**1. Introduction**

**1.1 Overview**

This Low-Level Design document outlines the implementation details for the Kafka-based Service Area (SA) change detection system. The system validates Transfer Orders (TOs) against Service Area changes and takes appropriate actions based on validation results.

**1.2 Scope**

The system handles:

* Real-time processing of Service Area snapshot events from Kafka
* Validation of Transfer Orders against Service Area changes
* Dynamic response to SA changes with appropriate TO actions (cancellation/updates)
* Integration with TOM (Transfer Order Management) database
* Event publishing for downstream systems

**1.3 Key Requirements**

* **Real-time Processing**: Process SA changes as they occur
* **Data Integrity**: Ensure TO validation accuracy
* **Scalability**: Handle high-volume message processing
* **Reliability**: Robust error handling and recovery mechanisms
* **Monitoring**: Comprehensive logging and metrics collection

**1.5 Historical Data Processing Challenge**

**1.5.1 Problem Statement**

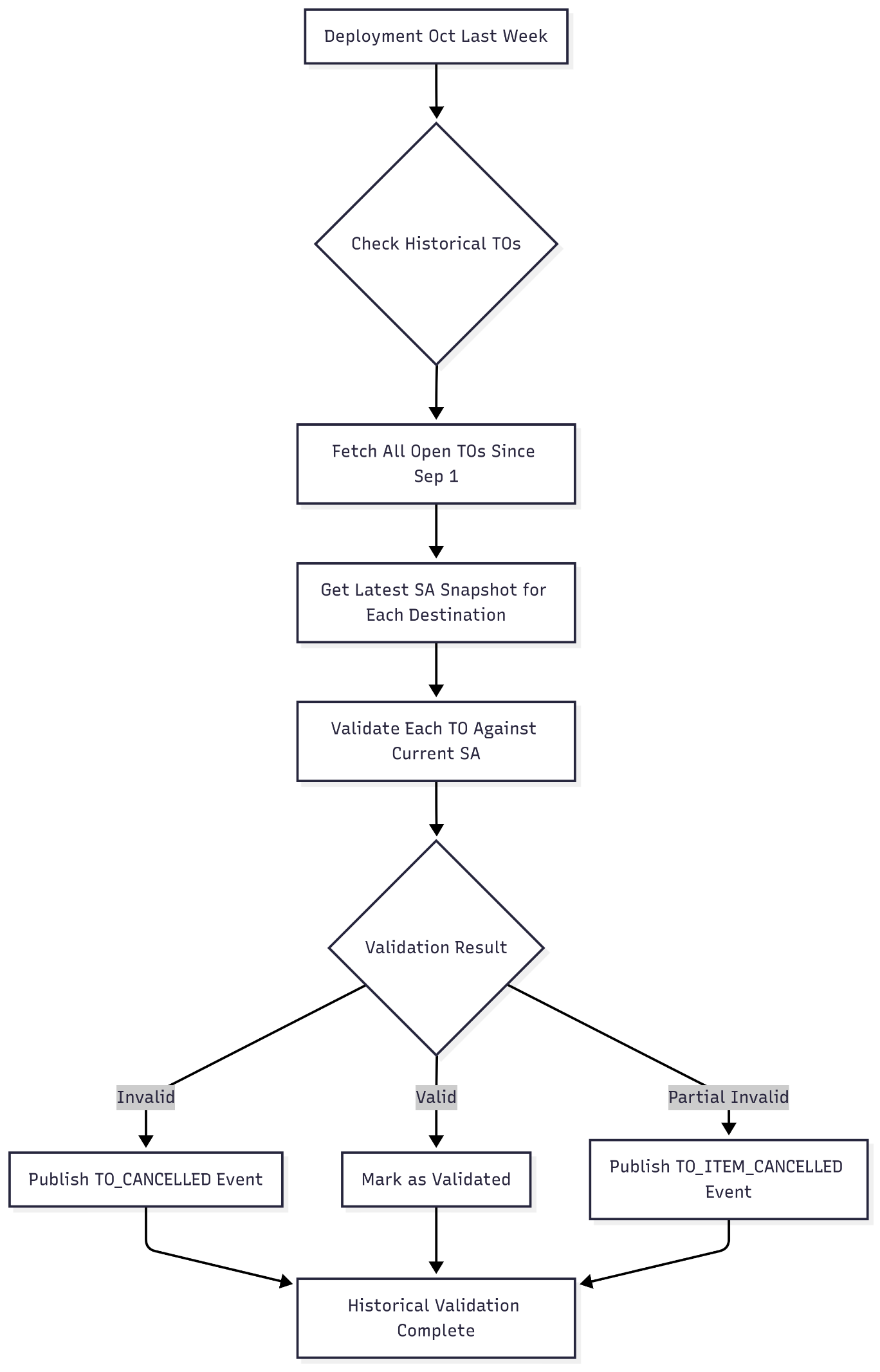
**Deployment Timeline**: Last week of October 2025  
**Historical Data Concern**: Transfer Orders created from September 1st week onwards

