20TH ANNIVERSARY EDITION

WSO2CON ASIA

PLATFORMLESS MODERNIZATION

Using EIPs to Solve Real-World Integration Challenges

Nadeesha Gamage
Director of Solutions Architecture
WSO2

Agenda

- Key challenges in integrating systems.
- Solving problems through patterns.
- Enterprise Integration patterns in a message flow.
- Example of different patterns and their use in real life.
- How to build these patterns with WSO2.

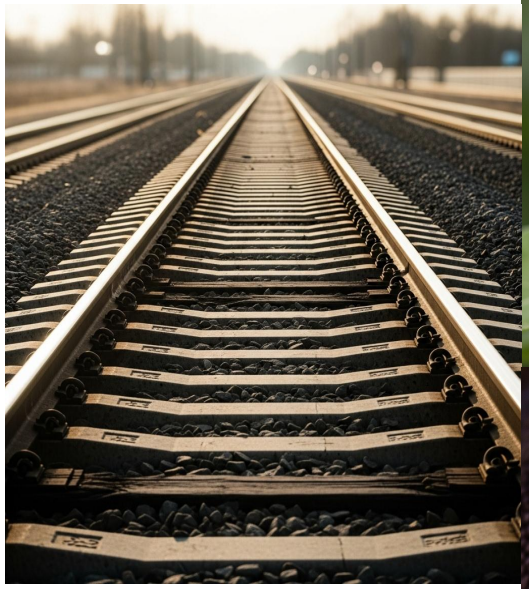
Key Challenges in System Integration

- Systems communicating in different standards and protocols.
- Systems process messages at different speeds.
- Systems are not available all the time.
- Integrations are too complex.
- Messages cannot be lost in transit.

Solving Problems Through Patterns

Patterns in Real Life

- Repeating or recurring arrangement of a design.
- Found everywhere in nature.
- Easily recognizable and predictable.
- Maintains its shape as it scales.

