



Building rock solid *relationships* and optimising technology every time.

## Open Source Enterprise Service Bus (ESB) Evaluation

This paper explores how vendors define the Enterprise Service Bus (ESB) and compares some popular open source ESBs.

An Enterprise Service Bus (ESB) is a loosely defined term that in most instances refers to a software product that implements a software architecture that provides fundamental services via an event-driven and standards-based messaging-engine. In most cases, the term 'ESB' is used to refer to Java-centric solutions, even though similar middleware products exist in the .Net world. This document focuses on the traditional Java ESB product offerings.

Most enterprise integration vendors and open-source communities claim to implement the ESB concept in their own products, and they provide several services and features that they associate with typical ESB functionality. Most ESBs provide overlapping features that are prevalent in concepts such as Business Process Management (BPM) and Business Activity Monitoring. This makes the task of evaluating and comparing different ESBs a non-trivial exercise, with a huge subset of features to consider and compare.

To overcome this problem, we can dissect how popular vendors and groups describe ESBs and their offerings to find a subset of common terms and attributes they use to define the Enterprise Service Bus.

## Vendors on their ESBs

### JBoss Enterprise Service Bus [1]

"JBossESB is the next generation of **EAI** - better and without the **vendor lock-in** characteristics of old. As such, many of the capabilities mirror those of existing EAI offerings: **Business Process Monitoring, Integrated Development Environment**, Human Workflow User Interface, **Business Process Management**, Connectors, Transaction Manager, Security, Application Container, **Messaging Service, Metadata Repository, Naming and Directory Service, Distributed Computing Architecture**.

Plus JBossESB is part of an SOI (Service Oriented Infrastructure).

### Oracle Enterprise Service Bus [2]

"Oracle Enterprise Service Bus provides everything you need for seamless integration of data and enterprise applications within your organization and with trading partners. Oracle Enterprise Service Bus is a key component of a **Service-Oriented-Architecture**, providing low-cost, standards-based integration between systems for greater IT flexibility and responsiveness. **SOA** allows organizations to more easily manage the complexity of their **heterogeneous environment**, without vendor lock-in to proprietary technologies."



Building rock solid *relationships* and optimising technology every time.

### **Sun OpenESB** [3]

"A large number of components are being developed in the OpenESB community: e.g. for accessing back-end systems and other ESBs or **SOA** platforms, for message **transformation**, etc. Unlike proprietary ESBs, OpenESB is purely based on open **standards** (e.g. JBI and Java EE). This prevents **vendor lock-in**.

The NetBeans based IDE makes it very easy to create **integration** solutions or **composite** applications."

### **Sun JavaCaps** [4]

"Java CAPS provides a **standards-based, open, extensible platform** for developing software infrastructures using a **service-oriented architecture** approach. This unified and comprehensive suite can help your enterprise create **composite** applications from existing investments as well as deliver new **business services** in a flexible SOA environment with no **vendor lock-in**."

### **Mule ESB** [5]

"Mule is a lightweight **integration platform** and service container that allows you to quickly and easily connect your application together. Mule provides a robust, secure and scalable platform to build enterprise applications offering an array of **transports** such as JMS, HTTP, Email, FTP, JDBC and many more. It also offers rich set of features for **web services, message routing, mediation, transformation** and **transaction management**. Designed around the ESB (Enterprise Service Bus) concept, Mule is the most widely used open source integration platform."

### **TIBCO ActiveMatrix Service Bus** [6]

"TIBCO ActiveMatrix® Service Bus is a lightweight enterprise service bus that helps organizations bridge the mediation gap in their **SOA infrastructure**. It **reduces complexity** and increases **flexibility** and reuse by replacing hard-coded service dependencies with configuration, and can be used to quickly onboard services from a third-party environment or a TIBCO-based infrastructure."

### **IBM WebSphere ESB** [7]

"WebSphere ESB provides Web services connectivity, JMS **messaging**, and **service-oriented integration** to power your SOA."

