITION_ROUTED, and COMPLETED. Grade outcomes from CONDITION_ASSESSED for regular returns are applied automatically: a full or unopened device transitions to CREDIT_ISSUED at the full refund amount; a device in acceptable but used condition transitions to CREDIT_ISSUED with an automatic condition-based deduction applied; a device in unacceptable condition transitions to NO_REFUND, a terminal state. Exception states include UNDER_REVIEW and CANCELLED, as shown in Fig. 2. CONDITION_MISMATCH and ADJUSTED_OFFER are specific to the trade-in sub-process and are described in Section IV.D. Channel-specific intermediate states are scoped to their originating sub-process — for example, IN_STORE_DROPOFF_CONFIRMED applies only to the in-store return sub-process, while SHIPMENT_LABEL_ISSUED and IN_TRANSIT_TO_FACILITY apply only to the ship-from-home sub-process.

The state machine enforces ordering constraints consistent with the forward fulfillment design: condition assessment cannot occur before the device reaches RECEIVED_AT_FACILITY or, for in-store returns, before in-store inspection is confirmed; credit issuance cannot occur before condition assessment completes. These constraints prevent premature credit issuance and provide the same correctness guarantees established for forward order processing.

C. Channel-Specific Return Entry Points

Channel-specific child process definitions handle return entry while converging on the shared post-receipt workflow. The in-store return sub-process captures device condition at the point of return through a store-facing interface, allowing immediate authorization and, in straightforward cases, immediate credit issuance. The ship-from-home sub-process captures a customer self-reported condition at initiation, issues a return shipping label through the third-party logistics integration, and transitions the return to AWAITING_PHYSICAL_RETURN — a state with no equivalent in the in-store flow, reflecting the multi-day window during which the device's actual condition remains unconfirmed.

A third entry path — the accepted-store-option flow — handles cases where a customer originally selected the ship-to-warehouse return method but subsequently walks into a store and requests to complete the return in person. In this scenario, the store agent accepts the device and processes the return in-store, effectively converting the open ship-from-home return into an in-store return without requiring a new return authorization. The BPMN process handles this through a boundary event on the AWAITING_PHYSICAL_RETURN state that detects the in-store acceptance signal and redirects the workflow to the in-store completion path. All three entry paths converge at the same post-receipt workflow — condition assessment, grade-based refund determination, and disposition routing — which is shared logic common to all channels. This convergence avoids duplicating downstream logic while preserving each channel's distinct entry characteristics.

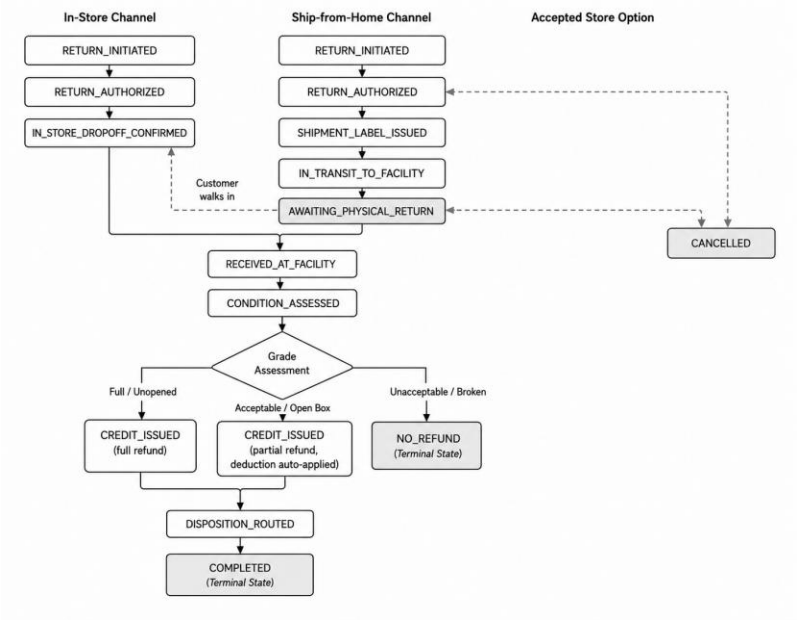

Fig. 2. Unified Return Lifecycle State Machine — Service-Cancellation, Device Exchange, and Device Upgrade Returns (Grade Outcomes Applied Automatically)