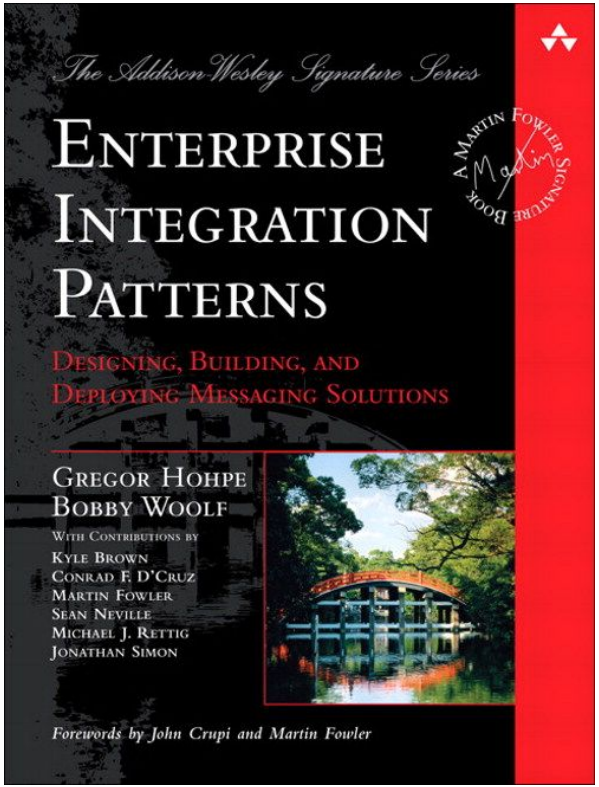


Why Use Patterns

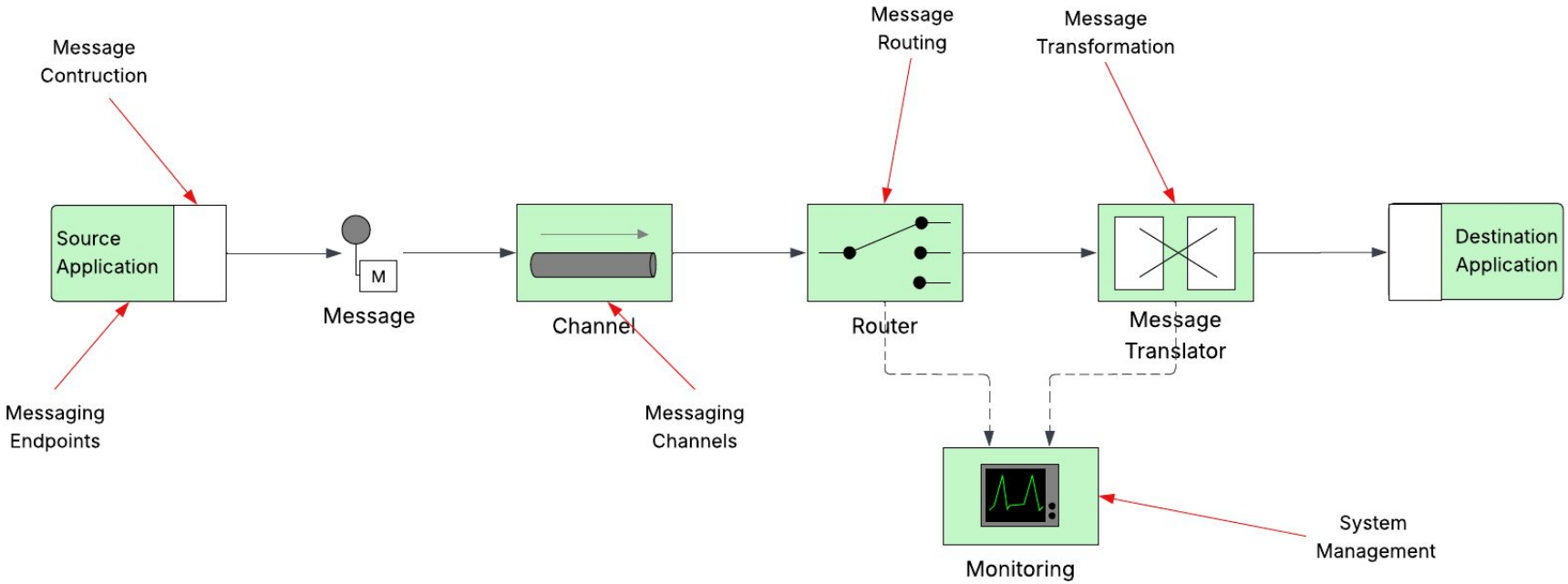
- Previously tested and optimized to solve a problem.
- Easy to build and scale.
- Easy to build tools to create and assemble patterns.
- Easy to template and apply in multiple places.

Enterprise Integration Patterns

- Set of patterns that solves common integration problems.
- Based on a book published in 2003 by Gregor Hohpe and Bobby Woolf.
- Defines 65 different patterns.
- Language and technology independent.
- Standardization of the solution.
- Loose coupling of services.



Enterprise Integration Patterns in a Typical Message Flow



Enterprise Integration Patterns

Message Endpoints

- Message Endpoint
- Messaging Gateway
- Messaging Mapper
- Transactional Client
- Polling Consumer
- Event-driven Consumer
- Competing Consumers
- Message Dispatcher
- Selective Consumer
- Durable Subscriber
- Idempotent Receiver
- Service Activator

Message Construction

- Generic Message
- Command Message
- Document Message
- Event Message
- Request-Reply
- Return Address
- Correlation Identifier
- Message Sequence
- Message Expiration
- Format Indicator

Message Channels

- Message Channel
- Point-to-Point Channel
- Publish-Subscr. Channel
- Datatype Channel
- Invalid Message Channel
- Dead Letter Channel
- Guaranteed Delivery
- Channel Adapter
- Messaging Bridge
- Message Bus

Message Routing

- Pipes-and-Filters
- Message Router
- Content-based Router
- Message Filter
- Dynamic Router
- Recipient List
- Splitter
- Aggregator
- Resequencer
- Composed Msg. Processor
- Scatter-Gather
- Routing Slip
- Process Manager
- Message Broker

Message Transformation

- Message Translator
- Envelope Wrapper
- Content Enricher
- Content Filter
- Claim Check
- Normalizer
- Canonical Data Model

