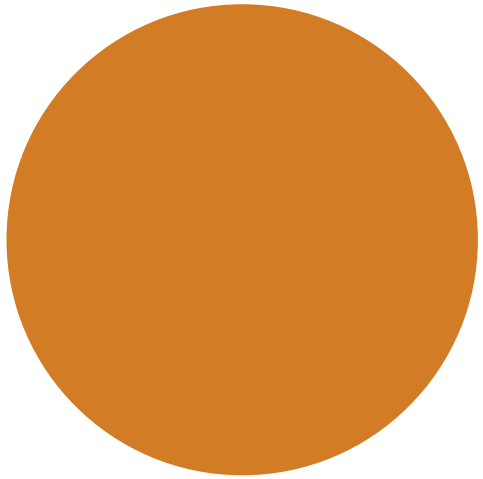
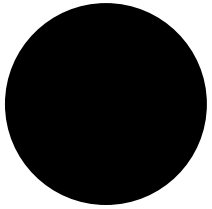
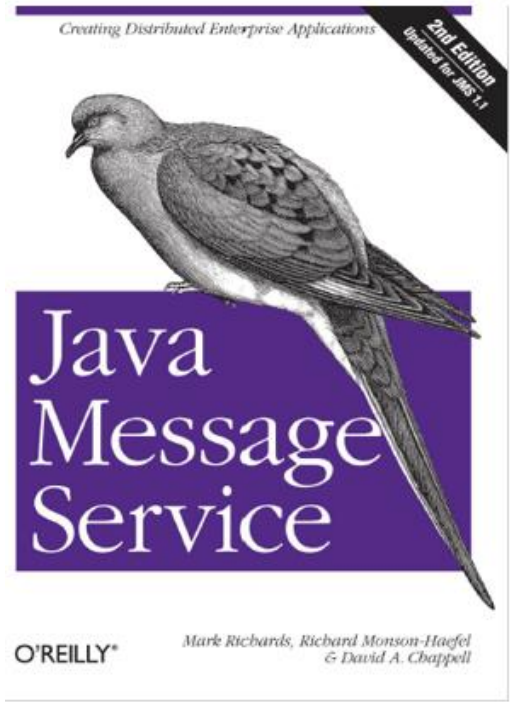




Enterprise Application Integration

Lesson 2
EAI Messaging patterns
&
Introduction to JMS

Hanzehogeschool Groningen
University of Applied Sciences

Mind map for this EAI course



Theory on software design principles & patterns

Theory on Integration Styles

Theory on IT architectures (SOA)

Knowledge of Enterprise Applications

Software Components

Integration

Performance as a whole

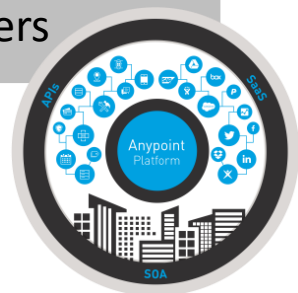
\+

Key Business goal for EAI => Agility which enables short Time to Market

Theory on EAI messaging patterns

Theory on EAI messaging technologies / standards

Knowledge on Middleware products from suppliers



Contents

- EAI Messaging Basics
- EAI Messaging Patterns
- Introduction to Java Message Service

Messaging basics

- Sender = producer = Provider
- Message has a Header & Body
- Message channel = queue = buffer
- Message is some sort of data structure = description of a command, event or other info.
- Messaging systems belong to the category of Message Oriented Middleware
- Sending Application => Messaging System => Receiving Application

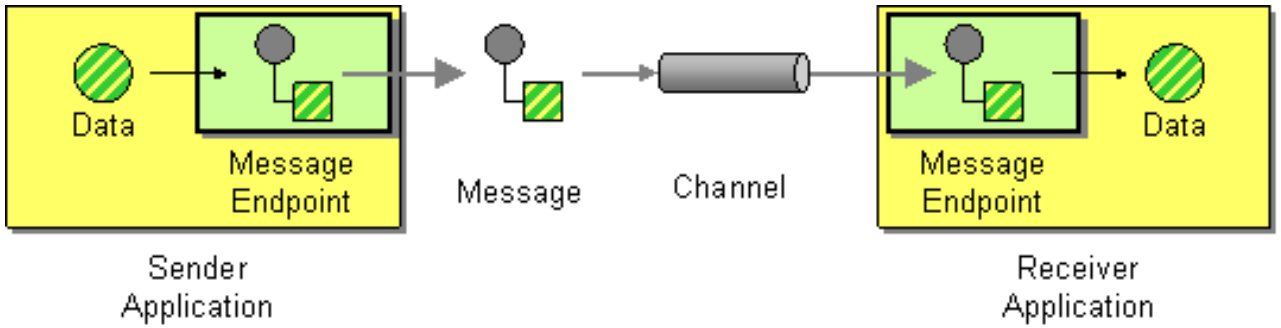
→ Receiver = Consumer

→ Body content = payload

→ Queue can be used concurrently by several applications.
Queue is a collection either in memory or on disk.

Messaging basics: Endpoints

- "It is the messaging endpoint code that takes that command or data, makes it into a message, and sends it on a particular messaging channel.
- It is the endpoint that receives a message, extracts the contents, and gives them to the application in a meaningful way."

