redhat

Enterprise Integration Patterns Flashcards

Jboss.org/products/fuse

The design patterns defined in this deck are based on the book Enterprise Integration Patterns[1] which documents the authors combined experience in the integration space and created this notation, which has since been adopted as the standard for describing messaging solutions.

www.eaipatterns.com

1 Hohpe, Gregory. Woolf, Bobb Enterprise Integration Patterns. Addison-Wesley Professional, 2003.

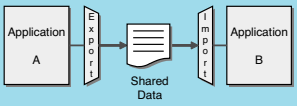
Get the tool: jboss.org/products/fuse

Apache Camel is a popular open source integration framework that makes it easy for developers to implement Enterprise Integration Patterns.

http://camel.apache.org

Get the tool: jboss.org/products/fuse

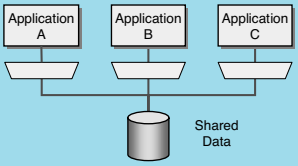
File Transfer



Have each application produce files that contain the information the other applications must consume. Integrators take the responsibility of transforming files into different formats. Produce the files at regular intervals according to the nature of the business.

Get the tool: jboss.org/products/fuse

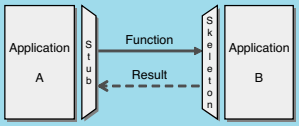
Shared Database



Integrate applications by having them store their data in a single Shared Database, and define the schema of the database to handle all the needs of the different applications.

Get the tool: jboss.org/products/fuse

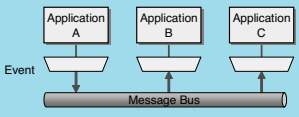
Remote Procedure Invocation



Develop each application as a large-scale object or component with encapsulated data. Provide an interface to allow other applications to interact with the running application.

Get the tool: jboss.org/products/fuse

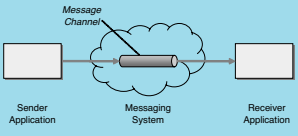
Messaging



Use Messaging to transfer packets of data frequently, immediately, reliably, and asynchronously, using customizable formats.

Get the tool: jboss.org/products/fuse

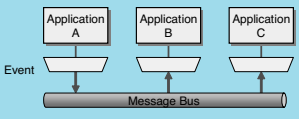
Message Channel



Connect the applications using a Message Channel, where one application writes information to the channel and the other one reads that information from the channel.

Get the tool: jboss.org/products/fuse

Message



Package the information into a Message, a data record that the messaging system can transmit through a Message Channel.

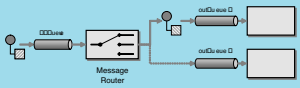
Get the tool: jboss.org/products/fuse

Pipes and Filters

Pipes and Filters describes a fundamental architectural style for messaging systems: Individual processing steps (filters) are chained together through the messaging channels (pipes).

Get the tool: jboss.org/products/fuse

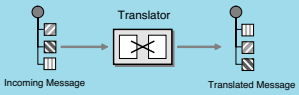
Message Router



Insert a special filter, a Message Router, which consumes a Message from one Message Channel and republishes it to a different Message Channel, depending on a set of conditions.

Get the tool: jboss.org/products/fuse

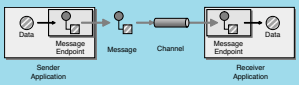
Message Translator



Use a special filter, a Message Translator, between other filters or applications to translate one data format into another.

Get the tool: jboss.org/products/fuse

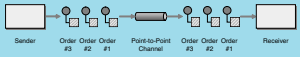
Message Endpoint



The Message Endpoint encapsulates the messaging system from the rest of the application and customizes a general messaging API for a specific application and task.

Get the tool: jboss.org/products/fuse

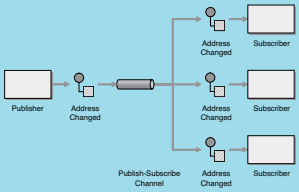
Point-to-Point Channel



A Point-to-Point Channel ensures that only one receiver consumes any given message. The channel can have multiple receivers that can consume multiple messages concurrently, but only one of them can successfully consume a particular message.

Get the tool: jboss.org/products/fuse

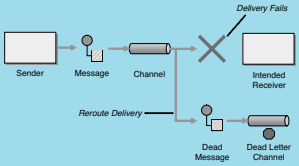
Publish-Subscribe Channel



Send the event on a Publish-Subscribe Channel, which delivers a copy of a particular event to each receiver.

