Institute of Architecture of Application Systems
University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Master's Thesis Nr. 3166

Extending an Open Source Enterprise Service Bus for Multi-Tenancy Support

Stefan Essl

Course of Studies	Information Systems
First Examiner	Prof. Dr. Frank Leymann
Second Examiner	Prof. Dr.-Ing. habil. Bernhard Mitschang
Advisor	Dipl.-Inf. Steve Strauch
	Dipl.-Inf. Tobias Binz
Commencement	April 1, 2011
Conclusion	November 4, 2011
CR-Classification	C.2.4, D.2.11, H.3.4, H.4.1, K.1

Abstract

Within the Cloud computing approach, Platform as a Service is a way to provide customers with the capability to deploy acquired or consumer-created applications onto the Cloud infrastructure. It relieves these of the need to install and run their own infrastructure or to manage and control the underlying Cloud infrastructure. Whereas providers of such services try to serve as many customers as possible to exploit economies of scale, especially small and medium businesses profit from this approach, because they can save the high up front and administrative cost of installing and running their own processing systems and applications.

In order to offer an Enterprise Service Bus as a proven technology known from the field of Service-Oriented Architectures as a Platform in the Cloud it has to be made multi-tenant aware. This fulfills the Platform as a Service providers' need to raise the overall utilization and to maximize revenue by serving multiple customers from one system instance.

This master's thesis develops a concept to extend an Enterprise Service Bus by multi-tenancy support with respect to communication and implements this concept in an open source product. The concept and implementation are evaluated by application to a scenario originating from the European project 4CaaSt.

Contents

List of Abbreviations

AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BPEL	Business Process Execution Language
CIFS	Common Internet File System
DOM	Document Object Model
DPR	Dynamic Policy Resolution
EAI	Enterprise Application Integration
EDI	Electronic Data Interchange
EIP	Enterprise Integration Pattern
ESB	Enterprise Service Bus
FIX	Financial Information Exchange
FTP	File Transfer Protocol
HL7 MLLP	Health Level Seven Minimal Lower Layer Protocol
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IAM	Identity and Access Management
IMAP	Internet Message Access Protocol
IRC	Internet Relay Chat
JAAS	Java Authentication and Authorization Service
JBI	Java Business Integration
JDBC	Java DataBase Connectivity
JMS	Java Message Service
JMX	Java Management Extensions
JTA	Java Transaction API
LDAP	Lightweight Directory Access Protocol
MSMQ	Microsoft Message Queuing
OSGi	Open Services Gateway Initiative Framework
PaaS	Platform as a Service
PGP	Pretty Good Privacy
POP	Post Office Protocol
QoS	Quality of Services
RMI	Remote Method Invocation
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SFTP	SSH File Transfer Protocol
SIP	Session Initiation Protocol
SMPP	Short Message Peer-to-Peer
SMTP	Simple Mail Transfer Protocol

SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
SOA	Service-Oriented Architecture
SQL	Structured Query Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
UUID	Universally Unique Identifier
WCF	Windows Communication Foundation
WebDAV	Web-Based Distributed Authoring and Versioning
WS	Web Service
WSDL	Web Services Description Language
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XSD	XML Schema
XSLT	Extensible Stylesheet Language Transformations

List of Figures

List of Tables

List of Listings

1 Introduction

Providing and accessing computing power as a utility like water or electricity has been a desire since decades [cf. Par66]. However, it has not been until the last years that crucial technologies to enable this approach became widespread, accessible, and rendered it profitable. Ubiquitous broadband network access enabled Cloud computing, an approach that allows enterprises, small and medium businesses, and even home users to utilize standardized services.

Because services are provided by a multitude of different systems that use different protocols, a component is needed to mediate between these systems. The Enterprise Service Bus (ESB) is known from Service-Oriented Architectures (SOA) to serve this exact purpose. However this proven technology cannot be provided sufficiently in a Cloud computing environment without any modifications.

1.1 Problem Statement

Cloud computing allows providers to serve several customers, called tenants in Cloud computing environments, from a single system instance by means of virtualization and multi-tenancy. This enables the provider to drive the utilization of the infrastructure and to reduce the cost to serve a specific tenant. This renders Cloud computing offerings cost-effective even for small and medium-sized companies.

With regards to service providers being able to build up large scale computing centers and profit from economies of scale, this approach is even able to offer tenants lower cost as if they ran their systems in-house [cf. AFG+09]. This enables the provider to serve the tenants more flexibly and catch a much larger market segment, because of the higher efficiency and lower cost per tenant.

However this approach requires that infrastructure and applications need to be multi-tenant aware. They have to be able to differentiate tenants, provide proper data and performance isolation, and allow tenants to be served by individually configured services.

A concept enabling multi-tenant aware communication is needed that allows the identification of tenants in incoming requests and the correlation of each request, message, or processing job to the correct tenant. Because tenants have different demands regarding their services, they have to be provided with customization and configuration options, which have to be correctly administered and integrated into each application.

The goal of this master's thesis is to extend an open source ESB by multi-tenancy support with respect to communication to allow its use as a Cloud computing application. The ESB needs to correlate each message to the correct tenant and route service requests and responses correctly. Out of scope of this master's thesis are data or performance isolation of the actual services the tenants deploy to the ESB, tenant configuration and administration, multi-tenant service configuration, or tenant based service discovery and allocation.

1.2 Research Design

In Cloud computing settings, a multitude of aspects have to be considered. Infrastructure is widely distributed, both geographically and organizational. The involved systems need to be load-balanced to cope with the high number of customers. All services need to be replicated to ensure availability, because outages would involve both a financial loss and a loss of customer trust. Besides all these aspects, multi-tenancy is important for the individual handling of each tenant.

This master's thesis pursues its goal of extending an ESB by multi-tenancy support with regards to communication through research on existing multi-tenancy approaches. It analyses the state of the art of ESBs and evaluates widespread ESB products. This evaluation shows both strengths and shortcomings of existing products and multi-tenancy approaches. A number of general specifications on how to realize a multi-tenancy approach within an ESB are derived from these strengths and shortcomings, and detailed into specifications for this master's thesis. It uses these specifications for a precise design, which is then implemented in an open source ESB product. The concept and implementation are evaluated based on a scenario originating from the European project 4CaaSt.

1.3 Motivating Scenario

The goal of this motivating scenario is to review the approach taken in this master's thesis at the end of the implementation in Section 6.3. The motivating scenario bases on a taxi service setting with basic and premium taxi companies. The users call the taxi company of their choice. Thus, the taxi company is the tenant within this motivating scenario. It then hands off the user request to the taxi service provider via SOAP over Hypertext Transfer Protocol (HTTP), Java Message Service (JMS), or e-mail and includes information on how to inform the customer about the taxi arrival. The service provider's multi-tenant aware ESB locates taxi drivers close to the customer with privileged handling of requests of the premium taxi company and informs the customer about the estimated time of arrival of the taxi. For the customer notification each taxi company has its own preferences about the way the customers are informed. This results in customers being informed for example by e-mail if they ordered a taxi with the first taxi company, while they might be informed by instant message via Extensible Messaging and Presence Protocol (XMPP) if they ordered a taxi with the second taxi company. Figure 1.1 shows this setting with the transport protocols to use.

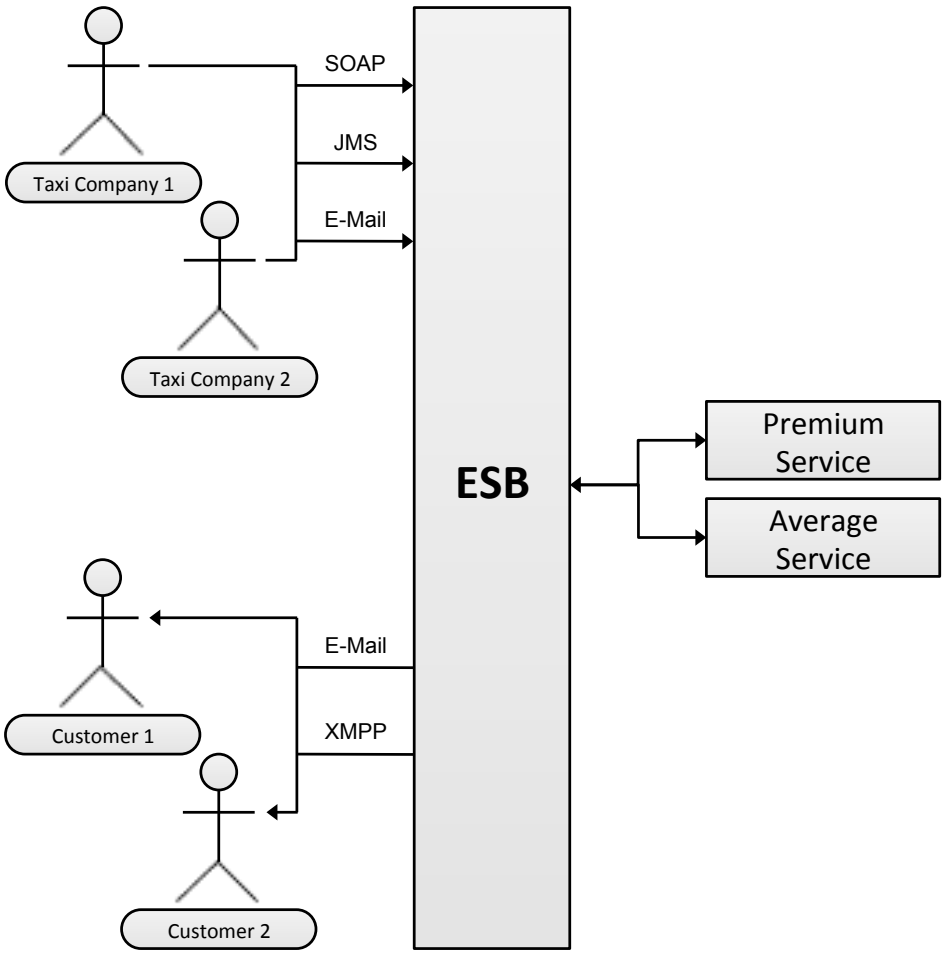

Figure 1.1: Evaluation Scenario

The multi-tenancy aspect within this motivating scenario is the need for the ESB to distinguish between basic and premium taxi companies when looking up nearby taxi drivers as well as taking into account the contact preferences of each tenant for customer notification.