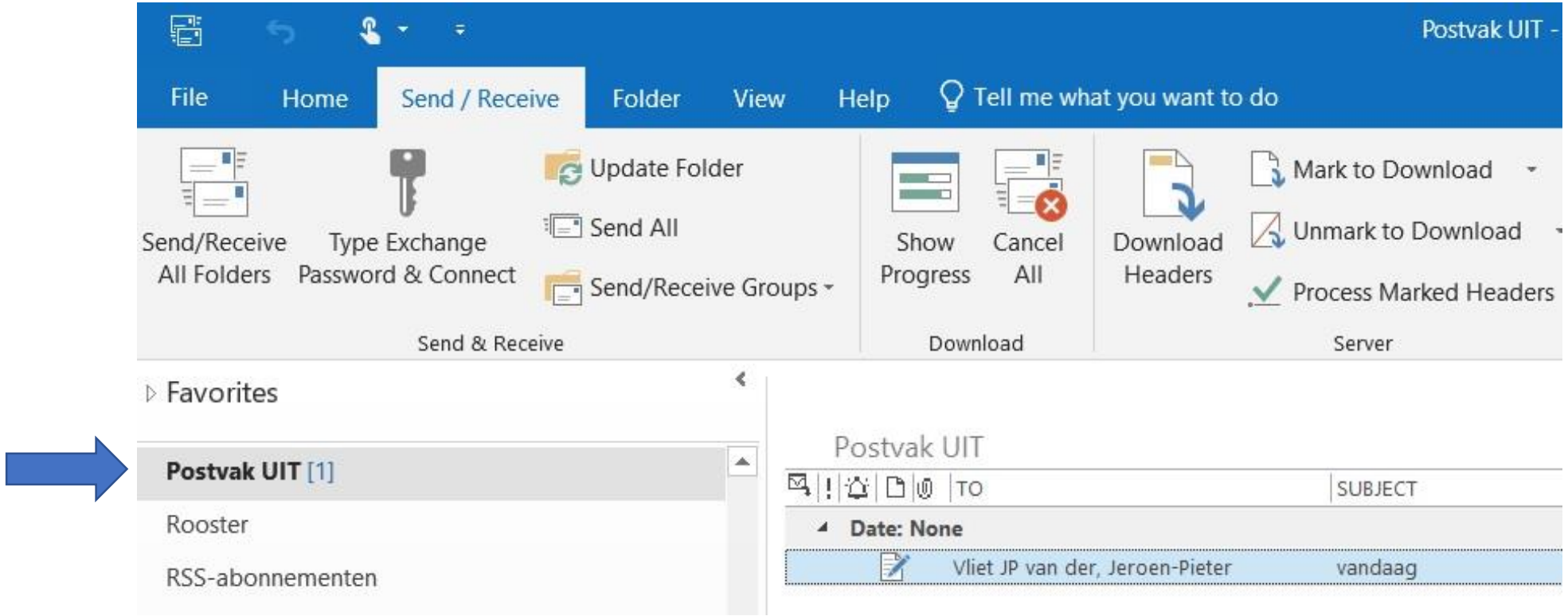


So an endpoint is more than just an url.

https://www.enterpriseintegrationpatterns.com/patterns/messaging/MessageEndpoint.html

Queue illustration

A queue



Messaging basics

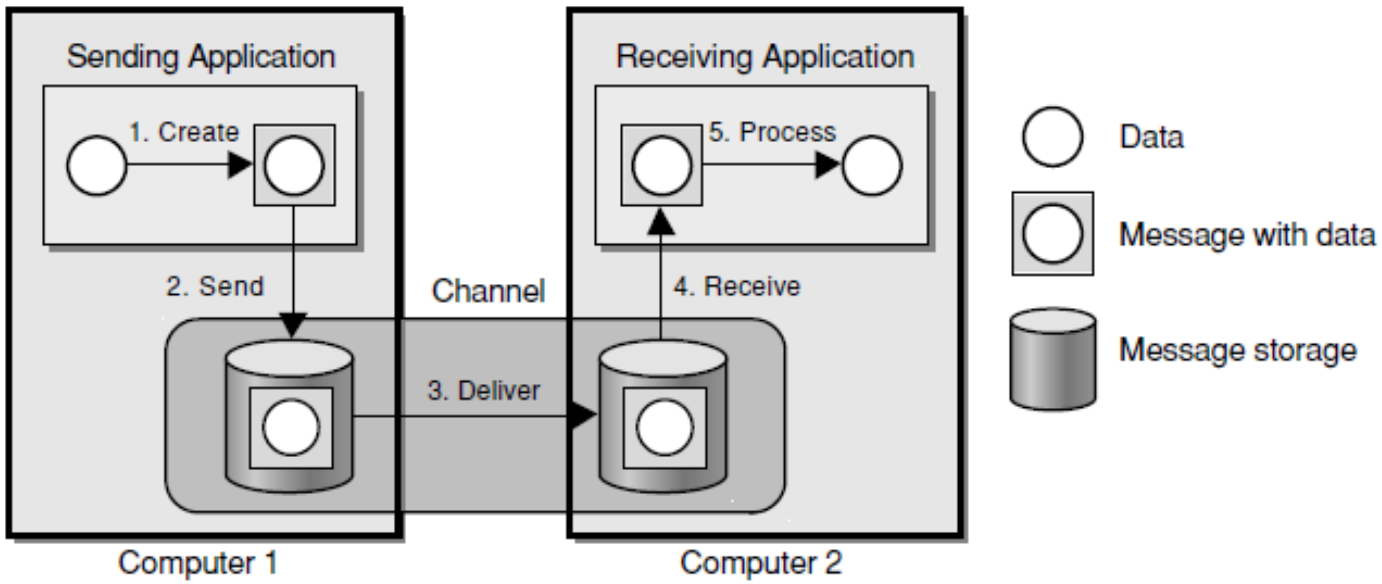
- Synchronous messaging

- Asynchronous messaging

Which one is better?