Get the tool: jboss.org/products/fuse

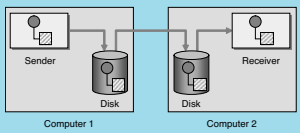
Dead Letter Channel



When a messaging system determines that it cannot or should not deliver a message, it may elect to move the message to a Dead Letter Channel.

Get the tool: jboss.org/products/fuse

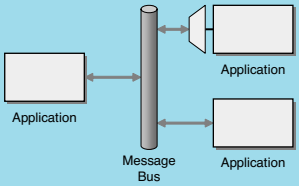
Guaranteed Delivery



The messaging system uses local datastores to persist messages. A send operation cannot complete successfully until the message is stored in the sender's local datastore. A message cannot be deleted from one datastore until it is forwarded and stored in the next datastore.

Get the tool: jboss.org/products/fuse

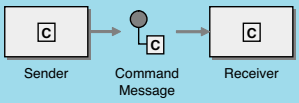
Message Bus



A Message Bus is a combination of a Canonical Data Model, a common command set, and a messaging infrastructure to allow different systems to communicate through a shared set of interfaces.

Get the tool: jboss.org/products/fuse

Command Message

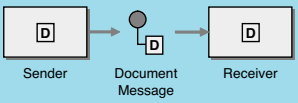


C = getLastTradePrice('DIS');

Use a Command Message to reliably invoke a procedure in another application.

Get the tool: jboss.org/products/fuse

Document Message

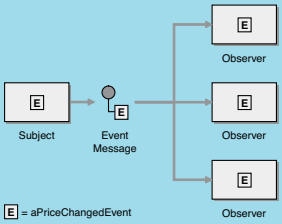


D = aPurchaseOrder

A Document Message just passes data and lets the receiver decide what, if anything, to do with the data. The data is a single unit of data, a single object or data structure that may decompose into smaller units.

Get the tool: jboss.org/products/fuse

Event Message

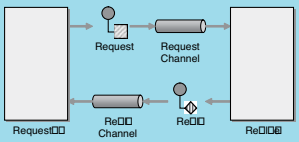


E = aPriceChangedEvent

Use an Event Message for reliable, asynchronous event notification between applications.

Get the tool: jboss.org/products/fuse

Request-Reply

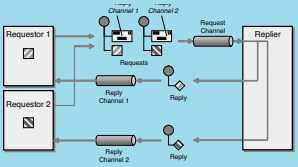


Request-Reply has two participants:

1. Requestor sends a request message and waits for a reply message.
2. Replier receives the request message and responds with a reply message.

Get the tool: jboss.org/products/fuse

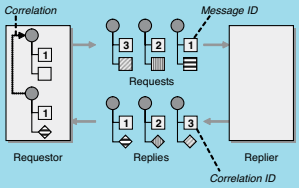
Return Address



A message's Return Address is analogous to the reply-to field in an e-mail message. The reply-to e-mail address is usually the same as the from address, but the sender can set it to a different address to receive replies in an account other than the one used to send the original message.

Get the tool: jboss.org/products/fuse

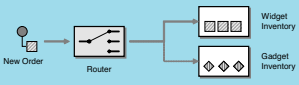
Correlation Identifier



A requestor assigns each request with a unique request ID. Replier's add the request ID to the reply so that the requestor can correlate the reply with the request that generated it.

Get the tool: jboss.org/products/fuse

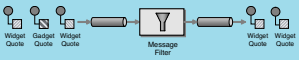
Content-Based Router



The Content-Based Router examines the message content and routes the message onto a different channel based on data contained in the message.

Get the tool: jboss.org/products/fuse

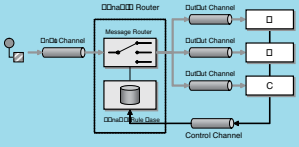
Message Filter



The Message Filter is a Message Router with a single output channel. If the content of an incoming message matches the criteria specified by the Message Filter, the message is routed to the output channel. If the message content does not match the criteria, the message is discarded.

Get the tool: jboss.org/products/fuse

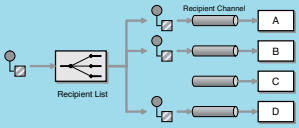
Dynamic Router



The Dynamic Router routes messages using a rule base that is generated based on input from potential message recipients. Communication between recipients and the router are done over a special control channel.

