FUSE Mediation Router™

The Open Source Solution for Routing and Mediation

Organizations need to integrate applications quickly, and many developers would like to take a code-first approach without having to work with domain-specific or platform-specific deployment details. With the FUSE Mediation Router, a Java developer can quickly develop integration patterns by using a simple API directly within a Java program. These integration patterns, or integration components, can also be deployed in a number of deployment platforms, including the FUSE ESB, providing a seamless migration path to a more formal SOA infrastructure.

FUSE Mediation Router, IONA's distribution of Apache Camel, is a powerful rule-based routing and process mediation engine that combines the ease of basic POJO development with the clarity of the standard Enterprise Integration Patterns. It can be deployed inside any container or be used stand-alone, and works directly with any kind of transport or messaging model to rapidly integrate existing services and applications.

FUSE: A Family of Open Source SOA Components

FUSE Mediation Router is one component in a family of components that include FUSE HQ, FUSE ESB, the FUSE Services Framework, and the FUSE Message Broker. The FUSE components are tested together, certified, and supported to combine the speed and innovation of open source software with the reliability and expertise of commercially provided enterprise services.

About IONA Technologies, Inc.

IONA's commitment to open source software is part of its 15-year heritage of solving the most complex integration problems by applying open, standards-based solutions. An industry leader in integration and SOA, IONA has the proven expertise to design a highly flexible, distributed SOA infrastructure for Global 2000 customers including Raymond James & Associates, Nokia, Zurich Insurance, Ericsson and Credit Suisse using standardized components.

Why Routing?

- **Ease of Use**
  FUSE Mediation Router uses a Java Domain Specific Language (DSL) in addition to Spring XML for configuring routing rules to create powerful and concise code and support type-safe smart completion, minimizing the need to work with large numbers of XML configuration files.
- **Rapid Prototyping**
  Developers do not need to know deployment or container configuration details. The FUSE Mediation Router provides a layer of abstraction to mask the details of JMS, JBI, and JAX-WS, and provides a Maven archetype so organizations can quickly use Enterprise Integration Patterns to integrate complex systems.
- **Interoperability**
  FUSE Mediation Router uses generics, annotations and Uniform Resource Identifiers (URIs) to easily work directly with any kind of transport or messaging model including HTTP, JMS, JBI, SCA, MINA or CXF Bus API without mandating a normalized message API, which can often lead to leaky abstractions.
- **Distributed**
  Like all FUSE components, the Mediation Router is a lightweight, small-footprint solution that can be easily embedded at endpoints, allowing distributed systems to intelligently interact without mandating a centralized server.

Get the leading open source mediation router now:

FUSE Mediation Router

open.IONA.com

IONA

Making Software Work Together™

Supported Enterprise Integration Patterns

The following Enterprise Integration Patterns are defined by Gregor Hohpe and Bobby Woolf.
For more information visit **http://www.enterpriseintegrationpatterns.com.**

Messaging Systems

Message Channel
How one application communicates with another using messaging

Message
How two applications connected by a message channel exchange a piece of information

Pipes and Filters
How to perform complex processing on a message while maintaining independence and flexibility

Message Router
How to decouple individual processing steps so that messages can be passed to different filters depending on a set of conditions

Message Translator
How systems using different data formats communicate with each other using messaging

Message Endpoint
How an application connects to a messaging channel to send and receive messages

Messaging Channels

Point to Point Channel
How the caller is sure that exactly one receiver will receive the document or perform the call

Publish Subscribe Channel
How the sender broadcasts an event to all interested receivers

Dead Letter Channel
What the messaging system does with a message it cannot deliver

Guaranteed Delivery
How the sender makes sure that a message will be delivered, even if the messaging system fails

Message Bus
An architecture that enables separate applications to work together, but in a de-coupled fashion such that applications can be easily added or removed without affecting the others

Message Routing

Content Based Router
How to handle a situation where the implementation of a single logical function is spread across multiple physical systems

Message Filter
How a component avoids receiving uninteresting messages

Recipient List
How to route a message to a list of dynamically specified recipients

Splitter
How to process a message if it contains multiple elements, each of which may have to be processed in a different way

Resequencer
How to get a stream of related but out-of-sequence messages back into the correct order

Message Transformation

Content Enricher
How to communicate with another system if the message originator does not have all the required data items available

Content Filter
How to simplify dealing with a large message, when interested only in a few data items

Normalizer
How to process messages that are semantically equivalent, but arrive in a different format

Messaging Endpoints

Messaging Mapper
How to move data between domain objects and the messaging infrastructure while keeping the two independent of each other

Event Driven Consumer
How an application automatically consumes messages as they become available

Polling Consumer
How an application consumes a message when the application is ready

Competing Consumers
How a messaging client processes multiple messages concurrently

Message Dispatcher
How multiple consumers on a single channel coordinate their message processing

Selective Consumer
How a message consumer selects which messages it wishes to receive

Durable Subscriber
How a subscriber avoids missing messages while not listening for them

Idempotent Consumer
How a message receiver deals with duplicate messages

Transactional Client
How a client controls its transactions with the messaging system

Messaging Gateway
How to encapsulate access to the messaging system from the rest of the application

Service Activator
How an application designs a service invoked both via various messaging technologies and via non-messaging techniques

System Management

Wire Tap
How to inspect messages that travel on a point-to-point channel

US Headquarters
IONA Technologies, Inc.
200 West Street Waltham, MA 02451 USA

European Headquarters
IONA Technologies PLC
The IONA Building
Shelbourne Road, Dublin 4, Ireland

Asia-Pacific Headquarters
Kioicho Bldg. 3-12 Kioicho,
Chiyoda-ku,
Tokyo 102-0094

To download FUSE visit
open.IONA.com/downloads

For more information visit
open.IONA.com

1-877-235-8491 (toll free)
1-310-437-4870 (direct)
opensource@iona.com

IONA

Making Software Work Together™

03854CJ03 12-07