#### **Title Page**

* **Title**: Low-Level Design (LLD) for Kafka Pain Areas Project
* **Author**: Sarthak Kanungo
* **Date**: 09-Sep-2025 Tue

#### **Table of Contents**

* Automatically generated based on headings.

#### **1. Introduction**

* **1.1 Overview**
* **1.2 Scope**

#### **2. Detailed Design**

* **2.1 Kafka Listener**
  + **2.1.1 Class Diagram**
  + **2.1.2 Sequence Diagram**
  + **2.1.3 Pseudocode**
* **2.2 TOM DB Query Service**
  + **2.2.1 Class Diagram**
  + **2.2.2 Sequence Diagram**
  + **2.2.3 Pseudocode**
* **2.3 Service Area Comparator**
  + **2.3.1 Class Diagram**
  + **2.3.2 Sequence Diagram**
  + **2.3.3 Pseudocode**
* **2.4 Event Publisher**
  + **2.4.1 Class Diagram**
  + **2.4.2 Sequence Diagram**
  + **2.4.3 Pseudocode**

#### **3. Error Handling**

* **3.1 Exception Handling Strategies**

#### **4. Integration Details**

* **4.1 System Integration**

#### **5. Conclusion**

* **5.1 Summary**
* **5.2 Next Steps**

**#### 1. Introduction**

**The Low-Level Design (LLD) document provides a detailed design for the Kafka-based system that handles reverse movement scenarios and cut-over historical data issues. This design breaks down the high-level components into detailed elements, including class diagrams, sequence diagrams, pseudocode, and error handling strategies.**

