BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Message Channels

Madhu Venkat

Guest Faculty
BITS, WILP

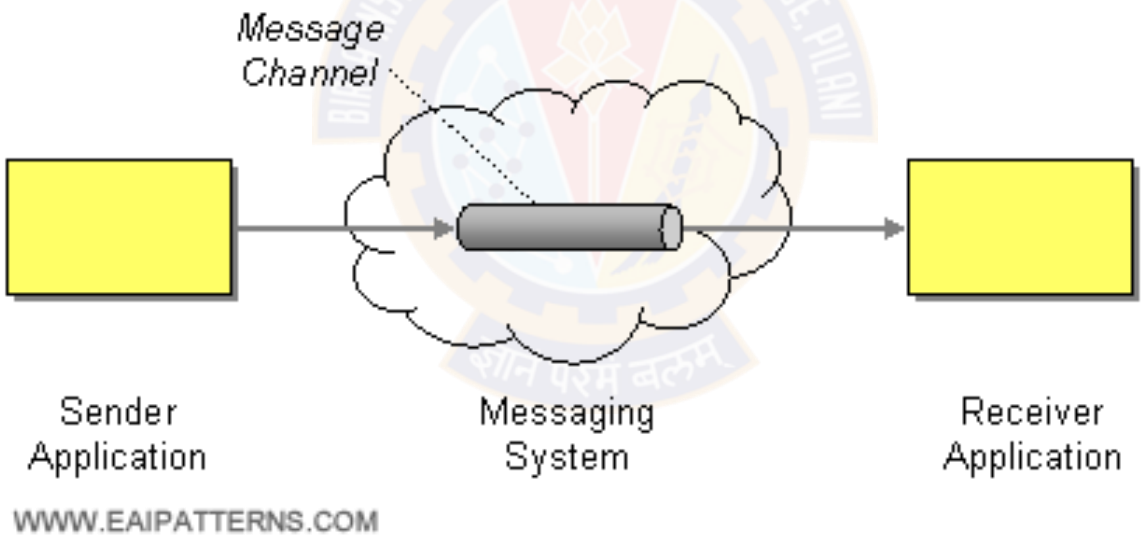
In this segment

Message Channels

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

Message Channel

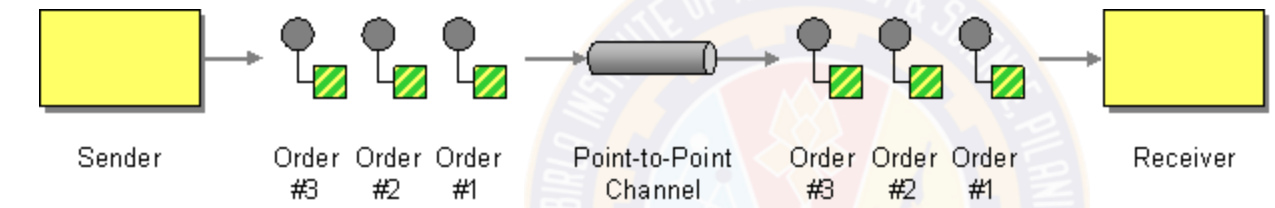
- A Message channel is a basic architectural pattern of a messaging system and is used fundamentally for exchanging data between applications. An application can use a channel to make a specific type of data available to any receiver applications that need to consume that type of data.