For IBM, an ESB[8] is:

"An Enterprise Service Bus (ESB) is a **flexible** connectivity infrastructure for integrating applications and services. An ESB can power your Services Oriented Architecture (SOA) by reducing the number, size, and **complexity** of **interfaces** between those applications and services.

An ESB performs the following:



Building rock solid *relationships* and optimising technology every time.

- **Routing messages** between services
- Converting **transport** protocols between requester and service
- **Transforming** message formats between requester and service
- Handling **business events** from disparate sources”

## Apache ServiceMix [9]

“Apache ServiceMix is an open source ESB (Enterprise Service Bus) that combines the functionality of a Service Oriented Architecture (**SOA**) and an Event Driven Architecture (EDA) to create an **agile**, enterprise ESB.

Apache ServiceMix is an open source distributed ESB built from the ground up on the Java Business Integration (JBI) specification JSR 208 and released under the Apache license. The goal of JBI is to allow components and services to be integrated in a **vendor independent** way, allowing users and vendors to **plug and play.**”

## Common attributes of an ESB

Using the above extracts we can determine that an ESB:

- Is key to Service Oriented Architecture and is part of a **Service Oriented Infrastructure** (SOI)
- Is the next generation **Enterprise Application Integration** (EAI) product
- Is **open standards-based** and avoids vendor lock-in
- **Increases flexibility** and **responsiveness** and assists in **managing complex** and **heterogeneous environments**



Key concepts of an ESB tag cloud



Building rock solid *relationships* and optimising technology every time.

We can also use the same approach to extrapolate evaluation criteria using the most common attributes of an ESB. The representation of this in a tag cloud highlights common themes and will form the base for our ESB evaluation.



*Typical services tag cloud*

From the tag cloud we can conclude that “Message Service” is one of the most mentioned services from ESB vendors and there is a very good reason for this. Most ESB’s are based on the Java Business Integration (JBI) specification that was developed under the Java Community Process (JCP). The JBI specification seeks to create a standard for EAI and business-to-business integration (B2B) and attempts to do so by “**adopting a SOA, which maximizes the decoupling between components, and creates well-defined interoperation semantics founded on standards-based messaging**” [10]. JBI defines an architecture that allows the construction of integration systems from plug-in components that interoperate through mediated message exchange.

JBI also differentiate between two distinct types of components:

- **Service Engine (SE):** SEs provide business logic and transformation services to other components as well as consume such services. SEs are in principal components that focus on business logic and processing. Tags from the tag cloud that fit broadly into this category are **Mediation, Routing, Transaction Management** and **Transformation**.
- **Binding Components (BC):** BCs provide connectivity to services external to a JBI environment. BC’s represents communication components. Tags that fit into this category are **Transport, Connectors** and **Message Services**



Building rock solid *relationships* and optimising technology every time.

## Evaluation considerations

Other important factors that should be considered in evaluating an ESB's are:

### Quality of documentation

It is of no use to have a brilliant piece of software without any knowledge of how to use it. Open Source Software (OSS) creators are notably bad at documenting their software products and care must be taken to ensure that a piece of software that is perceived to be free does not drain the budget by taking up project resources to reverse engineer OSS source code to determine how to operate it successfully.

Documentation that is recent and covers the following topic is a necessity:

- Basic architecture and principles
- Getting started guide
- Basic introduction tutorials
- Code examples

### Market visibility

Market visibility will most likely ensure adequate skills and other resource availability for the product in question. A bigger market footprint means more people developing, testing, debugging and writing books or tools for the product, and decreases the likelihood that an open source project gets abandoned. Visibility in your particular geography is important since it could indicate the availability of local resources and support.

### Tool Support

Connecting, transforming and routing your business processes is the activity that most time will be spend on in your new ESB. A graphical Integrated Development Environment (IDE) is in general more desirable than editing XML files. IDE support varies significantly between ESB vendors and can range from an integrated XML editor to an fully blown drag-and-drop orchestration and transformation tool. It is also important to consider interoperability with your current development tools, since this can decrease the time your development team needs to spend learning new tools.

