**OPTIMUS MAPS OPEX Task**

**Version:** 1.0  
**Date:** [26th March 2025]

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change Summary** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**



**1. Introduction**

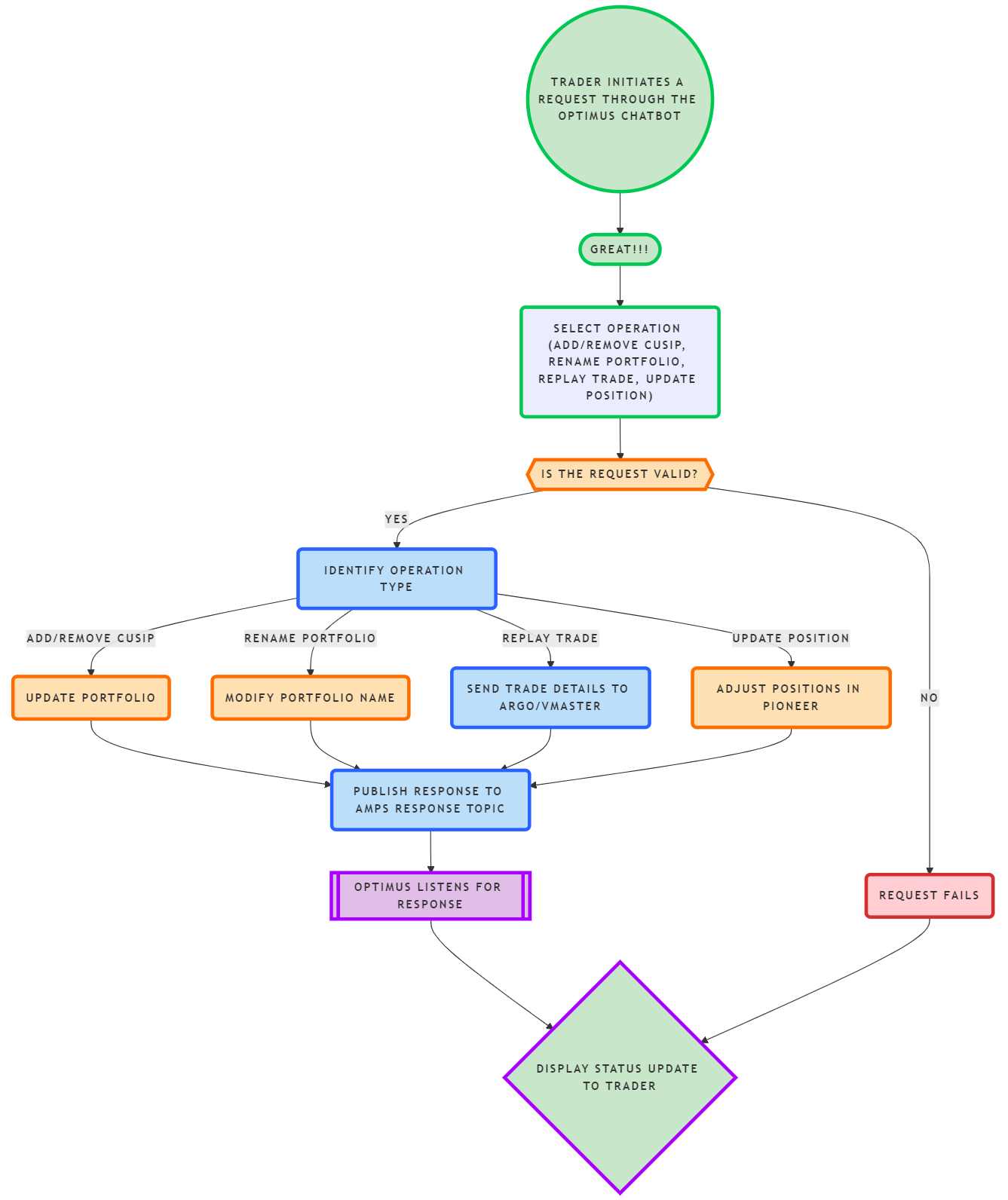
This document provides the **System Design** for the **OPTIMUS MAPS OPEX Task**. The system enables traders to interact with the **OPTIMUS** chatbot, which facilitates the execution of various functionalities related to portfolio management, trade reconciliation, and position updates. The requests made by traders are published to an **AMPS topic**, processed by a **subscriber service**, and responded to via a **response topic**, ensuring real-time interaction.

