



FUSE™ Product Family

**Getting Started with the FUSE™ Product Family**

**PROGRESS**  
SOFTWARE

# Getting Started with the FUSE<sup>™</sup> Product Family

Publication date 11 Aug 2009

Copyright © 2001-2009 Progress Software Corporation and/or its subsidiaries or affiliates.

## ***Legal Notices***

Progress Software Corporation and/or its subsidiaries may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this publication. Except as expressly provided in any written license agreement from Progress Software Corporation, the furnishing of this publication does not give you any license to these patents, trademarks, copyrights, or other intellectual property. Any rights not expressly granted herein are reserved.

Progress, IONA, IONA Technologies, the IONA logo, Orbix, High Performance Integration, Artix;, FUSE, and Making Software Work Together are trademarks or registered trademarks of Progress Software Corporation and/or its subsidiaries in the US and other countries.

Java and J2EE are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. CORBA is a trademark or registered trademark of the Object Management Group, Inc. in the US and other countries. All other trademarks that appear herein are the property of their respective owners.

While the information in this publication is believed to be accurate Progress Software Corporation makes no warranty of any kind to this material including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Progress Software Corporation shall not be liable for errors contained herein, or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

All products or services mentioned in this manual are covered by the trademarks, service marks, or product names as designated by the companies who market those products.

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of Progress Software Corporation. No third-party intellectual property right liability is assumed with respect to the use of the information contained herein. Progress Software Corporation assumes no responsibility for errors or omissions contained in this publication. This publication and features described herein are subject to change without notice. Portions of this document may include Apache Foundation documentation, all rights reserved.

# Table of Contents

<b>I. Introduction to the FUSE Product Family</b>	<b>7</b>
<b>1. Why Use Integration?</b>	<b>9</b>
<b>2. Why Use Open Source Software?</b>	<b>11</b>
<b>3. What is FUSE?</b>	<b>13</b>
Overview	14
FUSE ESB	15
FUSE Message Broker	16
FUSE Services Framework	17
FUSE Mediation Router	18
FUSE Integration Designer	19
FUSE HQ	20
<b>4. Which FUSE Products Should I Use?</b>	<b>21</b>
What Are the FUSE Development Models?	22
What Should I Download?	25
How Do I Run My Application?	26
<b>II. Logisticx FUSE Demo</b>	<b>29</b>
<b>5. Introduction to the Demo</b>	<b>31</b>
<b>6. Installing the Demo</b>	<b>33</b>
<b>7. Running the Demo</b>	<b>39</b>
<b>8. Understanding the Demo</b>	<b>47</b>
Demo Services and Applications	48
Sequence of Events	49
Client, Server, and Supply Side Integration	50
<b>9. Further Reading</b>	<b>51</b>
Index	53



# List of Figures

7.1. Initial Logisticx FUSE Demo Web Page .....	40
8.1. Logisticx Services and Applications .....	48
8.2. Client, Server, and Supply Side Integration .....	50



# Part I. Introduction to the FUSE Product Family

*The FUSE Product Family combines the best available open source software for developing and deploying applications in an integration infrastructure. All of the FUSE components are developed in the Apache open source community and are flexible, easy to use, and lightweight.*

<b>1. Why Use Integration?</b> .....	<b>9</b>
<b>2. Why Use Open Source Software?</b> .....	<b>11</b>
<b>3. What is FUSE?</b> .....	<b>13</b>
Overview .....	14
FUSE ESB .....	15
FUSE Message Broker .....	16
FUSE Services Framework .....	17
FUSE Mediation Router .....	18
FUSE Integration Designer .....	19
FUSE HQ .....	20
<b>4. Which FUSE Products Should I Use?</b> .....	<b>21</b>
What Are the FUSE Development Models? .....	22
What Should I Download? .....	25
How Do I Run My Application? .....	26





# Chapter 1. Why Use Integration?

*The FUSE Product Family provides open source software and tooling to develop and manage a robust, enterprise integration infrastructure that enables communication among disparate applications.*

## Reasons for integrating systems and applications

The main reasons for developing an integration infrastructure include:

- Handles various technologies, data formats, protocols, and application versions in a heterogeneous environment
- Provides a flexible IT environment to facilitate the delivery of new products
- Reduces the cost of managing and maintaining applications and integration connections
- Shares and transforms data

---

## Using messaging in integration

Messaging enables asynchronous communication and loosely coupled interfaces among applications. With message based integration, applications do not communicate with each other directly. Instead, they send and receive messages asynchronously through a message broker. For more information, see ["JMS messaging" on page 22](#) and ["FUSE Message Broker" on page 16](#).

---

## Using an ESB in integration

An enterprise service bus (ESB) combines loosely coupled interfaces and asynchronous interactions in its bus technology. Applications just have to plug into the bus, send data to the bus, and receive data from the bus. The bus delivers the data to the target application in its preferred data format. Applications can plug into and unplug from the bus without disrupting communication among other applications on the bus.

An ESB enables rapid prototyping and development of an infrastructure to connect applications to each other. It also solves integration challenges such as location transparency, transport protocol conversion, message routing, transformation, and enhancement. An ESB provides a highly distributed solution with strong qualities of service, such as clustering, load balancing, and security. ESBs use an extensible, loosely-coupled data model. With an ESB, applications handle changes flexibly, and systems can be upgraded incrementally and continuously.

An ESB can connect to various transports and technologies, including file-based, JMS messaging, JDBC database connectivity, mail-based (POP3 and SMTP), file transfer protocol (FTP), Enterprise JavaBeans (EJBs), and web service connectivity with SOAP. For more information, see ["FUSE ESB" on page 15](#).

---

### **Service oriented architecture**

An ESB is the layer of technology that makes *service oriented architecture* (SOA) possible. SOA is an integration methodology that uses loosely coupled, distributed services to make resources available to consumers in a standardized way. SOA services are small units of functionality that communicate by passing data in the form of messages. Each service has a well defined interface that is accessible across a distributed computing environment. SOA services are loosely coupled and reusable; changes to one service do not require changes to the whole system. With SOA, you define applications based on business requirements. The only implementation details you must define are the messages that the services exchange.

# Chapter 2. Why Use Open Source Software?

*Organizations are increasingly investing in open source software as part of their integration strategy to take advantage of the affordability, flexibility, and innovation provided by community-based development.*

## **Advantages of open source software**

Using open source software produces high-quality features faster than with traditional commercial models. Open source software is developed collaboratively and is owned by a large and diverse group of developers and users, enabling you achieve the efficiencies of integration without compromising your architecture or being locked into a single vendor. The source code is freely available, and users are encouraged and allowed to change, improve, and redistribute the software, subject to the terms of the open source license.

---

## **Apache open source software**

Apache open source software projects have a collaborative, consensus-based development process, a permissive open source software license, and a focus on creating high quality software that leads the way in its field.

Apache projects elect committers who are the only community members allowed to add code to a project and they must prove themselves to the community before they are elected. Because robustness and enterprise quality of service (QoS) are such high priorities, the process used to create Apache projects results in software that is appropriate for use in IT departments.

---

## **FUSE open source software**

The components in the FUSE Product Family are based on four Apache open source projects. The FUSE team employs many of the key committers on the four Apache projects, along with all of the project chairs. The FUSE team takes a stable and consistent snapshot of an Apache project and then creates a packaged code drop that goes through a standard release process.

The FUSE team performs extensive quality assurance tests on a broad range of platforms, and tests for backwards compatibility for each FUSE release. Testing is designed for typical enterprise configurations with long running processes, multiple clients, and multiple machines. The FUSE components are tested for interoperability, are certified, and are supported to combine the speed and innovation of open source software, with the reliability and expertise of commercially provided enterprise services.