Enterprise Application Integration is primarily based on Asynchronous messaging

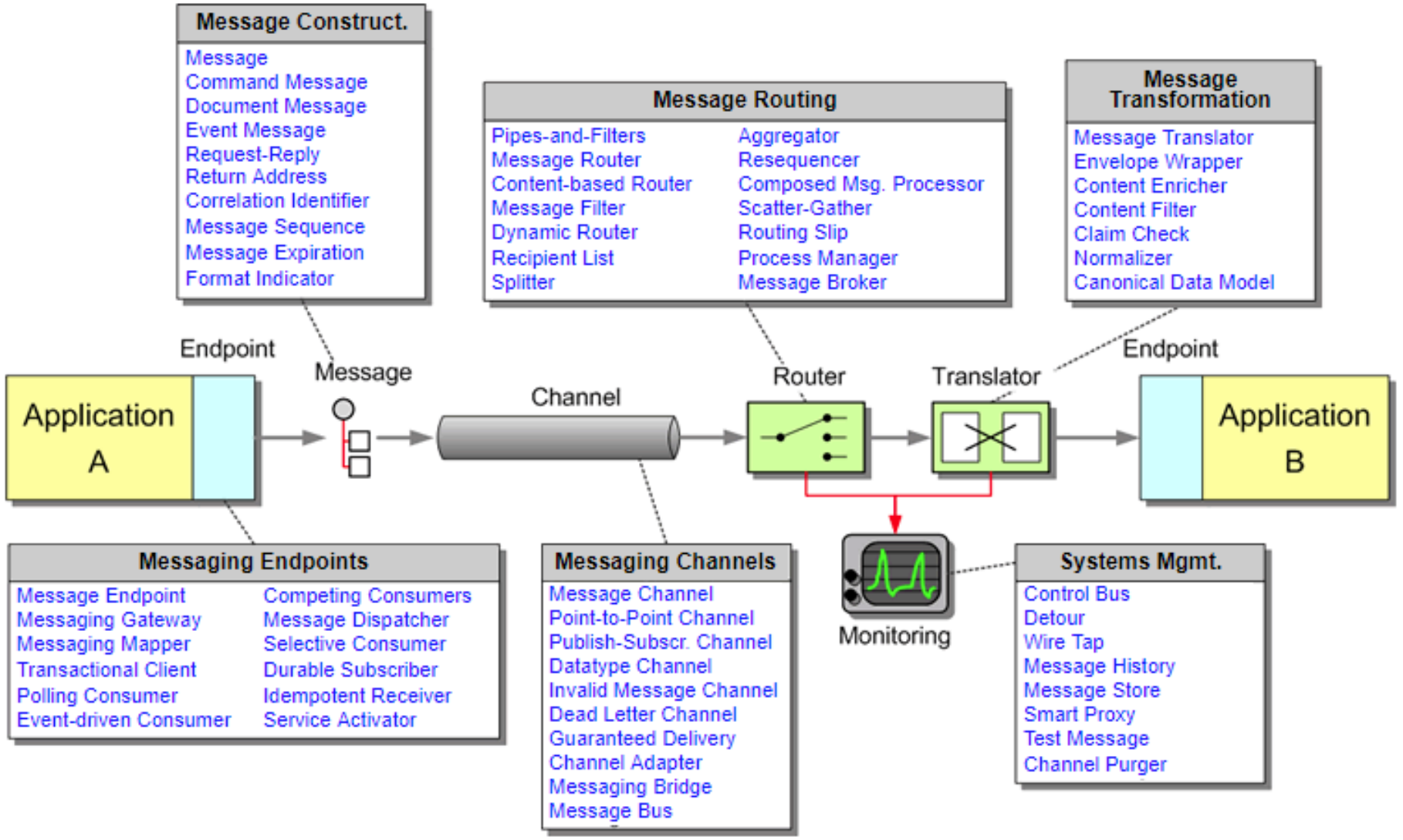
- Reliable delivery by sending messages repeatedly when needed.
- Allows a Send-and-Forget attitude of the provider => no need to wait (the thread is not blocked)
- This is Decoupling in time = apply the design principle of "loosely coupled systems"



