Advanced Programming Techniques - week 5

Introduction to Enterprise Integration

Why Integration Matters

Why integration matters in the enterprise world

Real-world Example: Webshop Integration

A webshop integrates with:

- Payment Gateway
- Inventory System
- Notification System

Types of Integration

- Point-to-Point vs. Hub-and-Spoke
- Synchronous vs. Asynchronous
- Batch vs. Real-Time

Common Integration Problems

Data duplication

Multiple systems storing the same information leading to inconsistencies

Inconsistent data across systems

Different versions of the same data existing in various applications

Tight coupling (codependency)

Systems that are highly dependent on each other's implementation details

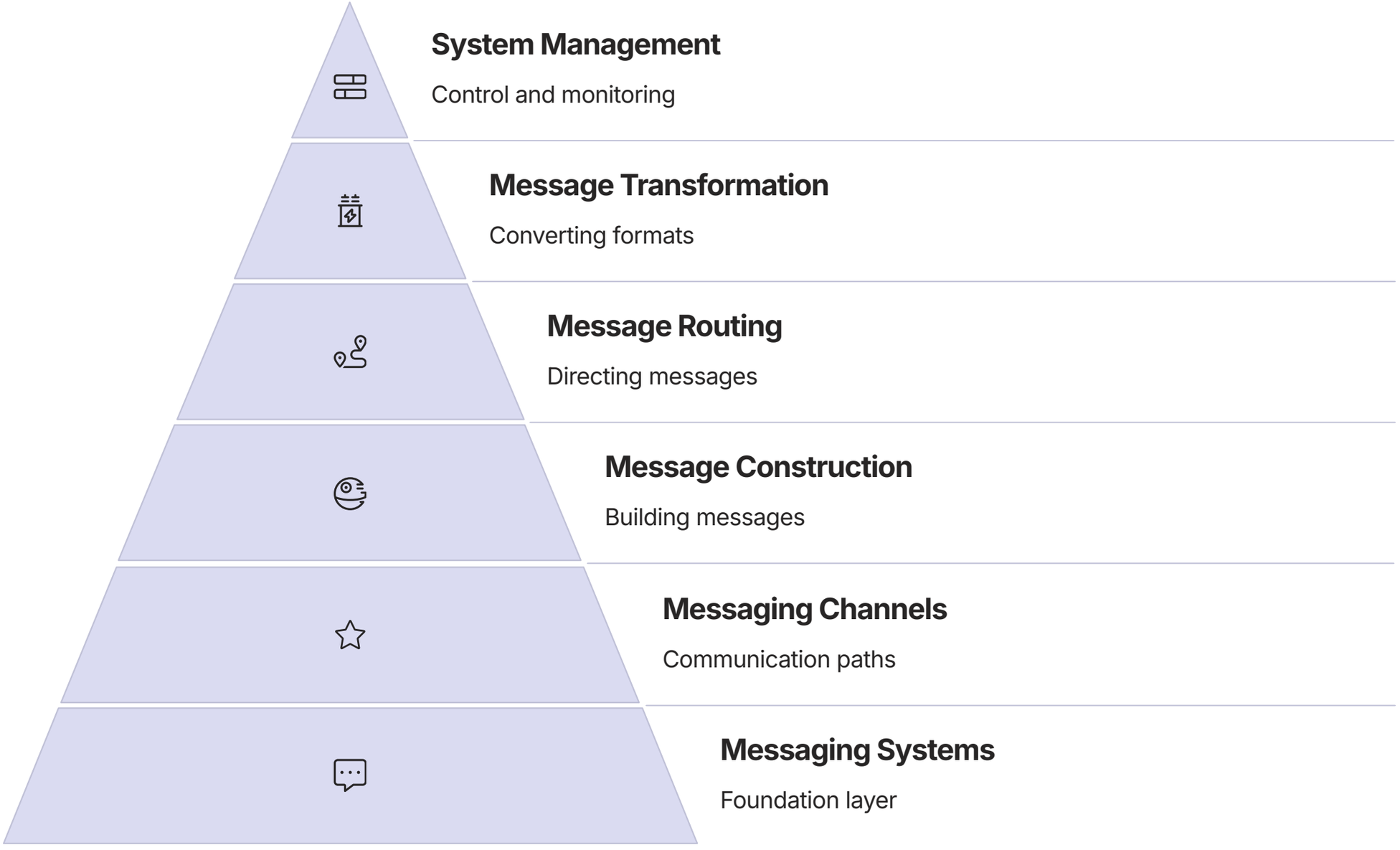
Communication failures