1.4 Outline

This document contains the following chapters.

Chapter 1 - Introduction motivates the topic of this master's thesis and illustrates a motivating application scenario for this work.

Chapter 2 - Background details the notions and terms, provides background information on concepts and technologies this master's thesis uses, and references related works.

Chapter 3 - State of the Art provides an extensive analysis of the state of the art and of ESB products.

Chapter 4 - Specification clarifies the shortcomings of existing products and specifies requirements a multi-tenant ESB needs to meet.

Chapter 5 - Design shows, which precise components of the overall product architecture need to be changed and how they need to be changed to meet the requirements.

Chapter 6 - Implementation shows the implementation of each of these components and gives exact details on how they are changed and extended. It also contains an evaluation to review the chosen design and implementation based on the motivating scenario introduced in Section 1.3.

Chapter 7 - Conclusion and Future Work concludes this master's thesis and shows some perspectives how the topic *Multi-Tenant Aware Enterprise Service Bus* could be further developed.

2 Background

This chapter gives an introduction into the topics of this master's thesis. It explains some basic terms and notions, provides background information on concepts and technologies this master's thesis uses, and references works that deal with the specific parts in detail.

2.1 Service-Oriented Architecture

Over the previous decade, the SOA paradigm gains importance for the design and operation of large scale distributed systems that cover a multitude of business processes. Its main concepts are "services, interoperability through an enterprise service bus, and loose coupling" [Jos07, p. 8] with the goal to improve flexibility of processes, departments, and whole companies including their business networks [cf. Jos07, p. 12]. SOA thereby tries to overcome difficulties that large and globally distributed enterprises and their business partners face with regards to IT architecture. These are challenges like distributed systems, different owners of systems or processes, and the heterogeneity of systems or networks [cf. Cha04, p. 1].

The central component in a SOA is the ESB. Its function is to connect service providers to service consumers and thereby ensure decoupling of these [cf. Cha04, p. 1]. The service consumer on the one hand should not need to know where each service it wants to use resides, or how its service requests get there. The service provider on the other hand should not have to take care of service management, security, or reliability individually for each particular service. An ESB may further provide data transformation, routing with complex rules, or monitoring and logging [cf. Jos07, p. 48 f.]. Figure 2.1 shows the SOA triangle, which represents the tasks of an ESB. To achieve the decoupling of requestors and services, requestors pose service requests to the ESB with an interface description rather than with a physical address of a service.

2.2 Cloud Computing

The concept of Cloud computing evolves since some years. In this setting, providers serve customers infrastructure, platforms, or applications as services over broadband network connections [cf. AFG+09, p. 1]. The terms of these service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [cf. MG09, p. 2]. According to these authors, IaaS provides the consumer with direct access to infrastructure services like storage or processing power with the ability to freely choose what to run on top of this infrastructure in terms of operating systems, middleware, or applications. PaaS

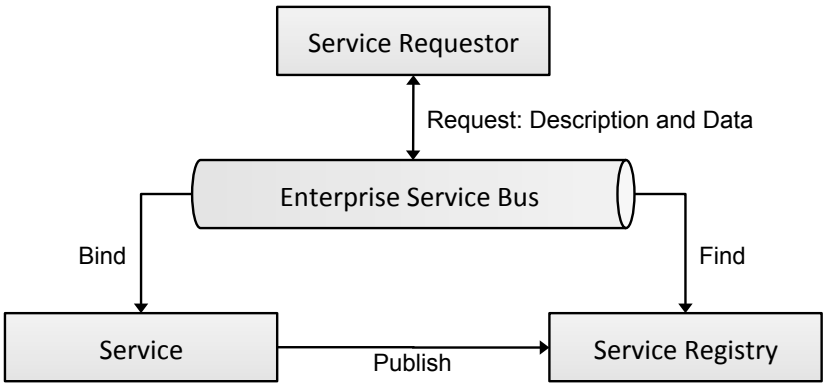

Figure 2.1: The SOA Triangle [WCL+05, p. 20]

does not allow the consumer to control the infrastructure, but provides access to a framework of services the consumer can build its application on. SaaS provides the consumer with a ready to run software over a broadband connection like the internet, but does not allow any manual changes to the underlying infrastructure or application other than a limited set of configuration settings each consumer or user can define. The three service models form a stack, because each level can be built on top of the lower level. Therefore, a PaaS provider can itself be an IaaS consumer and build its platform on the infrastructure it consumes. Figure 2.2 shows this stack with example service providers for each level.

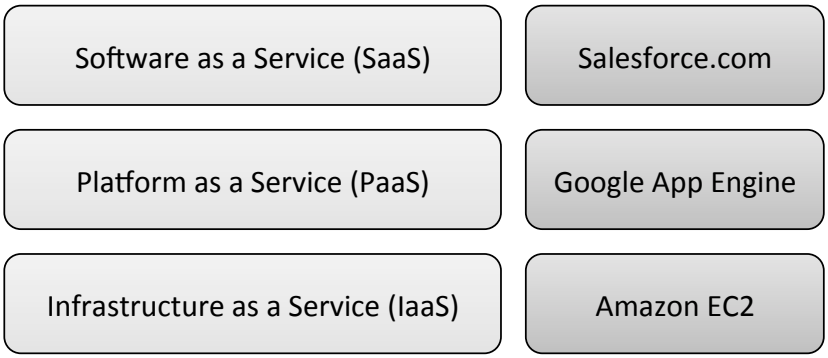

Figure 2.2: Cloud Computing Services Stack [MG09, p. 2]

Cloud computing comes with essential characteristics, as the illusion of infinite resources through pooling of virtualized resources, standardized on-demand self-services, broad network access, rapid elasticity, and flexible payment models based on measured services that typically include paying for only the resources the customer uses [cf. AFG+09, p. 1][cf. MG09, p. 1]. This way the provider can profit from economies of scale in a multitude of ways from the cost of electricity to the purchase of actual servers [cf. AFG+09, p. 1 f.]. Customers can benefit from these profits, because using such services could mean lower operation costs than each customer would be able to achieve in its own processing centre, because of missing economies of scale for each customer. This effect is even more important for small and medium-sized enterprises, because of the high up front costs of building its own processing centre and purchasing the hardware [cf. AFG+09, p. 2].

Another advantage of this approach is its flexibility. In a Cloud computing approach, requesting one machine for 100 hours costs the same as 100 machines for one hour, but finishes the task 100 times faster [cf. Got07]. Because the customer does not have to commit into long term arrangements, but can request and release computing power on demand, there is further no need to leave costly computing infrastructure unused. Without Cloud computing, lines of business with fluctuating demand would purchase computing power that is able to handle peak demand, which would then sit unused the rest of the time. If they purchase computing power for average or low demand, they would loose valuable sales. Therefore, the Cloud computing approach helps the customer to transfer risks to its service provider [cf. AFG+09, p. 11]. The service provider can profit from its large number of customers and assume that the peak and low demands of its customers even out.

Mell and Grance [MG09, p. 2] name four deployment models regarding Cloud computing. A private cloud consist of infrastructure that is available only for one organization, and is therefore comparable to existing processing facilities that are enhanced with the characteristics of Cloud computing. A community cloud targets a specific set of organizations that share requirements regarding the computing environment whereas a public cloud is accessible for the general public. Any combination of two different models is called hybrid cloud. An organization with a private cloud may for example outsource some of their computing needs into a public cloud for a short period of time if they see their infrastructure under peak demand.

As Cloud computing approaches are composed of standardized services that service consumers can access, each Cloud computing approach is itself a SOA [cf. BGK+11]. Cloud services are SOA services and SOA principles can be used and applied to Cloud computing approaches.

2.3 Multi-Tenancy

Within a Cloud computing setting each customer is called a tenant [cf. RCL09]. To leverage existing infrastructure, services and applications, Cloud computing providers extend their services by multi-tenancy. This means that an existing service is modified to serve several tenants in an isolated manner without these taking notice thereof. Because now the provider does not have to keep an extra service with all its requirements in stand by for each tenant, the costs to serve each tenant shrink. This enables the provider to address a larger market and catch the "long tail" [cf. CC06], which is a market share that is otherwise unaddressable, because of higher costs to reach it than profits to gain from it. Figures 2.3 and 2.4 show this setting before and after the provider could lower the cost for each customer by multi-tenancy.