D. Trade-In Valuation and Condition Assessment Workflow

The trade-in sub-process introduces an additional state, QUOTED_VALUE_HELD, entered when a customer accepts a trade-in promotion at the point of sale. The promotion is held as a conditional credit pending physical receipt and grading of the traded device. For in-store trade-ins, grading is performed by the store agent at the time of device submission. For mail-in trade-ins, the device is shipped to a third-party processing facility that performs the grade and returns the result to the platform via a callback event. Upon receiving the grade, the workflow evaluates it against the promotion tier through a configurable rule at a BPMN decision gateway, producing one of three outcomes: a full grade match confirms the full promotion and transitions to CREDIT_ISSUED; a partial or lower grade triggers an adjusted promotion offer to the customer, and if accepted, transitions to CREDIT_ISSUED with the reduced amount; an unacceptable grade — such as a broken or non-functional device — transitions to RETURNED_TO_CUSTOMER, the device is shipped back, and no promotion is applied.

E. Disposition Routing Engine

For regular device returns — service-cancellation, device exchange, and device upgrade — following condition assessment a configurable disposition rule, expressed as a structured DMN (Decision Model and Notation) decision table evaluated at a BPMN gateway and consistent with the rule engine pattern used for forward fulfillment routing, determines the outcome for the physical device: return to forward-facing inventory for unopened or pristine devices, route to refurbishment for repairable open-box devices, route to a resale channel for graded used devices, or remove

from inventory for devices in unacceptable condition. Trade-in devices, which may originate from any manufacturer or carrier, follow a separate disposition path managed by the third-party processing facility after the grade outcome is communicated to the platform. Fig. 3a illustrates the trade-in valuation and grade-outcome workflow. Fig. 3b illustrates the grade-based refund and disposition routing for regular device returns. Because disposition criteria change more frequently than the orchestration logic itself — reflecting shifts in refurbishment capacity, resale market conditions, or device-specific recycling requirements — externalizing this decision into a configurable rule allows disposition policy to be updated independently of the workflow definition.

When a device is routed back to forward-facing inventory, the service layer publishes a domain event to the integration layer used by forward fulfillment, making the recovered unit available for allocation without requiring a separate reconciliation process between reverse and forward systems.