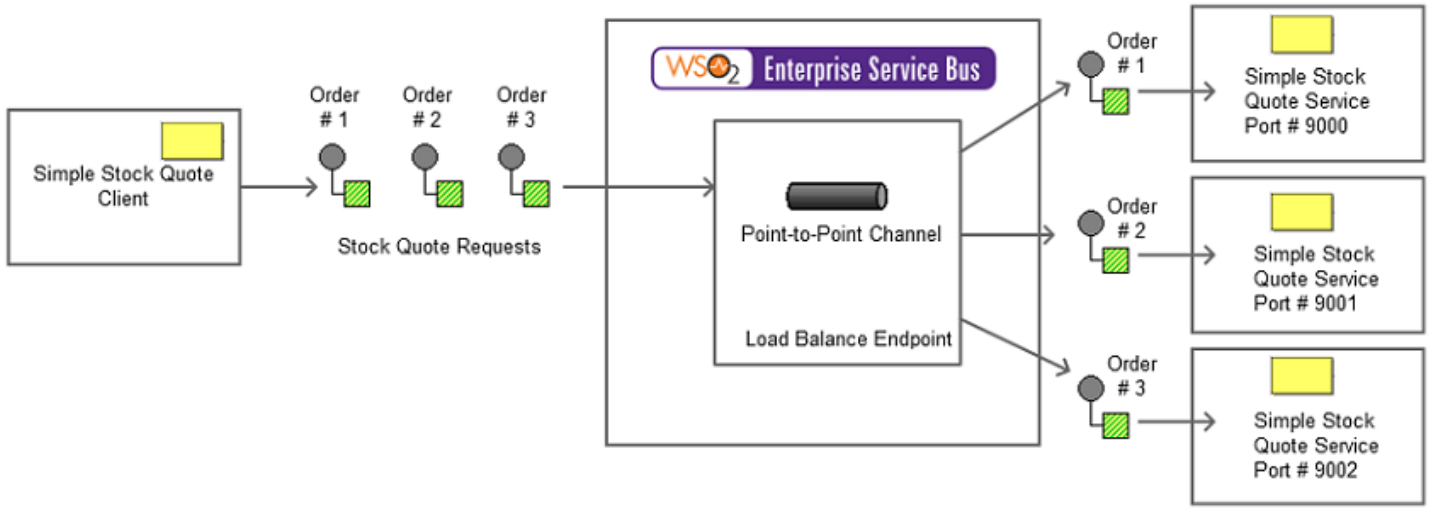


Message Channel

Point to Point Channel

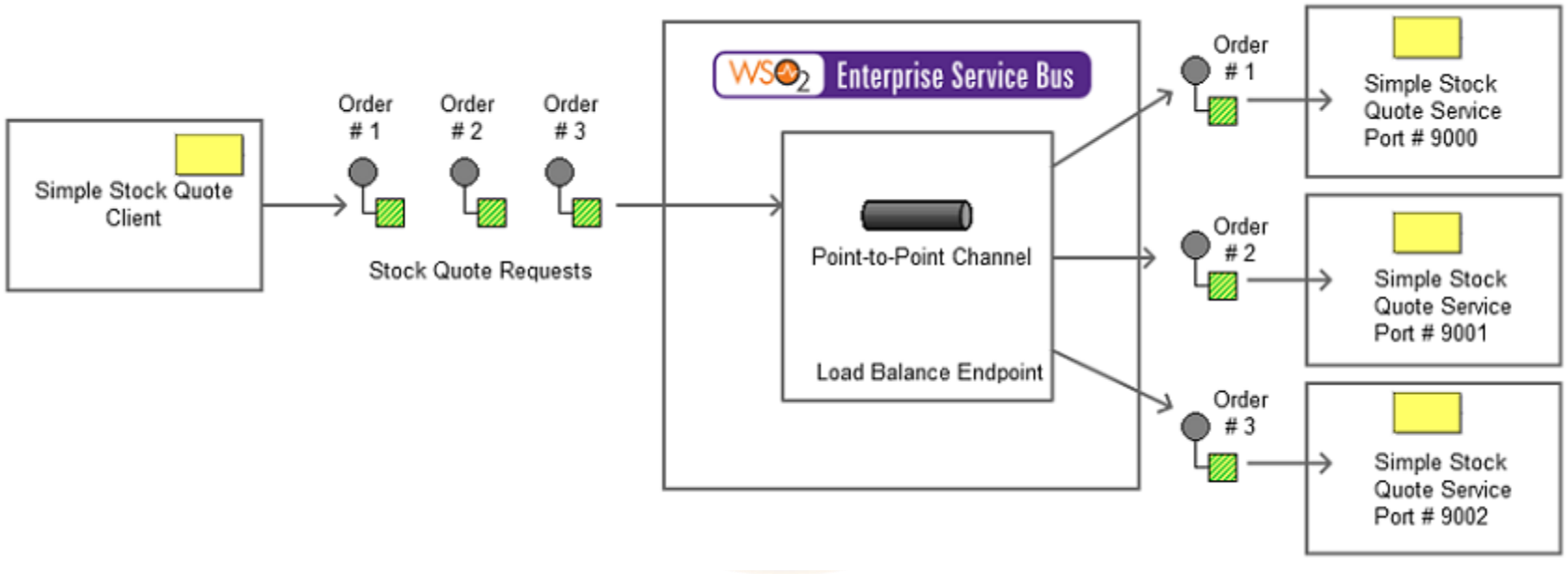
- The Point-to-Point Channel EIP ensures that only a single receiver consumes a message (when there are multiple receivers waiting to consume the message).