System Management

- Control Bus
- Detour
- Wire Tap
- Message History
- Message Store
- Smart Proxy
- Test Message
- Channel Purger

Enterprise Integration Patterns

Message Endpoints

- Message Endpoint
- Messaging Gateway
- Messaging Mapper
- Transactional Client
- Polling Consumer
- Event-driven Consumer
- Competing Consumers
- Message Dispatcher
- Selective Consumer
- Durable Subscriber
- Idempotent Receiver
- Service Activator

Message Construction

- Generic Message
- Command Message
- Document Message
- Event Message
- Request-Reply
- Return Address
- Correlation Identifier
- Message Sequence
- Message Expiration
- Format Indicator

Message Channels

- Message Channel
- Point-to-Point Channel
- Publish-Subscr. Channel
- Datatype Channel
- Invalid Message Channel
- Dead Letter Channel
- Guaranteed Delivery
- Channel Adapter
- Messaging Bridge
- Message Bus

Message Routing

- Pipes-and-Filters
- Message Router
- Content-based Router
- Message Filter
- Dynamic Router
- Recipient List
- Splitter
- Aggregator
- Resequencer
- Composed Msg. Processor
- Scatter-Gather
- Routing Slip
- Process Manager
- Message Broker

Message Transformation

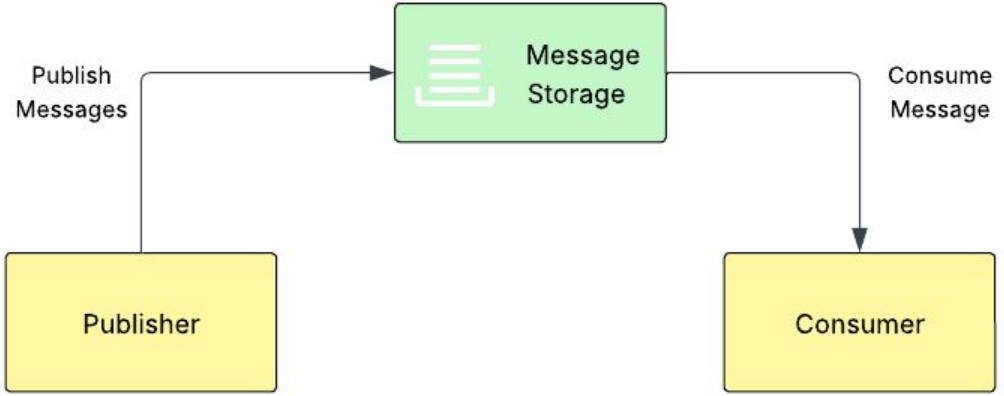
- Message Translator
- Envelope Wrapper
- Content Enricher
- Content Filter
- Claim Check
- Normalizer
- Canonical Data Model

System Management

- Control Bus
- Detour
- Wire Tap
- Message History
- Message Store
- Smart Proxy
- Test Message
- Channel Purger

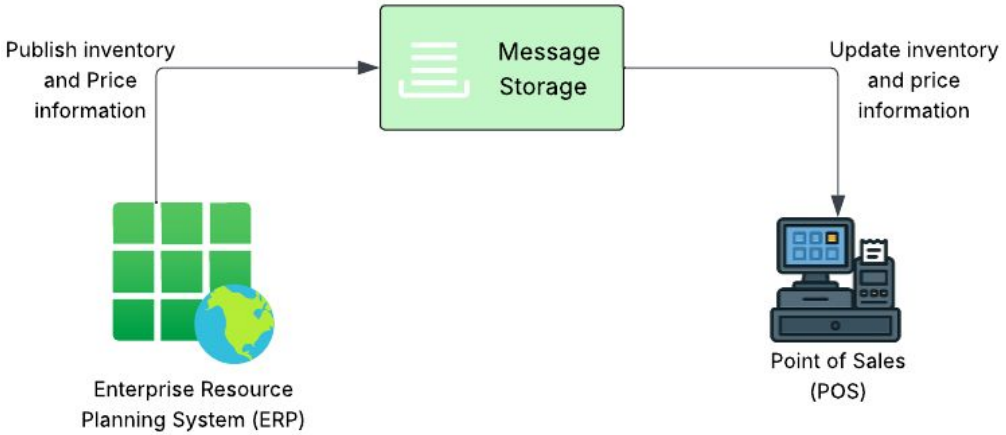
Durable Subscriber : Message Endpoints

- Subscriber’s doesn't miss messages if they are offline.
- Message broker retains messages for offline subscribers.
- When the subscriber reconnects, it receives all missed messages.