Breakdowns in the exchange of information between systems

Complex transformations

Difficult data conversions between different system formats

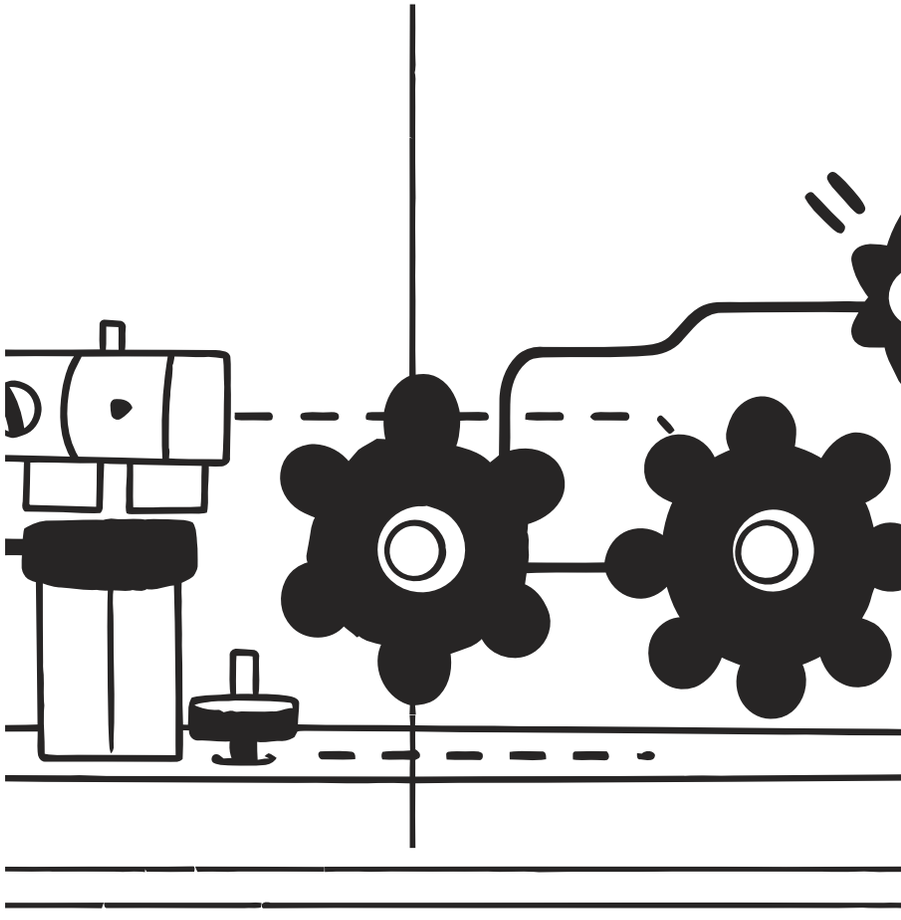
Enterprise Integration Patterns (EIP)



A catalog of messaging-based architectural solutions

Key Patterns Overview

Pattern	Description	Real-world example
Message Channel	Transmits messages	Kafka topic for logs
Message	Data unit sent	JSON with order details
Router	Routes messages	EU vs. Global system
Translator	Format converter	XML to JSON
Filter	Drops messages	Remove duplicates
Aggregator	Combines messages	Shipments → invoice
Splitter	Splits message	Order → items
Endpoint	Sender/Receiver	API pushing to queue

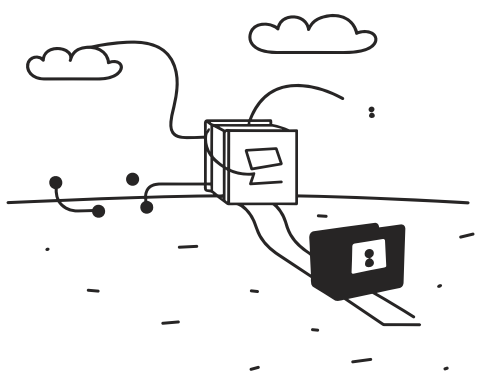


Messaging Systems

What is a Messaging System?