### Runtime tools

Runtime tools are typically tools that you use in your production environment to keep your process engine running in optimal fashion. Monitoring tools such as Business Activity Monitoring and runtime optimization tools falls into this category.



Building rock solid *relationships* and optimising technology every time.

## Standards

Most vendors claim their ESB to be standards-based in support of the obvious standards like SOAP and WSDL, but emphasis should be put on standards like BPEL and JBI. JBI in theory should allow you to use Binding and Service components from other vendors in you ESB.

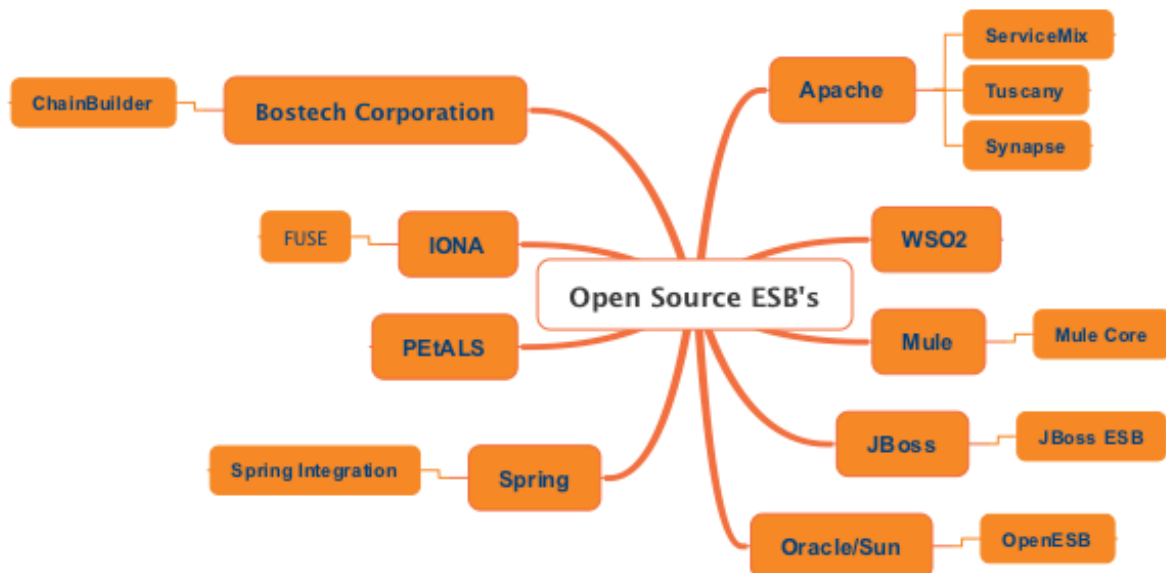
## Runtime and development platform support

Consider the development platform and runtime environment where the product will be developed and deployed. It is generally desirable that your newly acquired ESB runs on the same application server and OS as the rest of your environment.

## Product Support

Community support is in most part the norm for supporting Open Source Software and will in most instances take the form of Wikis and user and developer forums. The availability of commercial support must also be considered and will be of particular interest if the product is used in an enterprise setting. Note should also be made of the geographical nature of the support, since onsite support will be more expensive of even impossible if supplied from an international rather than local location.

## The open source ESB market place



### *Open Source ESB's and EIA application*

The number of available Open Source ESB and EAI products is now in double digits and the options are growing by the day. Subprojects and spin-offs of ESBs like ServiceMix also increase the size of the field.



Building rock solid *relationships* and optimising technology every time.

In the remainder of this document we will explore four open source ESBs. They were selected based on their perceived popularity and market visibility, variable development approaches and maturity.