**2. System Overview**

* Traders initiate requests via the **OPTIMUS chatbot**.
* The chatbot publishes the request to a designated **AMPS topic**.
* A **Subscriber Service** listens to the topic, retrieves the request, processes it accordingly, and executes the required action.
* Upon completion, the **Subscriber Service** sends a response back to another **AMPS topic**.
* **OPTIMUS** listens to the response topic and displays the results to the trader.

This architecture ensures a seamless, low-latency interaction between traders and the system, enabling automated handling of financial operations.

**3. Flow chart Diagram**

**

**4. Component Description**

* **OPTIMUS Chatbot**: A trading assistant that allows traders to perform key financial operations via text-based commands.
* **AMPS (Advanced Message Processing System)**: A message broker that facilitates real-time message publishing and subscribing for high-frequency trading operations.
* **Subscriber Service**: A backend component that listens to the AMPS topic, processes the incoming requests, executes the required actions, and publishes the response.
* **Response Topic**: An AMPS topic where processed responses are sent, which **OPTIMUS** listens to for displaying results to traders.

**5. Workflow**

1. A trader interacts with **OPTIMUS** and selects a functionality.
2. The chatbot publishes the request to the appropriate **AMPS topic**.
3. The **Subscriber Service** listens to the topic, retrieves the request, and determines the appropriate action.
4. The service executes the action (e.g., modifying a portfolio, replaying a trade, updating positions).
5. Once processing is complete, the **Subscriber Service** publishes a response to the **Response Topic**.
6. **OPTIMUS** listens to the response topic and displays the status/result of the operation to the trader.

**6. Functional Requirements**

The system supports the following key functionalities as per the MAPS team’s instructions via **OPTIMUS**:

**6.1 Adding or Removing CUSIP from User Portfolio**

* Traders can add or remove **CUSIP** (a unique identifier for securities) from their portfolios.
* Upon receiving the request, the system will update the relevant portfolio database and confirm the change.
* The response is sent back to **OPTIMUS**, confirming success or failure.

**6.2 Renaming Portfolios**

* Traders can rename their existing portfolios.
* The request is processed by the **Subscriber Service**, which updates the portfolio name in the database.
* A response is sent back to confirm the update.

**6.3 Replay Trade to ARGO/VMaster after STP Failures**

* If a trade fails during **Straight-Through Processing (STP)**, the trader can request a replay.
* The **Subscriber Service** fetches the failed trade and resends it to **ARGO/VMaster**.
* The status of the replay is published back to **OPTIMUS**.

**6.4 Updating Trading Positions in Pioneer**

* If there is a discrepancy between **Bloomberg** and **Pioneer**, traders can request a position update.
* The system validates and updates the trading position in **Pioneer**.
* A response is sent back indicating whether the update was successful.

**7. Error Handling & Logging**

* **Message Acknowledgment**: Ensures messages are processed successfully before being marked as complete.
* **Retry Mechanism**: If processing fails, messages are retried based on a configurable retry policy.
* **Logging**: Every request and response is logged for debugging and auditing.
* **Alerting**: If critical errors occur, alerts are generated for immediate investigation.

**8. Performance & Scalability**

* **Low Latency Processing**: Optimized message handling ensures real-time updates.
* **Horizontal Scaling**: The **Subscriber Service** can scale horizontally to handle high message volumes.
* **Load Balancing**: Distributes requests across multiple service instances to avoid bottlenecks.
* **High Availability**: The system is designed with redundancy to ensure continued operation in case of failures.

**9. Conclusion**

This document outlines the system design for processing trader requests via **OPTIMUS**, leveraging **AMPS** for real-time message passing, and a **Subscriber Service** for request execution. The architecture ensures high availability, security, and scalability for seamless financial operations.