- Enables apps to communicate through messages over channels

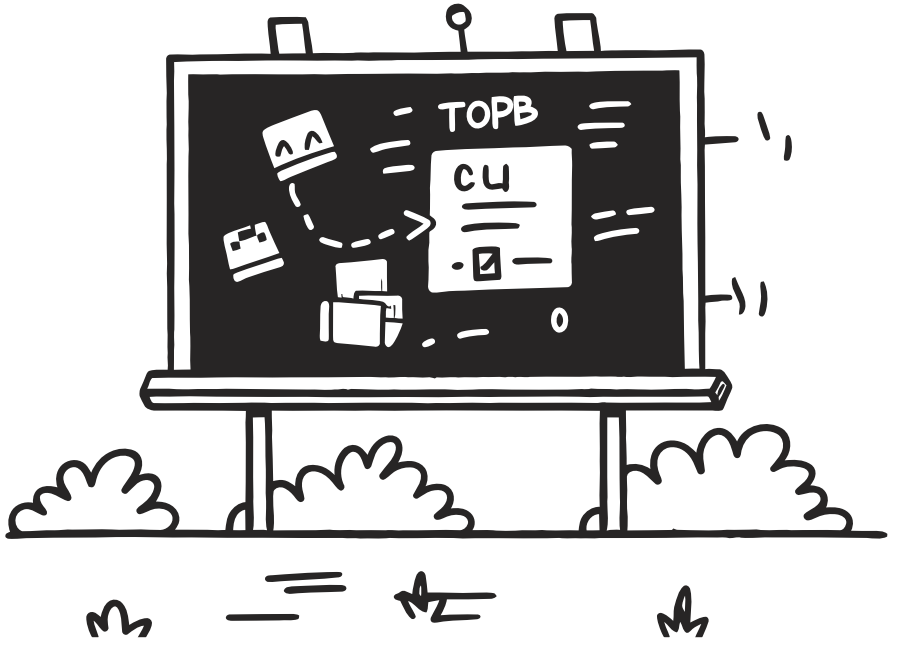
- Supports **asynchronous, decoupled, resilient** architecture



Key Concepts

- Message, Producer, Consumer
- Message Channel (Queue/Topic)
- Broker (e.g., RabbitMQ, Kafka)

Messaging Models



a) Point-to-Point (Queue)

- One producer, one consumer

- Message is removed after consumption.

Example: Order processing queue

b) Publish-Subscribe (Topic)

- One message sent to **all subscribers**

Example: Send order confirmation to multiple services

Messaging System Benefits

Decoupling of systems

Systems can operate independently without direct dependencies

Asynchronous processing

Operations can be performed without waiting for responses

Resilience to service failures

Systems can continue functioning even when components fail

Scalability via consumer groups

Easy to scale by adding more consumers to process messages

Auditability through persisted messages

Message history provides a record of all system interactions

Real-World Messaging Examples

E-commerce Scenario:

Cart → checkout event (message)

Order Service listens → Creates order

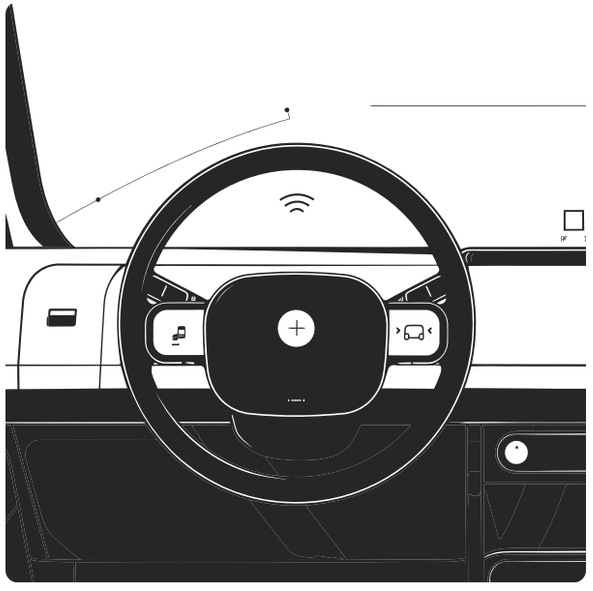
Payment Service → Processes payment

Inventory → Updates stock

Notification → Sends email

Monitoring Scenario:

- Services → Kafka → ELK → Dashboards + Alerts



Popular Messaging Platforms

RabbitMQ

Simple, transactional queues

Kafka

Event streaming, high-throughput

ActiveMQ

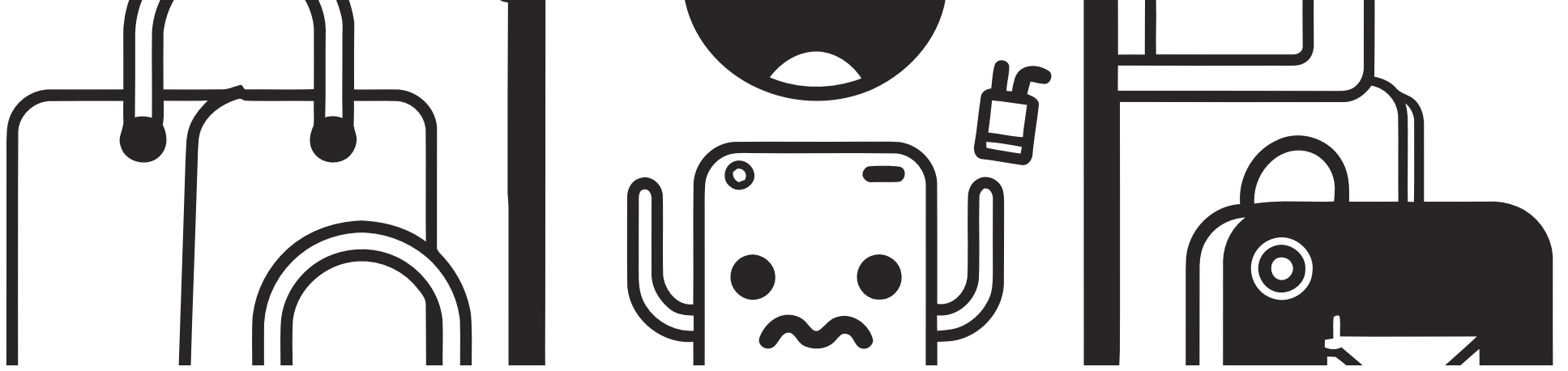
JMS-based enterprise messaging

SQS

AWS cloud-native queues

Azure Service Bus

Cloud-based pub/sub and queue support



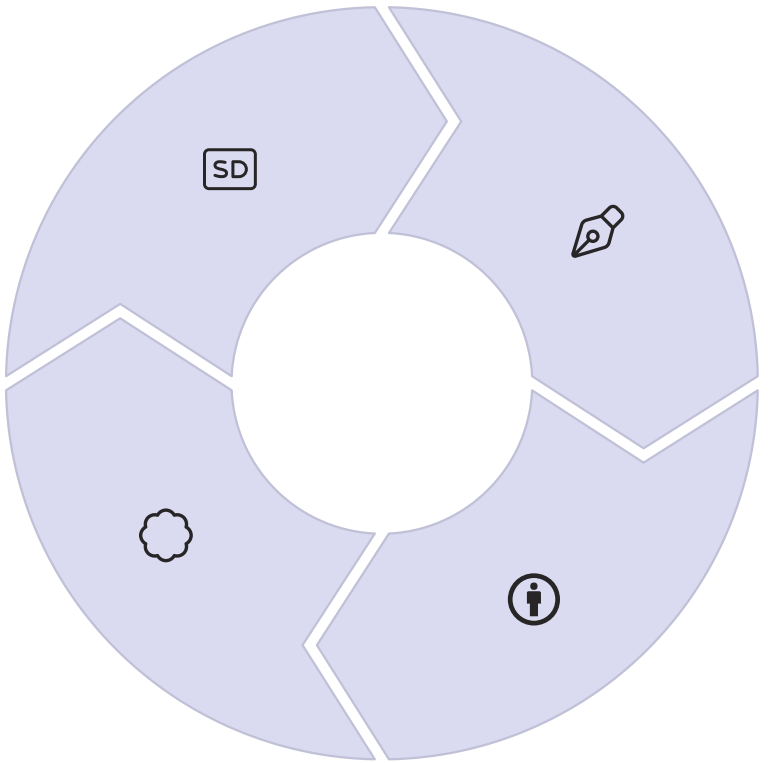
Challenges in Messaging

Problem	Solution
Message Ordering	Use partitioned channels
Duplicates	Ensure idempotent processing
Failures	Use retries and DLQs
Overflow	Scale consumers / add backpressure
Schema Evolution	Use versioned messages and validation