## OpenESB / GlassFish ESB v2.1

OpenESB is based on the JBI 1.0 specification and can run in a bare JVM or be embedded inside a Java EE application server. The platform consists of a core runtime, business and service components and design time support. GlassFish ESB takes the core runtime collocated with the GlassFish application server and the NetBeans IDE, and a subset of components pre-installed into an install bundle.

### Service Engines

There are 15 service engines available as part of the OpenESB project in various states (Stable, Beta or Incubator). Only components in stable state are distributed with GlassFish ESB but can be installed separately.

### Binding Components

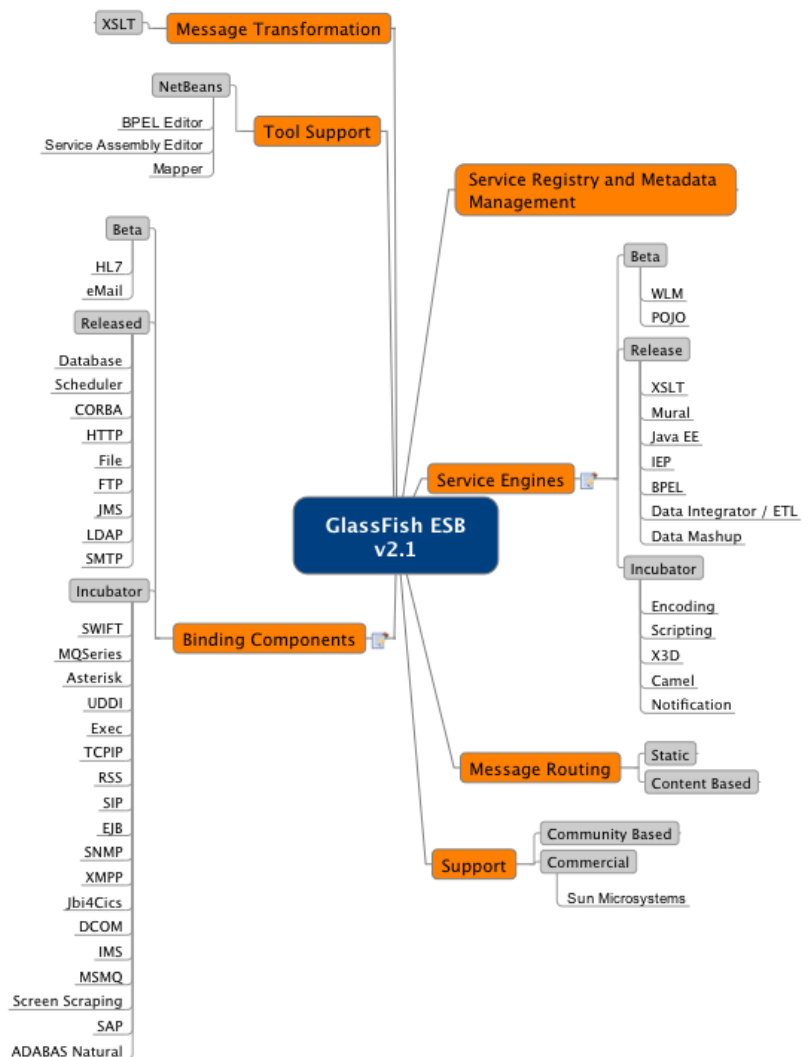
More than 30 binding components are available as part of the OpenESB project. Most of the commonly used binding components like HTTP, FTP etc. are in release state with the rest in Beta and Incubation state.

### Quality of documentation

Documentation seems in most part sufficient with a few video tutorials to get started. Components are also sufficiently documented to get them up and running.

### Tool Support

GlassFish ESB is packaged with NetBeans BPEL and Service Assembly editor and works surprisingly well. Almost all development can be done with the graphical interface without touching XML. It does however support BPEL editing using the built-in XML editor.





Building rock solid *relationships* and optimising technology every time.

## **Standards**

OpenESB/GlassFish ESB is built on the JBI (JSR208) and WSDL 1.1/2.0 standard.