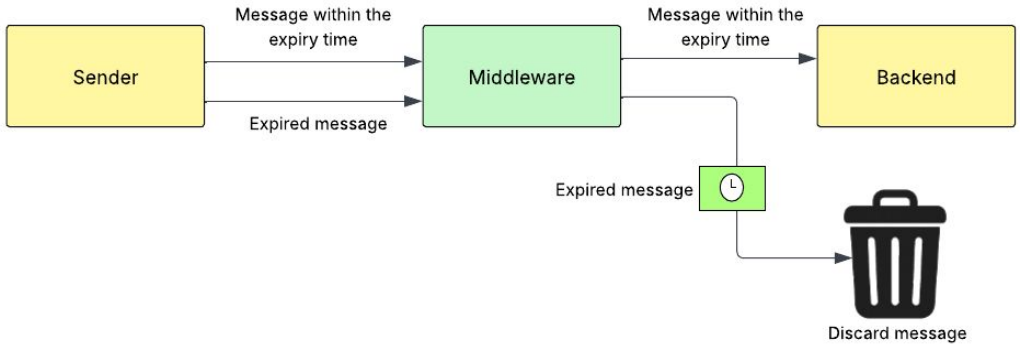
Durable Subscriber : Real World Usecase

- Inventory and price information syncing to remote POS Terminals.
- Unreliable connectivity.
- POS connects to a broker to pull all messages.
- Ensure guaranteed delivery of all messages.



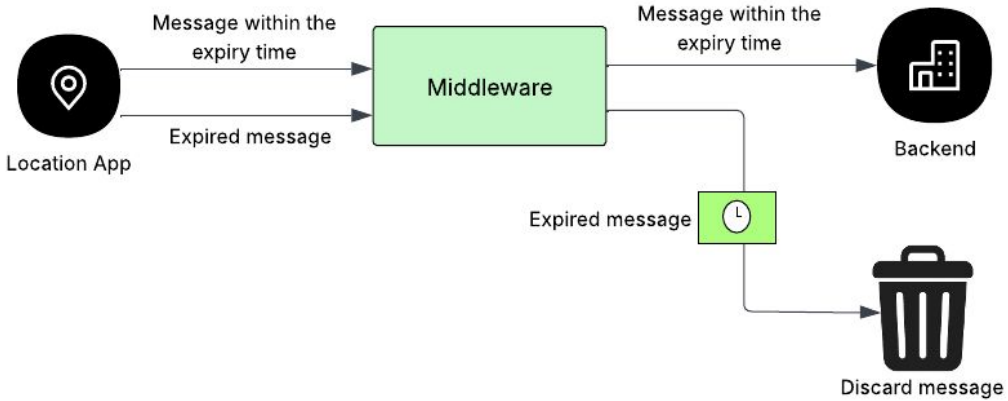
Message Expiry : Message Construction Pattern

- Producer sets an expiration time when sending the message.
- Middleware checks the expiry of the message before processing, and discard the expired messages.
- Avoids processing stale data.
- Reduce the load on the system.
- Ideal for delivering time sensitive messages.