There are several approaches to achieve multi-tenancy. Many concentrate on multi-tenant data architecture [cf. CCW06] whereas others provide several multi-tenancy patterns that services can use to enable them by multi-tenancy [cf. MUTL09]. Guo et. al. [GSH+07] concentrate on an architecture level and differentiate between multiple instances multi-tenancy and native multi-tenancy. Figure 2.5 shows both approaches.

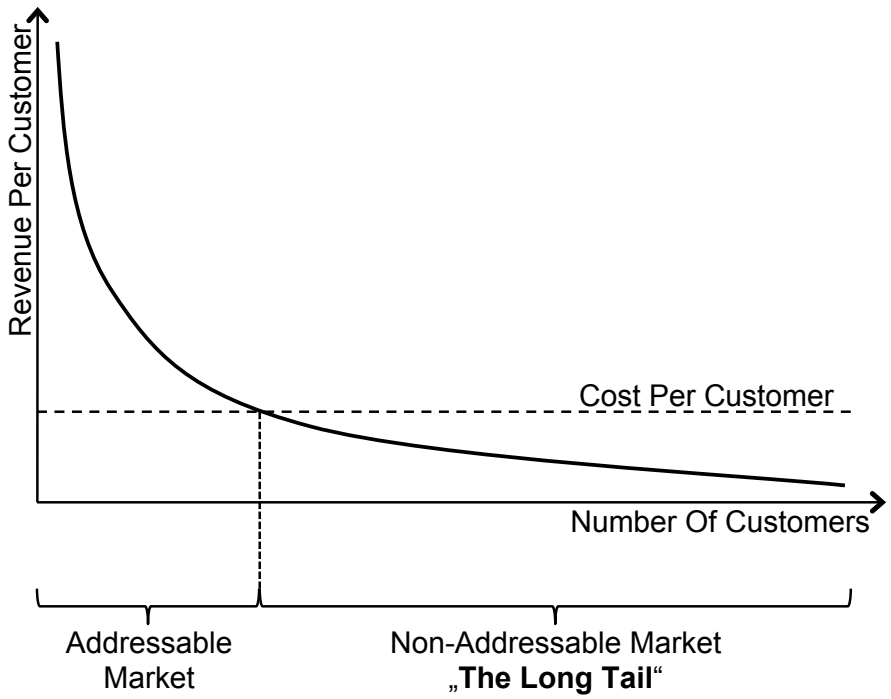

Figure 2.3: The Long Tail [cf. CC06]

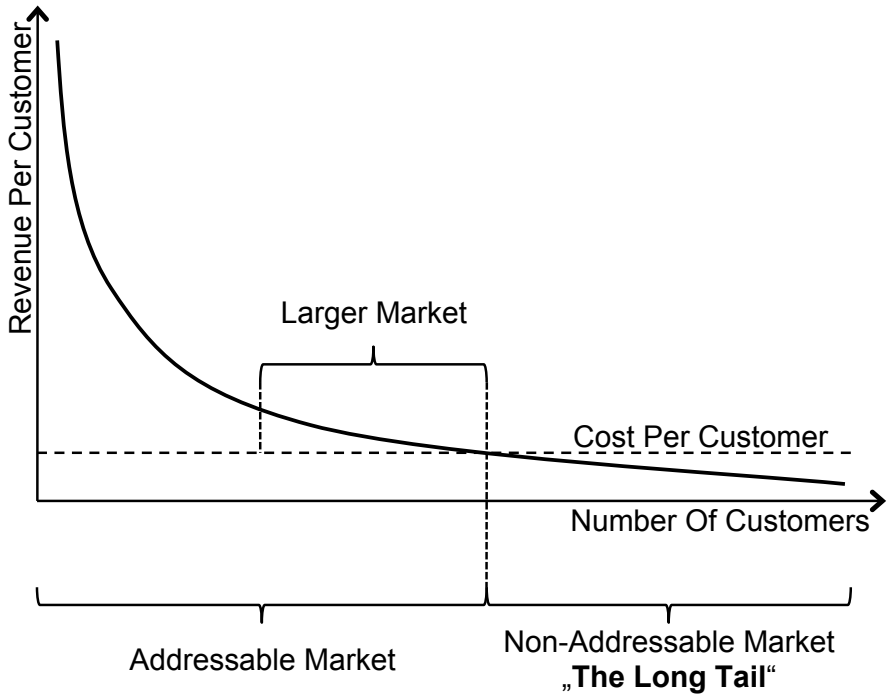

Figure 2.4: The Long Tail With Lower Cost per Customer [cf. CC06]

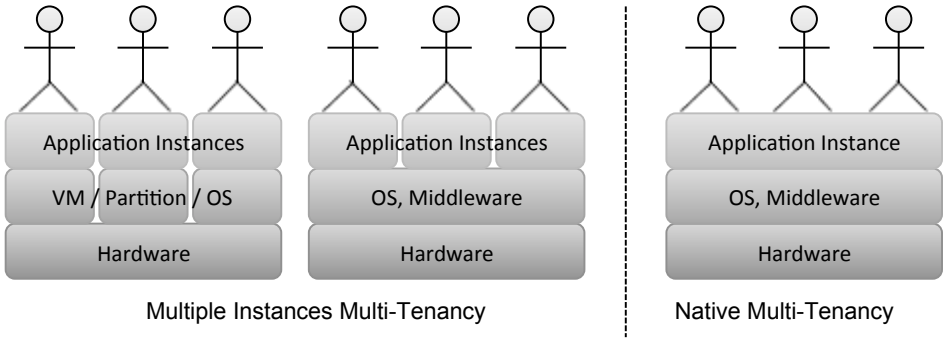

Figure 2.5: Approaches to Multi-Tenancy [GSH+07]

Whereas the multiple instances multi-tenancy approach provides each tenant with its own application instances and even virtual machine or operating system, the native-multi-tenancy approach can handle tenants natively within the application itself. This helps to serve more tenants from the existing infrastructure and provides more efficient scaling in large environments, because the overhead of multiple instances drops out. The multiple instances approach however is easier to set up and requires much less effort to ensure that tenants do not affect the Quality of Services (QoS) of other tenants, because of its isolated nature [cf. GSH+07].

Because of the Cloud computing focus of this master's thesis, it aims at the native multi-tenancy approach, because of the importance of appropriate scaling capabilities. The native approach benefits providers and tenants, because of higher profit margins for the provider and lower delivery cost for the consumer [cf. GSH+07] in settings where there are a high number of tenants with one provider. With regards to the long tail that Figures 2.3 and 2.4 illustrate, this approach even renders services accessible for small and medium-sized enterprises that would otherwise not be profitable for these.

With regards to SOA, the ESB is such a service that is profitable only at larger scales. Because SOA and with it all components around the concept of SOA need a high initial effort for set up and implementation [cf. Mal06], especially small and medium-sized enterprises profit from the ability to use an ESB in a PaaS setting without the need of the high up front investment for setting up its own ESB.

This leads to the goal of this master's thesis, extending an open source ESB for multi-tenancy support with respect to communication. To achieve this goal, this master's thesis makes use of several technologies, which the following sections describe.

2.4 Java Business Integration

To ease the implementation of SOA environments, the Java Community developed the Java Business Integration (JBI) standard. It provides a framework for building an environment that contains components that typically exist in a SOA environment and is extensible by plug-ins. JBI uses a messaging model that bases on the Web Services Description Language

(WSDL). The Normalized Message, which is a Normalized Message Format, is used for all the communication inside the ESB. Therefore, the central routing component within the ESB is called Normalized Message Router. Binding Components transform every message that arrives from external sources into such Normalized Messages and vice versa. Because of the extensible JBI environment, Binding Components can be implemented and plugged into the JBI environment for arbitrary protocols. Service Engines transform these messages, provide business logic, integrate other Java applications, or consume such services themselves. For systems management, JBI defines structures that base on the Java Management Extensions (JMX). JBI is located in the Java package `javax.jbi` [cf. Jav].

2.5 Service Engine

The description of JBI in Section 2.4 introduces the concept of Service Engines. Their tasks are to provide business logic, routing and transformation, or to integrate other Java applications. This work uses a Service Engine to integrate Apache Camel and make use of its extensive support of Enterprise Integration Patterns (EIP) [cf. Theg]. Hohpe and Woolf [HW03] originally introduced these patterns to guideline enterprise architects through integration challenges of large, distributed, and complex systems.

2.5.1 Apache Camel

Apache Camel is an integration framework released as open source by the Apache Software Foundation [cf. Thea]. It bases on the EIPs of Hohpe and Woolf [HW03] and offers the possibility to implement these patterns in standard Java objects through Bean Integration [cf. Thec]. The components that come with Apache Camel allow connections to an extensive set of transports [cf. Theb]. Within the Apache projects it can integrate into other Apache applications like the open source ESB Apache ServiceMix as a routing and mediation engine [cf. Thea].

2.6 Binding Components

The description of JBI in Section 2.4 further introduces the concept of Binding Components. These accept messages from external sources, transform them into Normalized Messages and dispatch these to the Normalized Message Router of the ESB and vice versa. This master's thesis uses the Binding Components for the protocols SOAP over HTTP, JMS, e-mail, and XMPP. The following subsections introduce these protocols.