## **Runtime and development platform support**

OpenESB runs out of the box on GlassFish with support for JBoss and Sun Java Application Server Enterprise Edition.

## **Product Support**

Support is mostly community-based but commercial support is available for GlassFish ESB through Sun Microsystems.

## **License**

OpenESB is mostly covered under the Common Development and Distribution License (CDDL) v1.0 but some other licenses are used for Service and Binding components. GlassFish and Netbeans are covered under the GlassFish and NetBeans license.

## **Summary**

GlassFish ESB is highly recommended for a development team invested in Sun Microsystems technology. It utilises the popular Netbeans IDE that a large number of developers will be familiar with. The availability of video tutorials and examples also makes it easy to learn.



Building rock solid *relationships* and optimising technology every time.

## Mule Community Edition v2.1

Mule is one of the few ESB products that is not based on the JBI specification, and instead implements its own model focusing on productivity and ease of development. Mule's main building blocks consists of business Components, Transport, Transformers, Inbound and Outbound Routers. Business components represent business logic and could be Plain Old Java Objects (POJO's), Spring Beans or REST services.

### Service Engines

Mule supports a range of out-of-the-box filters and routers that covers most traditional Service Engine functionality and can be implemented using Java or Spring Beans.

### Binding Components

Binding components are referred to as Transport components and most frequently-used protocols are covered out of the box.

### Quality of documentation

A few introductory and getting started guides are available, but not enough to become proficient with Mule. The suggested method of learning Mule is by configuring new services and studying code examples. Commercial-quality user guides and manuals are available with the commercial Mule Enterprise Edition.

### Tool Support

Mule is in its basic form a collection of POJO's and XML configuration files. There is an Eclipse plug-in called Mule IDE available but it is of limited use. It essentially only provides an interface to generate the initial XML configuration file and provide functionality to deploy and run your application on Mule standalone.

### Standards

Mule support SOAP, JMS and other common, but does not adhere to the JBI and BPEL specifications from an implementation perspective. It does, however, provide JBI connectivity via a JBI adaptor implementation.

### Runtime and development platform support

All mainstream Java platforms are supported as run-time environments. The lack of an extensive development IDE also makes it portable to almost all possible Java development platforms.

### Product Support

Support is community-based but commercial support is available in the form of Mule Enterprise Edition.

### License

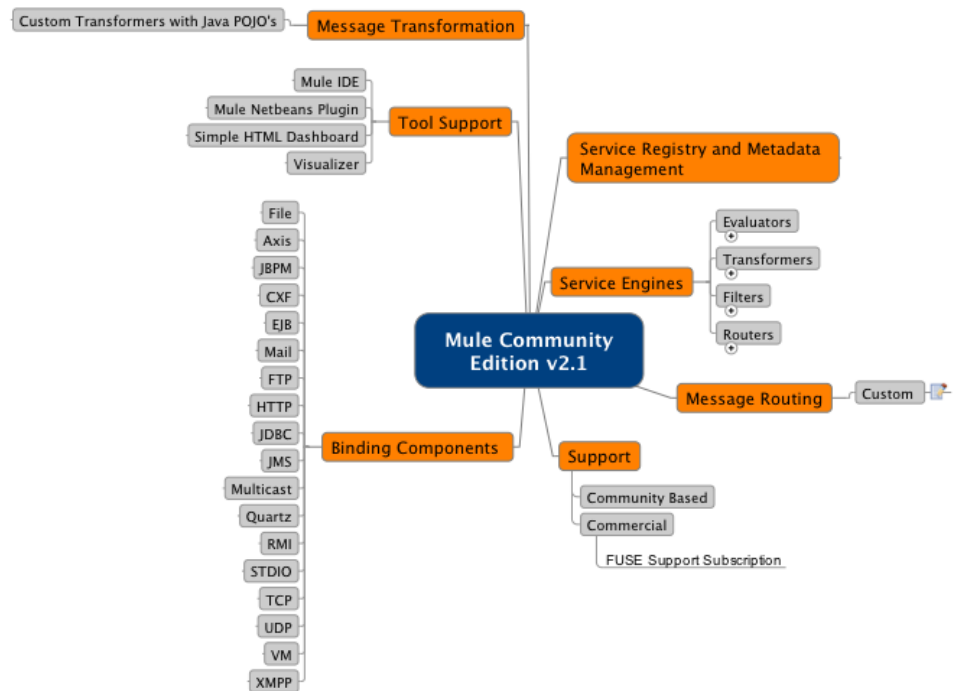
Common Public Attribution License (CPAL) v1.0.