Message Transmission Step-by-Step

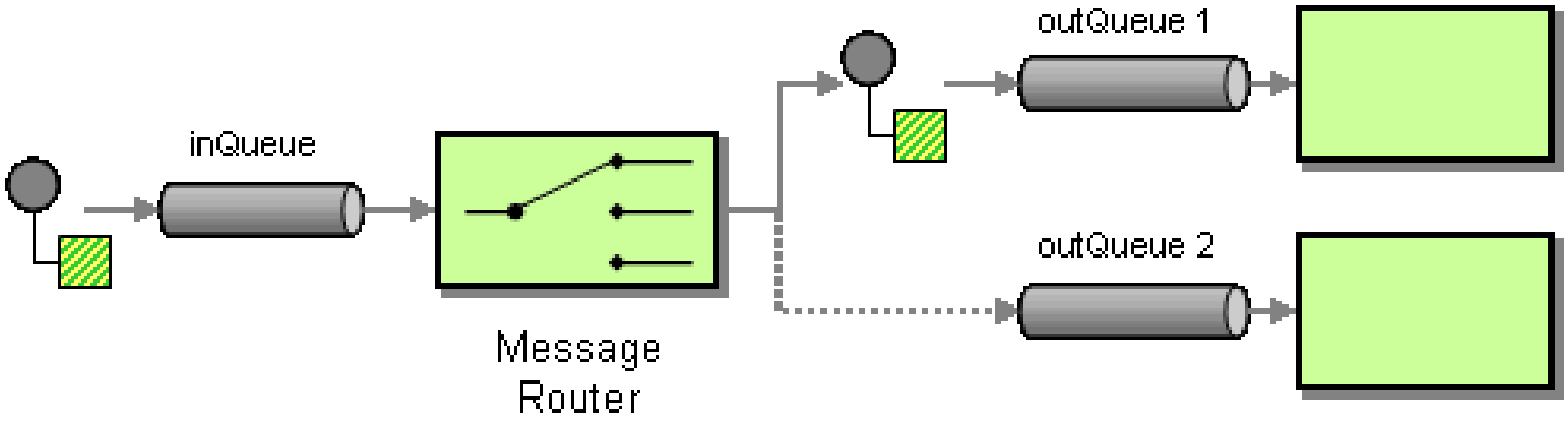
Source Hohpe & Woolf, 2004

Overview of 65 messaging patterns

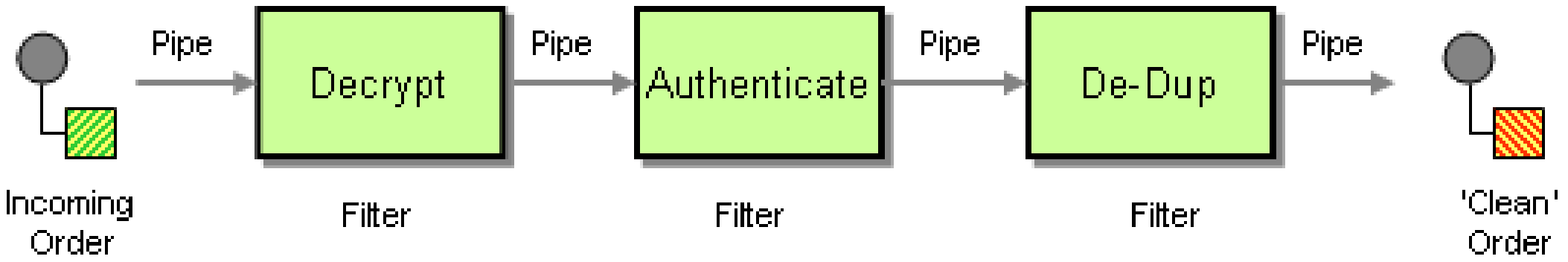


Source: https://www.enterpriseintegrationpatterns.com/patterns/messaging/

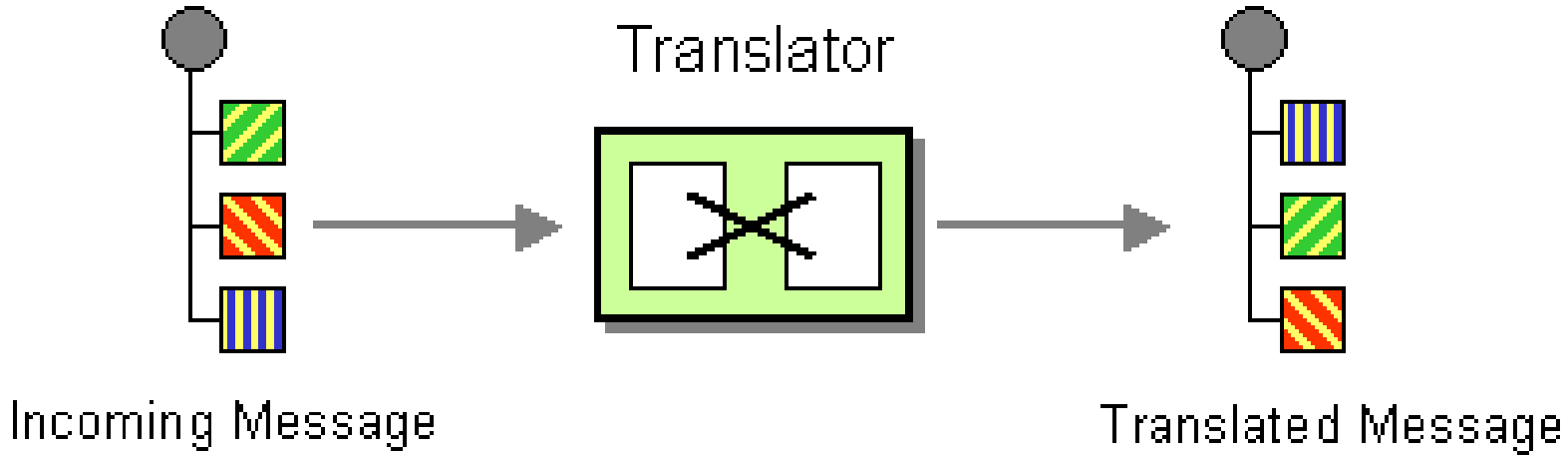