2.6.1 SOAP over HTTP

Weerawarana et. al. [WCL+05, p. 63 ff.] introduce SOAP as a messaging framework to access Web services in loosely coupled infrastructures. It bases on the Extensible Markup Language (XML) and each SOAP message consists of a SOAP envelope with at least zero SOAP headers and a SOAP body. The headers target each receiver on a SOAP message path and for example contain request identification or user authentication data. The body carries the actual message payload or business data. If some error occurs, a participating party can return a SOAP message that contains a SOAP fault.

SOAP messages themselves are transported on top of an underlying network protocol. This master's thesis uses HTTP as a standardized protocol, but other standardized or proprietary protocols could be used. The roles that take part in a SOAP message exchange are called SOAP sender for transmitting systems, SOAP receiver for receiving systems, or SOAP intermediary for systems that receive and transmit messages.

2.6.2 Java Message Service

Oracle [Ora] introduces the Java Message Service (JMS) as a standard for highly distributed message based communication. It aims at providing loosely coupled, reliable and asynchronous communication. JMS provides point-to-point and publish/subscribe communication. In point-to-point communication, the sender submits its messages to a specific queue that persistently stores these messages until a receiver requests the message from the queue. As soon as the receiver successfully retrieved the message, it is removed from the queue. In publish/subscribe communication the sender posts a message to a specific topic that an arbitrary number of receivers can subscribe on. As soon as the sender's message arrives, each of the subscribers receives a copy of this message.

2.6.3 E-Mail

The Internet Message Format standard [cf. They] specifies e-mail as a syntax for text messages between computer users. Each e-mail consists of a header and a body. The header carries mandatory data about the sender and the creation date of the e-mail, and optional data that can even be individually defined. This optional data carries for example a signature of each system the e-mail passed on its way to the receiver or a value that states the encoding of the e-mail content. The body carries the actual content the sender wants to transmit. This may for example be a SOAP message.

The network protocols around e-mail are the Simple Mail Transfer Protocol (SMTP) for sending and forwarding messages among systems, whereas the Post Office Protocol (POP) and the Internet Message Access Protocol (IMAP) are typically used by e-mail applications to retrieve messages from an inbox on the e-mail server.

2.6.4 XMPP

The XMPP Standards Foundation defines the Extensible Messaging and Presence Protocol (XMPP) as an open and XML based technology for real-time communication [cf. Fou]. Its widest application is instant messaging, but XMPP enables voice or video calls and further collaboration between users. This master's thesis uses its instant messaging capabilities. XMPP is also known under the term Jabber.

3 State of the Art

This chapter extensively analyses the state of the art regarding ESBs. Section 3.1 references several works that deal with frameworks to evaluate ESB products, extends them by multi-tenancy criteria and elaborates an evaluation framework this master's thesis will work with. Section 3.2 researches ESB products to evaluate in this work whereas Section 3.3 operationalizes the evaluation framework into measurable dimensions. Section 3.4 evaluates ESB products in great detail and provides facts on each of them. In Section 3.5, an ESB product is chosen for the concept and implementation of this master's thesis. Section 3.6 describes the architecture of the chosen product and explains the components that are important for the rest of this work.

3.1 Criteria for Evaluating ESB Products

Depending on the point of view of the author, the set of criteria taken into account to evaluate ESB products varies a lot. Ahlberg [cf. Ahl10] on the one hand concentrates on reliable message transfer, thus only listing criteria that are important in this exact context. Astrova et. al. [cf. AKK10] on the other hand emphasize high availability whereas others [cf. Sch10][cf. Bay08][cf. RD08][cf. TNB06][cf. Gup08][cf. Dav09] try to capture important criteria in a broader way. Woolley [cf. Woo06, p. 35] even provides an entire ESB evaluation framework.

Yet none of them considers multi-tenancy within their evaluation. With it being the key aspect, this master's thesis makes use of the criteria based on the works of these authors and extends them by multi-tenancy criteria. Table 3.1 shows the final evaluation model.

3.2 Candidates for ESB Product Evaluation

A first collection of existing ESBs to evaluate within this master's thesis leads to the list of potential candidates shown in Table 3.2. Further research on this extensive list shows that some of them are not even an ESB in a closer context. In the words of Chappell [Ker05] regarding Apache Synapse:

> "This project [N.B.: Synapse] is related to ESB, but it is not in itself an ESB. [...] What Synapse brings to the table is a mediation framework that allows users to get in the middle between service requesters and providers and perform various tasks - including transformation and routing and that helps to promote loose coupling between services."

Category	Criterion	Explanation
A Messaging	A.1 Support for EIP	Which patterns are supported?
	A.2 Routing	Which routing capabilities are offered?
	A.3 Message Transformation	Which transformation capabilities are offered?
	A.4 Normalized Message Format	Does the product offer message normalization?
B Integration	B.1 BPEL support	Is the Business Process Execution Language (BPEL) supported?
	B.2 WS-* standards	Are common Web Service standards supported?
	B.3 JBI support	Is JBI support offered?
	B.4 Adapters	Are there adapters to common products?
	B.5 Custom Adapters	Is there a possibility to implement custom adapters?
	B.6 Transport Mechanisms	Which transport mechanisms are supported?
C Runtime	C.1 Dynamic Policy Resolution	Through which means is dynamic policy resolution offered?
	C.2 Service Registry Support	How is the registration and de-registration of services supported?
D Security	D.1 Access Control	Which means are offered to maintain access control?
	D.2 Secure Message Transport	To which extent is secure message transport offered?
E QoS	E.1 Availability and Scalability	How are availability and scalability ensured?
	E.2 Transactions	Is there support for transactions?
F Multi-Tenancy	F.1 Data Isolation	In which way is data isolated for different tenants?
	F.2 Performance Isolation	Which means are offered to ensure performance isolation?
	F.3 Tenant Based IAM	How is tenant based identity and access management (IAM) offered?
G Extensibility	G.1 Plug-In Mechanism	Is there a possibility to use plug-in based extensions?
	G.2 Documentation	How comprehensive is the documentation?
	G.3 Community	Is there an active community supporting the product?
	G.4 Training	Which means of training are offered?
	G.5 Licensing	Is the product open source or proprietary?

Table 3.1: Criteria to Evaluate ESB Products

Product Name	Vendor	Type
Apache Camel	Apache Software Foundation	Open Source
Apache CXF	Apache Software Foundation	Open Source
Apache ServiceMix	Apache Software Foundation	Open Source
Apache Synapse	Apache Software Foundation	Open Source
Apache Tuscany	Apache Software Foundation	Open Source
BizTalk Server	Microsoft	Proprietary
Business Works/Active Matrix	Tibco	Proprietary
ChainBuilder ESB	Chain Builder ESB Integration Community	Open Source
Fiorano ESB	Fiorano Software	Proprietary
FUSE ESB	FUSE Open Source Community	Open Source
JBossESB	JBoss	Open Source
Mule ESB	Mulesoft	Open Source
OpenAdapter	OpenAdapter	Open Source
Open ESB	Oracle, former Sun Microsystems	Open Source
Oracle ESB	Oracle	Proprietary
Oracle Service Bus, former AquaLogic Service Bus	Oracle, former BEA	Proprietary
Petals ESB	OW2 Consortium	Open Source
Progress Artix ESB	Progress Software	Proprietary
Progress Sonic ESB	Progress Software	Proprietary
SAP SOALogix Confero	SAP	Proprietary
SOPERA ASF	SOPERA	Open Source
Spring Integration	SpringSource	Open Source
Sun Java CAPS	Sun Microsystems	Proprietary
webMethods ESB-Plattform	Software AG	Proprietary
WebSphere ESB	IBM	Proprietary
WSO2 ESB	WSO2	Open Source
xBus ESB		Open Source

Table 3.2: List of ESB Products to evaluate

Thus, the listed products are entered in a preselection that focuses on different aspects:

- Accordance to the requirements an ESB has to meet (see Section 2.1)
- Attention and Adoption in Research and Industry
  - User Base
  - Customer Size
  - Researches
- Set of Features
  - Compliance to WS-* standards
  - BPEL support
  - Adapter Count
- Attention within the Research Community
- Community (N.B.: only open source products)
  - Member Count
  - Activity of Community
  - Roadmap
- Customer Support

These narrow down the list of products to enter the detailed evaluation to the relevant ones shown in Table 3.3.

Product Name	Vendor	Type
Apache ServiceMix	Apache Software Foundation	Open Source
BizTalk Server	Microsoft	Proprietary
JBossESB	JBoss	Open Source
Mule ESB	Mulesoft	Open Source
Petals ESB	OW2 Consortium	Open Source
WebSphere ESB	IBM	Proprietary
WSO2 ESB	WSO2	Open Source

Table 3.3: Preselected List of ESB Products for Later Evaluation

3.2.1 Interlude - Situation of Open ESB

Open ESB is in any way considered a candidate for the detailed evaluation within this master's thesis. Unfortunately the project is going through immense changes at the moment. After the acquisition of Sun Microsystems by Oracle, the latter announced not to continue development of Open ESB any further. At the time of writing of this master's thesis a community establishes to ensure Open ESB's future with Oracle's support for this transition. Nonetheless, the changes take place with few details being available. Therefore, a well-founded evaluation is not possible at the moment [cf. Log].

3.3 Operationalization of Criteria

To attain an objective comparison the criteria listed in Table 3.1 are operationalized rendering them countable. To achieve this purpose, products are ranked on a scale from 0 to 4 points regarding each criterion and depending on the level of fulfillment. Only full points are given.