**---**

**#### 2. Detailed Design**

**##### 2.1 Kafka Listener**

**\*\*Class Diagram:\*\***

**- \*\*Attributes:\*\***

**- `kafkaTemplate`: Template for sending messages to Kafka.**

**- `tomDbQueryService`: Service for querying the TOM DB.**

**- `serviceAreaComparator`: Comparator for service area responses.**

**- `eventPublisher`: Publisher for different types of events.**

**- \*\*Methods:\*\***

**- `listenToTopic()`: Subscribes to the Kafka topic.**

**- `processMessage(KafkaMessage message)`: Processes the incoming Kafka message.**

**- `validateAndAct(ServiceAreaResponse response)`: Validates the service area response and takes appropriate action.**

**\*\*Sequence Diagram:\*\***

**1. `KafkaListener` receives a message from the `service\_area\_snapshot` topic.**

**2. `KafkaListener` processes the message payload.**

**3. `KafkaListener` calls `TomDbQueryService` to fetch relevant TOs.**

**4. `TomDbQueryService` queries the TOM DB and returns the TOs.**

**5. `KafkaListener` calls `ServiceAreaComparator` to compare the service area response.**

**6. `ServiceAreaComparator` validates the TOs and determines the course of action.**

**7. `ServiceAreaComparator` calls `EventPublisher` to publish the appropriate event.**

**\*\*Pseudocode:\*\***

**```plaintext**

**function listenToTopic():**

**// ... existing code ...**

**while true:**

**message = receiveMessageFromTopic("service\_area\_snapshot")**

**processMessage(message)**

**function processMessage(message):**

**// ... existing code ...**

**serviceAreaResponse = parseMessagePayload(message)**

**tos = tomDbQueryService.fetchOpenTOs(serviceAreaResponse.destinationLocation)**

**validateAndAct(serviceAreaResponse, tos)**

**function validateAndAct(response, tos):**

**// ... existing code ...**

**for to in tos:**

**if not serviceAreaComparator.isOriginValid(response, to):**

**serviceAreaComparator.handleInvalidServiceArea(to)**

**```**

**##### 2.2 TOM DB Query Service**

**\*\*Class Diagram:\*\***

**- \*\*Attributes:\*\***

**- `jdbcTemplate`: Template for executing SQL queries.**

**- \*\*Methods:\*\***

**- `fetchOpenTOs(String destinationLocation)`: Fetches open and relevant TOs based on the destination location.**

**- `getTOById(Long toId)`: Retrieves a specific TO by its ID.**

**\*\*Sequence Diagram:\*\***

**1. `KafkaListener` calls `TomDbQueryService.fetchOpenTOs()`.**

**2. `TomDbQueryService` constructs the query based on the destination location.**

**3. `TomDbQueryService` executes the query using `jdbcTemplate`.**

**4. `TomDbQueryService` returns the list of TOs to `KafkaListener`.**

**\*\*Pseudocode:\*\***

**```plaintext**

**function fetchOpenTOs(destinationLocation):**

**// ... existing code ...**

**query = "SELECT \* FROM transfer\_orders WHERE status = 'OPEN' AND destination\_location = ?"**

**return jdbcTemplate.query(query, [destinationLocation])**

**function getTOById(toId):**

**// ... existing code ...**

**query = "SELECT \* FROM transfer\_orders WHERE id = ?"**

**return jdbcTemplate.queryForObject(query, [toId])**

**```**

**##### 2.3 Service Area Comparator**

**\*\*Class Diagram:\*\***

**- \*\*Attributes:\*\***

**- None**

**- \*\*Methods:\*\***

**- `compareServiceAreaResponse(ServiceAreaResponse response, List<TransferOrder> tos)`: Compares the response with TO data.**

**- `isOriginValid(ServiceAreaResponse response, TransferOrder to)`: Checks if the origin location is valid.**

**- `handleInvalidServiceArea(TransferOrder to)`: Handles cases where the service area is invalid.**

**\*\*Sequence Diagram:\*\***

**1. `KafkaListener` calls `ServiceAreaComparator.compareServiceAreaResponse()`.**

**2. `ServiceAreaComparator` iterates through each TO and checks the origin location.**

**3. `ServiceAreaComparator` calls `isOriginValid()` for each TO.**

**4. `ServiceAreaComparator` decides the course of action based on the validation result.**

**5. `ServiceAreaComparator` calls `handleInvalidServiceArea()` if the origin is invalid.**

**\*\*Pseudocode:\*\***

**```plaintext**

**function compareServiceAreaResponse(response, tos):**

**// ... existing code ...**

**for to in tos:**

**if not isOriginValid(response, to):**

**handleInvalidServiceArea(to)**

**function isOriginValid(response, to):**

**// ... existing code ...**

**for node in response.servicingNodes:**

**if node.status == "ACTIVE" and node.location == to.originLocation:**

**return true**

**return false**

**function handleInvalidServiceArea(to):**

**// ... existing code ...**

**eventPublisher.publishCancelledEvent(to, "INVALID SERVICE AREA")**

**```**

**##### 2.4 Event Publisher**

**\*\*Class Diagram:\*\***

**- \*\*Attributes:\*\***

**- `kafkaTemplate`: Template for sending messages to Kafka.**

**- \*\*Methods:\*\***

**- `publishCancelledEvent(TransferOrder to, String reason)`: Publishes a `TO\_CANCELLED` event.**

**- `publishUpdateEvent(TransferOrder to, String reason)`: Publishes a `TO\_UPDATE` event.**

**\*\*Sequence Diagram:\*\***

**1. `ServiceAreaComparator` calls `EventPublisher.publishCancelledEvent()` or `publishUpdateEvent()`.**

**2. `EventPublisher` constructs the event message.**

**3. `EventPublisher` sends the event message using `kafkaTemplate`.**

**\*\*Pseudocode:\*\***

**```plaintext**

**function publishCancelledEvent(to, reason):**

**// ... existing code ...**

**event = constructEvent("TO\_CANCELLED", to, reason)**

**kafkaTemplate.send("events\_topic", event)**

**function publishUpdateEvent(to, reason):**

**// ... existing code ...**

**event = constructEvent("TO\_UPDATE", to, reason)**

**kafkaTemplate.send("events\_topic", event)**

**```**

**#### 3. Error Handling**

**\*\*Exception Handling Strategies:\*\***

**- \*\*SaveAndPublishException\*\*: Catch exceptions during save and publish operations, log the error, and increment the exception count for monitoring.**