# Chapter 3. What is FUSE?

*The FUSE Product Family is a collection of open source components and tools used to build, deploy, and monitor applications in an integration infrastructure.*

Overview .....	14
FUSE ESB .....	15
FUSE Message Broker .....	16
FUSE Services Framework .....	17
FUSE Mediation Router .....	18
FUSE Integration Designer .....	19
FUSE HQ .....	20

# Overview

## Introduction

The FUSE Product Family provides open source software components and tools to integrate enterprise applications. These products provide:

- Ease of use
  - Flexibility of deployment
  - Many connectivity options
  - Mix of features
  - Customizable alternatives
- 

## Open source components

FUSE includes four open source components based on Apache projects:

- **FUSE ESB** — Based on Apache ServiceMix (see ["FUSE ESB" on page 15](#))
- **FUSE Message Broker** — Based on Apache ActiveMQ (see ["FUSE Message Broker" on page 16](#))
- **FUSE Services Framework** — Based on Apache CXF (see ["FUSE Services Framework" on page 17](#))
- **FUSE Mediation Router** — Based on Apache Camel (see ["FUSE Mediation Router" on page 18](#))

All of the FUSE Product Family open source components use industry standards such as Java, SOAP, HTML, and JMS, making it easier for both the community and the users to add new features.

---

## Development and management tools

The FUSE Product Family also provides development and management tools to develop, deploy, and monitor an enterprise integration infrastructure:

- **FUSE Integration Designer** — Eclipse tooling for FUSE (see ["FUSE Integration Designer" on page 19](#))
- **FUSE HQ** — Based on Hyperic HQ Enterprise and included with any FUSE support subscription (see ["FUSE HQ" on page 20](#))

# FUSE ESB

## Description

FUSE ESB is an open, standards based integration platform based on [Apache ServiceMix](http://servicemix.apache.org)<sup>1</sup> and is used to deploy and manage services on an *enterprise service bus* (ESB). An ESB combines messaging, web services, data transformation, and intelligent routing to reliably connect and coordinate applications across an extended enterprise. An ESB handles the delivery of messages between services and provides management, monitoring, and mediation services such as routing, service discovery, and transaction processing.

---

## JBIs support

FUSE ESB supports the Java Business Integration (JBI) specification for packaging, deploying, and managing components deployed on an ESB. With JBI, plugin components interoperate through mediated message exchange. FUSE ESB manages and deploys the components developed using FUSE Services Framework and FUSE Mediation Router and other JBI-compliant components like BPEL. See ["Integration using JBI" on page 24](#).

---

## OSGi support

FUSE ESB 4 also supports the OSGi standard, which provides simplified deployment and mechanisms for controlling the life cycle of components, and managing dependencies between application components. With OSGi, you can dynamically update, replace, and remove components in a running system without shutting down the system. See [Using OSGi in FUSE ESB](#).<sup>2</sup>

---

## Usage in the Demo

The Logisticx FUSE Demo demonstrates the use of FUSE ESB as part of a business solution developed using the FUSE Product Family. In the Demo, services are deployed on FUSE ESB to provide order processing and inventory management capabilities to a supply-chain company with globally distributed customers. See [Logisticx FUSE Demo on page 29](#) for information on how FUSE ESB is used in the Demo along with instructions for running the Demo.

---

## More information

For more information, see the *Getting Started with FUSE ESB* guides and other FUSE ESB documentation for [FUSE ESB 3](#)<sup>3</sup> and [FUSE ESB 4](#)<sup>4</sup>.

---

<sup>1</sup> <http://servicemix.apache.org>

<sup>2</sup> <http://fusesource.com/documentation/fuse-esb-documentation/>

<sup>3</sup> <http://fusesource.com/documentation/fuse-esb-documentation/>

<sup>4</sup> <http://fusesource.com/products/enterprise-servicemix4/#documentation>

# FUSE Message Broker

## Description

Based on [Apache ActiveMQ](http://activemq.apache.org/)<sup>5</sup>, FUSE Message Broker is a flexible messaging platform for reliably executing transactions, moving data, efficiently scaling operations, and connecting processes across heterogeneous database and application environments. FUSE Message Broker is high performance middleware that loosely couples services in time and location. FUSE ESB uses FUSE Message Broker as its underlying messaging infrastructure for connectivity, reliable messaging, and JMX management and instrumentation.

---

## Capabilities

FUSE Message Broker fully implements the Java Message Service (JMS) 1.1 specification, providing full message broker capabilities, including Publish and Subscribe (PubSub) and Point to Point (PTP) messaging. FUSE Message Broker is standards-based, supporting JMS, J2EE, JNDI, AJAX, REST, and HTTP. See ["JMS messaging" on page 22](#).

FUSE Message Broker simplifies application development by providing:

- High performance and mission-critical reliability for distributed enterprise applications and systems
  - Durable, fault tolerant connections, clustering, and failover to ensure high availability
  - Reliable connectivity to remote clients and protection against transactions being abandoned or data being lost if a system is unavailable
  - Authentication and authorization, supports custom and third party solutions, and integrates with your existing security investments
- 

## Usage in the Demo

The Logisticx FUSE Demo demonstrates the use of FUSE Message Broker to send status messages to a client web application that provides a web interface for customers. (See [Logisticx FUSE Demo on page 29](#).)

---

## More information

For more information, see *Exploring JMS with FUSE Message Broker*, *Getting Started with FUSE Message Broker*, and other [FUSE Message Broker documentation](http://fusesource.com/documentation/fuse-message-broker-documentation/)<sup>6</sup>.

---

<sup>5</sup> <http://activemq.apache.org/>

<sup>6</sup> <http://fusesource.com/documentation/fuse-message-broker-documentation/>



# FUSE Services Framework

## Description

FUSE Services Framework is an open source Java web services framework based on [Apache CXF](http://cxf.apache.org/)<sup>7</sup>. FUSE Services Framework provides design-time tools and a technology-neutral runtime infrastructure for creating and integrating services across heterogeneous environments. Use FUSE Services Framework to service-enable new and legacy applications in an enterprise integration infrastructure.

---

## Capabilities

FUSE Services Framework is a small footprint engine for creating reusable services as part of an integration solution. Java developers can use JAX-WS, JAX-RS, JavaScript, or plain old Java objects (POJOs) to create web services and RESTful services. FUSE Services Framework supports multiple protocols including SOAP, XML, HTTP and RESTful HTTP, and it works over a variety of transports such as HTTP/S and JMS. See ["Developing services" on page 22](#).

---

## Usage in the Demo

The Logisticx FUSE Demo demonstrates the use of FUSE Services Framework to implement external web services as part of a business solution created with the FUSE Product Family. In the Demo, external warehouse web services determine and return order status. See [Logisticx FUSE Demo on page 29](#) for information on how FUSE Services Framework is used in the Demo, and for instructions for running the Demo.

---

## More information

For more information, see *Getting Started Developing Services* and other [FUSE Services Framework documentation](#)<sup>8</sup>.

---

<sup>7</sup> <http://cxf.apache.org/>

<sup>8</sup> <http://fusesource.com/documentation/fuse-service-framework-documentation/>

# FUSE Mediation Router

## Description

FUSE Mediation Router is a powerful open source framework based on [Apache Camel](#)<sup>9</sup>. It uses routing rules to mediate between services. Java developers can quickly implement [Enterprise Integration Patterns](#)<sup>10</sup> (EIPs) using a code-first approach and plain old Java objects (POJOs). See ["Integration using EIPs" on page 23](#).

---

## Capabilities

With FUSE Mediation Router you can implement both routing rules and mediation in a Java-based Domain Specific Language (DSL) (or Fluent API), using Spring-based XML configuration files, or using the Scala DSL. FUSE Mediation Router provides a layer of abstraction to hide the details of JMS, JBI, and JAX-WS, and provides Maven archetypes to implement EIPs for integrating complex systems.

FUSE Mediation Router has a small library with minimal dependencies, making it easy to embed in any Java application. Since FUSE Mediation Router is lightweight and has a small footprint, you can embed it at endpoints, allowing distributed systems to intelligently interact without a centralized server. Use FUSE Mediation Router directly with any kind of transport or messaging model to rapidly integrate existing services and applications.

---

## Usage in the Demo

The Logisticx FUSE Demo demonstrates the use of FUSE Mediation Router to route messages to the appropriate services based on message content. See [Logisticx FUSE Demo on page 29](#) for information on how FUSE Mediation Router is used in the Demo and the instructions for running the Demo.

---

## More information

For more information, see *Getting Started with FUSE Mediation Router* and other [FUSE Mediation Router documentation](#)<sup>11</sup>.

