Software System Design Document

Contents

1. Purpose

The purpose of this document is to design and implement a messagebased system that leverages multiple concurrent collections and priority execution to manage concurrent operations effectively. The system will consist of a central component and multiple dependent components that communicate in realtime. It aims to demonstrate robust commandresponse handling and continuous data streaming capabilities.

2. Scope

This design document outlines the architecture, components, and operations of a messagebased system with the following features:

- Realtime communication between a central component and dependent components.
- Prioritybased execution and management of concurrent operations.
- Continuous data streaming capabilities.
- Robust commandresponse handling.
- Integration with SignalR for realtime communication.

The system is designed to be modular and extensible, allowing engineers to adapt and extend it as needed for their specific use cases.

3. Audience

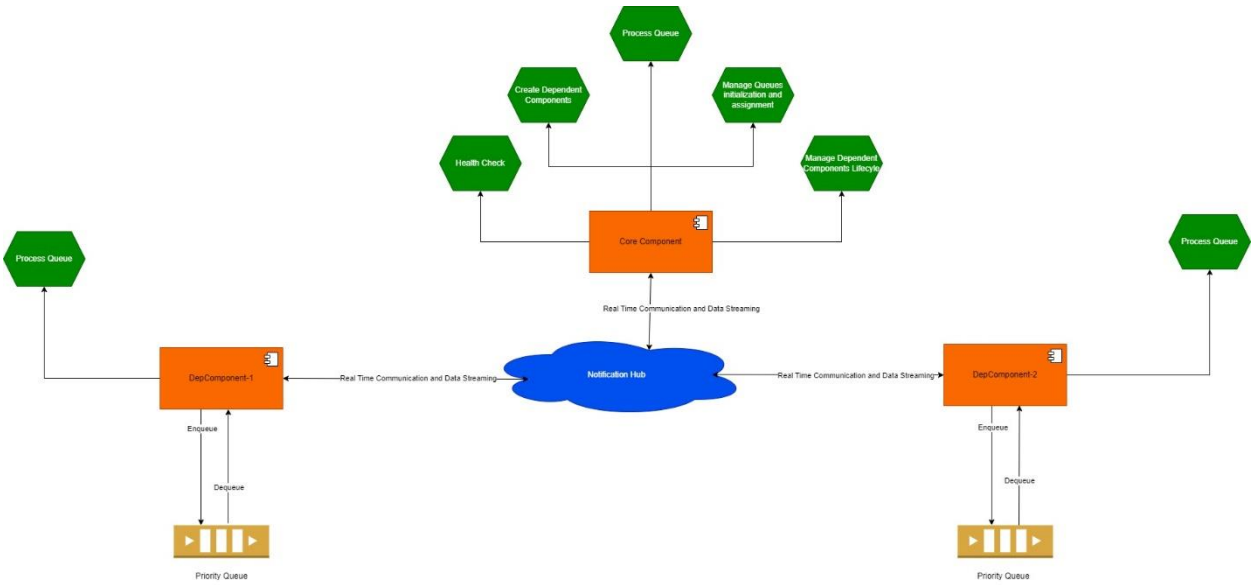
This document is intended for software engineers who will implement, maintain, or further modify the system. It provides a comprehensive overview of the system's design, including component details, architecture, and applications.

4. Assumptions

The .NET 8 runtime is installed on the systems where the software will be deployed.

Engineers using or modifying the system will have a foundational understanding of C# and .NET.

5. Architectural Diagram



The architectural diagram illustrates the components of the system and their interactions:

- SignalR Server: Handles realtime communication and routing of messages.
- Notification Hub: Manages message routing between components.
- Core Component: Manages dependent components and processes messages.
- Dependent Components: Process messages and response.

6. Component Details

6.1 Core Component

Responsibilities:

- Creates and manages dependent components.
- Assigns and manages queues for each component to maintain their state.
- Performs health checks on dependent components. If an issue is detected, it destroys and creates a new component.
- Receives realtime messages from the Notification Hub.
- Processes and replies to messages using respective component queues.
- Enable/Disable data streaming.

Implementation:

- Utilizes a ‘ComponentManager’ class to handle the creation and management of dependent components.
- Implements a priority queue to manage message processing based on urgency.

6.2 Dependent Component

Responsibilities:

- Receives realtime messages from the Notification Hub.
- Processes messages and maintains its own state.
- Replies to messages using its assigned queue.

Implementation:

- Contains a ‘DependentComponent’ class responsible for message processing.
- Process high priority messages first
- Uses concurrent collections to handle realtime message processing.

6.3 Notification Hub

Responsibilities:

- Enables realtime communication between components.
- Routes messages between the Core Component and Dependent Components using WebSocket as the main communication protocol.

Implementation:

- Utilizes SignalR for message routing and realtime communication.

Request and Response Payload Specification

Payload Structure

The system uses a standardized payload format for communication between components. This format includes headers, payload data, error information, state, priority.

Payload Definition:

```json
{
  "Header": {
    "MessageId": "12345",
    "Timestamp": "2024-08-01T12:00:00Z",
    "SenderId": "sender1",
    "RecipientId": "recipient1",
    "Priority": 0,
    "CorrelationId": "correlation123"
  },
  "Payload": "{\"key\":\"value\"}",  // This can be any JSON string
  "Error": {
    "Code": "404",
    "Message": "Not Found"
  },
  "State": 1,
  "Priority": 0
}
```

Field Descriptions

Header:

- MessageId (string): Unique identifier for the message.
- Timestamp (string): ISO 8601 formatted date and time when the message was sent.
- SenderId (string): Identifier for the sender of the message.
- RecipientId (string): Identifier for the recipient of the message.
- Priority (integer): Priority level of the message (e.g., 1 for normal, 0 for high).
- CorrelationId (string): Identifier used to correlate request and response messages.

Payload:

- Content (string): A JSON formatted string containing the data or command details.

Error:

- Code (string): Error code indicating the type of error (e.g., "404" for not found).
- Message (string): Descriptive error message.

State:

- State (integer): Represents the current state of the request or response (e.g., 1 for success, 2 for failure).

Priority:

- Priority (integer): Indicates the priority level of the message.

Usage Notes:

- The 'Header' section is used for routing and managing messages.
- The 'Payload' section should contain the actual data or commands intended for processing.
- The 'Error' section is included if there is an issue with processing the message.
- 'State', 'Priority' provide additional context for handling the message appropriately.

7. Applications

7.1 SignalRServer

Description: A deployable application that acts as the server, handling realtime communication and message routing.

Deployment: Can be deployed on any system with SignalR support.

7.2 CommunicationComponentLib

Description: A library containing code that can be used by any application requiring messagebased communication and realtime features.

Usage: Provides core functionalities and components for building applications that integrate with the messagebased system.

Technical Stack

NET8 Framework:

- Utilize .NET 8 and C# Core for developing the system.
- Ensure adherence to best practices in asynchronous programming and data handling.

Conclusion

This document provides a comprehensive design for a messagebased system with realtime communication and concurrent processing capabilities. It serves as a guide for engineers to implement and extend the system according to their needs.