Message Channel

Point to Point Channel – Example



Message Channel

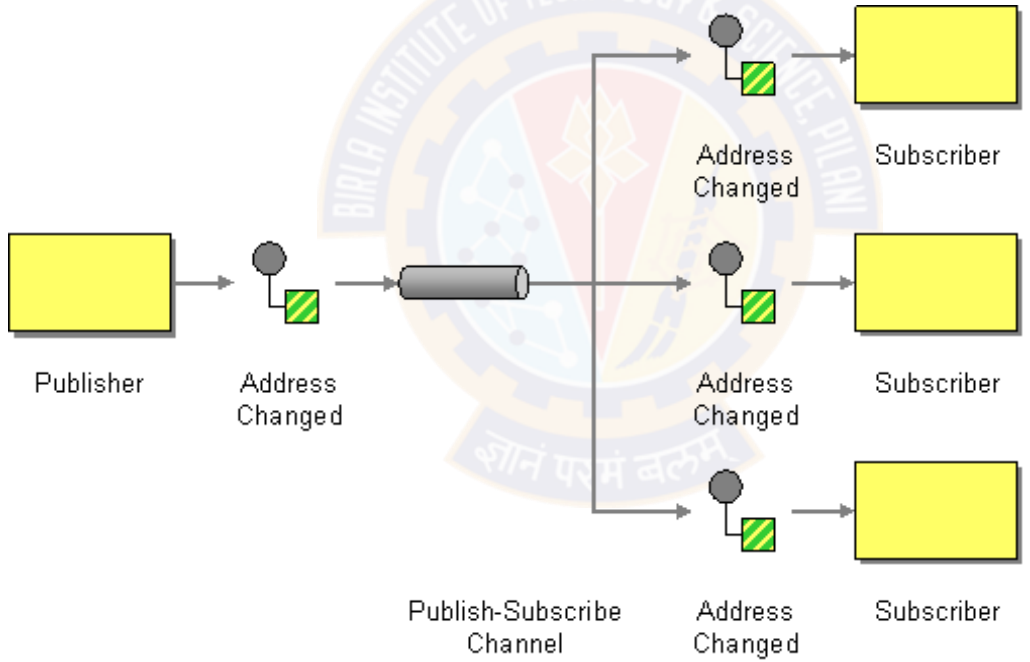
Point to Point Channel

Point-to-Point EIP (Figure 1)	Point-to-Point Channel Example Scenario (Figure 2)
Sender	Stock Quote Client
Receiver	Three instances of the Stock Quote service
Point to Point Channel	Load-balance Endpoint
Order	Stock Quote Requests

Message Channel

Publish-Subscribe Channel

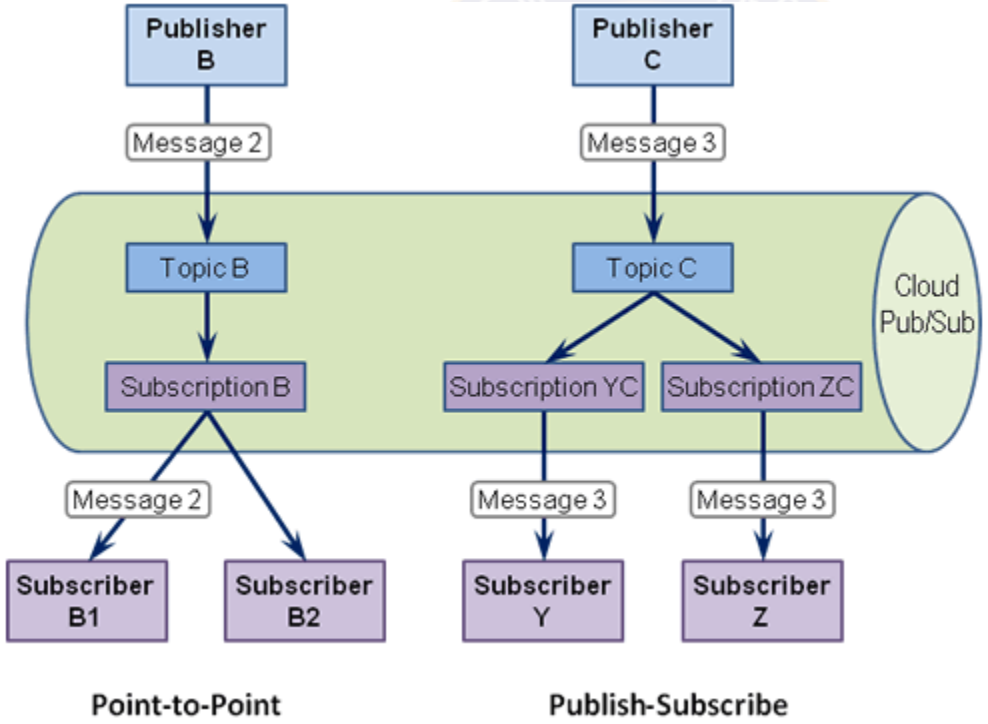
- The Publish-Subscribe Channel EIP receives messages from the input channel, and then splits and transmits them among its subscribers through the output channel. Each subscriber has only one output channel.



Message Channel

Publish-Subscribe Channel - Example

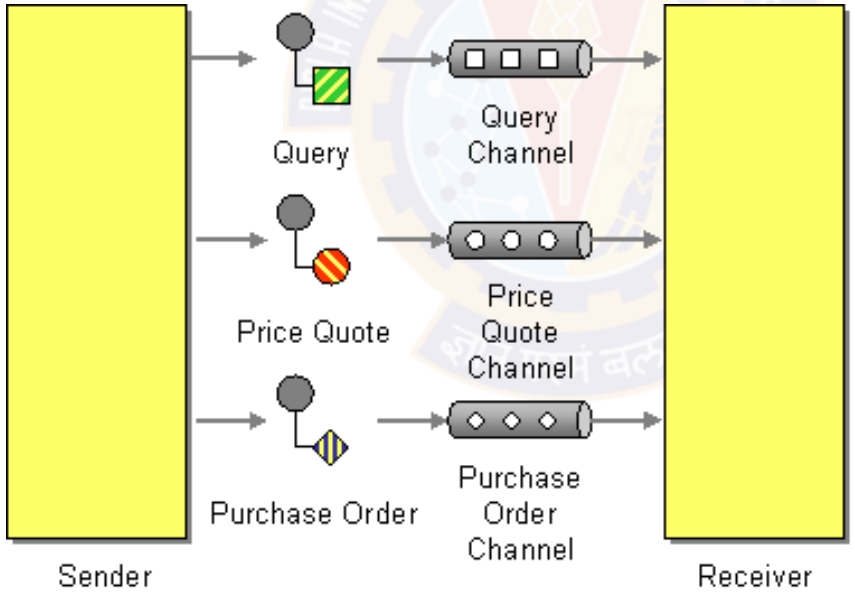
- In this example, both *Subscriber Y* and *Subscriber Z* each receive a copy of *Message 3* as they are subscribing to the same *Topic C*, but through separate subscriptions.