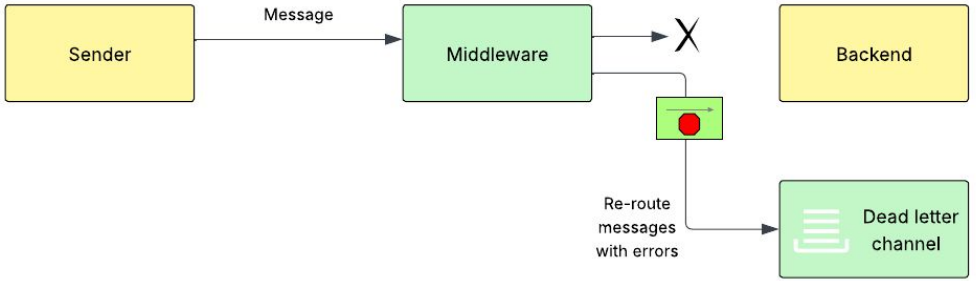
Message Expiry : Real World Usecase

- A geo-location app that sends location to a remote server.
- Pre-defined message expiry set for the message.
- If the location information arrives later than the expiry time, it is discarded without processing.
- Middleware only passes valid messages to the backend.



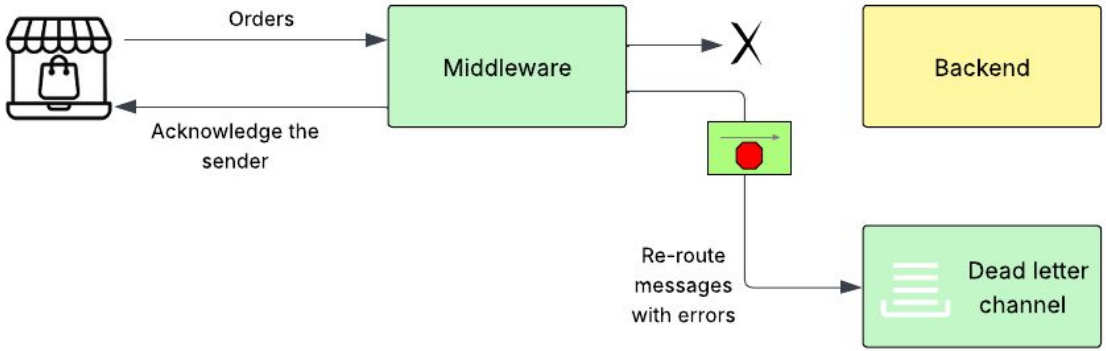
Dead Letter Channel : Message Channel Pattern

- Middleware stores unprocessed or problematic messages.
- Allows failed messages to be isolated, preventing disruption of the main flow.
- Ensures fault tolerance in message processing and store failed messages.



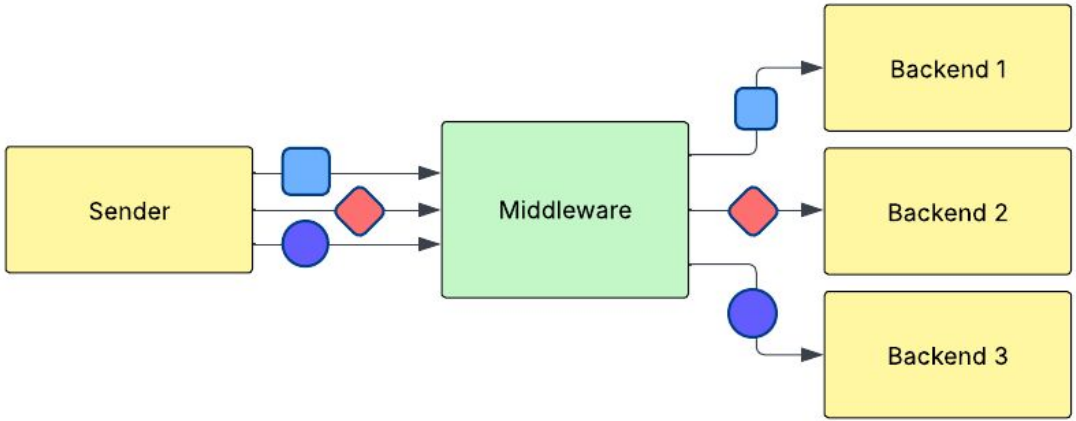
Dead Letter Channel : Real World Usecase

- An online store takes orders and updates the backend with order information.
- Some orders may fail due to unprocessable addresses.
- Failures are gracefully handled without impacting the user experience.