3.3.1 Operationalizing Messaging

One of the main functions of an ESB is its Messaging System. It routes, transforms, and delivers messages within an enterprise and beyond ensuring non-functional requirements like security, availability, reliability et cetera.

A.1: **Enterprise Integration Patterns (EIP)** are based on the book by Hohpe and Woolf [cf. HW03]. The authors offer patterns that describe best practices for the integration of existing application systems within an enterprise. The more patterns an ESB product supports, the less effort Enterprise Application Integration (EAI) needs. Table 3.4 shows the resulting points scale. Whether the ESB provides these patterns by itself or through extra components is important. An extra component typically includes an advanced set of functionalities, because it fully focuses on the problem it solves whereas the integrated approach helps to lower administrative complexity. Because either way offers upsides and downsides that need evaluation in the context of the specific use case, this difference is not taken into account regarding this ranking. Nonetheless, the description of each product offers details about which approach the product takes.

Points	Requirement
0	ESB does not support any patterns
2	ESB offers limited support for patterns
4	ESB fully supports patterns

Table 3.4: Points for Supporting Enterprise Integration Patterns

A.2: **Routing** is a means to ensure efficient transfer of messages among participating parties. To make sure messages are received by all obligatory endpoints, but thereby not distributed any more than needed, the ESB has to offer Routing Patterns. The Scatter-Gather pattern for example is a typical pattern used for auctions. The ESB broadcasts the request for a bid to the tenderers and aggregates their responses to provide the requestor with a single response. Table 3.5 shows how products are ranked regarding this criterion. The Routing Capabilities that Hohpe and Woolf [cf. HW03, 225 ff.] list, are the basis of the ones this evaluation demands. Offering these either on its own or through extra components are both valid ways for a product to get ranked accordingly.

Points	Requirement
0	ESB does not offer any advanced Routing Capabilities
2	ESB offers limited Routing Capabilities
4	ESB offers full support for Routing Capabilities

Table 3.5: Points for Routing Capabilities

A.3: **Message Transformation** is another important aspect. ESBs should offer means to convert format as well as content of messages to ensure that applications using different message designs can communicate with one another. Table 3.6 shows the ranking of products regarding Message Transformation.

Points	Requirement
0	ESB does not offer any Message Transformation
1	ESB offers a given set of Message Transformations via extra components
2	ESB offers a given set of Message Transformations on its own
3	ESB offers Message Transformations in any way needed via extra components
4	ESB offers Message Transformations in any way needed on its own

Table 3.6: Points for Message Transformation

A.4: **Normalized Message Format** is a message design the ESB uses internally. It translates each incoming message into this normalized format, routes it, and, on leaving the ESB infrastructure, transforms it into the format the receiver needs to be able to process the message. Table 3.7 shows how this criterion influences the ranking.

Points	Requirement
0	ESB does not use a Normalized Message Format
4	ESB uses a Normalized Message Format

Table 3.7: Points for Normalized Message Format

3.3.2 Operationalizing Integration

Because another one of an ESBs purposes is the integration of existent systems within an enterprise and with systems of suppliers, customers, and partners, this is another category deemed important for this evaluation.

B.1: **BPEL Support**: BPEL is the de facto standard used to specify actions within business processes [cf. Cob04]. Therefore, its support is an important feature to be able to design business processes by using the services known to the ESB. Table 3.8 shows how the products are ranked regarding this criterion.

Points	Requirement
0	ESB does not offer the design of business processes at all
1	ESB offers means to design business processes without BPEL
2	ESB offers means to convert BPEL descriptions to an internal format
3	ESB offers the integration of a BPEL engine
4	ESB offers an integrated BPEL engine

Table 3.8: Points for BPEL Support

B.2: **WS-* Standards** are aside from BPEL an important set of specifications regarding Web services. WS-Policy for example is a specification that allows Web services to claim their needs regarding QoS. To be able to serve functional and non-functional requirements that these specifications describe, it is fundamental that ESB products support as many as possible. Table 3.9 shows how this evaluation ranks products for the support of the number of WS-* standards.

Points	Requirement
0	ESB supports up to three WS-* standards
2	ESB supports up to six WS-* standards
4	ESB supports more than six WS-* standards

Table 3.9: Points for Supporting WS-* Standards

B.3: **JBI Support**: As Section 2.4 mentions, JBI is an important specification with regards to a SOA approach. Therefore, its support is another important criterion for ESB products being evaluated in this master's thesis. Table 3.10 shows the points that are given in this category.

B.4: **Adapters** to common products are another important feature to enable integration. These offer functionality to connect application systems to the ESB without the need of developing custom integration components. The offering of as many Adapters as possible by the ESB product lowers the effort customers need to implement their own adapters and integrate the ESB with their existing systems. Table 3.11 shows how products are ranked regarding the availability of Adapters.

Points	Requirement
0	ESB does not support JBI
1	ESB supports JBI through extra components
2	ESB supports JBI on its own
3	ESB supports JBI compliant to the official specification, but is not certified
4	ESB fully supports JBI and is certified by the official specification

Table 3.10: Points for JBI Support

Points	Requirement
0	ESB provides no Adapters
2	ESB offers a limited set of Adapters
4	ESB provides an extensive set of Adapters

Table 3.11: Points for Providing Adapters

B.5: **Custom Adapters** offer a way to integrate existing non-standard applications with the ESB aside from the Adapters already provided. Table 3.12 shows how products are ranked for providing this possibility.

Points	Requirement
0	ESB does not offer a possibility to implement Custom Adapters
2	ESB offers a possibility to implement Custom Adapters with limitations
4	ESB offers a possibility to implement Custom Adapters in any way needed

Table 3.12: Points for Providing the Possibility to Implement Custom Adapters

B.6: **Transport Mechanisms** are the means that ESB products offer to receive, send, and carry messages. The more possibilities an ESB product offers, the less effort has to be put into designing adapters for supporting additional transport mechanisms. The basic and most common standard is SOAP over HTTP, which every product in this ranking provides. Therefore, this is seen as a basic requirement. The standards rated in this ranking are JMS, XMPP, e-mail, and the Session Initiation Protocol (SIP). Each product rates one point higher for each of these four transports supported.

3.3.3 Operationalizing Runtime

To be able to evaluate the ESB products regarding their manageability, different aspects are taken into account within the category Runtime.

C.1: **Dynamic Policy Resolution** (DPR): Policies are non-functional requirements a service demands for being processed. Examples are Quality of Services (QoS) such as security, scalability, or the compliance to certain rules. Because hard-coding these requirements into the services and the ESB renders the whole approach inflexible whereas flexibility is one of

the main goals of a SOA, DPR at runtime should be available. With this approach the ESB would read the policy requirements that a request states and would map this request to a provider able to meet the demands. Table 3.13 shows how points are distributed for this criterion.

Points	Requirement
0	ESB does not offer DPR at all
2	ESB can be extended to support DPR
4	ESB offers native DPR

Table 3.13: Points for Dynamic Policy Resolution

C.2: **Service Registry**: ESBs need to offer a Service Registry to allow services to make themselves known to the ESB. Important factors therein are:

- WSDL support
- Support for adding metadata to services
- Registration of services at runtime without the need to restart the ESB or the registry (on-the-fly registration)
- A front end for service management

Therefore, the ESBs are able to earn one point for each factor they support.

3.3.4 Operationalizing Security

As already noted, an ESB has to meet non-functional requirements with one of them being security. Especially in the context of publicly available services within approaches like Cloud computing, security is one of the key aspects.

D.1: **Access Control**, which typically includes means to authenticate, authorize, and audit users and actions is a typical measure to avoid unauthorized usage and modification of an ESB. There even exist frameworks like the Java Authentication and Authorization Service (JAAS). Table 3.14 shows how points are given regarding this criterion.

Points	Requirement
0	ESB does not provide Access Control at all
1	ESB can be extended with component-specific Access Control
2	ESB-specific Access Control is provided
3	ESB can be extended with standardized Access Control by a component
4	ESB provides a standardized Access Control Framework like JAAS

Table 3.14: Points for Access Control

D.2: **Secure Message Transport**: Another security threat is wire tapping, which means that an attacker may record the message flow and get access to confidential information. Therefore, the ESB should provide secure message transport to ensure non-disclosure of transmitted messages. Points given in this context are shown in Table 3.15.

Points	Requirement
0	ESB does not offer Secure Message Transport at all
2	ESB can be extended for Secure Message Transport
4	ESB natively supports Secure Message Transport

Table 3.15: Points for Secure Message Transport

3.3.5 Operationalizing Quality of Services

Aside from security other non-functional requirements are included under the term Quality of Services (QoS) and explained in the following.

E.1: **Availability** and **Scalability** depend on clustering support and are key success factors for service providers. With respect to Cloud computing and hosting of multi-tenant aware services, each downtime could lose a company money (a 1998 report stating as much as $6.5 million an hour [IBM98, p. 1]), customers, trust, and reputation. Modern software makes use of techniques like redundancy and fault instrumentation to avoid or at least reduce downtime. Furthermore, the reliable offering of services depends on the ability to grow or shrink available computing power according to growth and shrinking of the user base. Not growing fast enough leads to unserved and unsatisfied customers whereas not shrinking fast enough leads to unused computing capacity being occupied. Table 3.16 shows how ESBs providing clustering support to achieve availability and scalability are rated.