---

<sup>9</sup> <http://camel.apache.org/index.html>

<sup>10</sup> <http://camel.apache.org/enterprise-integration-patterns.html>

<sup>11</sup> <http://fusesource.com/documentation/fuse-mediation-router-documentation/>

# FUSE Integration Designer

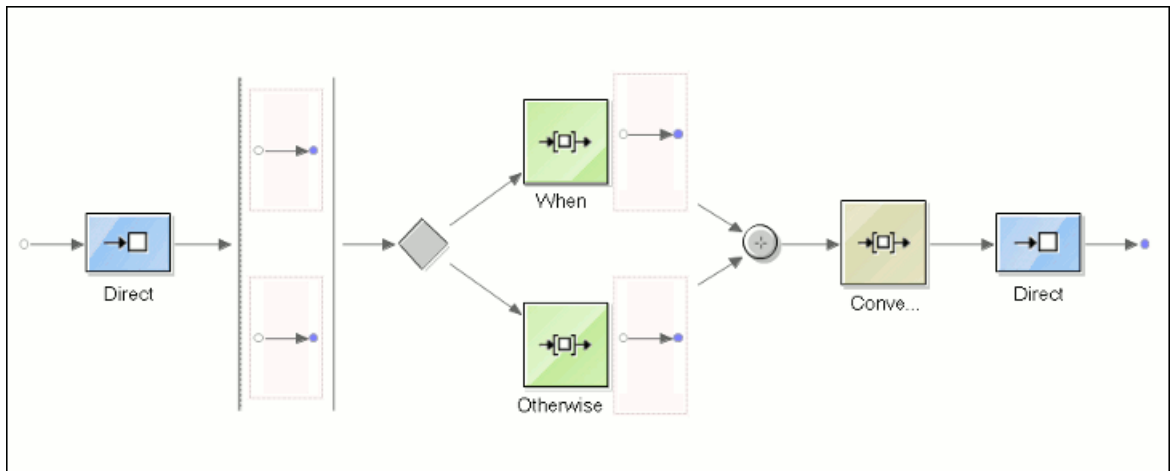
## Description

FUSE Integration Designer is Eclipse tooling for creating and integrating web services using Enterprise Integration Patterns (EIPs). It includes a canvas for visualizing, creating, and debugging EIPs, and wizards for generating and deploying services.

## Capabilities

With FUSE Integration Designer, you can create EIP diagrams based on EIPs from FUSE Mediation Router, generate an EIP diagram from a FUSE Mediation Router configuration file, and export EIP diagrams to FUSE Mediation Router Spring-based XML configuration files. You can also create web services based on FUSE Services Framework.

This example shows a diagram in the EIP editor:



For more information, see the online help and the [FUSE Integration Designer documentation](http://fusesource.com/documentation/fuse-integration-designer-documentation/)<sup>12</sup>.

<sup>12</sup> <http://fusesource.com/documentation/fuse-integration-designer-documentation/>

# FUSE HQ

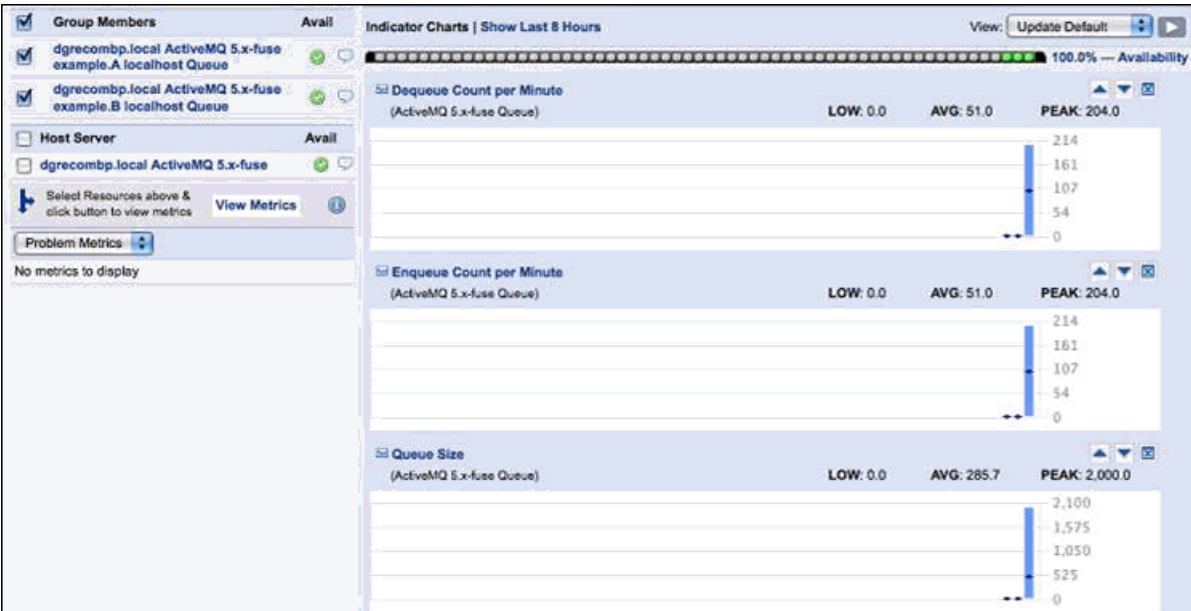
## Description

FUSE HQ is an enterprise management and monitoring system based on Hyperic HQ Enterprise. FUSE HQ uses the JMX-based reporting capabilities of the FUSE Product Family components to provide administrators with real-time administration, control, and advanced reporting capabilities. FUSE HQ is available as part of a FUSE support subscription.

## Capabilities

Powerful dashboards display real-time and historical metrics to monitor the health of systems and services, and review relationships between FUSE servers and other managed resources. Role based access controls allow you to manage user permissions. You can define intelligent alerts to proactively identify problems before they occur, and to ensure high availability by allowing multiple FUSE HQ servers to assume workloads in the event of a server failure. For more information, see the [FUSE HQ documentation](#)<sup>13</sup>.

This example shows FUSE Message Broker group alerts in FUSE HQ:



<sup>13</sup> <http://fusesource.com/wiki/display/HDOX/Home>

# Chapter 4. Which FUSE Products Should I Use?

*You can use the FUSE components either individually or together. They do not require a monolithic middleware environment and they are deployable in several containers including Tomcat and Spring. This pluggability makes FUSE Product Family a customizable integration solution that you can configure to meet your requirements.*

What Are the FUSE Development Models? .....	22
What Should I Download? .....	25
How Do I Run My Application? .....	26

## What Are the FUSE Development Models?

### Development models

Deciding which FUSE product or products to use depends on your development model. There are three main development models to use with the FUSE products:

- Developing message oriented middleware using ["JMS messaging"](#)
  - Developing [web services and RESTful services](#)
  - Integration using [enterprise integration patterns](#) and JBI
- 

### JMS messaging

With message oriented middleware (MOM), you can decouple applications that send and receive messages from the messaging system asynchronously. The messaging system manages the connections and communication channels among the applications.

Message oriented middleware uses the Java Message Service (JMS), a widely accepted messaging specification. With JMS, application components create, send, receive, and read messages. JMS messaging enables distributed communication that is loosely coupled, reliable, and asynchronous. See [Exploring JMS with FUSE Message Broker](#)<sup>1</sup> to get started with JMS messaging.

[FUSE Message Broker](#) provides a reliable, flexible messaging broker for the asynchronous exchange of critical business data and events throughout an enterprise. With FUSE Message Broker, brokers can be clustered to provide load balancing and high availability, improved performance, and scalability. FUSE Message Broker also provides fault tolerance and advanced routing using security domains.

---

### Developing services

FUSE ESB provides a powerful environment for developing web services and RESTful services using front end APIs such as JAX-WS and JAX-RS. FUSE ESB supports both contract first development with WSDL, and code first development with plain old Java objects (POJOs).