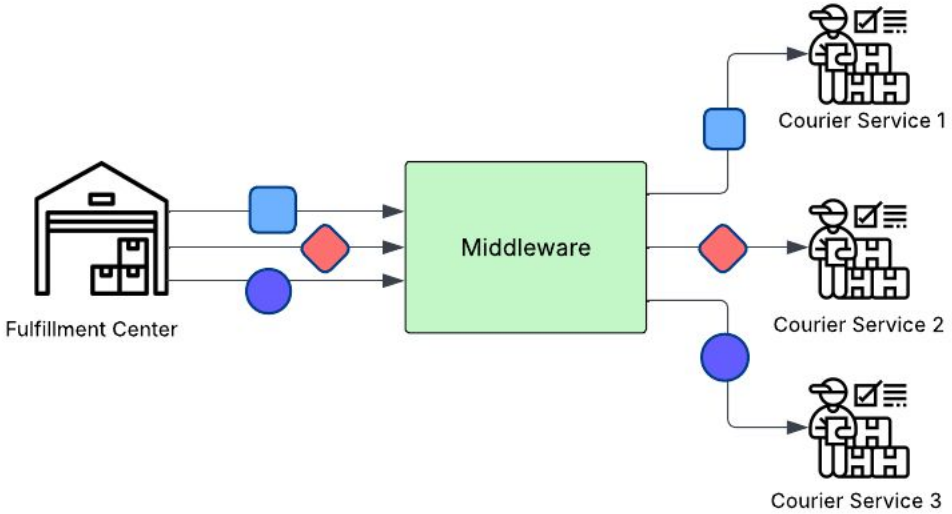
Content-based Routing : Message Transformation Pattern

- Routes messages to different channels based on the content of the message.
- Enables dynamic decision-making in message routing logic.
- Routing logic can be based on the content or header.
- Provides high flexibility in routing decisions.



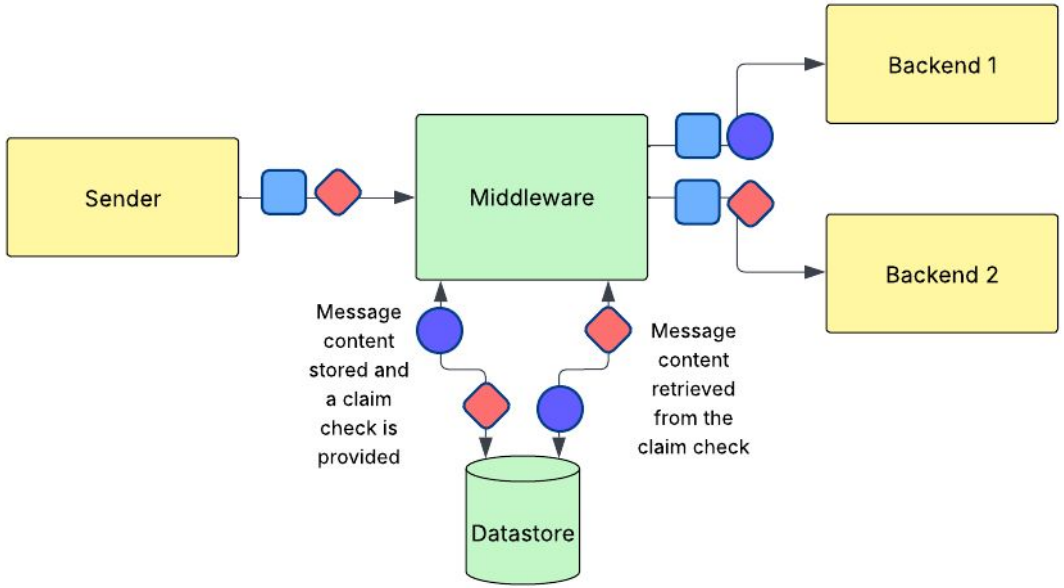
Content-based Routing : Real World Usecase

- A fulfillment center routing messages to different couriers based on the destination country.
- Routing logic is maintained in the middleware and can be easily changed based on the business requirement.