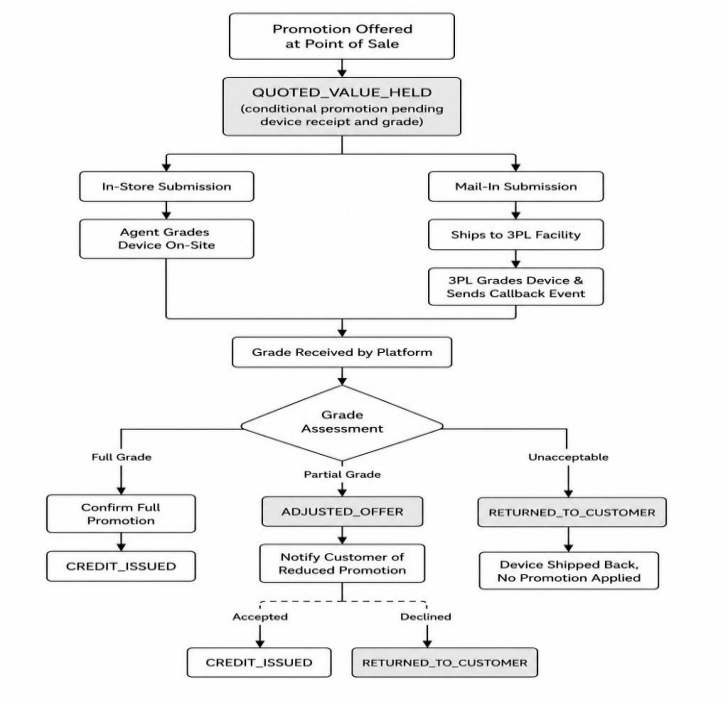

Fig. 3a. Trade-In Valuation Workflow — Grade-Dependent Promotion Outcome

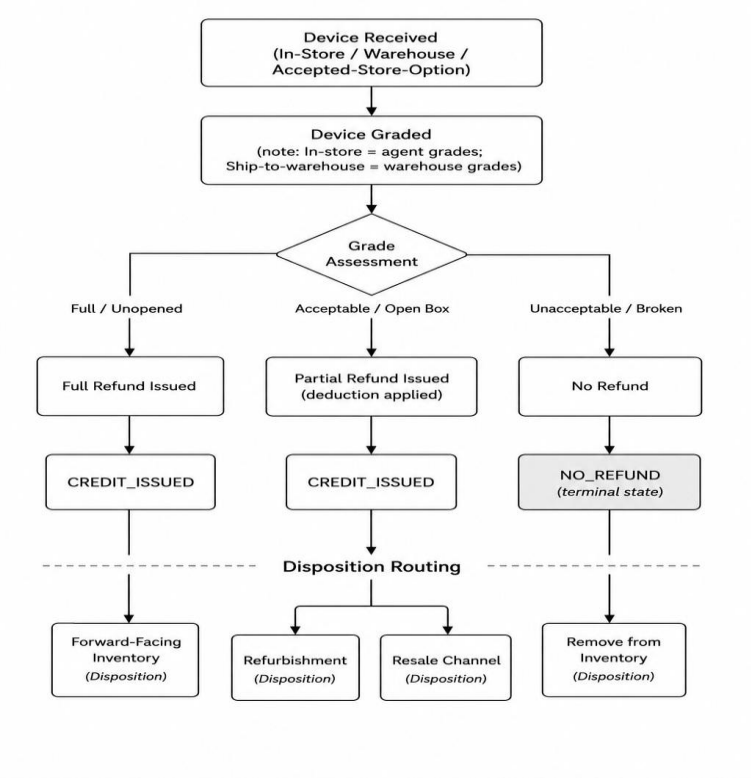

Fig. 3b. Regular Return Refund and Disposition Routing — Service-Cancellation, Device Exchange, and Device Upgrade Returns

F. Billing Credit and Refund Integration

Credit and refund issuance follows the same event-driven integration pattern established for forward fulfillment: the service layer publishes a billing action request to a designated Kafka topic, a dedicated integration adapter translates the request into the appropriate billing platform API call, and the response is published to a corresponding callback topic, where it is correlated with the waiting BPM process instance and used to resume the workflow. This consistency means operational teams troubleshoot reverse logistics billing integration using the same patterns and tooling used for forward fulfillment billing.

G. Exception Handling

Exceptions are classified into three categories, consistent with the taxonomy established for forward fulfillment: integration exceptions (logistics or billing system failures and timeouts), condition exceptions (grade outcome diverges from expected condition — for trade-ins this triggers the three-path resolution workflow described in Section IV.D; for regular returns this triggers the grade-based refund logic described in Section IV.E), and operational exceptions (device never received within a configured window, suspected fraud or abuse patterns). Integration exceptions are handled through Flowable's error boundary events with configurable retry sub-processes. Operational exceptions — such as a return authorized but never physically returned within a configured window — are detected through timer boundary events that escalate the case to UNDER_REVIEW for manual investigation, rather than allowing the return to remain indefinitely open with no operational visibility.

5. IMPLEMENTATION

A. Technology Stack

The reverse logistics ecosystem reuses the existing production technology stack, summarized in Table I. The reverse logistics workflow definitions run on the same Flowable cluster used for forward order orchestration, with Java and Spring Boot providing the microservices foundation for the service layer, consistent with established microservices decomposition and independent-deployability principles [8][9] and with event-driven communication patterns identified as a key enabler of service decoupling [12]. Apache Kafka serves as the event streaming backbone for all external integrations, and the platform is deployed on the same cloud-native infrastructure — Cloud Foundry and AWS — used by the broader order management platform.

Table I. Reverse Logistics Technology Stack

Component	Technology
BPM Engine	Flowable (BPMN 2.0, DMN)
Microservices	Java, Spring Boot, Spring Cloud
Event Streaming	Apache Kafka
Cloud Platform	Cloud Foundry, AWS
Observability	Splunk, Kibana, AppDynamics
CI/CD	Jenkins, GitOps pipelines

Prior to this architecture, return and trade-in processing relied on a combination of store-level manual processes and standalone return-authorization tooling with no shared state model connecting in-store and ship-from-home channels, and no systematic linkage between a return's condition assessment outcome and its eventual billing or disposition treatment. The reverse logistics workflows described in this paper were introduced as child process definitions within the existing Fulfillment Workflow Service, adding a formal return lifecycle without requiring changes to the parent order process or the underlying integration layer.