Get the tool: jboss.org/products/fuse

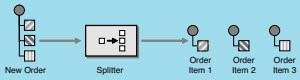
Recipient List



Define a channel for each recipient. Then use a Recipient List to inspect an incoming message, determine the list of desired recipients, and forward the message to all channels associated with the recipients in the list.

Get the tool: jboss.org/products/fuse

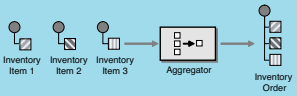
Splitter



Use a Splitter to break out the composite message into a series of individual messages, each containing data related to one item.

Get the tool: jboss.org/products/fuse

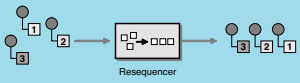
Aggregator



Use a stateful filter, an Aggregator, to collect and store individual messages until it receives a complete set of related messages. Then, the Aggregator publishes a single message distilled from the individual messages.

Get the tool: jboss.org/products/fuse

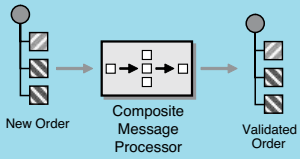
Resequencer



The Resequencer can receive a stream of messages that may not arrive in order. It stores out-of-sequence messages in an internal buffer until a complete sequence is obtained, and then publishes the messages to the output channel in the proper sequence.

Get the tool: jboss.org/products/fuse

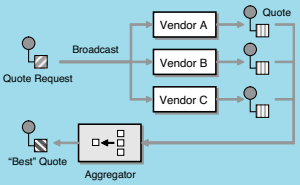
Composed Message Processor



Use a Composed Message Processor to process a composite message. The Composed Message Processor splits the message up, routes the submessages to the appropriate destinations, and reaggregates the responses back into a single message.

Get the tool: jboss.org/products/fuse

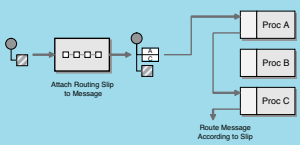
Scatter-Gather



Use a Scatter-Gather that broadcasts a message to multiple recipients and reaggregates the responses back into a single message.

Get the tool: jboss.org/products/fuse

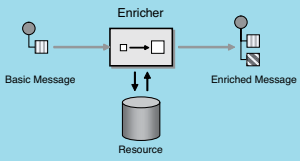
Routing Slip



Attach a Routing Slip to each message, specifying the sequence of processing steps. Wrap each component with a special message router that reads the Routing Slip and routes the message to the next component in the list.

Get the tool: jboss.org/products/fuse

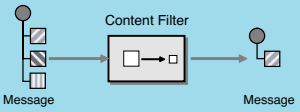
Content Enricher



The Content Enricher uses information inside the incoming message (e.g., key fields) to retrieve data from an external source. After the Content Enricher retrieves the required data from the resource, it appends the data to the message.

Get the tool: jboss.org/products/fuse

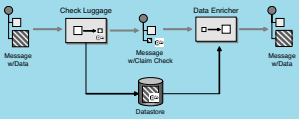
Content Filter



Use a Content Filter to remove unimportant data items from a message, leaving only important items.

Get the tool: jboss.org/products/fuse

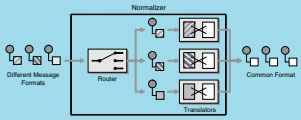
Claim Check



Store message data in a persistent store and pass a Claim Check to subsequent components. These components can use the Claim Check to retrieve the stored information.

Get the tool: jboss.org/products/fuse

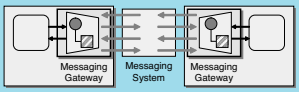
Normalizer



The Normalizer features one Message Translator for each message format and routes the incoming message to the correct Message Translator via a Message Router.

Get the tool: jboss.org/products/fuse

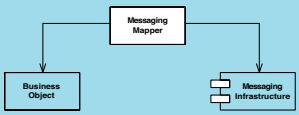
Messaging Gateway



The Messaging Gateway encapsulates messaging-specific code (e.g., the code required to send or receive a message) and separates it from the rest of the application code.

Get the tool: jboss.org/products/fuse

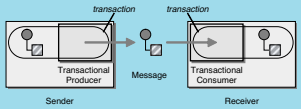
Messaging Mapper



The Messaging Mapper accesses one or more domain objects and converts them into a message as required by the messaging channel. It also performs the opposite function, creating or updating domain objects based on incoming messages.