**- \*\*DBQueryException\*\*: Handle exceptions during database queries, log the error, and retry the operation.**

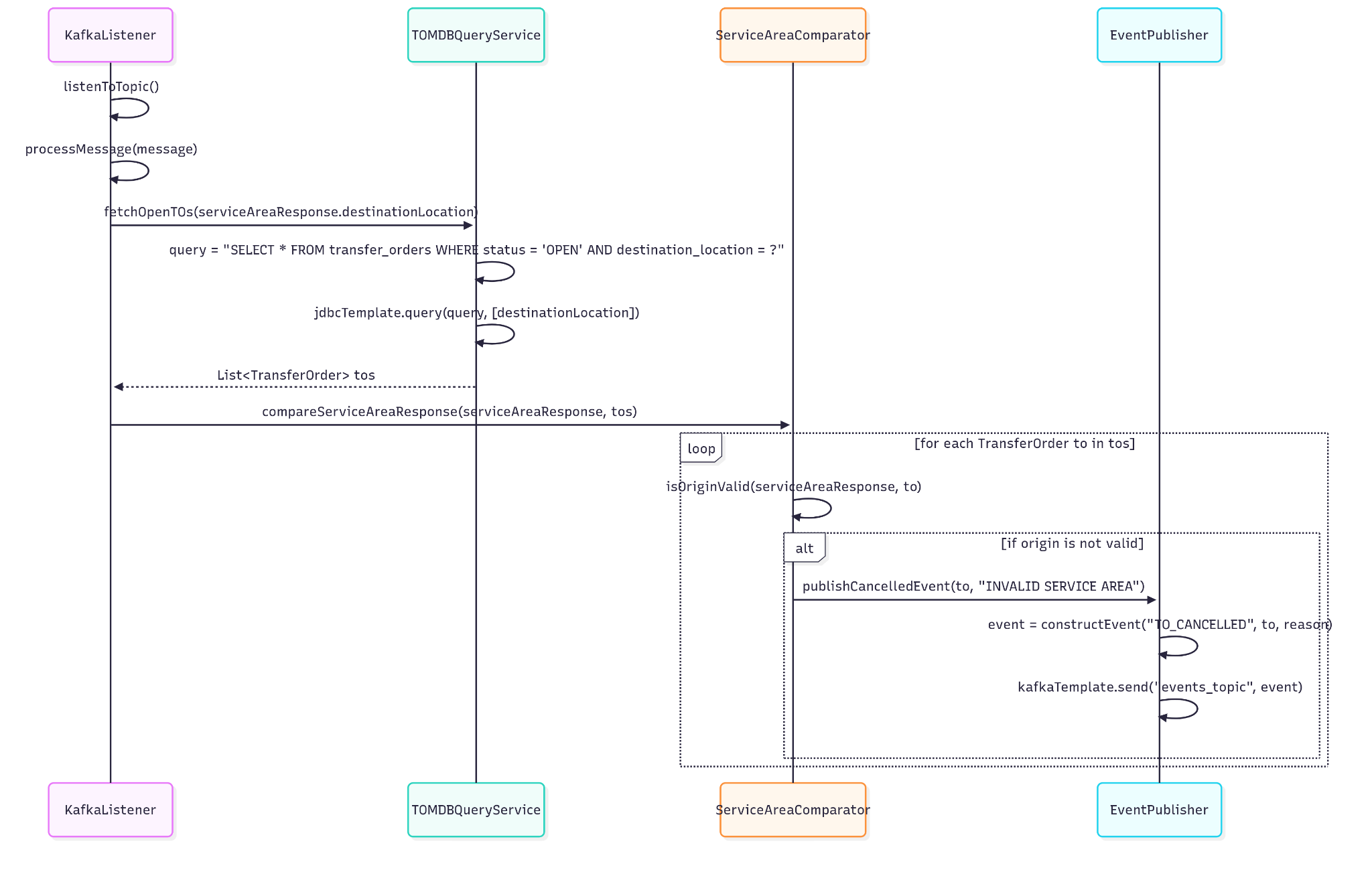
**- \*\*KafkaSendException\*\*: Catch exceptions during Kafka message sending, log the error, and retry sending the message.**