EAI Messaging Pattern: Router



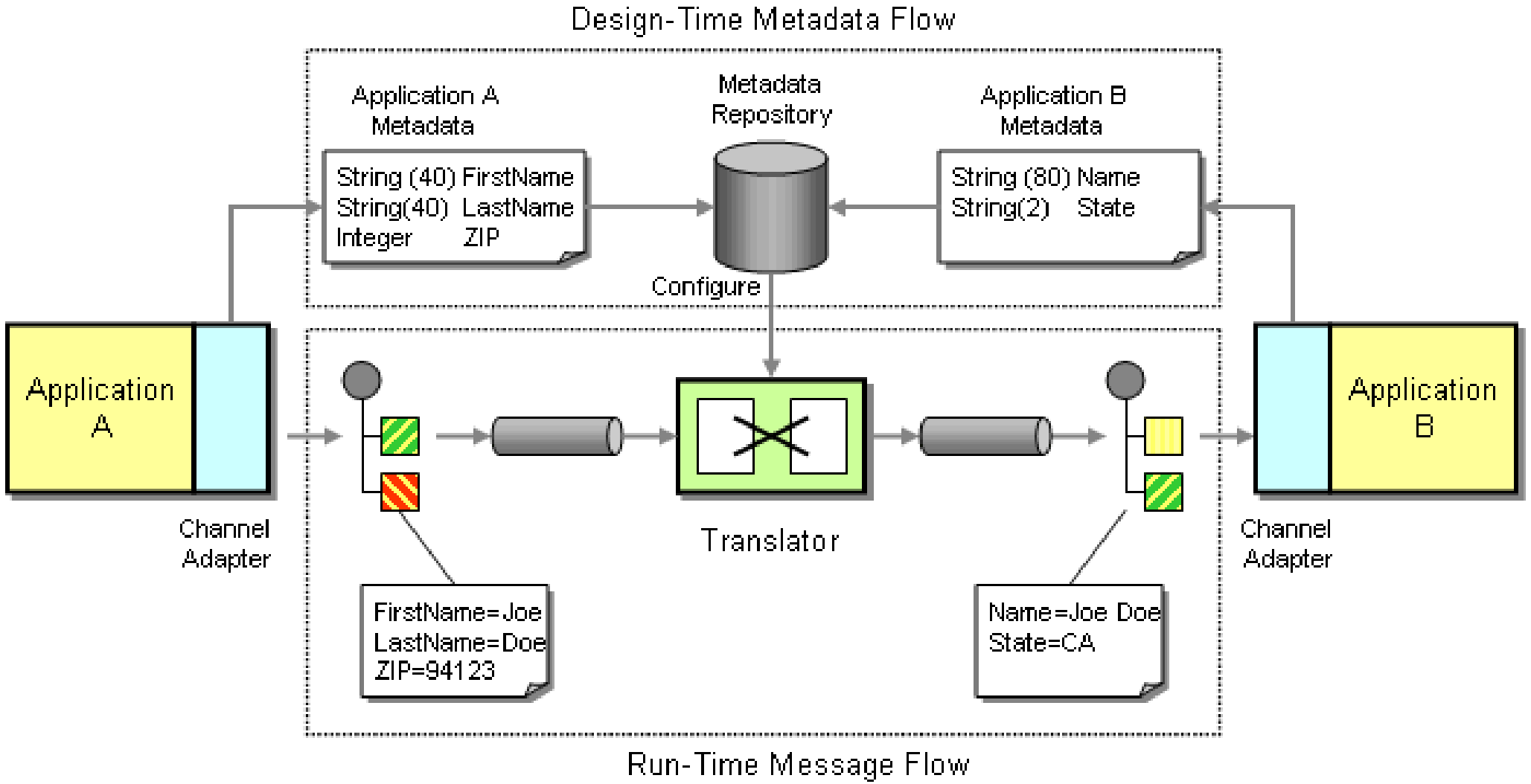
EAI Messaging Pattern: Pipes and Filters



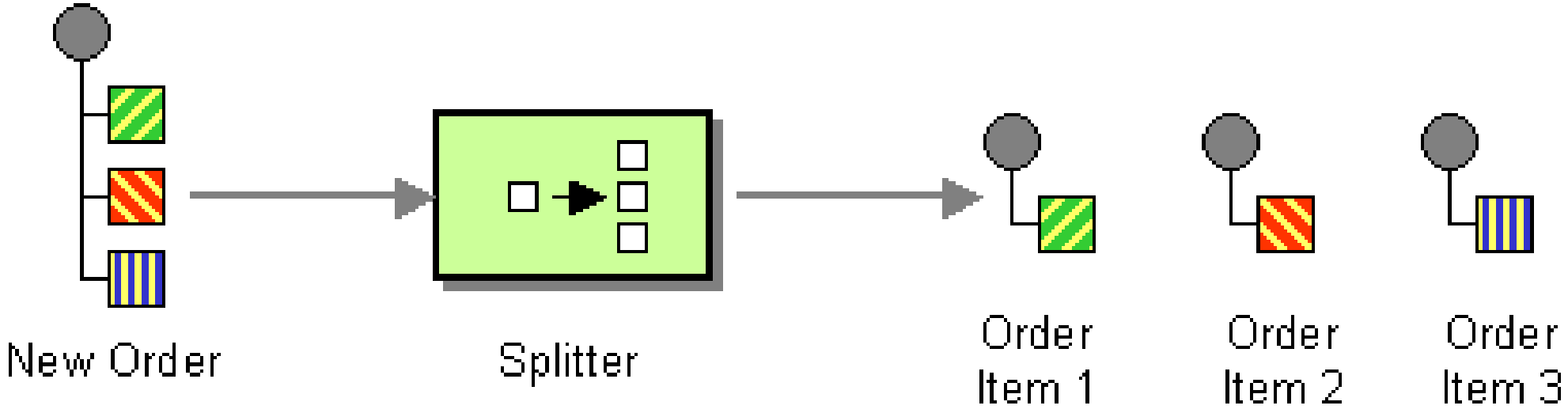
EAI Messaging Pattern: Translator



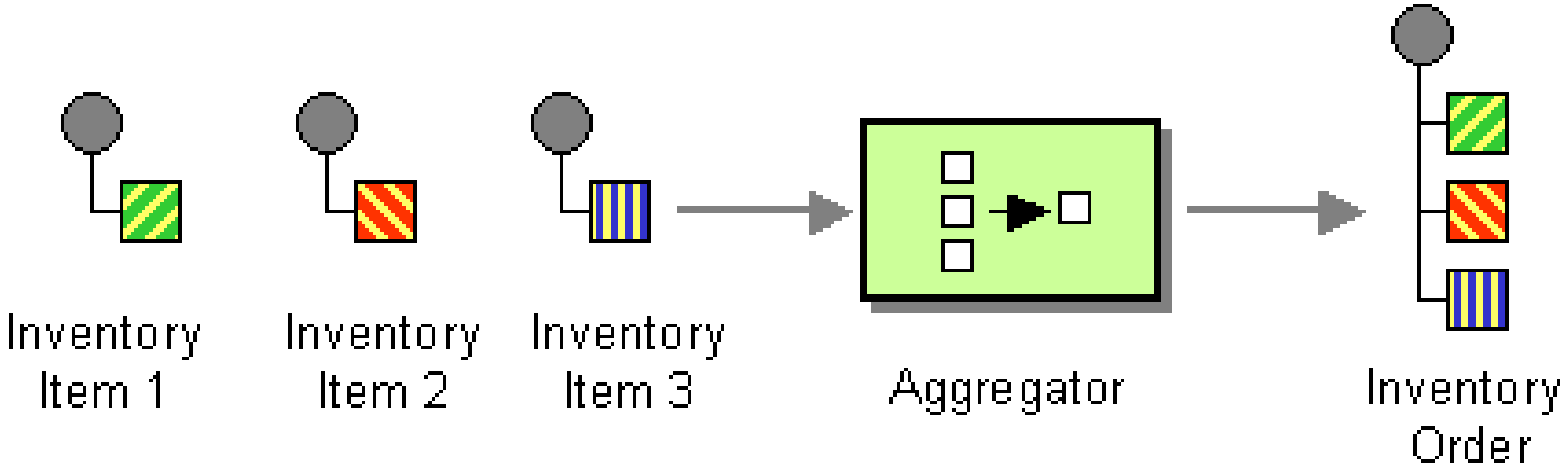
EAI Messaging Pattern: Translator example