B. Workflow Design Patterns

The implementation reuses the canonical BPM patterns established for forward fulfillment, applied to reverse logistics scenarios. Event sub-processes handle return-specific exceptions — a timeout event sub-process attached at the top level of the ship-from-home return process interrupts the flow if the device is not received within a configured window, escalating to operational review. Message correlation connects the BPM process to the Kafka integration layer for both logistics callbacks (shipment tracking, facility receipt confirmation) and billing callbacks (credit and refund confirmation), using the same order or return identifier as the correlation key established in the forward fulfillment architecture. Timer boundary events at the QUOTED_VALUE_HELD state enforce a maximum hold duration, after which an unreceived trade-in device automatically triggers customer notification and conditional credit review.

C. Production Deployment

The reverse logistics capability described in this paper was rolled out and validated in production within the mobile commerce platform of a major U.S. telecommunications carrier, building on top of the same Fulfillment Workflow Service used for ship-to-home, in-store, store pickup, on-demand delivery, and pre-order fulfillment channels described in prior work. Reverse logistics process definitions were introduced incrementally — the ship-from-home return process first, followed by in-store returns, and subsequently trade-in valuation and disposition routing — with each addition deployed as a new child process definition without modifying the parent order process, the service layer, or the Kafka integration topics already in production use for forward fulfillment.

The primary implementation consideration was ensuring that the shared Fulfillment Workflow Service could accommodate reverse logistics process volume alongside its existing forward fulfillment load without requiring separate infrastructure, validating the architectural premise that reverse and forward logistics can be engineered as extensions of a single orchestration platform rather than as independent systems.

6. EVALUATION AND RESULTS

A. Return State Visibility

Prior to this architecture, return and trade-in status existed only as fragmented records across store-level systems and standalone authorization tooling, with no unified state model spanning channels. The unified return lifecycle state machine described in Section IV.B provides operational teams with a single, queryable source of truth for every service-cancellation return, device exchange return, device upgrade return, and trade-in in progress, regardless of originating channel — a capability that did not previously exist. This visibility allows operations to distinguish, for any given return, exactly which step has completed and which step is pending, rather than inferring status from disconnected systems.

B. Trade-In Condition Mismatch Handling

The structured grade-outcome workflow described in Section IV.D replaced ad hoc handling of trade-in grade results. Previously, the outcome of a trade-in grade — whether to apply the full promotion, offer a reduced promotion, or return the device with no promotion — had no standardized resolution path, leading to inconsistent outcomes depending on which operator, store location, or processing facility handled the transaction. With the structured workflow, every grade result is routed through the same three-path decision logic: full grade confirms the full promotion, a lower grade triggers a consistent adjusted-offer process, and an unacceptable grade initiates the same device-return and no-promotion flow regardless of channel.

C. Grade-Based Refund Processing

For service-cancellation, device exchange, and device upgrade returns, the refund amount is determined by the device grade assessed at the point of return. A device returned in full, unopened condition receives a full refund. A device returned in used but acceptable condition receives a partial refund with a condition-based deduction applied through the configurable DMN rule engine. A device returned in unacceptable condition — such as a broken or non-functional device — receives no refund. This grade-to-refund mapping, previously applied inconsistently across channels, is now governed by a single configurable decision table that produces consistent refund outcomes regardless of whether the device was returned in-store, shipped to a warehouse, or accepted at a store under the accepted-store-option flow.

D. Operational Fallout Reduction

Applying the same retry and timer boundary event patterns established for forward fulfillment to reverse logistics integrations reduced the category of fallout caused by transient third-party logistics or billing system unavailability. A return shipment tracking callback that does not arrive within a configured window now triggers an automatic status query to the logistics provider rather than leaving the return in an indeterminate state pending manual follow-up. Only returns that fail this automated recovery path escalate to manual review, consistent with the fallout reduction pattern observed in the forward fulfillment platform.

E. Architecture Comparison

Table II compares the BPM-driven reverse logistics architecture against the prior fragmented approach across key operational and engineering dimensions.