Message Channel

Datatype Channel

- This EIP creates a separate channel for each type of data so that all the messages on a given channel will contain the same data type. The sender, who knows the data type, should select the appropriate channel on which to send the message. The receiver identifies the type of data a message contains, based on the channel in which it is received.

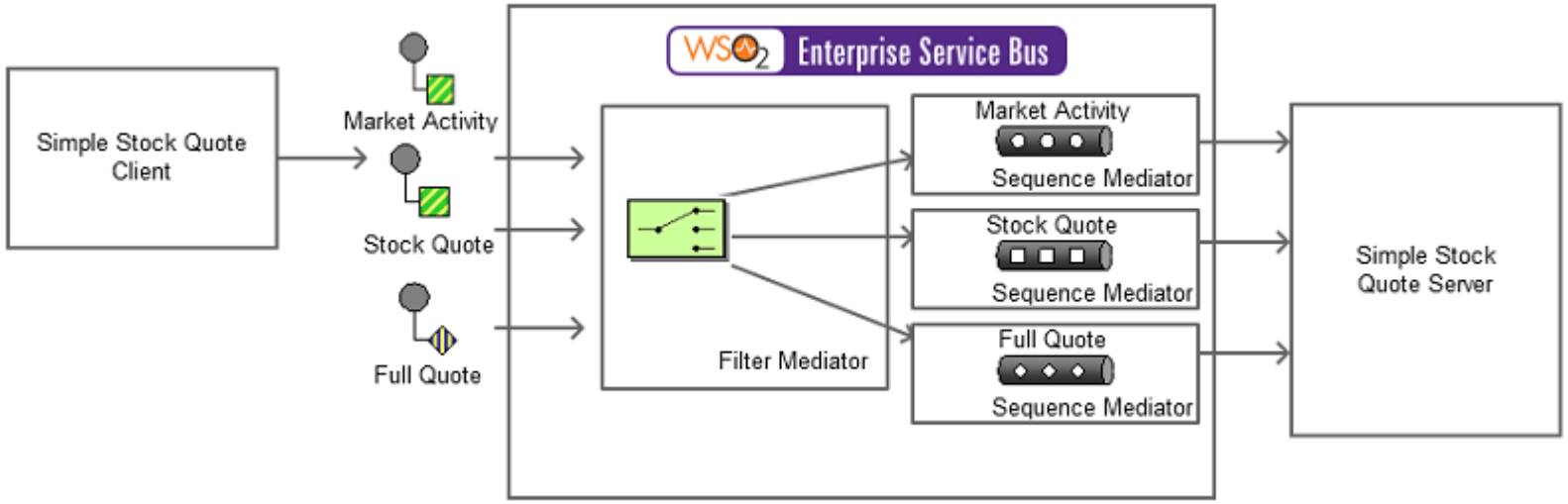


This example scenario depicts a Stock Quote service deployed in Axis2 server. It offers several service operations to the user. The ESB uses the filter mediator to identify each actio

Message Channel

Datatype Channel - Example

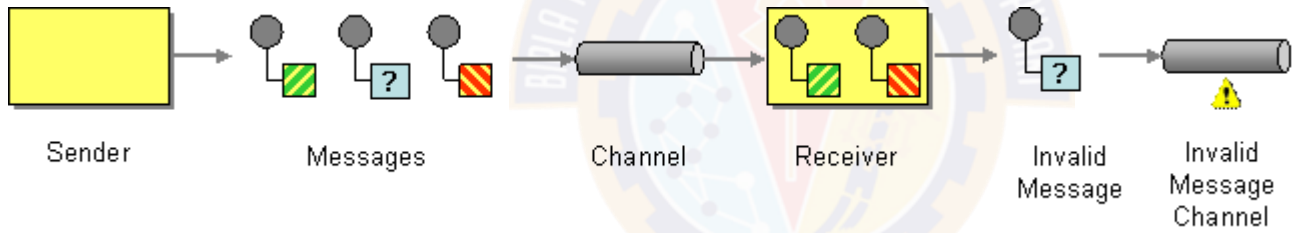
- This EIP creates a separate channel for each type of data so that all the messages on a given channel will contain the same data type. The sender, who knows the data type, should select the appropriate channel on which to send the message. The receiver identifies the type of data a message contains, based on the channel in which it is received.