**The Challenge**:

* TOs created between Sep 1st week and Oct last week (deployment) won't have been validated against SA changes
* Service Area changes that occurred during this period may have invalidated existing TOs
* Need to ensure data integrity for all open TOs at the time of system deployment

**1.5.2 Historical Data Processing Strategy**

**Approach 1: One-Time Historical Validation (Recommended)**



**Implementation Details**:

java

@Component

public class HistoricalDataProcessor {

@Autowired

private TomDbQueryService tomDbQueryService;

@Autowired

private ServiceAreaValidator validator;

@Autowired

private EventPublisher eventPublisher;

*/\*\**

\* One-time historical validation during deployment

*\*/*

@PostConstruct

@ConditionalOnProperty(name = "historical.validation.enabled", havingValue = "true")

public void processHistoricalTOs() {

log.info("Starting historical TO validation for period: Sep 1 - Oct deployment");

LocalDate startDate = LocalDate.of(2025, 9, 1);

LocalDate endDate = LocalDate.now();

List<TransferOrder> historicalTOs = tomDbQueryService

.fetchTOsCreatedBetween(startDate, endDate);

log.info("Found {} historical TOs to validate", historicalTOs.size());

processHistoricalTOsInBatches(historicalTOs);

}

private void processHistoricalTOsInBatches(List<TransferOrder> historicalTOs) {

int batchSize = 100;

for (int i = 0; i < historicalTOs.size(); i += batchSize) {

List<TransferOrder> batch = historicalTOs.subList(

i, Math.min(i + batchSize, historicalTOs.size())

);

processBatch(batch);

try {

Thread.sleep(1000); *// 1 second delay between batches*

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

log.error("Historical processing interrupted", e);

break;

}

}

}

private void processBatch(List<TransferOrder> batch) {

for (TransferOrder to : batch) {

try {

*// Get current SA snapshot for TO's destination*

ServiceAreaSnapshot currentSnapshot =

serviceAreaService.getCurrentSnapshot(to.getDestinationLocation());

*// Validate TO against current SA state*

ValidationResult result = validator.validateTransferOrder(currentSnapshot, to);

*// Take appropriate action based on validation*

handleHistoricalValidationResult(to, result);

} catch (Exception e) {

log.error("Failed to process historical TO: {}", to.getId(), e);

*// Continue with other TOs*

}

}

}

private void handleHistoricalValidationResult(TransferOrder to, ValidationResult result) {

switch (result.getAction()) {

case CANCEL\_TO:

eventPublisher.publishCancelledEvent(to, "HISTORICAL\_SA\_VALIDATION: " + result.getReason());

log.info("Historical TO cancelled: {} - {}", to.getId(), result.getReason());

break;

case CANCEL\_ITEMS:

result.getInvalidItems().forEach(item ->

eventPublisher.publishItemCancelledEvent(item, "HISTORICAL\_SA\_VALIDATION: " + result.getReason())

);

log.info("Historical TO items cancelled: {} - {} items", to.getId(), result.getInvalidItems().size());

break;

case NO\_ACTION:

*// Mark as validated in database for audit purposes*

tomDbQueryService.markAsHistoricallyValidated(to.getId());

log.debug("Historical TO validated successfully: {}", to.getId());

break;

}

}

}

**1.5.3 Database Query for Historical TOs**

sql

*-- Query to fetch historical TOs that need validation*

SELECT

t.id,

t.origin\_location,

t.destination\_location,

t.status,

t.created\_date,

COUNT(tli.id) as line\_item\_count

FROM transfer\_orders t

LEFT JOIN to\_line\_items tli ON t.id = tli.to\_id

WHERE t.created\_date >= '2025-09-01 00:00:00'

AND t.created\_date < NOW()

AND t.status IN ('OPEN', 'IN\_PROGRESS', 'ALLOCATED')

AND t.historical\_validation\_status IS NULL *-- Not yet validated*

GROUP BY t.id

ORDER BY t.created\_date ASC;