---

<sup>1</sup> <http://fusesource.com/documentation/fuse-message-broker-documentation/>

For information on service development, see the following guides:

- [\*Using the FUSE Services Framework Service Engine\*](#)<sup>2</sup> — Describes how to use FUSE ESB to develop services using POJOs, and to deploy them in a FUSE ESB container
- [\*Developing and Deploying JAX-WS Services with FUSE ESB\*](#)<sup>3</sup> — Describes how to use the native FUSE Services Framework integration to develop and deploy JAX-WS services
- [\*Developing Applications Using JAX-WS\*](#)<sup>4</sup> — Provides detailed information for using FUSE Services Framework to develop services using the JAX-WS APIs
- [\*Developing RESTful Services\*](#)<sup>5</sup> — Describes how to develop and deploy RESTful services using FUSE Services Framework

## Integration using EIPs

Use FUSE Mediation Router to implement [Enterprise Integration Patterns](#)<sup>6</sup> (EIPs). Common EIPs for routing messages include:

- Content based routing (CBR) — Use message content to route messages to specific destinations. The Logisticx FUSE Demo uses content based routing to route orders, as shown in step 5 in "[Demo Services and Applications](#)" on page 48, and in "[Sequence of Events](#)" on page 49.
- Message filtering — Prevents particular messages from reaching specific destinations.
- Recipient list routing — Routes one message to multiple destinations.
- Message transformation — Transforms the message format between applications, usually with XSLT.

<sup>2</sup> <http://fusesource.com/products/enterprise-servicemix4/#documentation>

<sup>3</sup> <http://fusesource.com/products/enterprise-servicemix4/#documentation>

<sup>4</sup> <http://fusesource.com/documentation/fuse-service-framework-documentation/>

<sup>5</sup> <http://fusesource.com/documentation/fuse-service-framework-documentation/>

<sup>6</sup> <http://camel.apache.org/enterprise-integration-patterns.html>

For information on implementing EIPs, see:

- [Enterprise Integration Patterns \(JB1 Container\)](#)<sup>7</sup> — Describes how to use the native FUSE Mediation Router integration in FUSE ESB to deploy EIP-based integration solutions in a JBI container
- [Enterprise Integration Patterns \(OSGi Container\)](#)<sup>8</sup> — Describes how to use the native FUSE Mediation Router integration in FUSE ESB to deploy EIP-based integration solutions in an OSGi container
- [Defining Routes](#)<sup>9</sup> — Provides an introduction to using FUSE Mediation Router to define routes with Java-based Domain Specific Language (DSL) and Spring XML syntax
- [Implementing Enterprise Integration Patterns](#)<sup>10</sup> — Describes how to use FUSE Mediation Router to implement EIPs

---

## Integration using JBI

With Java Based Integration (JBI), you can build components to plug into the JBI environment. FUSE ESB uses a normalized message router (NMR) to carry messages between the components.

Binding components (BCs) provide protocol independence by interfacing with external protocols and transports, such as file, HTTP, SOAP, JMS, and FTP.

Service engines (SEs) provide services, such as message transformation, orchestration, implementation of enterprise integration patterns (EIPs), and capabilities such as scheduling jobs, implementing WS-Notification, and using the SMPP protocol.

See [Using JBI in FUSE ESB](#)<sup>11</sup> for an overview of JBI and how to use the JBI tooling provided by FUSE ESB.

---

<sup>7</sup> <http://fusesource.com/products/enterprise-servicemix4/#documentation>

<sup>8</sup> <http://fusesource.com/products/enterprise-servicemix4/#documentation>

<sup>9</sup> <http://fusesource.com/documentation/fuse-mediation-router-documentation/>

<sup>10</sup> <http://fusesource.com/documentation/fuse-mediation-router-documentation/>

<sup>11</sup> <http://fusesource.com/products/enterprise-servicemix4/#documentation>



# What Should I Download?

## Overview

After determining your development model, you can decide which FUSE products to download. (See ["What Are the FUSE Development Models?" on page 22.](#))

---

## FUSE products for JMS messaging

FUSE ESB includes a runtime installation of FUSE Message Broker. However, for JMS messaging, you should download [FUSE Message Broker](#). Message flows manage how messages are transferred between components. FUSE Message Broker supports more message flows than FUSE ESB. For example, you must use FUSE Message Broker for a JMS flow, which can communicate between distributed containers and can be transparently clustered.

---

## FUSE products for service development

Download [FUSE ESB](#) if you plan to deploy and manage services on an enterprise service bus (ESB). FUSE ESB handles the delivery of messages between services, and provides management, monitoring, and mediation services such as routing, service discovery, and transaction processing. FUSE ESB also manages and deploys applications developed using FUSE Services Framework and FUSE Mediation Router.

FUSE ESB supports JBI for packaging, deploying, and managing components deployed on an ESB. FUSE ESB 4 also supports OSGi, which provides simplified deployment life cycle and dependency management. With OSGi, you can dynamically update, replace, and remove components in a running system without taking the whole system down.

Since FUSE ESB includes a runtime installation of FUSE Services Framework, you can develop web services and RESTful services in FUSE ESB. Downloading [FUSE Services Framework](#) provides additional transports, front ends, samples, and development tools.

---

## FUSE products for integration

Since FUSE ESB includes a runtime installation of FUSE Mediation Router, you can implement enterprise integration patterns (EIPs) and JBI with [FUSE ESB](#). Downloading [FUSE Mediation Router](#) provides support for additional Enterprise Integration Patterns and samples. FUSE Mediation Router can be used outside JBI, and it supports Java DSL and XML configuration.

## How Do I Run My Application?

### Running and deploying applications

When you just run an application, you must rerun it every time you stop and restart it. When you deploy an application into the runtime environment (FUSE ESB or application server), the environment instantiates a running instance of the application automatically. Then you can start and stop the application without having to redeploy it.

With the FUSE products, you can run or deploy an application either:

- ["Standalone in an ESB container"](#)
  - ["In an application server"](#)
- 

### Standalone in an ESB container

You can deploy your application standalone in a FUSE ESB container. For example:

- With FUSE ESB 3, you can deploy applications into a JBI container. FUSE ESB provides a Maven plug-in and Maven archetypes to facilitate developing, packaging, and deploying applications.
- With FUSE ESB 4, you can deploy applications in an OSGi container, which provides the flexibility and ease of use of Spring Dynamic Modules. ([Spring](#)<sup>12</sup> is an environment for deploying and running Java applications.)

The server-side applications in the Logisticx FUSE Demo are deployed in FUSE ESB. (See ["Client, Server, and Supply Side Integration"](#) on page 50.)

---

<sup>12</sup> <http://www.springsource.org/>

For information on standalone deployment, see:

- [\*Using the FUSE Services Framework Service Engine\*](#)<sup>13</sup> — Describes how to use FUSE ESB to develop services using POJOs and to deploy them in the FUSE ESB container
- [\*Managing the FUSE ESB Runtime\*](#)<sup>14</sup> — Describes how to deploy FUSE ESB containers and manage them using the command console
- [\*Enterprise Integration Patterns \(JBI Container\)\*](#)<sup>15</sup> — Describes how to deploy a FUSE Mediation Router application as a standalone application.
- [\*FUSE Mediation Router Deployment Guide\*](#)<sup>16</sup> — Describes how to deploy a FUSE Mediation Router application as a standalone application

Running FUSE ESB as a standalone server is ideal for packaging applications and launching them in different execution environments in an identical, pre-configured manner.

---

## In an application server

You can package your application and deploy it directly in an application server. You can also package the application and deploy it in FUSE ESB and then deploy FUSE ESB in the application server.

You can also embed your application in J2EE application servers, containers, and several Java web application servers and servlet containers, including Apache Tomcat, Apache Geronimo, Jetty, and JBoss. FUSE Services Framework also provides a Spring container where you can deploy any Spring-based application.

The client-side applications in the Logisticx FUSE Demo are deployed on a web server (see "[Client, Server, and Supply Side Integration](#)" on page 50).