Message Channels

Definition

Logical path through which messages travel between sender and receiver



Purpose

Decouples systems, supports asynchronous delivery

Guarantees

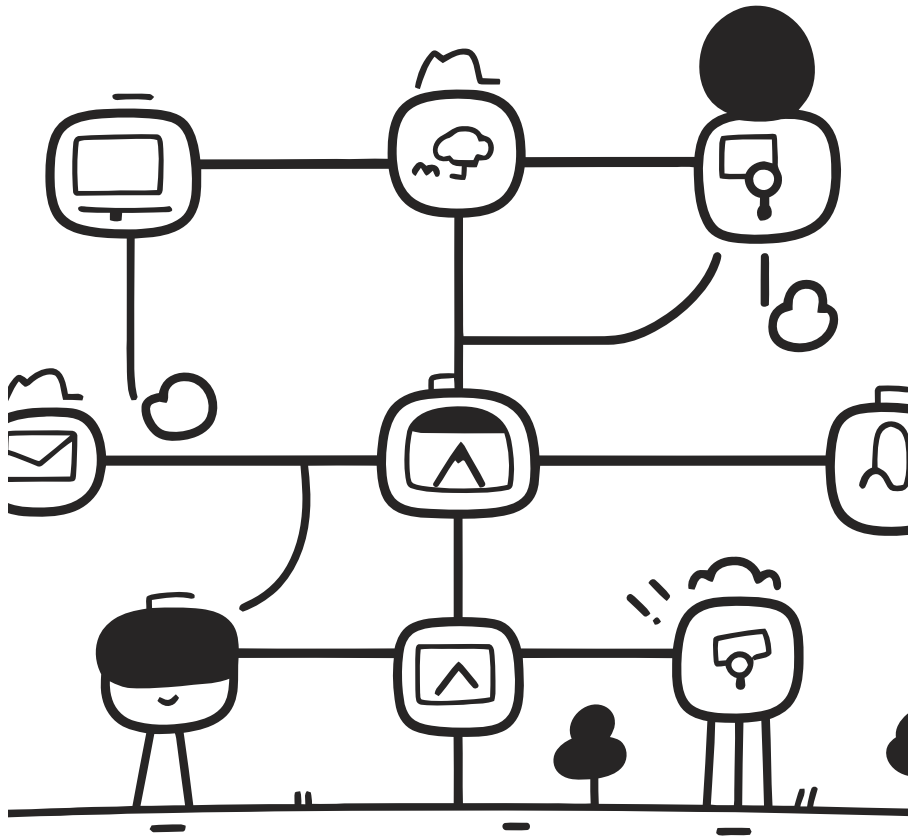
Ordering guarantees, Buffering behavior

Key Attributes

Direction, Type, Persistence

Types of Message Channels

Type	Description	Example
Point-to-Point	One consumer	Order queue
Publish-Subscribe	Broadcast to many	Kafka topic
Datatype Channel	Restricts message type	Only OrderPlaced
Invalid Message Channel	Handles malformed input	Logs parsing failures
Dead Letter Channel	Catches failed deliveries	Retry queue
Guaranteed Delivery	Retry + Ack	Ensures reliability



Message Channel Analogy

Point-to-point

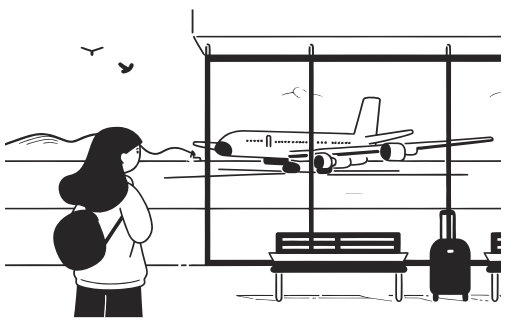
Taxi (one passenger)

Pub-sub

Radio (everyone tuned in)

Dead Letter

Lost-and-found



Invalid Message

Airport security screening