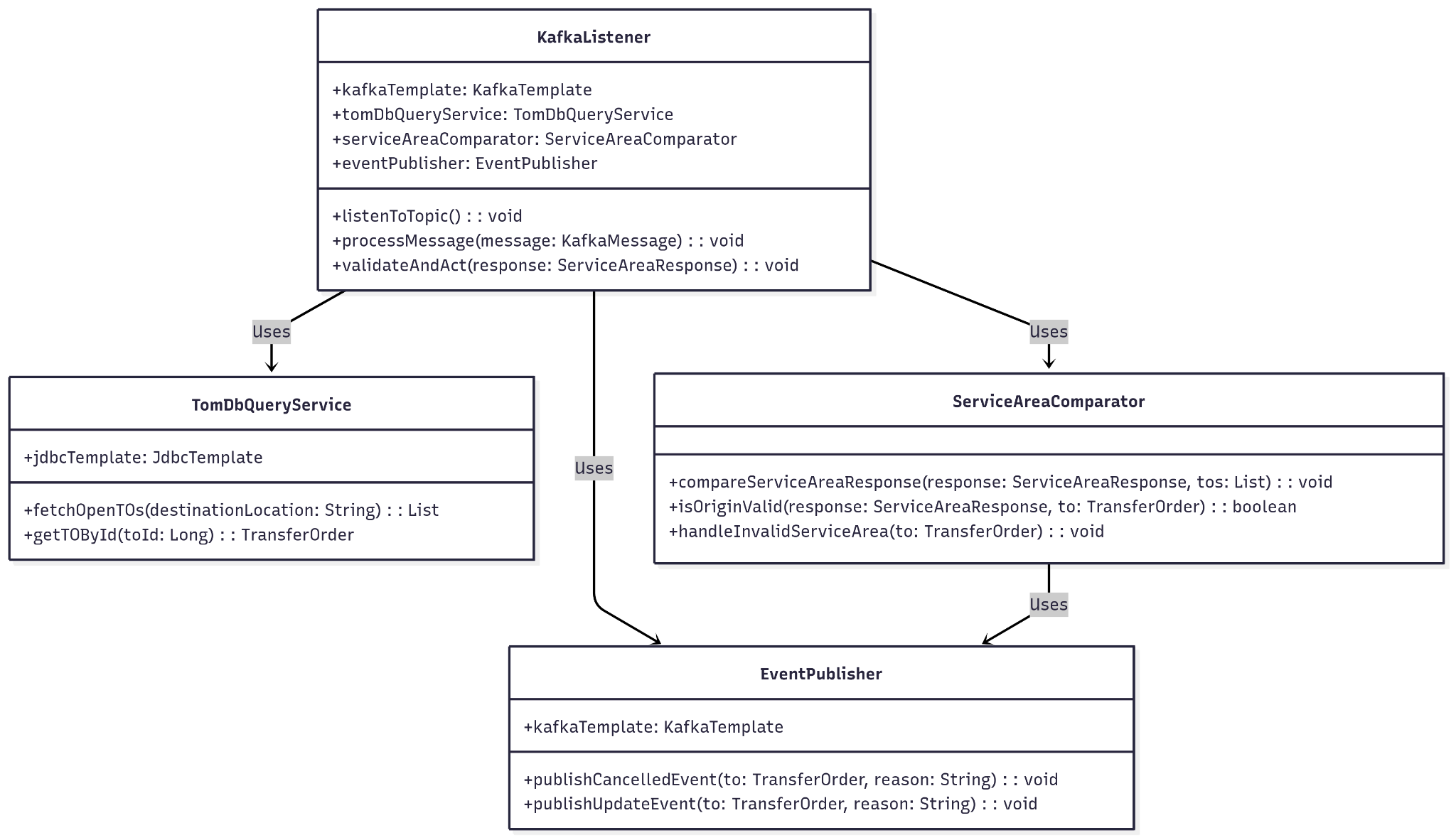
**#### 4. Integration Details**

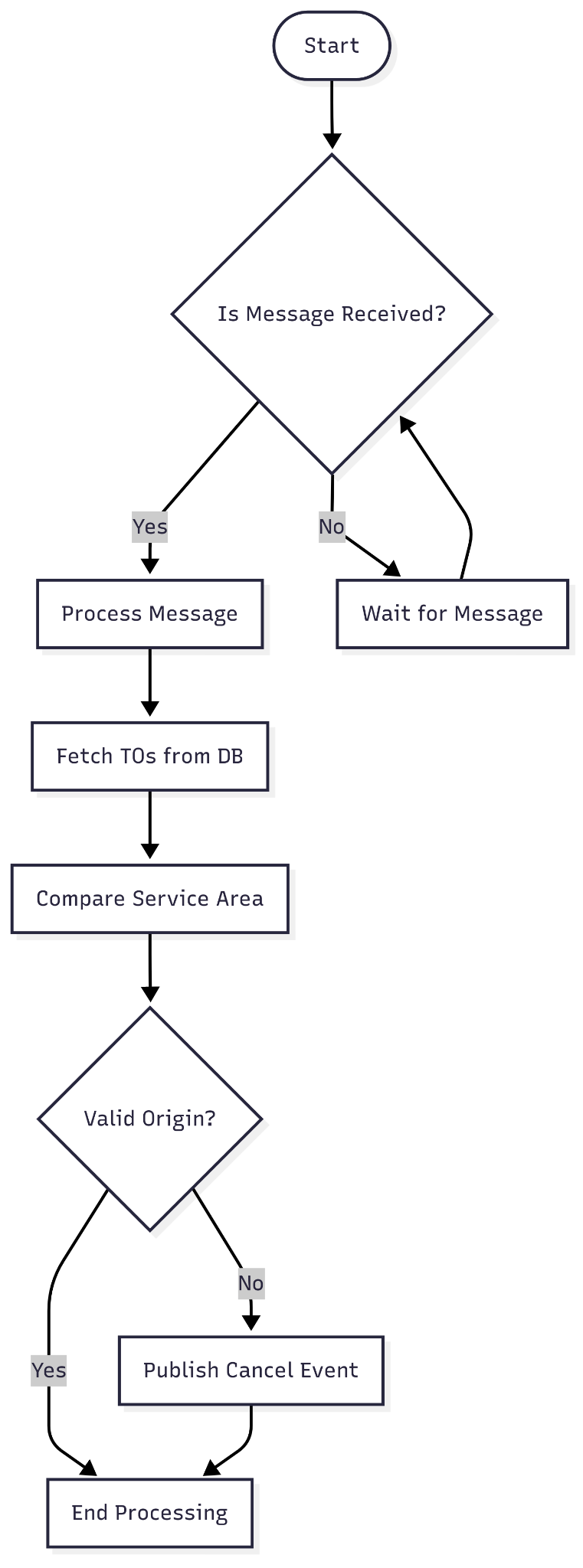
**This module integrates with the existing system by subscribing to the Kafka topic `service\_area\_snapshot`, querying the TOM DB for relevant TOs, comparing service area responses, and publishing events based on validation results.**

**#### 5. Conclusion**

**This LLD document provides a comprehensive design for the Kafka-based system, detailing each component's structure, behavior, and interactions. The design ensures real-time event processing, accurate TO validation, and dynamic reaction to SA changes, meeting the project's requirements and constraints.**