---

<sup>13</sup> <http://fusesource.com/products/enterprise-servicemix4/#documentation>

<sup>14</sup> <http://fusesource.com/products/enterprise-servicemix4/#documentation>

<sup>15</sup> <http://fusesource.com/products/enterprise-servicemix4/#documentation>

<sup>16</sup> <http://fusesource.com/documentation/fuse-mediation-router-documentation/>

For information on deploying applications in application servers and containers, see:

- [\*Configuring and Deploying FUSE Services Framework Endpoints\*](#)<sup>17</sup> — Provides detailed information on how to configure the FUSE Services Framework runtime and how to package applications for deployment in various containers.
- [\*Using FUSE Services Framework in J2EE Environments\*](#)<sup>18</sup> — Provides detailed information on using FUSE Services Framework with J2EE application servers
- [\*FUSE Mediation Router Deployment Guide\*](#)<sup>19</sup> — Describes how to deploy a FUSE Mediation Router application in a Spring container.

---

<sup>17</sup> <http://fusesource.com/documentation/fuse-service-framework-documentation/>

<sup>18</sup> <http://fusesource.com/documentation/fuse-service-framework-documentation/>

<sup>19</sup> <http://fusesource.com/documentation/fuse-mediation-router-documentation/>

# Part II. Logisticx FUSE Demo

*The Logisticx FUSE Demo demonstrates the capabilities of the open source components in the FUSE Product Family. In this demo, the Logisticx company provides supply-chain management services to globally distributed customers.*

<b>5. Introduction to the Demo .....</b>	<b>31</b>
<b>6. Installing the Demo .....</b>	<b>33</b>
<b>7. Running the Demo .....</b>	<b>39</b>
<b>8. Understanding the Demo .....</b>	<b>47</b>
Demo Services and Applications .....	48
Sequence of Events .....	49
Client, Server, and Supply Side Integration .....	50
<b>9. Further Reading .....</b>	<b>51</b>



# Chapter 5. Introduction to the Demo

## Logisticx company

Logisticx is a fictitious company that provides supply-chain management services that include warehousing, inventory management, and shipping to a range of globally distributed customers. To meet the systems integration challenges of this business model, the Logisticx company uses the FUSE Product Family to implement services that handle all of the functions required of this company.

The services deployed in this demo perform the following tasks:

- Take customer orders and return order status
- Route customer orders based on the content of the order message
- Query and update the company database
- Determine if the warehouse has the items in stock, or if they are on order
- Poll a file system to determine if stock is in inventory and request an order for new inventory if necessary

---

## Flexibility of integration

The Logisticx company requires flexibility to integrate with customers having various protocols and data formats. FUSE ESB provides the technology platform for the Logisticx company's customer-facing needs, as well as for its own back end management, monitoring, and services. FUSE ESB's flexibility enables the Logisticx company to handle various protocols and data formats in its client, server, and supply side integration.

Client, server, and supplier side interactions include:

- A customer placing an order with a partner
- A partner managing inventory with the Logisticx company
- The Logisticx company managing shipping and inventory on behalf of its partners

The Logisticx company and its customers and partners have the following roles in this demo:

- **Customer** — Customers place orders with a partner of the Logisticx company using a branded web site provided by the partner. The customers receive feedback from the partner about order status and shipping. Customers also receive branded communications from both the partner and the Logisticx company.
- **Logisticx company** — The Logisticx company stocks the products for the partners and ships the products directly to the partners' customers on their behalf. Orders are routed to and shipped from the warehouse closest to the customer. Logisticx notifies partners of each order so the partner can accurately predict demand in real time.
- **Partner** — Partners, or vendors, are retail merchants that sell products through their online web storefront. Partners provide a branded storefront that is actually hosted by the Logisticx company. All orders for merchandise are routed through the Logisticx provider before notification is sent to the partners. Partners do not stock large quantities of their merchandise, and instead rely on the Logisticx company to stock and ship their product.

---

### Using the FUSE Product Family in the demo

FUSE ESB includes runtime installations of FUSE Message Broker, FUSE Mediation Router, and FUSE Services Framework, and provides all of the FUSE Product Family runtime capabilities required in this demo. Installing each product separately provides complete development support, including samples, but it is not required for this demo.



# Chapter 6. Installing the Demo

## Overview

The following sections describe the prerequisites and the installation of the Logisticx FUSE Demo:

1. ["Prerequisites for the Logisticx FUSE Demo"](#)
  2. ["Removing Logisticx tutorial files and service assemblies"](#)
  3. ["Downloading the Logisticx FUSE Demo files"](#)
  4. ["Deploying the server-side applications to FUSE ESB"](#)
  5. ["Deploying the web application to a web server"](#)
- 

## Prerequisites for the Logisticx FUSE Demo

The Logisticx FUSE Demo is a complete, ready-to-run application, so running the Demo requires you to install only the FUSE ESB runtime environment.

To run the Demo, install the following software:

- JDK 1.5 or higher
- FUSE ESB 3.3.0.6 or higher.

You can download FUSE ESB 3.4.x or FUSE ESB 4.0.x from the FuseSource Downloads page at <http://fusesource.com/downloads>.

Follow the instructions for installing the required version of FUSE ESB:

- FUSE ESB 3.4 — See [http://fusesource.com/docs/esb/3.4/install\\_guide/](http://fusesource.com/docs/esb/3.4/install_guide/)
- FUSE ESB 4 — See [http://fusesource.com/docs/esb/4.0/install\\_guide/](http://fusesource.com/docs/esb/4.0/install_guide/)



## Note

These instructions explain running the demo with FUSE ESB 3.x. You might notice some differences if you use FUSE ESB 4.x.

## Removing Logisticx tutorial files and service assemblies

If you previously downloaded the Logisticx tutorial source files and deployed any service assemblies to FUSE ESB as described in the [Logisticx Tutorial Guide](#)<sup>1</sup>, or if you deployed any other service assemblies to FUSE ESB, you must do the following before running the Logisticx demo application:

- Delete the `\data` directory from your FUSE ESB installation.
- Remove any service assemblies you deployed to the FUSE ESB 3 `\hotdeploy` directory, or to the FUSE ESB 4 `\deploy` directory.

## Downloading the Logisticx FUSE Demo files

To download the Logisticx FUSE Demo files:

1. Download the binaries for your platform and version of FUSE ESB from the Product Demo section of the FuseSource Downloads page at <http://fusesource.com/downloads>.
2. Expand the archive to a directory on your hard drive. The expanded archive shown contains five ZIP archives for deploying to FUSE ESB and three WAR files for deploying to a web server or servlet container:

Name	Size	Type
README.txt	5 KB	Text Document
logisticx-web-gwt.war	17,691 KB	WAR File
warehouse.war	15,954 KB	WAR File
warehouse-soapjms.war	18,416 KB	WAR File
derby-sa-1.1.zip	2,882 KB	WinZip File
order-processor-sa-1.1.zip	11,252 KB	WinZip File
order-route-sa-1.1.zip	6,896 KB	WinZip File
order-sa-1.1.zip	12,449 KB	WinZip File
stock-processor-sa-1.1.zip	3,735 KB	WinZip File

<sup>1</sup> <http://fusesource.com/docs/logistix/>

## Deploying the server-side applications to FUSE ESB

---

To deploy the server-side applications to FUSE ESB:

1. Copy the five ZIP archives from the `\logisticx-demo-1.1\deployables` directory to one of the appropriate directories:

- For FUSE ESB 3: `\hotdeploy`
- For FUSE ESB 4: `\deploy`

2. Start FUSE ESB by opening a console window or a terminal at the level of the FUSE ESB installation directory:

- On Windows, enter:

```
bin\servicemix.bat
```

- On UNIX, enter:

```
/opt/fuse-esb $ ./bin/servicemix
```

---

## FUSE ESB 3 startup

When FUSE ESB 3 starts you should see output to the console. When the output stops, FUSE ESB is started and ready for use.

Starting the FUSE ESB container automatically starts FUSE Message Broker. The broker configuration file, `activemq-broker.xml`, is located in the FUSE ESB 3 installation directory.

FUSE ESB starts the server-side applications you added to your `\hotdeploy` directory. The console displays the service startup messages.

You can also view the `servicemix.log` log file in the `\data\log\` directory for more information about the events that occur during startup.

---

## FUSE ESB 4 startup

When FUSE ESB 4 starts you should see the ServiceMix shell. FUSE ESB continues to deploy in the background, and is ready for use soon after you see the shell (you might have to wait a minute or two).