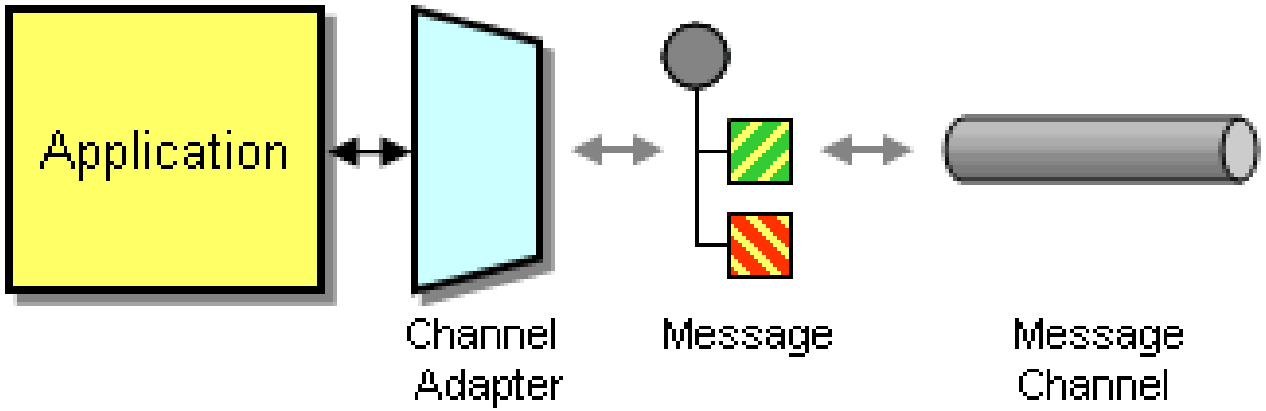
EAI Messaging Pattern: Splitter



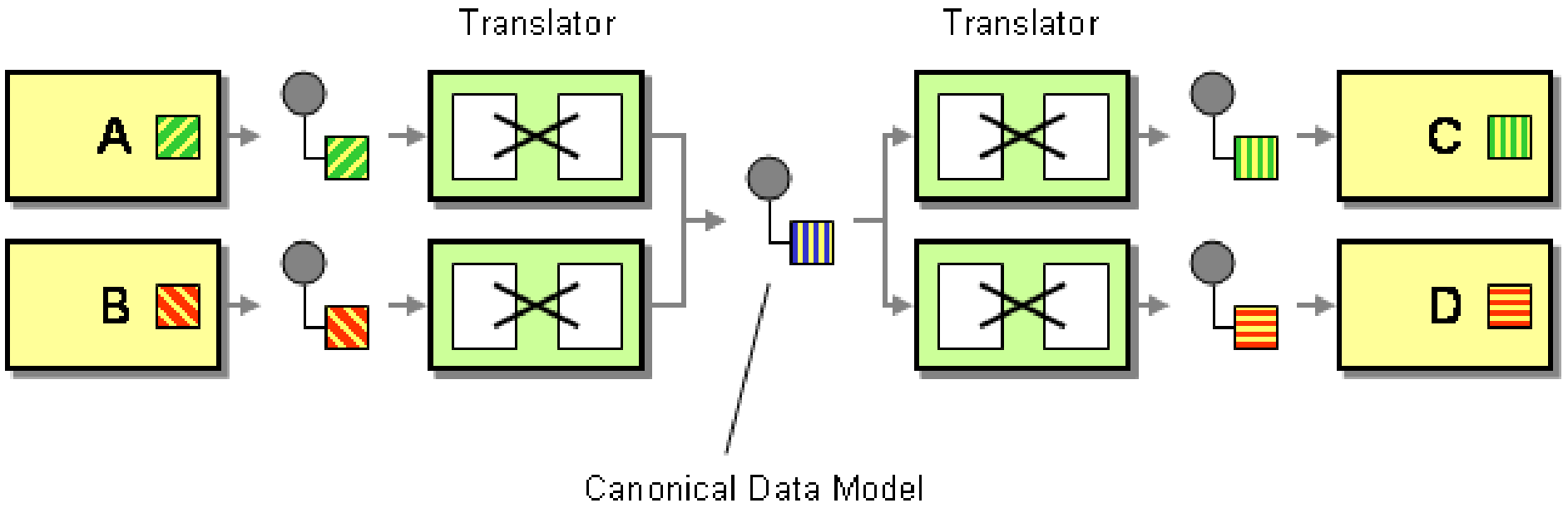
EAI Messaging Pattern: Aggregator



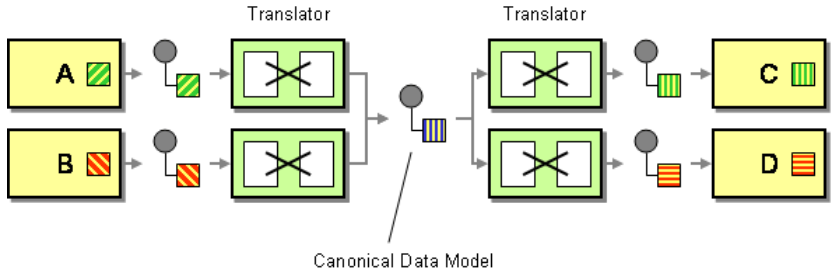
EAI Messaging Pattern: Adapter



EAI Messaging Anti-Pattern: Canonical Data Model



Anti-Pattern: Canonical Data Model



In this pattern, each message should use a company standard as data model standard / format.