Message Channel

Invalid Message Channel

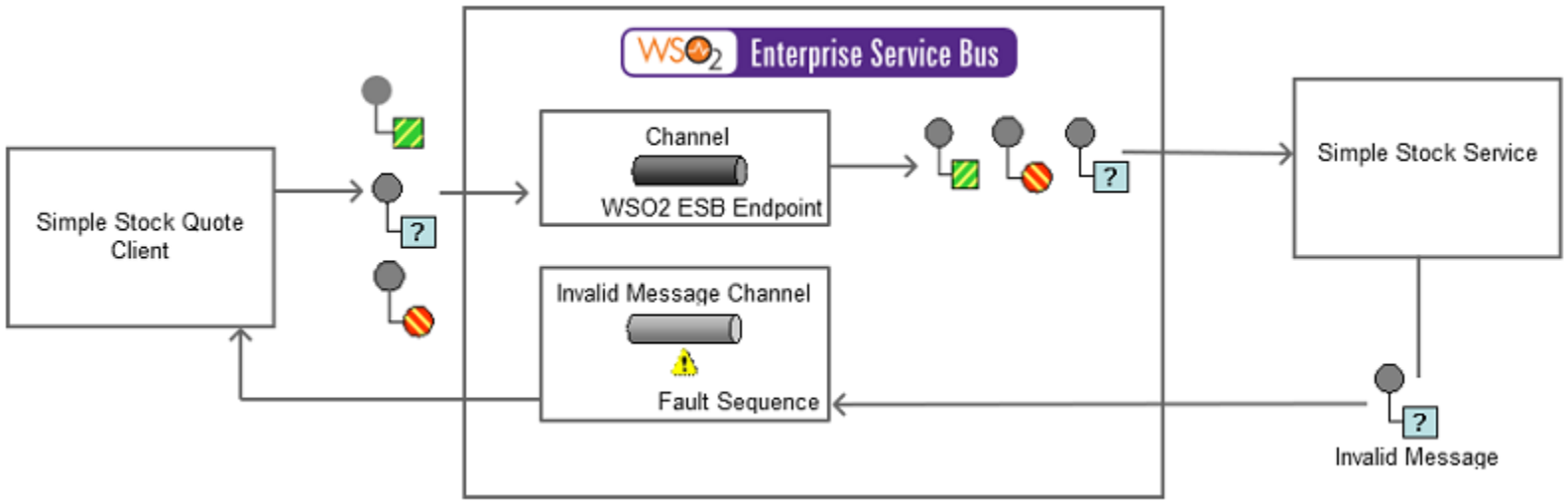
- The Invalid Message Channel EIP pattern allows administrators to define an error indication that appears when an endpoint fails to process a request.



Message Channel

Invalid Message Channel - Example

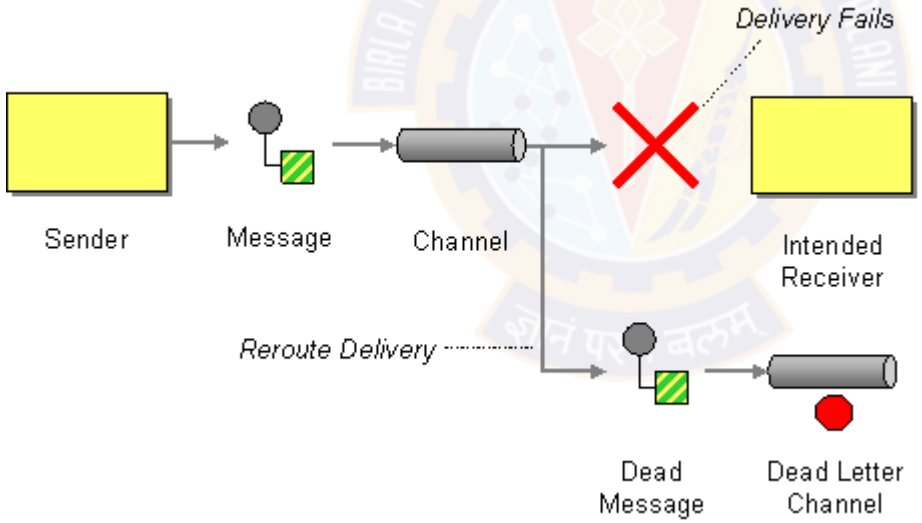
- The example scenario creates a deliberate error situation to demonstrate how the ESB handles errors on message failures.