Points	Requirement
0	ESB does not provide Clustering Support at all
2	ESB can be extended to provide Clustering Support
4	ESB provides native Clustering Support

Table 3.16: Points for Clustering Support

E.2: **Transactions** are means to run critical operations and thereby ensure ACID properties namely atomicity, consistency, isolation, and durability. ESBs supporting this are ranked as shown in Table 3.17.

Points	Requirement
0	ESB does not provide Transactions at all
2	ESB provides Transactions dependent on the transport protocol used
4	ESB provides Transactions independent of other criteria

Table 3.17: Points for Transactions

3.3.6 Operationalizing Multi-Tenancy

With regards to Cloud computing and provisioning an ESB as PaaS, multi-tenancy is a key factor to achieve profitable use of existent computer hardware.

F.1: **Data Isolation** is critical to achieve isolation of tenants and prevents users and tenants from accessing information that others own. Table 3.18 shows, which means of Data Isolation influence the ranking in which way.

Points	Requirement
0	ESB does not offer Data Isolation support
4	ESB offers Data Isolation support

Table 3.18: Points for Data Isolation

F.2: **Performance Isolation** is another important aspect regarding isolation of tenants. None of the tenants should be able to use processing power in such an exhaustive way that other tenants are negatively affected. Table 3.19 shows the distribution of points regarding this criterion.

Points	Requirement
0	ESB does not offer Performance Isolation support
4	ESB offers Performance Isolation support

Table 3.19: Points for Performance Isolation

F.3: **Tenant Based Identity and Access Management** (IAM) enables effective administration of tenants. Table 3.20 shows the points given.

3.3.7 Operationalizing Extensibility

As soon as the functionality of an ESB is no longer enough, customers have to develop custom components. This renders extensibility of a given product important.

G.1: **Plug-In Mechanism** is the offering of well-described interfaces to achieve extensibility by providing the ability to develop customized components. It aims at lowering the effort needed to extend the ESB by further features not being delivered with the product itself. Different to

Points	Requirement
0	ESB does not support tenant based IAM
2	ESB offers at least support for different users
4	ESB fully supports IAM

Table 3.20: Points for Tenant Based Identity and Access Management

the criterion "Custom Adapters", plug-ins are means to extend the product itself and not only the adapters to other systems. Table 3.21 shows the points given for this criterion.

Points	Requirement
0	ESB does not provide a Plug-In Mechanism at all
4	ESB provides a Plug-In Mechanism

Table 3.21: Points for Providing a Plug-In Mechanism

G.2: **Documentation** needs to be detailed and well written to enable the development of extensions and custom components for an ESB product. Points for it being available are given according to Table 3.22.

Points	Requirement
0	Product does not offer any Documentation at all
1	The Documentation offered describes only basic features
2	The Documentation offered is fragmentary and out of date
3	The Documentation offered lacks only detailed topics
4	The product offers an extensive, well written and up to date Documentation

Table 3.22: Points for Providing a Documentation

G.3: **Community** is a key factor in the context of open source products, but can also be important for setting up and customizing proprietary products. Table 3.23 shows how the evaluation takes this into account.

G.4: **Training** is important to get to know a product and learn how to handle it. The evaluation takes into account which means are offered regarding this criterion as shown in Table 3.24.

G.5: **Licensing** is important with regards to the operation of an ESB product. Aside from proprietary licenses that involve costs, open source licenses offer the use and modification of the product without any expenses. Table 3.25 shows how the evaluation considers this criterion.

Points	Requirement
0	There seems to be no active Community
1	The Community is small, activity takes place sporadically
2	There is a small, but active Community
3	The Community is numerous and active, but not supporting
4	The Community is numerous, active, and supporting

Table 3.23: Points for Community

Points	Requirement
0	No means of Training are offered
4	Training is offered

Table 3.24: Points for Providing Training

Points	Requirement
0	The product is offered with a proprietary license and charged for
1	The product is offered as free software, but with a proprietary license
2	The product is offered with a custom free software license
3	A custom, but widely standard derived free software license is offered
4	The product is offered with a standard free software license[a]

[a](e.g., Apache License, BSD License, MIT License, GPL)

Table 3.25: Points for Licensing

3.4 Evaluation of ESB Products

After the operationalization of criteria in Section 3.3 this section shows the ranking of each product including some details on the individual evaluations and preliminary notes. The roundup in Subsection 3.4.8 summarizes the results in a table.

3.4.1 Apache ServiceMix

> "Apache ServiceMix is a flexible, open-source integration container that unifies the features and functionality of Apache ActiveMQ, Camel, CXF, ODE, Karaf into a powerful runtime platform you can use to build your own integrations solutions. It provides a complete, enterprise ready ESB exclusively powered by OSGi." [Theo]

The Apache Software Foundation maintains and releases Apache ServiceMix under the Apache License v2.0 [cf. Theq]. This section explains its rating regarding the different categories of criteria. In total it ranks at 66 points within this evaluation, Table 3.27 bundles the ranking in a comprehensive overview.

Preliminary Notes

Apache ServiceMix (further: ServiceMix) is fully Java based. It can run standalone and within any Java EE application server or servlet engine [cf. Thew]. The documentation provides detailed instructions on how to integrate ServiceMix with Geronimo, JBoss, JOnAS and Tomcat [cf. Thew].

Evaluation of Messaging

Because ServiceMix focuses on the integration with other Apache projects, it integrates well with Apache Camel. This renders the full scale of EIPs available that are supported by Apache Camel, which counts nearly all patterns that Hohpe and Woolf [cf. HW03] describe [cf. Theg]. Furthermore, Apache Camel provides close to all routing patterns and message transformations the same authors describe [cf. Theg], these are also available to ServiceMix. This ranks ServiceMix at four points regarding the patterns and three points for the need of an extension to provide message transformation. Table 3.26 shows the patterns Apache Camel supports. Furthermore, the product makes use of a Normalized Message Router to decouple service providers and consumers and keeping the opportunity to do further processing on the messages [cf. Thes], ranking the product at four points regarding this criterion.

Messaging Systems	Message Routing	Message Transformation	Messaging Endpoints
Message Channel	Content Based Router	Content Enricher	Messaging Mapper
Message	Message Filter	Content Filter	Event Driven Consumer
Pipes and Filters	Dynamic Router	Claim Check	Polling Consumer
Message Router	Recipient List	Normalizer	Competing Consumers
Message Translator	Splitter	Sort	Message Dispatcher
Message Endpoint	Aggregator	Validate	Selective Consumer
	Resequencer		Durable Subscriber
	Composed Message Processor		Idempotent Consumer
	Scatter-Gather		Transactional Client
	Routing Slip		Messaging Gateway
	Throttler		Service Activator
	Sampling		
	Delayer		
	Load Balancer		
	Multicast		
	Loop		

Messaging Channels	Message Construction	System Management
Point to Point Channel	Event Message	Detour
Publish Subscribe Channel	Request Reply	Wire Tap
Dead Letter Channel	Correlation Identifier	Log
Guaranteed Delivery	Return Address	
Message Bus		

Table 3.26: Patterns and Transformations Apache ServiceMix Supports

Evaluation of Integration

Again ServiceMix benefits from its tight integration with other Apache products. In this case it can integrate with the BPEL engine Apache ODE for full support of business processes written according to the WS-BPEL standard [cf. Thek], ranking the product at three points. Similarly ServiceMix can access the full potential of Apache CXF, which leads to broad support of WS-* standards [cf. Thei], namely WS-Addressing, WS-Policy, WS-ReliableMessaging, WS-Security, WS-SecurityPolicy, WS-SecureConversation, and WS-Trust with more planned. This leads to four points for providing WS-* standards. Because ServiceMix is JBI based, it fully supports this standard, but is not certified accordingly [cf. Thep], ranking the product at three points. Regarding the adapters count, ServiceMix already provides an extensive set on its own, but can again integrate with Apache Camel to provide even more [cf. Thej][cf. Theb], ranking ServiceMix at four points. As soon as the adapters provided are not enough anymore, the JBI based architecture supports the development of custom adapters [cf. Thep], ranking the product at another four points. Regarding the transport mechanisms deemed important in this master's thesis, ServiceMix offers bundled components for SOAP over HTTP and components earning points in this evaluation, namely JMS, XMPP, and e-mail [cf. Thej]. It further offers support for File System, File Transfer Protocol (FTP), Short Message Peer-to-Peer (SMPP), and WS-Notification. Again this list is highly extensible through the integration of Apache Camel leading to another point for supporting the SIP protocol and ranking at four points in total regarding transport protocol count. Via Apache Camel, ServiceMix supports the following protocols [cf. Theb]:

- Advanced Message Queuing Protocol (AMQP)
- Atom
- Amazon Web Services
- Several Event Stream Processing protocols (Esper, Spring ApplicationEvents, Open Services Gateway Initiative Framework (OSGi) EventAdmin)
- Google App Engine
- Health Level Seven Minimal Lower Layer Protocol (HL7 MLLP)
- Internet Relay Chat (IRC)
- JavaSpaces
- JMX notifications
- Kestrel
- Remote Method Invocation (RMI)
- RSS
- Secure Shell (SSH) File Transfer Protocol (SFTP)
- Simple Network Management Protocol (SNMP)

- Several databases via native and Java DataBase Connectivity (JDBC) connectors
- Any Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) connection via Apache MINA

Evaluation of Runtime

ServiceMix does not support dynamic resolution of policy requirements, therefore ranking the product at zero points. Based on its open source foundation it can be extended to offer such functionality, but it needs custom development to achieve this. ServiceMix does not offer a service registry [cf. Thex], resulting in a ranking of zero points.