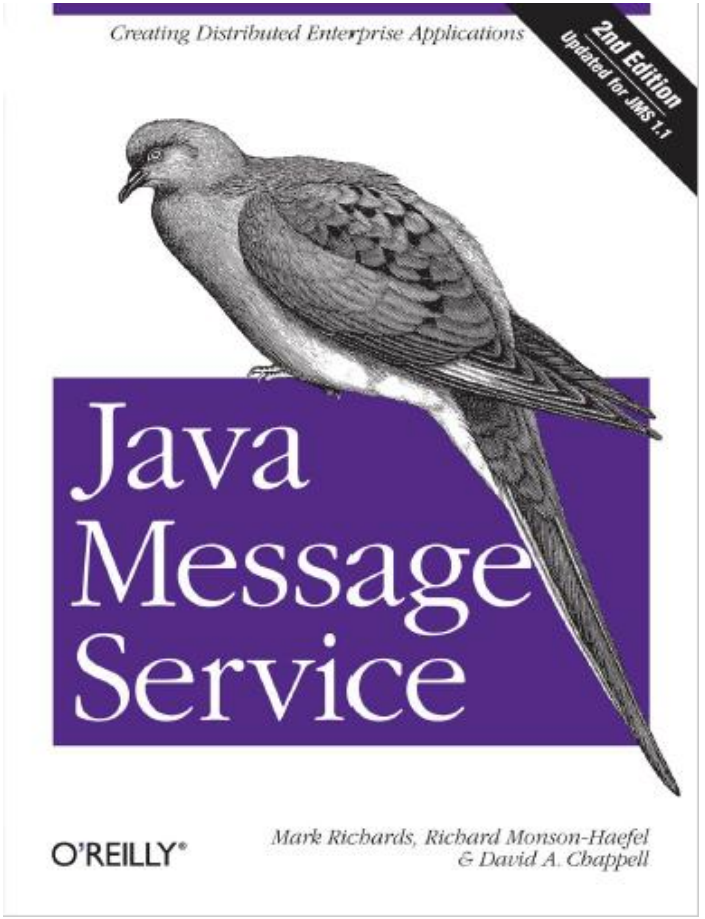
According to DOWALIL (2018, p. 86), this is often tried and it has often failed.

This considered an anti-pattern because:

- It is a sin against the principle of loosely coupled systems. It tightens the coupling by imposing additional requirements.
- It is usually a huge (communication) effort to unify to a certain standard. This effort will be repeated when changes occur.
- There is no pressing need for it, so it is a waste of effort and money.

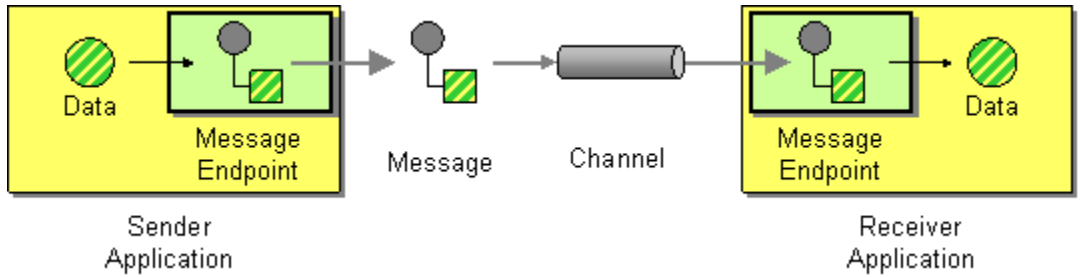
JMS - Java Message Service

- JMS is a specification for messaging
- Several suppliers have made Software products which implement this. Amongst others Apache's Active Message Queue, ActiveMQ



Sending a message with JMS; What should the software code of the provider endpoint* do?

1. Prepare the content you want to send in your own Java code
2. Put message content in a message object
3. Have this message sent through a channel so it can be delivered to the other endpoint
4. When needed: monitor whether the message was received
5. When needed: get a reply from the "endpoint at the other side of the line"
6. When needed: relate a reply to a previously sent message by using a correlation ID
7. etc.



* See endpoint definition slide in the beginning of this pptx presentation

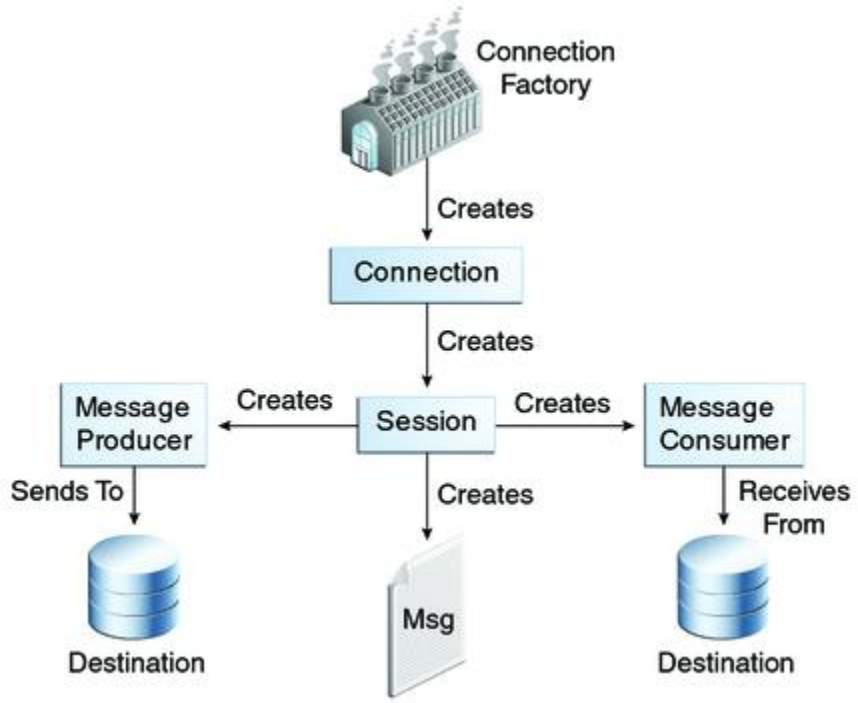
Set up for messaging with JMS

- Before sending messages you have to create the needed environment.