****

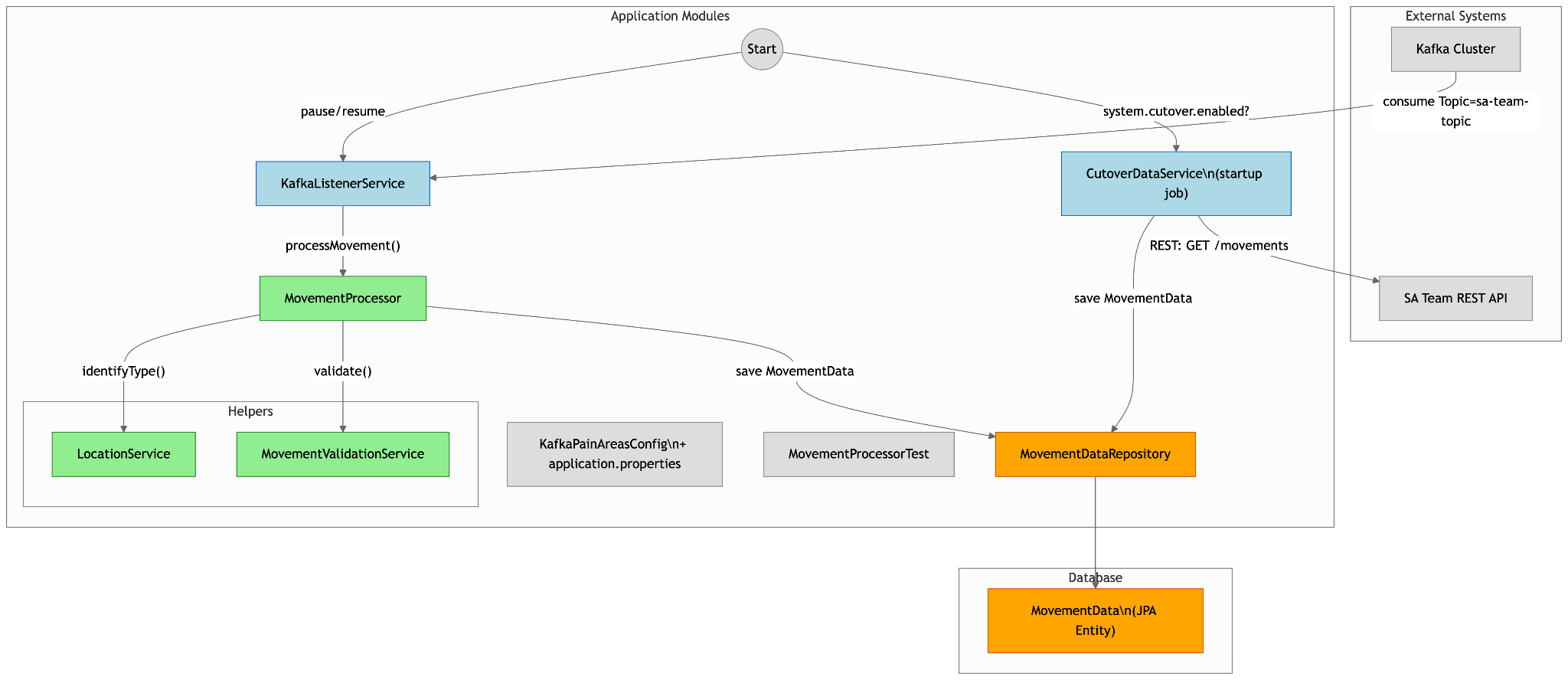
****

****

<https://github.com/sarthak1/kafka-pain-areas.git>

<https://uithub.com/sarthak1/kafka-pain-areas.git>

<https://gitdiagram.com/sarthak1/kafka-pain-areas>



<https://sarthak1.github.io/kafka-pain-areas/>

<https://docs.google.com/document/d/1zgdU1hg2GbGIbfFBMk47dErStJDG7BlTyhhfn9VCR7o/edit?usp=sharing>