Building rock solid *relationships* and optimising technology every time.

## Summary

Mule is not a good choice if your development team's preference is to work with a well-documented rich graphical IDE, but it could be a suitable if they are used to the Java Spring Framework development environment and comfortable working with Spring XML configuration files. Mule will also be a good choice if you need a very lightweight ESB that can be embedded into your application.



## JBoss ESB 4.6

JBossESB provides a basic core ESB framework that allows everything else to be configurable with a pluggable architecture. All JBoss subsystems such as messaging and transformation can be swapped with alternatives. Smooks 1.0 is the default transformation engine, but the product can also support XSLT. It supports BPEL 2.0 and include a built-in Service Registry based on JAX-R. Drools rule engine is also included and used for content-based routing and business rules.

## Service Engines

Most popular Service Engine implementations are included with JBoss ESB. JBoss is also the only ESB in this review that comes with a pre-installed business rules engine.

## Binding Components

Most common binding components, including JMS and SOAP, are included out-of-the-box.

## Quality of documentation

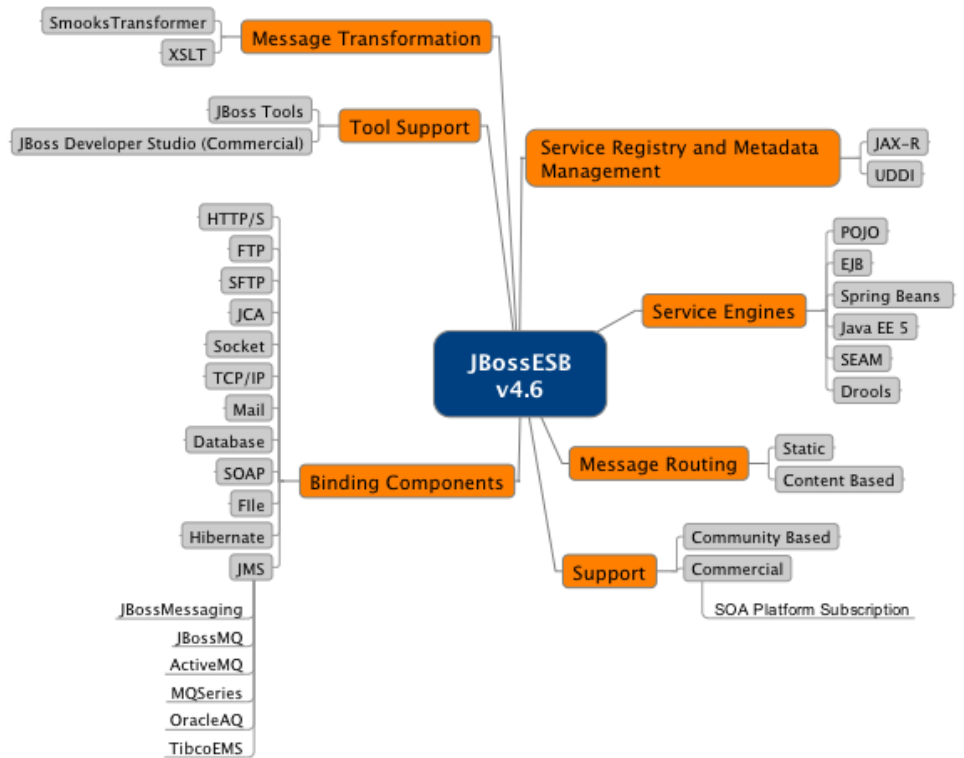
Introduction and getting started guides are available with examples to cover most binding components and service engines. A Programmer and Administration guide is also available and seems comprehensive and up-to-date with the latest releases of the ESB.