Message Channel

Dead Letter Channel

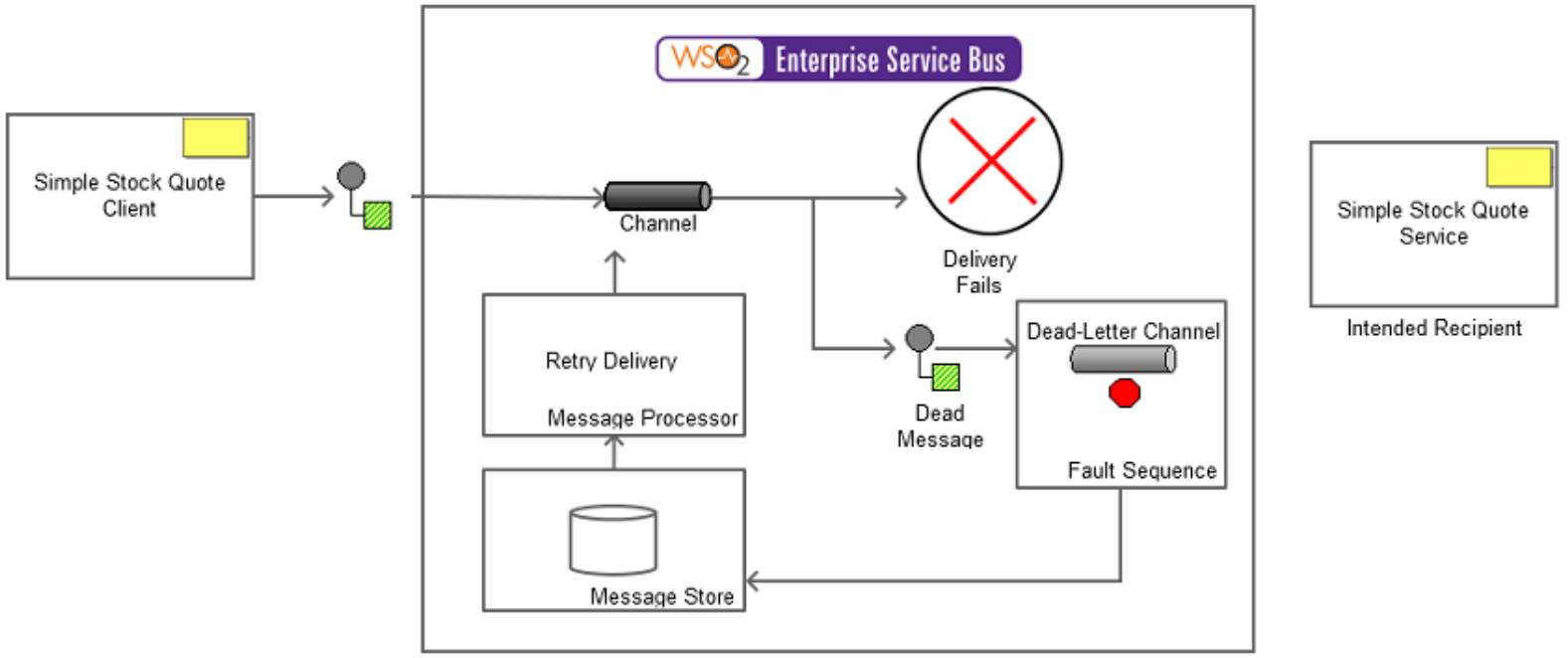
- The Dead Letter Channel (DLC) EIP outlines how messaging systems can handle messages that cannot be delivered to the recipient. Due to system/network failures or failures at the recipient's end, messages sometimes do not get delivered to the target. In such cases, the messaging system can deliver the message to a DLC.



Message Channel

Dead Letter Channel - Example

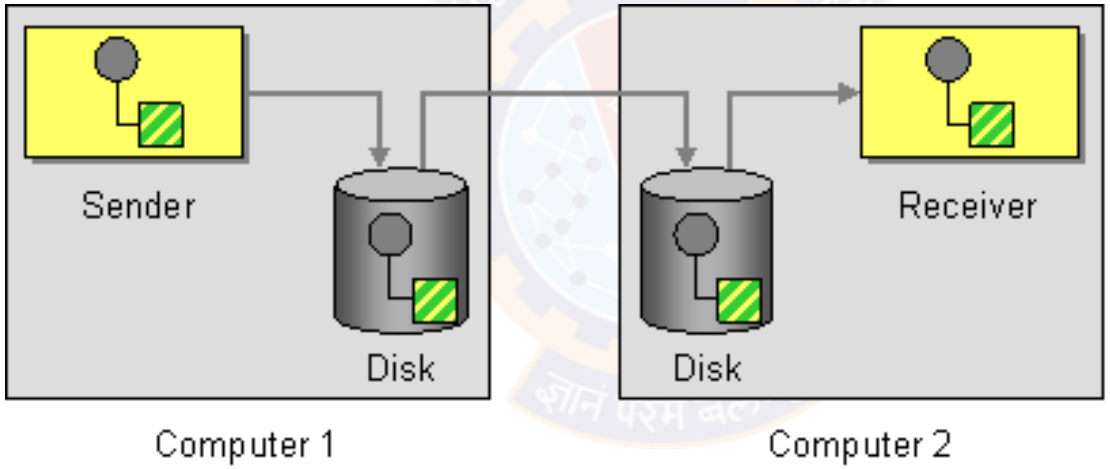
- This example takes a proxy service called StockQuoteProxy, which fronts a service by the name SimpleStockQuoteService. As long as the SimpleStockQuoteService is running, the clients calling StockQuoteProxy service got response. But if the SimpleStockQuoteService is down or a failure occurs while trying to send message to the SimpleStockQuoteService, the faultsequence will get involved and the message will be delivered to the dead letter channel.