Starting the FUSE ESB container automatically starts FUSE Message Broker. The broker configuration file, `activemq-broker.xml`, is located in the FUSE ESB 4 `installation_directory\deploy` directory.

FUSE ESB starts the server-side applications you added to your `\deploy` directory. You can run a command to confirm that the services started. For example:

- With FUSE ESB 4.0 — Run `jbi list`
- With FUSE ESB 4.1 and higher — Run `jbi/list`

You can also view the `servicemix.log` log file in the `\data\log\` directory for more information about the events that occur during startup.

---

### Deploying the web application to a web server

The web application is distributed as three WAR files that can be deployed to any Java web application server or servlet container, including Tomcat and Jetty.

Make sure that FUSE ESB and the service assemblies are running and deployed before deploying the web application.

To deploy the web application to a **Tomcat** server:

1. Download Tomcat 6.0.x from <http://tomcat.apache.org/download-60.cgi/>.

If you are installing on Windows, FUSE recommends the Windows Service Installer.

2. Install Tomcat to a directory of your choice, then follow the setup instructions at <http://tomcat.apache.org/tomcat-6.0-doc/setup.html> to set your `JAVA_HOME` and `CATALINA_HOME` environment variables.
3. Copy the three WAR files from the `\logisticx-demo-1.1\deployables` directory to the `\Tomcat-6.0.x\webapps` directory.
4. From the `Tomcat-6.0.x` directory, run Tomcat as follows:
  - On Windows, enter:

```
C:\Tomcat6.0> bin\tomcat6.exe start
```

- On UNIX, enter:

```
/opt/tomcat6.0/ $ ./bin/tomcat6.sh start
```

Tomcat starts and loads the Logisticx web application and warehouse services.

To deploy the web application to a **Jetty** server:

1. Download Jetty 6.1.x. from <http://dist.codehaus.org/jetty/jetty-6.1.6/jetty-6.1.6.zip>.
2. Copy the three WAR files from the \logisticx-demo-1.1\deployables directory to the \jetty-6.1.x\webapps directory.
3. Execute the following command from the jetty-6.1.x directory:

- On Windows, enter:

```
C:\jetty-6.1.x> java -jar start.jar etc\jetty.xml
```

- On UNIX, enter:

```
/opt/jetty-6.1.x $ java -jar start.jar etc/jetty.xml
```

Jetty starts and loads the Logisticx web application and warehouse services.

---

## Next steps

With your server- and client-side applications deployed and running, now you are ready to run the Logisticx FUSE Demo. See ["Running the Demo" on page 39](#) for instructions.



# Chapter 7. Running the Demo

## Introduction

After completing the steps in [Installing the Logisticx FUSE Demo on page 33](#), you are ready to run the Logisticx FUSE Demo. You access the demo via a web browser running a client application. This enables you to place orders and to view order status. You can also view the events that occur with the server- and client-side applications in console windows as you run the demo.

---

## Getting started

To confirm that you have deployed and started the Logisticx FUSE Demo applications for:

- **Server-side applications** — See ["Deploying the server-side applications to FUSE ESB" on page 35](#). These applications are deployed and running on FUSE ESB, and include services for:
  - Order processing
  - Order routing
  - Database interaction
  - Stock processing
- **Client-side applications** — See ["Deploying the web application to a web server" on page 36](#). These applications are running on your web application server, and provide the following:
  - Order placement via the web application that runs the Logisticx FUSE Demo browser
  - Warehouse services to determine order status

When you place orders, the ServiceMix and client consoles display the activity in the server- and client-side applications, respectively. Arrange your console windows so you can monitor both consoles simultaneously as you run the Logisticx FUSE Demo.



## Note

These instructions explain running the demo with FUSE ESB 3.x. You might notice some differences if you use FUSE ESB 4.x.

See ["Understanding the Demo" on page 47](#) for detailed descriptions of the Logisticx FUSE Demo components and events that occur when running the demo.

### Opening the web application

Open a web browser to the URL <http://localhost:8080/logisticx-web-gwt/com.iona.fuse.demo.logisticx.web.customer.Logisticx/Logisticx.html>. You should see the Logisticx FUSE Demo web page, as shown in [Figure 7.1](#):

**Figure 7.1. Initial Logisticx FUSE Demo Web Page**

Select demo role: **Customer (Create Order)** [Sign Out](#) [About](#)

## Logisticx Fuse Demo

Customer Information

Customer ID:  Customer PO#:

Order Detail

Product Description: **Fuse Video Game System** ▼

Product ID:

Product Qty:  ▼

[Submit Order](#)

Customer PO#	Order ID	Comments
--------------	----------	----------

This web page shows the Logisticx Client web application running in a web browser. The web application provides access to the Logisticx company frontend and enables customers to submit orders via SOAP/HTTP to the Order service.



## Placing orders

To place orders:

1. To place an order, enter the Customer ID and Purchase Order number, then select the product and quantity that you want to order. For example:

Customer Information			
Customer ID:	<input type="text" value="567"/>	Customer PO#:	<input type="text" value="123"/>
Order Detail			
Product Description:	<input type="text" value="Fuse Sound System 700B"/>		
Product ID:	<input type="text" value="1009"/>		
Product Qty:	<input type="text" value="1"/>		

2. Then click **Submit Order**. You should receive an initial response back with your Order ID:

Customer PO#	Order ID	Comments
123	1051	Thank you for your order! Your order# is 1051

As orders are processed, the application shows order status updates from the warehouse services indicating whether or not the product is in stock:

Customer PO#	Order ID	Comments
123	1051	Your order 1051 is in Stock

Messages in the ServiceMix console show the order processing steps performed by the services deployed on FUSE ESB and the update to the order status. For example:

```
...
791 order-processor TRACE ... INSERT INTO OrderStatus
(statusId, comments, statusCode, ORDER_ID) VALUES (?, ?, ?, ?) [params=(long) 1151, (String)
Thank you for your order! Your order# is 1051, (short) 0, (long) 1051]
...
```

This status update is sent to the Logisticx Client application and is displayed in the web browser.

Messages in the client console show that the order is placed via the web browser, and that the warehouse service determines and returns order status to the server-side order processing service. The client console shows the outbound SOAP message that contains the order information. For example:

```
INFO: Outbound Message
-----
Encoding: UTF-8
Headers: {SOAPAction=[""], Accept=[*]}
Messages:
Payload: <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body><ns2:putOrder xmlns:ns2="http://logisticx.demo.fuse.iona.com/orderService/">
<arg0><customerId>567</customerId><customerPoNumber>123</customerPoNumber><lineItem>
<description>Fuse Sound System 700B</description><itemId>1009</itemId>
<lineItemId>0</lineItemId><price>0.0</price><quantity>1</quantity>
<totalPrice>0.0</totalPrice></lineItem></arg0></ns2:putOrder></soap:Body>
</soap:Envelope>
```

### What happened

The events that occur when you place an order demonstrate how the FUSE Product Family is used to implement the server- and client-side services in the Logisticx FUSE ESB Demo. For example:

- **FUSE ESB** — Services deployed on FUSE ESB receive and process the customer order, sending it on to a JMS queue and a content-based routing service.
- **FUSE Message Broker** — The order status is sent to a JMS queue using FUSE Message Broker. The Logisticx Client web application registers a handler with FUSE Message Broker to listen on the queue. Order status updates are processed from the queue to update the web browser.
- **FUSE Mediation Router** — A content-based routing service implemented with FUSE Mediation Router sends the order to the appropriate warehouse service based on message content.
- **FUSE Services Framework** — External warehouse web services, implemented using FUSE Services Framework, determine and return order status.

["Understanding the Demo" on page 47](#) provides a more detailed description of the Logisticx services and order processing events that occur during this stage of the Demo.