Building rock solid *relationships* and optimising technology every time.

## Tool Support

JBoss Tools is an Eclipse plug-in that supports JBoss and related technologies. It serves as the primary free development environment for JBoss ESB. JBoss Developer Studio is the preferred development environment, but it is not free and is part of JBoss Enterprise SOA Platform.



## Standards

JBossESB support SOAP 1.2, WSDL 1.1 and BPEL 2.0.

## Runtime and development platform support

JBossESB can be deployed to JBoss Application Server, Tomcat or run in a standalone configuration.

## Product Support

Support is community-based through the use of user and developer forums, a Wiki and blogs. Commercial support is available via JBoss Enterprise SOA Platform subscription.

## License

GNU Lesser General Public License (LGPL) v2.1.

## Summary

JBoss ESB is more or less on par with Glassfish ESB in most respects and could be a better choice if you have already invested in other JBoss products. It will also appeal to development teams familiar with the Smooks Java transformation engine.

## Apache ServiceMix 4.0.0

Apache ServiceMix is an ESB built from the ground up on the JBI specification under the Apache license. It is lightweight and easily embeddable inside any server or client or can be run as a standalone ESB provider. It can be run in any Java SE or EE application server. FUSE ESB is a "productized" version of ServiceMix with a support subscription.



Building rock solid *relationships* and optimising technology every time.

## Service Engines

ServiceMix supports most popular Service Engines.

## Binding Components

Most common binding components are included out of the box like JMS, SOAP, etc.

## Quality of documentation

Documentation from ServiceMix is fairly comprehensive with a Getting Started guide that covers the basic introduction to JBI and server Installation. The tutorial section on the ServiceMix web site is especially comprehensive with sections targeted at the beginner to intermediate level. The "Article" section also proves to be a useful resource with various links to presentations and articles on ServiceMix and related concepts.

## Tool Support

There are no graphical tools per se for ServiceMix but the Eclipse BPEL Designer plug-in can be used to edit and deploy Apache ODE BPEL processes. Fuse Integration Designer can also be used but requires a FUSE support subscription.

## Standards

ServiceMix support JBI 1.0 and WS-BPEL 2.0 via Apache ODE.

## Runtime and development platform support

Any Java SE or EE environment

## Product Support

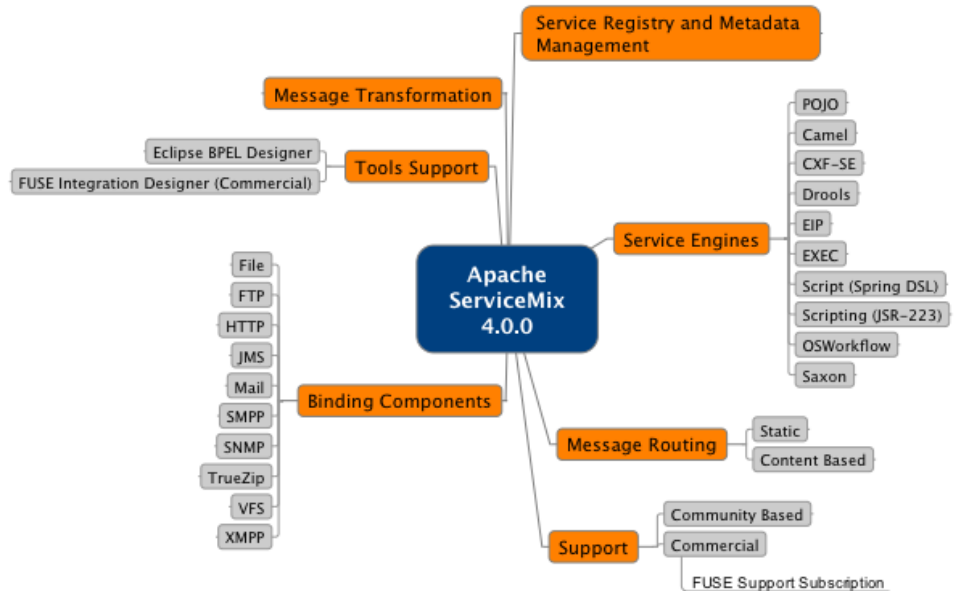
Support is once again community-based through the use of user and developer forums, Wikis and blogs. Commercial support is available through IONA FUSE ESB at fusesource.com.