Evaluation of Security

ServiceMix makes use of the JAAS framework [cf. Theu], therefore delivering security features as expected and ranking at four points. The Lightweight Directory Access Protocol (LDAP) is another option to maintain access control. It is able to secure the message transport either via Secure Sockets Layer (SSL) or WS-Security with the latter not being fully supported yet [cf. Thet]. The security features rank ServiceMix at another four points.

Evaluation of QoS

ServiceMix provides native clustering support in several ways to provide availability and scalability [cf. Thel], which leads to a ranking at four points. Transactions are only partly supported depending on the transport protocol used [cf. Thev]. This downside ranks the product at only two points.

Evaluation of Multi-Tenancy

Because ServiceMix is not yet tenant aware, no measures to support multi-tenancy are offered by the product and no points are given regarding multi-tenancy functionality.

Evaluation of Extensibility

The JBI based design of ServiceMix makes it easy to extend the product by further functionality [cf. Thep], ranking it at four points. The documentation is extensive, but still misses or is outdated for some detailed topics [cf. Then]. This leads to three points. The community behind ServiceMix is noteworthy and includes several developers working with large corporations [cf. Them], but there is no training offered. Thus, ServiceMix earns four points for its community, but zero points for not providing any training. As stated at the beginning of this section, ServiceMix is developed and distributed under the Apache License v2.0, ranking it at four points.

Category	Criterion	Points	Short Statement
A Messaging	A.1 Support for EIP	4	Many patterns supported
	A.2 Routing	4	
	A.3 Message Transformation	3	Arbitrary transformations via extension
	A.4 Normalized Message Format	4	Normalized Message Format used
B Integration	B.1 BPEL support	3	Integration of BPEL engine possible
	B.2 WS-* standards	4	Several important supported
	B.3 JBI support	3	Built on JBI, but not certified
	B.4 Adapters	4	Extensive set of adapters, further extensible
	B.5 Custom Adapters	4	Completely supported
	B.6 Transport Mechanisms	4	JMS, XMPP, e-mail, SIP plus many more
C Runtime	C.1 Dynamic Policy Resolution	0	Not supported
	C.2 Service Registry Support	0	No service registry
D Security	D.1 Access Control	4	WS-Security, JAAS, SSL, LDAP
	D.2 Secure Message Transport	4	
E QoS	E.1 Availability and Scalability	4	Native clustering support
	E.2 Transactions	2	Supported dependent on transport protocol
F Multi-Tenancy	F.1 Data Isolation	0	No multi-tenancy support
	F.2 Performance Isolation	0	
	F.3 Tenant Based Identity and Access Management	0	
G Extensibility	G.1 Plug-In Mechanism	4	Fully extensible based on JBI
	G.2 Documentation	3	Lacks only detailed topics
	G.3 Community	4	Numerous, supporting, and active
	G.4 Training	0	No training offered
	G.5 Licensing	4	Apache License v2.0
Total		**66**	

Table 3.27: Ranking of Apache ServiceMix

3.4.2 BizTalk Server

> "BizTalk Server is Microsoft's Integration and connectivity server solution. [...] BizTalk Server 2010 provides a solution that allows organizations to more easily connect disparate systems." [cf. Mica]

Microsoft develops and sells BizTalk Server, one of the proprietary ESB products evaluated in this master's thesis. The rankings within each category of criteria are discussed in detail in this section with BizTalk Server ranking at 53 points in total within this evaluation. Table 3.29 bundles the ranking in a comprehensive overview.

Preliminary Notes

Microsoft's BizTalk Server (further: BizTalk Server) bases on the .NET framework with Visual Studio being its development platform. Delivering some extra database supported functionality, the product further relies on Microsoft SQL Server [cf. Mica].

Evaluation of Messaging

EIPs are not by default supported, but manual orchestration of Web services via the tools bundled with the product can achieve a similar purpose [cf. HT04, p. 12 f.]. This leads to a ranking of two points. Routing patterns are supported out of the box, but come in a limited number of variations [cf. Mick]. Message transformations are offered in an even more limited way, but are also provided with BizTalk Server [cf. Mici]. The resulting ranking leads to two points for both criteria, Table 3.28 shows the patterns and transformations BizTalk Server supports. The product offers means to normalize messages, but does not make use of a Normalized Message Format for its own messaging purposes [cf. Micj]. This ranks BizTalk Server at zero points regarding this criterion.

Routing Patterns	Transformation Patterns
Message Router	Message Translator
Content-Based Router	Normalizer
Routing Slip	Content Enricher
Scatter-Gather	
Recipient List	
Splitter	

Table 3.28: Patterns and Transformations BizTalk Server Supports

Evaluation of Integration

BizTalk Server's support for BPEL is limited. It is possible to import and export BPEL, but internally the product works with its own solution [cf. Micd]. Because it does not implement a real BPEL engine, BizTalk Server ranks at two points. WS-* standards however are supported to an extensive degree, namely WS-Addressing, WS-Policy, WS-Security, WS-Trust, WS-SecureConversation, WS-ReliableMessaging, WS-AtomicTransaction, and WS-Coordination [cf. Mico], resulting in four points. Because JBI is a Java based standard with BizTalk Server being built on the .NET platform, it offers no JBI support whatsoever, leading to a ranking of zero points. The product comes with numerous adapters targeted on wide spread and smaller systems [cf. Micc]. Nonetheless, it even offers a framework lowering the effort needed to develop custom adapters [cf. Micf], resulting in four points regarding both criteria. The connectors to different transport mechanisms are also called adapters by Microsoft and are provided in limited ways [cf. Micc], leading to only one point for supporting the basic SOAP over HTTP protocol and e-mail. Apart from the transports evaluated in this master's thesis, BizTalk Server further offers support of Electronic data interchange (EDI), File, FTP, Microsoft Message Queuing (MSMQ), TIBCO Enterprise Message Service, TIBCO Rendezvous, WebSphere MQ, Windows Communication Foundation (WCF), Windows SharePoint Services, and databases of IBM and Oracle [cf. Micc].

Evaluation of Runtime

Microsoft's BizTalk Server offers native dynamic policy resolution at runtime [cf. Mich]. Its service registry is able to handle WSDL, offers a graphical interface, can handle metadata, and offers hot-deployment. Therefore, the product ranks at four points regarding both criteria.

Evaluation of Security

BizTalk Server offers access control in many ways, but all are specific to the product [cf. Micb], resulting in two points. Furthermore, the product provides native secure message transport [cf. Micl] and therefore ranks at four points.

Evaluation of QoS

Availability and scalability can be achieved by clustering the underlying Microsoft Windows Servers. This represents an extension of the BizTalk Server [cf. Mice] and results in two points. Transactions however are natively supported independent of any other criteria [cf. Micn] and rank the product at four points.

Category	Criterion	Points	Short Statement
A Messaging	A.1 Support for EIP	2	Manual orchestration needed
	A.2 Routing	2	Limited support
	A.3 Message Transformation	2	
	A.4 Normalized Message Format	0	No Normalized Message Format used
B Integration	B.1 BPEL support	2	Conversion of BPEL to internal format
	B.2 WS-* standards	4	Several important supported
	B.3 JBI support	0	No JBI support
	B.4 Adapters	4	Extensive set of adapters
	B.5 Custom Adapters	4	Fully supported
	B.6 Transport Mechanisms	1	E-mail plus few more
C Runtime	C.1 Dynamic Policy Resolution	4	Natively supported
	C.2 Service Registry Support	4	WSDL, metadata, hot-registration, front end
D Security	D.1 Access Control	2	Product specific means offered
	D.2 Secure Message Transport	4	Natively supported
E QoS	E.1 Availability and Scalability	2	Clustering via extension possible
	E.2 Transactions	4	Natively supported
F Multi-Tenancy	F.1 Data Isolation	0	No multi-tenancy support
	F.2 Performance Isolation	0	
	F.3 Tenant Based Identity and Access Management	0	
G Extensibility	G.1 Plug-In Mechanism	0	No plug-in mechanism provided
	G.2 Documentation	4	Comprehensive, detailed, up to date
	G.3 Community	4	Numerous, active, and supporting
	G.4 Training	4	Available in many different ways
	G.5 Licensing	0	Proprietary
Total		**53**	

Table 3.29: Ranking of BizTalk Server

Evaluation of Multi-Tenancy

Because BizTalk Server is not multi-tenant aware, none of the criteria within this category are served and no points are achieved.

Evaluation of Extensibility

Apart from the custom adapters that Microsoft offers a framework for, BizTalk Server does not offer a plug-in mechanism. Therefore, it cannot be extended and ranks at zero points. The documentation is comprehensive and extensive, up to date, and fully detailed [cf. Micg], ranking BizTalk Server at four points. The community is numerous, active, and supporting, ranking BizTalk Server at four points regarding this criterion. Microsoft offers training in manifold ways [cf. Micm] whereas the license is proprietary and charged for. Therefore, BizTalk Server ranks at four points for providing training, but zero points for its proprietary license.

