Asynchronous API Design Document

# 1. Overview

This document outlines the detailed design of the Asynchronous API system, which consists of an InventoryService (server) and a SimpleClient (client). The system demonstrates asynchronous processing using Kafka for message queuing and webhooks for notifications.

# 2. System Architecture

## 2.1 Components

* InventoryService (Server): A .NET Core Web API that handles inventory processing requests
* SimpleClient (Client): A .NET Core console application that interacts with the server
* Kafka (Simulated): A message broker for asynchronous job processing
* Webhook System: A notification mechanism for job completion

## 2.2 Communication Flow

1. Client registers a webhook with the server.
2. Client sends a request to process inventory.
3. Server acknowledges the request with a 202 Accepted response.
4. Server publishes the job to Kafka.
5. Server processes the job asynchronously.
6. When the job completes, server notifies the client via the registered webhook.

# 3. Server Design (InventoryService)

## 3.1 Components

### 3.1.1 Controllers

InventoryController

* Handles HTTP requests for inventory processing
* Endpoints:
* POST /api/inventory/processInventory – Initiates asynchronous inventory processing
* POST /api/inventory/registerWebhook – Registers a webhook for notifications
* Responsibilities:
* Create inventory jobs
* Accept error flags for testing
* Publish jobs to Kafka
* Register webhooks

### 3.1.2 Models

InventoryJob

* JobId, CreatedAt, CompletedAt, Status, Result, ErrorCode, ErrorMessage, HasError

WebhookRegistration

* WebhookId, CallbackUrl

ErrorCodes

* ValidationError, NotFoundError, AuthorizationError, DatabaseError, ProcessingError

### 3.1.3 Services

KafkaProducerService

* Publishes inventory jobs to Kafka topics
* Logs publishing events

KafkaTopicService

* Manages Kafka topics and message simulation
* Subscribes, commits messages, simulates functionality in-memory

InventoryProcessingService

* Consumes messages from Kafka
* Processes jobs asynchronously
* Updates job status
* Notifies webhooks on completion

WebhookService

* Registers webhooks
* Stores registrations
* Sends notifications
* Handles failures

## 3.2 Error Handling

* Input Validation: Validates incoming requests
* Error Simulation: Supports flags for simulated error scenarios
* Error Propagation: Communicates errors via Kafka
* Webhook Notification: Includes error details in callbacks
* Logging: Logs errors at appropriate levels

## 3.3 Data Flow

1. POST /api/inventory/processInventory
2. Controller creates InventoryJob
3. Publishes job via KafkaProducerService
4. Returns 202 Accepted with job ID
5. InventoryProcessingService processes job
6. Job status updated
7. Webhooks notified via WebhookService

## Component Diagram

A screenshot of a computer

AI-generated content may be incorrect.

## Sequence Diagram