EAI Pattern / component	JMS name	Related (Java) example code / windows command
Message Channel	Destination	Destination destination = session.createQueue(subject);
Point-to-point channel	Queue	,,
Publish-subscribe channel	Topic	
Sender endpoint	Producer	MessageProducer producer = session.createProducer(destination);
Receiver endpoint	Consumer	
	Connection	Connection connection = connectionFactory.createConnection(); connection.start();
	Session	
Message Broker	JMS Provider or Broker	.\activemq.bat start

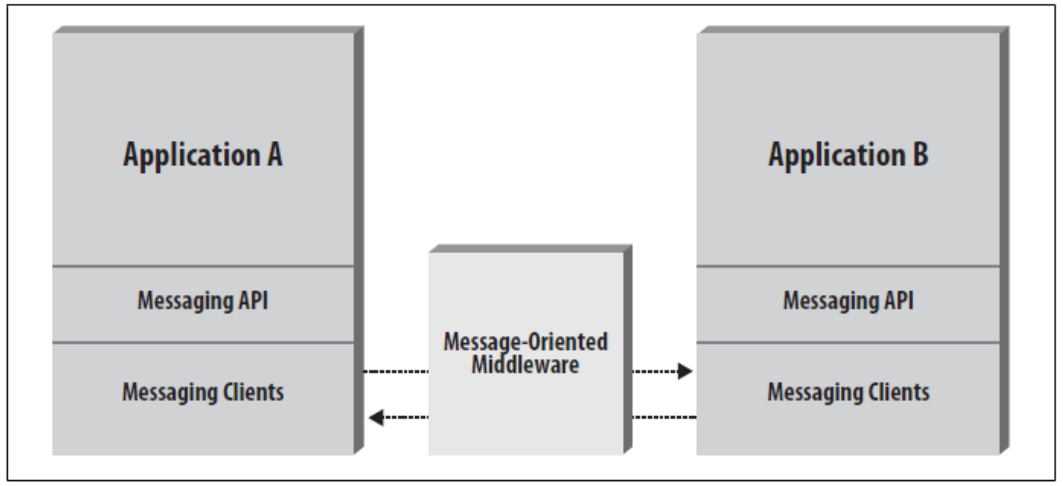
”In the Point to point messaging domain, destinations are called queues. In the pub/sub messaging domain, destinations are called topics” Source

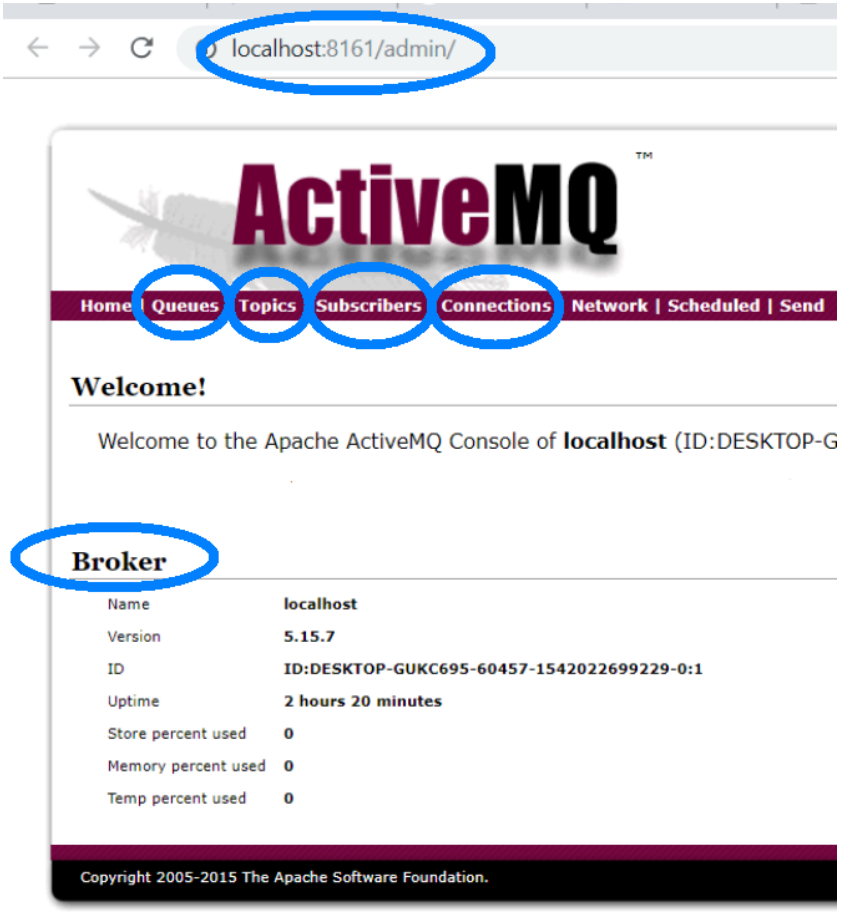
Set up
for messaging with JMS



EAI Pattern / component	JMS Name	
	Connection Factory	ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(url);

Getting started with JMS, learning task 2





Anypoint +

ActiveMQ