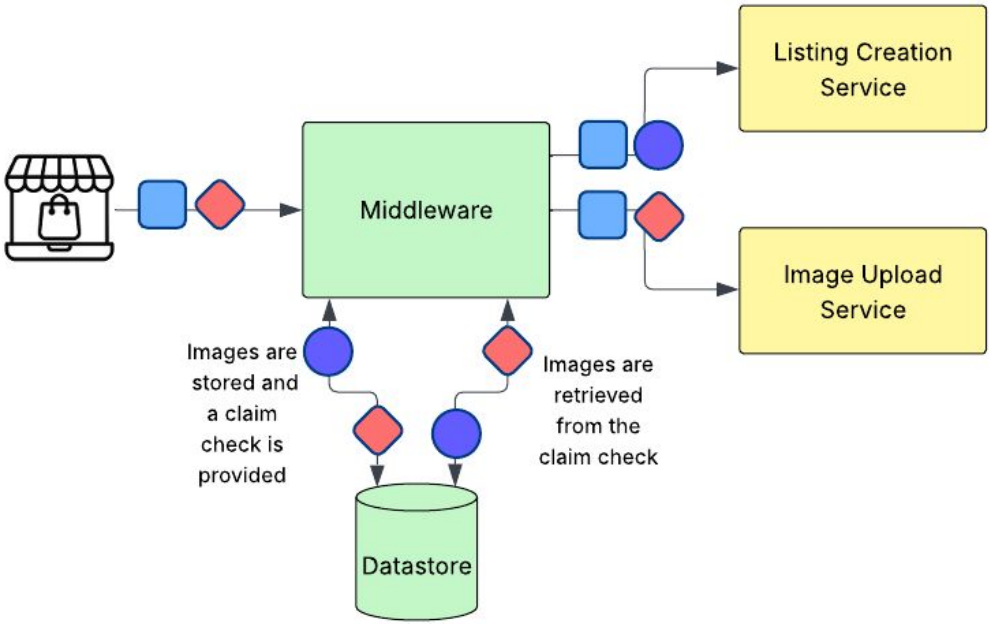
Claim Check : Message Transformation Pattern

- Extract the payload and store it for retrieval later in the message flow.
- Ideal for scenarios where large content is sent in a message.
- The claim check replaces the actual payload and is passed in the message.
- Improves performance, as large content doesn't need to be passed through the message flow.



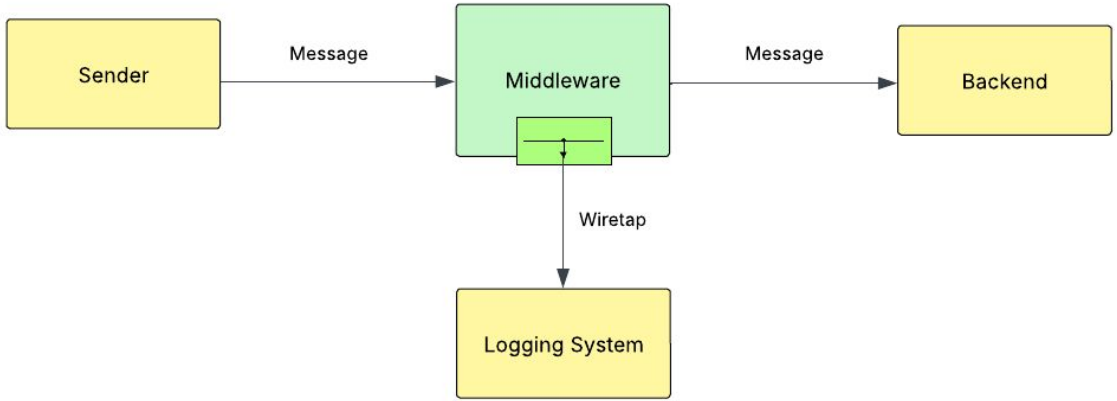
Claim Check : Real World Usecase

- E-commerce site require uploading of high-res product images.
- Product images are stored and a claim check is passed to backend systems.
- Once the listing is added, a backend system would retrieve the images using the claim check and process them.