## Placing additional orders

1. To place additional orders, enter the Customer ID and Purchase Order number, then select the product and quantity to order, as you did before (see ["Placing orders" on page 41](#)):

Customer Information			
Customer ID:	<input type="text" value="567"/>	Customer PO#:	<input type="text" value="124"/>
Order Detail			
Product Description:	<input type="text" value="Fuse Laptop EAA300"/>		
Product ID:	<input type="text" value="1001"/>		
Product Qty:	<input type="text" value="3"/>		

2. Click **Submit Order**. You receive responses back with your Order ID. The response also shows whether or not the product is in stock:

Customer PO#	Order ID	Comments
124	1251	Your order 1251 is in Stock
123	1051	Your order 1051 is in Stock

3. Since the application starts with a quantity of 20 of each product in stock, it can run out of stock. Try placing an order that will exceed the inventory:

Customer Information			
Customer ID:	<input type="text" value="567"/>	Customer PO#:	<input type="text" value="125"/>
Order Detail			
Product Description:	<input type="text" value="Fuse Laptop EAA300"/>		
Product ID:	<input type="text" value="1001"/>		
Product Qty:	<input type="text" value="20"/>		

4. You should receive the following response indicating that the item is out of stock:

Customer PO#	Order ID	Comments
125	1351	Your order 1351 is awaiting Stock
124	1251	Your order 1251 is in Stock
123	1051	Your order 1051 is in Stock

In this case, messages in the client console indicate that the required number of items exceeds the amount currently in stock. For example:











```
Processing Stock Order
...
Required = 20 Total in stock = 14
```

Messages in the ServiceMix console show the order status update: "Your order is awaiting stock." This status update is displayed in the Logictix Demo web browser.

## Replenishing inventory

The back end system polls the `\StockInventory` directory for stock availability. This directory is located in your FUSE ESB installation directory. You can replenish the inventory by adding an XML file to the `\StockInventory` directory.

The expanded archive that you downloaded has a `\stock-inventory` directory containing XML files you can use to replenish the inventory. For example:

Name	Size	Type
 FuseCamera500L.xml	1 KB	XML File
 FuseFlatScreenTV42.xml	1 KB	XML File
 FuseGPS.xml	1 KB	XML File
 FuseLaptopEAA300.xml	1 KB	XML File
 FuseMobileLKK20.xml	1 KB	XML File
 FuseMP3Player.xml	1 KB	XML File
 FusePCD920.xml	1 KB	XML File
 FuseSoundSystem700B.xml	1 KB	XML File
 FuseVideoGameSystem.xml	1 KB	XML File
 FuseVideoRecorder100X.xml	1 KB	XML File

To replenish the inventory of the out of stock item:

1. Copy `FuseLaptopEAA300.xml` from `\stock-inventory` to the `StockInventory` directory in your FUSE ESB installation.
2. Observe the messages in the ServiceMix and client consoles. When a file polling service deployed on FUSE ESB polls this directory, the stock processing service updates the quantity of stock available in the inventory, simulating delivery of more stock at the warehouse services. Shortly after you add the files to your `StockInventory` directory, messages in the client console indicate that the application is processing new stock, and shows the amount of old and new stock available. For example:

```
Processing new Stock
...
OldStock = 20 NewStock = 34
```

3. Click **Submit Order**.

You should receive a response that your order is now in stock. For example:

Customer PO#	Order ID	Comments
125	1551	Your order 1551 is in Stock
125	1351	Your order 1351 is awaiting Stock
124	1251	Your order 1251 is in Stock
123	1051	Your order 1051 is in Stock

See ["Understanding the Demo" on page 47](#) for more details about the file polling and stock processing services.



# Chapter 8. Understanding the Demo

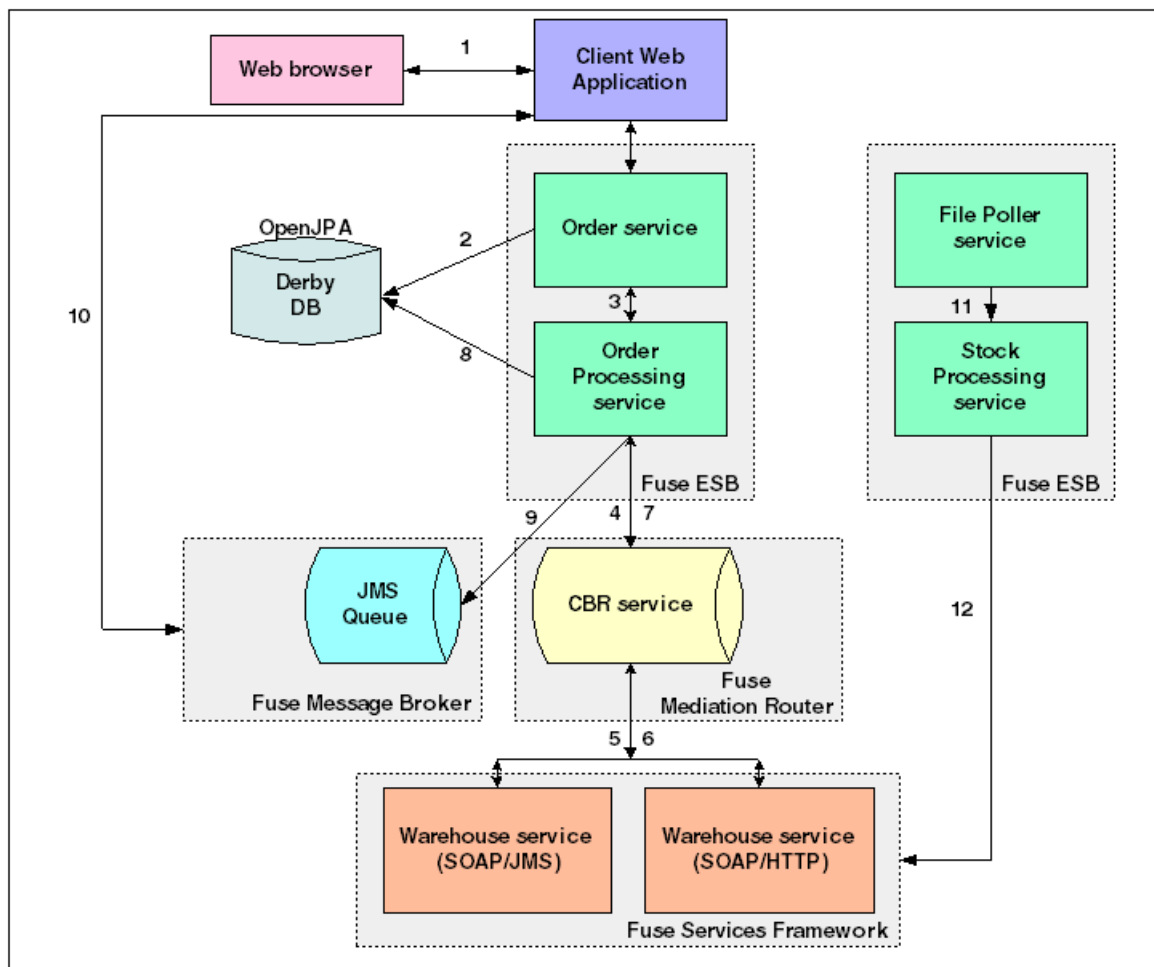
*This chapter describes the services and sequence of events in the Logisticx FUSE demo in the following sections:*

Demo Services and Applications .....	48
Sequence of Events .....	49
Client, Server, and Supply Side Integration .....	50

## Demo Services and Applications

Figure 8.1 on page 48 provides an overview of how the services and applications in the Logisticx FUSE demo work together. These components provide all the functionality behind the Logisticx FUSE Demo, and are implemented using the FUSE Product Family.