## License

Apache License v2.0.

## Summary

ServiceMix will appeal to development teams that need to embed or run an ESB on all Java-based platforms or application servers. The optional use of the Eclipse BPEL editor and JBI





Building rock solid *relationships* and optimising technology every time.

support could also make it more preferable than Mule, especially for teams not familiar with the Spring framework.

## Conclusion

Differentiating between traditional ESBs, EAI platforms and Business Process Management Suites (BPMS) is becoming more difficult with ESBs expanding to include non-traditional functions such as BAM and System-to-Human workflow support. It is important to understand the primary intent of the ESB and the objectives that it needs to address when selecting an ESB, and care must be taken to also consider external factors like the development team's current development preferences and skills. There is not an outright best ESB but rather a "horses for courses" scenario, and even some instances where an ESB is not the appropriate tool in the first place.

Selecting any of the above Open Source ESBs will almost always depend on your existing technology stack and development skill. GlassFish and JBoss ESB will appeal more to a graphically- oriented "drag and drop" style of development, while Mule sits on the other side of the spectrum and ServiceMix in between. All traditional integration APIs and protocols are covered by all the mentioned ESBs, with additional plug-ins available in cases where they are not provided out-of-the-box. Using a JBI-compliant ESB could be of some benefit where an out of the ordinary connector is required, since vendors can develop JBI connectors that can be used in any compliant JBI container.

## Summary

	OpenESB / GlassFish v2.1	Mule Community Edition v2.1	JBossESB v4.6	Apache ServiceMix v4.0.0
<b>Standards</b>	JBI	JBI compatible	JBI, BPEL 2.0	JBI
<b>Development Tools</b>	Netbeans	Mule IDE	JBoss Tools, JBoss Developer Studio	Eclipse BPEL Designer
<b>Documentation</b>	4	2	4	
<b>Support</b>	Community/Paid	Community/Paid	Community/Paid	Community/Paid
<b>Development Platform</b>	Windows/Linux/Mac	Windows/Linux/Mac	Windows/Linux/Mac	Windows/Linux/Mac
<b>Run-time container</b>	GlassFish, JBoss or Sun ES	Weblogic 10, Tomcat 6 Jetty 6.1	JBossESB, JBossAS, Tomcat, Standalone	Any Java SE or EE environment
<b>Run-time OS</b>	Windows/Linux/Unix	Windows/Linux/Unix	Windows/Linux/Unix	Any Java supported OS
<b>License</b>	CDDL v1.0	CPAL v1.0	LGPL v2.0	Apache License v2.0



Building rock solid *relationships* and optimising technology every time.

## References

1. <http://www.jboss.org/jbossesb/>
2. <http://www.oracle.com/appserver/esb.html>
3. <http://open-esb.dev.java.net/>
4. <http://www.sun.com/software/javaenterprisesystem/javacaps/index.jsp>
5. <http://www.mulesource.org/display/MULE/Home>
6. <http://www.tibco.com/software/soa/activematrix-service-bus/>
7. <http://www.ibm.com/developerworks/websphere/zones/businessintegration/wesb.html>
8. <http://www.ibm.com/developerworks/web/library/wa-soaesb/>
9. <http://servicemix.apache.org/home.html>
10. Java Business Integration 1.0 final release