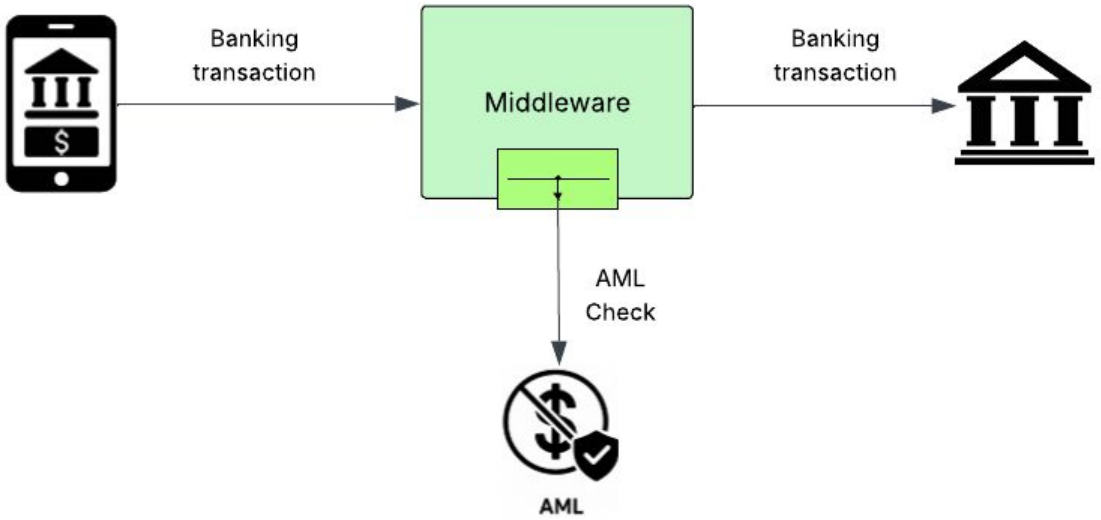
Wiretap : System Management Pattern

- Copy of the message sent to a secondary destination.
- Main message continues to the original destination unaffected.
- Done without interrupting the main processing flow and asynchronously to avoid latency.
- Ideal pattern for monitoring, auditing, or logging.



Wiretap : Real World Usecase

- A bank wants to run all its transactions via an anti money laundering (AML) system.
- No disruptions or latency should be added to the transaction.
- AML job should run asynchronously for each transaction.



How Do You Build These Patterns with WSO2

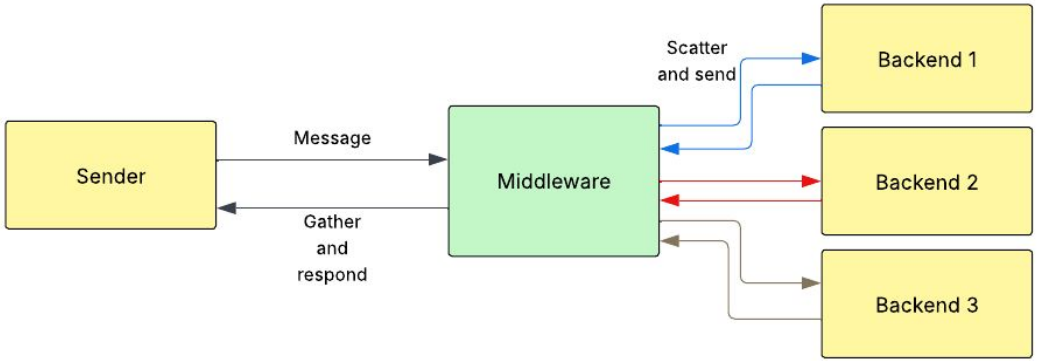
- WSO2 Integration Stack has 100% coverage on all EIP Patterns.
- Patterns are available as re-usable templates with WSO2.
- Try it today!

Question Time!


Thank you!
20TH ANNIVERSARY EDITION
WSO2CONASIA
PLATFORMLESS MODERNIZATION

Scatter-Gather : Message Routing Pattern

- Request is split (scattered) and sent to multiple recipients.
- Responses are collected (gathered) and combined into a single result.
- Tasks are performed concurrently by different services or components.
- Responses are gathered and typically merged using a defined strategy.



Scatter-Gather : Real World Usecase

- An app to check the doctor availability in multiple clinics at the same time.
- One single API call to the middleware. Middleware will make multiple calls to different clinics.
- Responses are gathered, sorted and send to the application.