**Figure 8.1. Logisticx Services and Applications**





# Sequence of Events

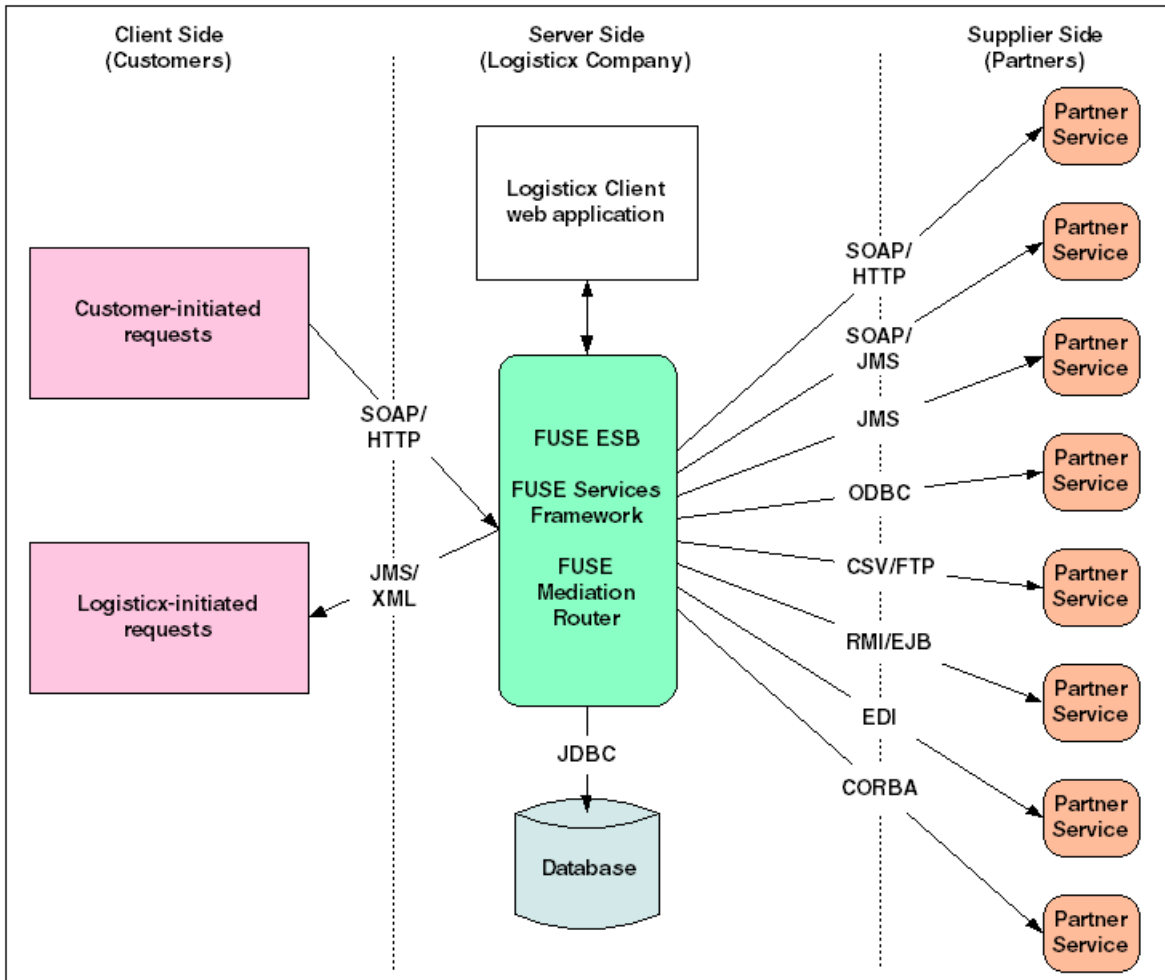
The following events occur in the Logisticx FUSE Demo shown in [Figure 8.1](#):

1. A customer places an order using a **web browser**. A Logisticx Client **web application** provides access to the Logisticx company front end, enabling customers to submit orders via SOAP/HTTP to the Order service.
2. The **Order service** requests order status from the Derby database and returns the initial order status to the customer. The Order service is deployed on FUSE ESB.
3. The **Order service** forwards the order to the Order Processing service, also deployed on FUSE ESB.
4. The **Order Processing service** processes the order and sends it to a content-based routing (CBR) service.
5. The **CBR service** routes the orders to one of the Warehouse services. The CBR service determines the routing based on whether the orders are SOAP/JMS or SOAP HTTP messages. The CBR service is implemented using FUSE Mediation Router.
6. The **Warehouse service** receives the order, as determined by the routing in the previous step. The Warehouse service returns the order status to the CBR service. The Warehouse service is an external web service implemented using FUSE Services Framework.
7. The **CBR service** receives the order status and returns the status to the Order Processing service.
8. The **Order Processing service** updates the order status in the Derby database.
9. The **Order Processing service** sends the status to a JMS queue using FUSE Message Broker.
10. The **Logisticx Client web application** listens on the JMS queue and updates the order status in the web browser.
11. The **File Poller service** polls the stock inventory and sends a request to the Stock Processing service to order new inventory.
12. The **Stock Processing service** sends a request to the Warehouse service to increase the quantity of the inventory by 20.

## Client, Server, and Supply Side Integration

The Logisticx Demo application demonstrates how the FUSE Product Family enables integration across a variety of client, server, and supplier protocols and data formats. Figure 8.2 shows the variety of protocols and formats required by the customer, partner, and Logisticx company applications.

Figure 8.2. Client, Server, and Supply Side Integration



# Chapter 9. Further Reading

## FUSE Product Family

The complete set of documentation for the FUSE Product Family is available at the [FUSE Product Documentation](http://fusesource.com/documentation/)<sup>1</sup> web site.

---

## Logisticx Tutorial Guide

The [Logisticx Tutorial Guide](http://fusesource.com/docs/logisticx/)<sup>2</sup> describes how to create and deploy the services used in the Logisticx Demo.

---

## Introductory documentation

The following guides provide introductory documentation for the FUSE open source products:

- [Getting Started with FUSE ESB](http://fusesource.com/documentation/fuse-esb-documentation/)<sup>3</sup>
- [Getting Started with FUSE ESB](http://fusesource.com/documentation/fuse-esb-documentation/)<sup>4</sup>
- [Exploring JMS with FUSE Message Broker](http://fusesource.com/documentation/fuse-message-broker-documentation/)<sup>5</sup>
- [Getting Started with FUSE Message Broker](http://fusesource.com/documentation/fuse-message-broker-documentation/)<sup>6</sup>
- [Getting Started Developing Services](http://fusesource.com/documentation/fuse-service-framework-documentation/)<sup>7</sup>
- [Getting Started with FUSE Mediation Router](http://fusesource.com/documentation/fuse-mediation-router-documentation/)<sup>8</sup>

---

<sup>1</sup> <http://fusesource.com/documentation/>

<sup>2</sup> <http://fusesource.com/docs/logisticx/>

<sup>3</sup> <http://fusesource.com/documentation/fuse-esb-documentation/>

<sup>4</sup> <http://fusesource.com/products/enterprise-servicemix4/#documentation>

<sup>5</sup> <http://fusesource.com/documentation/fuse-message-broker-documentation/>

<sup>6</sup> <http://fusesource.com/documentation/fuse-message-broker-documentation/>

<sup>7</sup> <http://fusesource.com/documentation/fuse-service-framework-documentation/>

<sup>8</sup> <http://fusesource.com/documentation/fuse-mediation-router-documentation/>



# Index

## D

- deploying
  - Logisticx FUSE Demo, 35
  - to a Jetty server, 37
  - to a Tomcat server, 36
  - to FUSE ESB, 35
- deployment options, 26
- development models, 22
- download decisions, 25

## E

- EIPs, 18, 23
- endpoints, 15
- enterprise integration patterns, 18, 23
- enterprise service bus, 9, 15
- ESB, 9, 15

## F

- FUSE ESB, 15
  - startup, 35
- FUSE HQ, 20
- FUSE Integration Designer, 19
- FUSE Mediation Router, 18
- FUSE Message Broker, 16
- FUSE Services Framework, 17

## I

- installing
  - FUSE ESB, 33
  - Jetty server, 37
  - Logisticx FUSE Demo, 33
  - Tomcat server, 36
- integration, 9

## J

- Java Based Integration, 24
- JBIs, 24
- JMS messaging, 22

## L

- Logisticx FUSE Demo
  - deploying, 35
  - description of events, 49
  - diagram, 48
  - installing, 33
  - introduction, 31
  - prerequisites, 33

## M

- message oriented middleware, 22
- messaging, 22
- MOM, 22

## O

- open source software, 11

## R

- RESTful services, 22

## S

- Service oriented architecture, 9
- SOA, 9

## W

- web services, 22