Message Channel

Guaranteed Delivery Channel

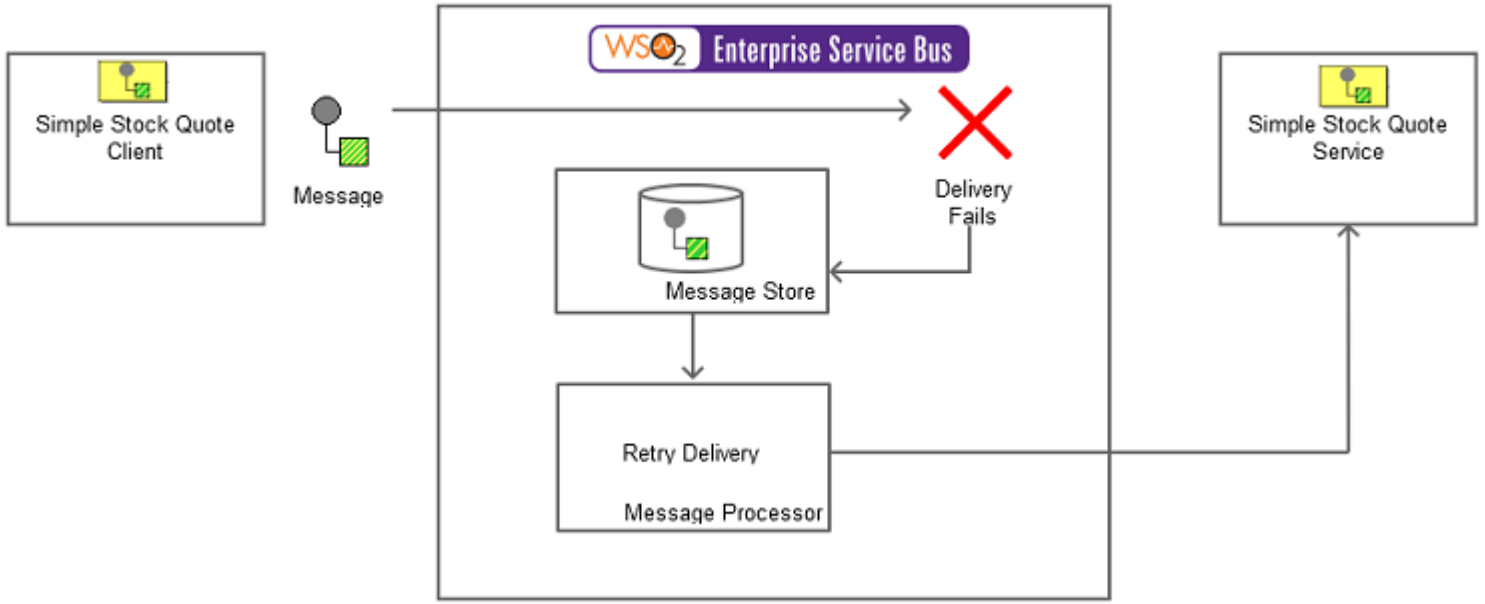
- The Guaranteed Delivery EIP ensures safe delivery of a message by storing it locally and transmitting it to the receiver's data store. Even when the receiver is offline, the EIP ensures that the message goes through when the receiver comes online.



Message Channel

Guaranteed Delivery Channel - Example

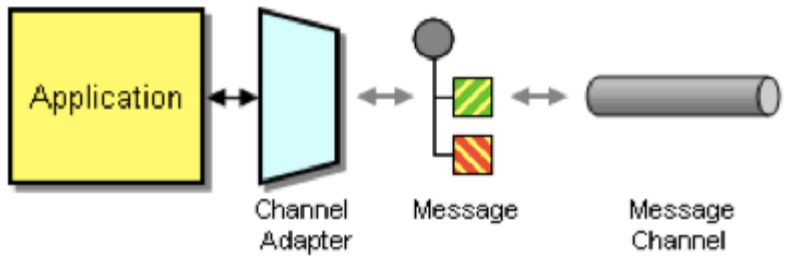
- This example is a stock quote service where a stock quote request is sent to a specific endpoint when the receiver is offline. An Axis2 server acts as the receiver. The ESB stores the request message in a JMS message store provided by the ESB. In this scenario, a Message Broker acts as the JMS message store. Here, the Message Broker can either be the WSO2 Message Broker or ActiveMQ.



Message Channel

Channel Adapter Channel

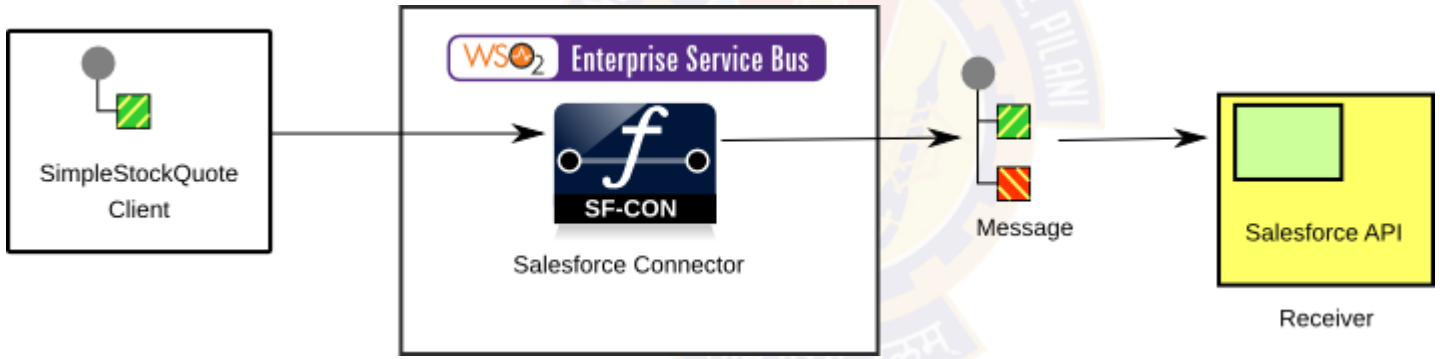
- Channel Adapter accesses an application's API or data and publishes messages on a channel based on this data. Also, it receives messages and invokes functionality inside the application.