Get the tool: jboss.org/products/fuse

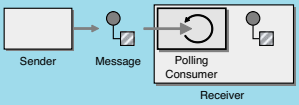
Transactional Client



Use a Transactional Client—make the client's session with the messaging system transactional so that the client can specify transaction boundaries.

Get the tool: jboss.org/products/fuse

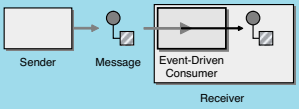
Polling Consumer



A Polling Consumer is an object that an application uses to receive messages by explicitly requesting them. When the application is ready for another message, it polls the consumer, which in turn gets a message from the messaging system and returns it.

Get the tool: jboss.org/products/fuse

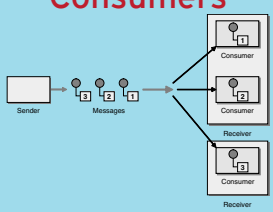
Event-Driven Consumer



An Event-Driven Consumer is an object that is invoked by the messaging system when a message arrives on the consumer's channel. The consumer passes the message to the application through a callback in the application's API.

Get the tool: jboss.org/products/fuse

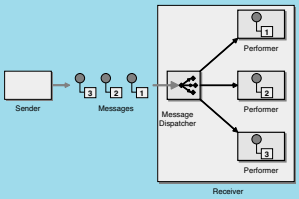
Competing Consumers



Competing Consumers are multiple consumers that are all created to receive messages from a single Point-to-Point Channel. When the channel delivers a message, any of the consumers could potentially receive it.

Get the tool: jboss.org/products/fuse

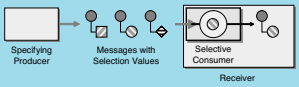
Message Dispatcher



When a Message Dispatcher receives a message, it obtains a performer and dispatches the message to the performer to process it.

Get the tool: jboss.org/products/fuse

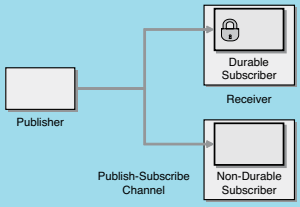
Selective Consumer



When a message arrives, a Selective Consumer tests the message's selection value to see if the value meets the consumer's selection criteria. If so, the consumer receives the message and passes it to the application for processing.

Get the tool: jboss.org/products/fuse

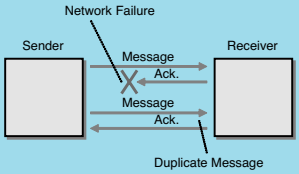
Durable Subscriber



Use a Durable Subscriber to make the messaging system save messages published while the subscriber is disconnected.

Get the tool: jboss.org/products/fuse

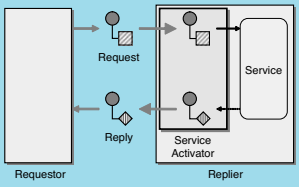
Idempotent Receiver



Design a receiver to be an Idempotent Receiver, one that can safely receive the same message multiple times.

Get the tool: jboss.org/products/fuse

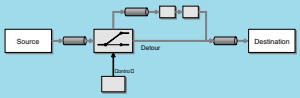
Service Activator



A Service Activator handles all of the messaging details and invokes the service like any other client, such that the service doesn't even know it's being invoked through messaging.

Get the tool: jboss.org/products/fuse

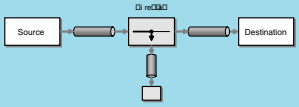
Detour



Construct a Detour with a Context-Based Router controlled via the Control Bus. In one state, the router routes incoming messages through additional steps, while in the other it routes messages directly to the destination channel.

Get the tool: jboss.org/products/fuse

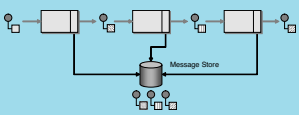
Wire Tap



The Wire Tap (also known as tee) is a fixed Recipient List with two output channels. It consumes messages off the input channel and publishes the unmodified message to both output channels.

Get the tool: jboss.org/products/fuse

Message Store



Use a Message Store to capture information about each message in a central location.

Get the tool: jboss.org/products/fuse

redhat.

Red Hat is the leader in open source integration and messaging products.

Please visit
redhat.com

#10611447_V1_0313