Table Ii. Bpm-Driven Vs. Prior Reverse Logistics Approach

Dimension	Prior Approach	BPM-Driven
Return state visibility	Fragmented	Unified, queryable
Channel handling	Separate systems	Shared state machine
Condition mismatch	Ad hoc	3-path grade outcome
Disposition routing	Manual decision	Configurable rule
Forward-reverse coupling	Manual reconciliation	Event-driven, automatic
New disposition outcome	Code change	Rule configuration

7. CONCLUSION

This paper presented a BPM-driven architecture for an enterprise reverse logistics ecosystem in telecom mobile device commerce, extending an existing BPM-driven order management platform rather than building reverse logistics as an independent system. The architecture addresses the structural inadequacy of treating returns and trade-ins as channel-specific, loosely tracked processes when faced with the complexity of multi-channel return entry, condition-assessment uncertainty, and disposition routing requirements.

The proposed architecture, combining a unified return lifecycle state machine, a trade-in valuation and condition assessment workflow, and a configurable disposition routing engine, demonstrates three key advantages. First, channel unification: in-store and ship-from-home returns converge on shared workflow logic despite differing entry characteristics. Second, reconciliation integrity: trade-in value mismatches are resolved through a consistent, auditable workflow rather than ad hoc adjustment. Third, disposition agility: routing rules for recovered devices are externalized from the orchestration layer, allowing disposition policy to evolve independently of workflow code.

Rolling this capability out within a major U.S. telecommunications carrier's mobile commerce platform, as an addition to an already-running BPM-driven order management system, confirmed that reverse logistics does not need to be built as a separate platform — it can be engineered as a natural extension of the same forward fulfillment orchestration.

Future work will explore three directions: machine learning-assisted condition assessment from customer-submitted device imagery at the point of return initiation; fraud and abuse pattern detection integrated at BPM decision gateways within the return authorization sub-process; and closed-loop optimization that jointly considers forward inventory allocation and reverse logistics disposition routing to maximize recovered value from returned and traded-in devices.

REFERENCES

1. D. S. Rogers and R. S. Tibben-Lembke, "An examination of reverse logistics practices," J. Bus. Logist., vol. 22, no. 2, pp. 129–148, 2001.
2. V. D. R. Guide Jr. and L. N. Van Wassenhove, "OR Forum—The evolution of closed-loop supply chain research," Oper. Res., vol. 57, no. 1, pp. 10–18, Jan./Feb. 2009.
3. Z. Hua, H. Hou, and Y. Bian, "Optimal shipping strategy and return service charge under no-reason return policy in online retailing," IEEE Trans. Syst., Man, Cybern., Syst., vol. 47, no. 12, pp. 3189–3206, Dec. 2017.
4. Flowable Project, "Flowable Open Source Documentation," 2023. [Online]. Available: https://www.flowable.com/open-source/docs/
5. Object Management Group (OMG), "Business Process Model and Notation (BPMN) Version 2.0," OMG Standard, Jan. 2011. [Online]. Available: https://www.omg.org/spec/BPMN/2.0
6. J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in Proc. NetDB Workshop, Athens, Greece, Jun. 2011, vol. 11, pp. 1–7.
7. W. M. P. van der Aalst, A. ter Hofstede, and M. Weske, "Business process management: A survey," in Proc. Int. Conf. Business Process Management (BPM 2003), LNCS, vol. 2678, Springer, 2003, pp. 1–12.
8. S. Newman, Building Microservices: Designing Fine-Grained Systems, 2nd ed. Sebastopol, CA: O'Reilly Media, 2021.
9. J. Thönes, "Microservices," IEEE Software, vol. 32, no. 1, p. 116, Jan./Feb. 2015.
10. A. Megargel, C. M. Poskitt, and V. Shankararaman, "Microservices orchestration vs. choreography: A decision framework," in Proc. IEEE 25th Int. Enterprise Distributed Object Computing Conf. (EDOC), Oct. 2021, pp. 134–141.

11. A. Madhu, "Understanding order management systems in retail and e-commerce," J. Inf. Syst. Eng. Manage., vol. 10, no. 59s, 2025. [Online]. Available: https://jisem-journal.com/index.php/journal/article/view/12997
12. A. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in Proc. IEEE 9th Int. Conf. Service-Oriented Computing and Applications (SOCA), 2016, pp. 44–51.
13. TM Forum, "Business Process Framework (eTOM), GB921," TM Forum Standard, Release 21.5, 2021. [Online]. Available: https://www.tmforum.org/resources/suite/gb921-business-process-framework-etom-suite/