Message Channel

Channel Adapter Channel - Example

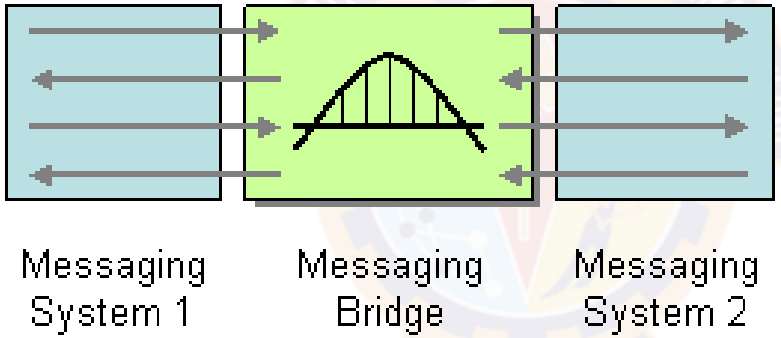
- This example demonstrates WSO2 ESB's Salesforce connector transferring a message coming from a stock quote client to Salesforce API and then sends the queried response that comes from Salesforce back to the client.



Message Channel

Messaging Bridge Channel

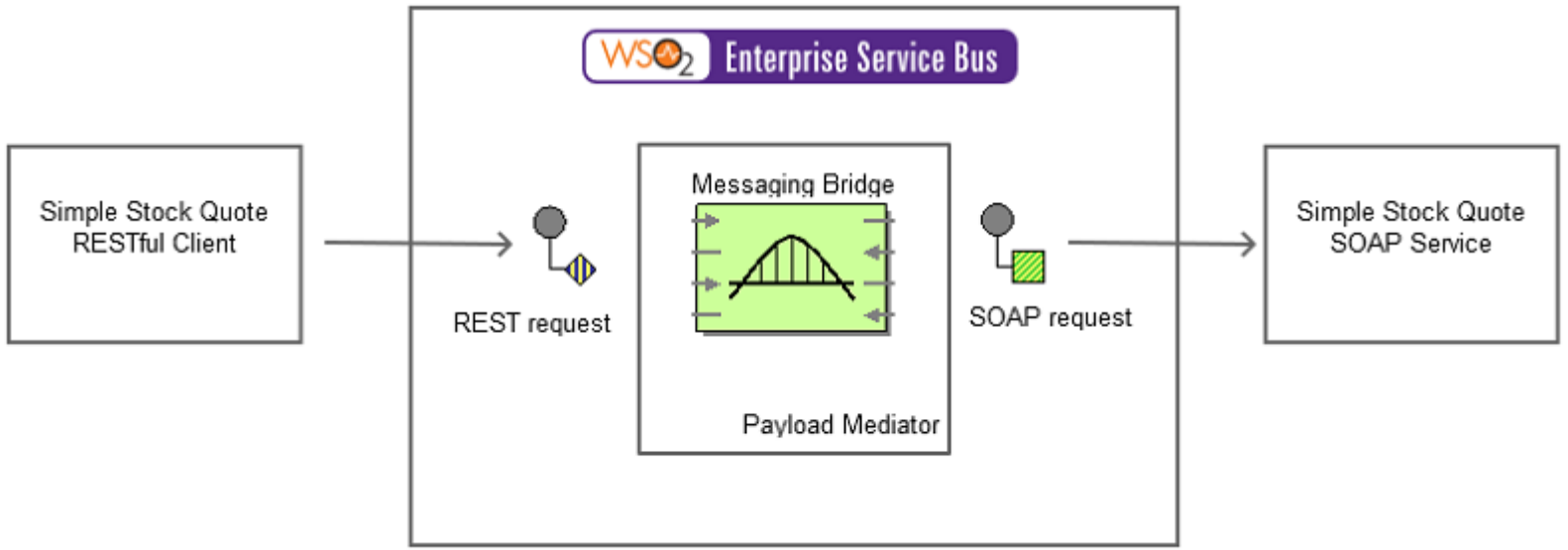
- The Messaging Bridge EIP facilitates the connection between messaging systems and replicates messages between them by transforming message formats from one system to the other.



Message Channel

Messaging Bridge Channel - Example

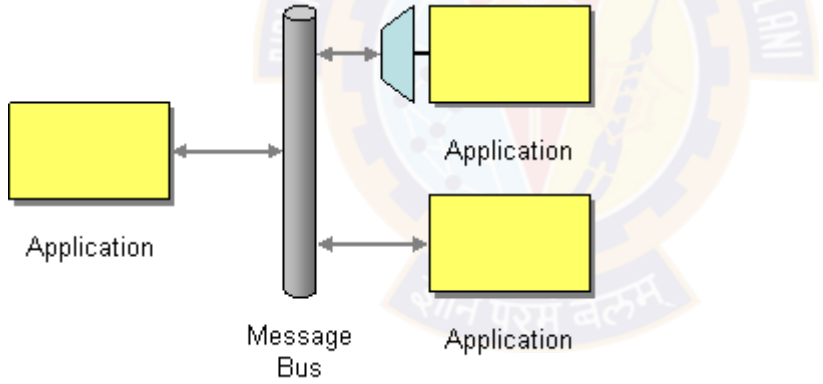
- This example illustrates the ESB as a message bridge, which accepts a REST message from the client, transforms the REST message to SOAP format, and sends the SOAP message to the back-end Axis2 server.



Message Channel

Messaging Bus Channel

- The Message Bus EIP enables separate applications to work together in a decoupled manner so that applications can be easily added or removed without affecting each other. This approach makes maintenance and testing smoother, since editing or removing an application will not affect the functionality of any other application.



Thank You!

In our next session:

Message Routing