3.4.3 JBossESB

> "JBossESB is the next generation of EAI [...]. 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." [JBob]

RedHat develops and distributes JBossESB as a part of the JBoss SOA Platform. It offers the product under the GNU Lesser General Public License v2.1 (LGPL v2.1) [cf. JBol] and scores 53 points in this ranking. This section shows the fully detailed evaluation regarding this product, Table 3.31 bundles the ranking in a comprehensive overview.

Preliminary Notes

JBossESB is Java based and runs standalone or integrated into a standard Java application server [cf. JBol]. Apart from the Java runtime and the Java based software build automation tool Apache Ant there are no further prerequisites needed to run JBossESB.

Evaluation of Messaging

JBossESB offers limited support for EIPs. Some are bundled with the product, but others have to be implemented by the customer [cf. JBoi], ranking it at two points. The same applies to routing capabilities [cf. JBoo], yet they can be extended by sophisticated business rules via integration with JBoss Drools [cf. JBoh], leading to four points for this criterion. Table 3.30

shows the patterns supported. Regarding message transformations JBossESB again makes use of the integration with other components. The Smooks Data Transformation/Processing Framework plugs into the ESB for versatile message transformation support [cf. JBom], ranking the product at three points. JBossESB further comes with a fully defined message format described in [JBon], resulting in four points for the support of a Normalized Message Format.

Messaging Channels	Message Routing	Message Transformation
Dead Letter Channel	Content-Based Routing	Content Filter
	Message Filter	
	Dynamic Router	
	Recipient List	

Messaging Endpoints	System Management
Dead Letter Service	Message Store
	Wire Tap

Table 3.30: Patterns and Transformations JBossESB Supports

Evaluation of Integration

Although JBossESB does not come with integrated BPEL support, another JBoss product named jBPM can extend it for support of BPEL and its own process definition language jPDL [cf. JBoc]. Therefore, it ranks at three points regarding BPEL support. It does not support any WS-* standards by default, but JBossWS extends it for the support of WS-Security, WS-Addressing, WS-ReliableMessaging, WS-Eventing, and WS-Policy [cf. JBor][cf. JBos]. There is no JBI support at the moment, but may be available in the future [cf. JBor]. This results in two points for supporting up to six WS-* standards and zero points for JBI support. JBossESB comes with no adapters to standard systems, but allows the implementation of custom adapters within certain limitations [cf. JBof], resulting in zero points for adapter count and two points for custom adapter development support. The transport mechanisms provided include the basic SOAP over HTTP protocol and the ranked JMS and e-mail protocols, but no XMPP or SIP [cf. JBoj]. This results in two points. Apart from the ranking, JBossESB supports InVm, TCP, File, and databases without mentioning further details on specific protocols.

Evaluation of Runtime

JBossESB does not support any policy resolution by default, but JBossWS extends it to enable this feature [cf. JBok], ranking the product at two points. The integrated service registry offers WSDL support, on-the-fly registration of services, and limited metadata support, but it does not offer a front end [cf. JBop], resulting in a ranking of three points.

Category	Criterion	Points	Short Statement
A Messaging	A.1 Support for EIP	2	Limited number of patterns supported
	A.2 Routing	4	Extensible for business rule support
	A.3 Message Transformation	3	Extensible for arbitrary transformations
	A.4 Normalized Message Format	4	Normalized Message Format used
B Integration	B.1 BPEL support	3	Integration of BPEL engine possible
	B.2 WS-* standards	2	Extensible to support some important
	B.3 JBI support	0	No JBI support (planned in the future)
	B.4 Adapters	0	No adapters bundled
	B.5 Custom Adapters	2	Limited support
	B.6 Transport Mechanisms	2	JMS, e-mail plus few more
C Runtime	C.1 Dynamic Policy Resolution	2	Support extensible
	C.2 Service Registry Support	3	WSDL, metadata, hot-registration
D Security	D.1 Access Control	2	Product specific
	D.2 Secure Message Transport	2	Extensible for WS-Security/WS-Trust
E QoS	E.1 Availability and Scalability	4	Native clustering support
	E.2 Transactions	2	Dependent on transport protocol
F Multi-Tenancy	F.1 Data Isolation	0	No multi-tenancy support
	F.2 Performance Isolation	0	
	F.3 Tenant Based Identity and Access Management	0	
G Extensibility	G.1 Plug-In Mechanism	0	No plug-in mechanism provided
	G.2 Documentation	4	Extensive, detailed, up to date
	G.3 Community	4	Numerous, supporting, and active
	G.4 Training	4	Available in many different ways
	G.5 Licensing	4	GNU Lesser General Public License v2.1
Total		**53**	

Table 3.31: Ranking of JBossESB

Evaluation of Security

JBossESB provides product specific access control on the one hand [cf. JBoq] and security protocols like WS-Security/WS-Trust via the extension by JBossWS on the other hand [cf. JBok]. Therefore, the product ranks at two points each regarding access control and secure message transport.

Evaluation of QoS

JBossESB comes with full clustering capabilities to ensure availability and scalability [cf. JBod], ranking the product at four points for full clustering support. Furthermore, it is able to run transactions with JMS and Structured Query Language (SQL) resources [cf. JBoj], therefore ranking at two points for transaction support dependent on the transport protocol used.

Evaluation of Multi-Tenancy

Because JBossESB is not multi-tenant aware, it ranks at zero points regarding this category of criteria.

Evaluation of Extensibility

Apart from the custom adapters mentioned, JBossESB does not come with a plug-in mechanism. Therefore, it is not extensible and ranks at zero points. The product documentation is extensive, up to date and understandable [cf. JBog], the associated community is numerous, active, and supporting [cf. JBoe]. Training is available by certified instructors around the globe [cf. JBoa]. JBossESB comes with an open source license, namely the GNU Lesser General Public License v2.1 (LGPL v2.1) [cf. JBol]. This results in a ranking of four points for each of the four criteria mentioned.

3.4.4 Mule ESB

> "Mule ESB is a lightweight Java-based enterprise service bus (ESB) and integration platform that allows developers to connect applications together quickly and easily, enabling them to exchange data. Mule ESB enables easy integration of existing systems, regardless of the different technologies that the applications use, including JMS, Web Services, JDBC, HTTP, and more." [Mulp]

MuleSoft maintains and releases Mule ESB under the Common Public Attribution License v1.0 (CPAL v1.0) [cf. Mulj]. In total the product scores 57 points in this ranking. This section explains the detailed evaluation regarding each category of criteria, Table 3.33 bundles the ranking in a comprehensive overview.

Preliminary Notes

As Mule ESB is Java based, a Java runtime environment is mandatory. It can either run standalone or within several Java EE application servers. The documentation mentions that Tcat Server, Tomcat, WebLogic, WebSphere, Geronimo, JBoss, Jetty, and Resin are known to work with Mule ESB [cf. Muln]. There are no specific database requirements, known to work with the product are Derby, MySQL, Oracle, Informix, and Sybase databases [cf. Muln].

Evaluation of Messaging

Table 3.32 shows the various EIPs and routing patterns Hohpe and Woolf [cf. HW03] describe that Mule ESB supports [cf. Mulg]. This ranks the product at four points for each criterion. It further comes with many message transformers and still offers the possibility to implement custom transformations in any way needed, therefore also achieving four points regarding this criterion [cf. Mulk]. Mule ESB does not make use of a Normalized Message Format, ranking it at zero points.

Evaluation of Integration

Mule ESB does not support BPEL by default, but offers the integration with BPM engines capable of BPEL that provide a Java Application Programming Interface (API) like JBoss jBPM [cf. Mulb], resulting in a ranking of three points. It supports the WS-* standards WS-Security, WS-Addressing, WS-Policy, and WS-Integration [cf. Muln], which leads to a ranking of two points for supporting up to six standards. Although Mule ESB does not base on JBI itself, it comes with endpoints that enable it to integrate into a JBI environment [cf. Mulh], leading to two points regarding this criterion. Mule ESB comes with an extensive set of adapters to applications [cf. Mula], ranking it at four points. Nonetheless, it still offers means to implement custom adapters and share them with the Mule ESB community if desired [cf. Mule] leading to four points for custom adapter support. Mule ESB further offers support for SOAP over HTTP and the points earning transports JMS, XMPP, and e-mail, but does not offer SIP [cf. Muld], leading to three points for supporting three of the four transport protocols evaluated in this master's thesis. Beyond these, Mule ESB supports File, FTP, TCP, and UDP [cf. Muld].

Evaluation of Runtime

Mule ESB makes use of a service registry, called Galaxy, that MuleSoft develops. It can save policies, but does not ensure dynamic policy resolution at runtime [cf. Muli]. Therefore, it ranks at zero points. It handles WSDL and service metadata, can register and publish services to Mule ESB at runtime, and offers a Web interface [cf. Muli]. Therefore, it provides all four factors demanded in this evaluation and ranks at four points regarding the service registry.

Inbound Routers	Outbound Routers	Async-Reply Routers	Catch-All Strategies
No Router	Pass-through	Single	Forwarding
Selective Consumer	Filtering	Collection	Custom Fowarding
Idempotent Receiver	Recipient List Routers	Custom	Logging
Idempotent Secure Hash Receiver	Multicasting		Custom
Collection Aggregator	Chaining		
Message Chunking Aggregator	List Message Splitter		
Custom Correlation Aggregator	Filtering XML Message Splitter		
Correlation Resequencer	Expression Splitter Router		
Forwarding	Message Chunking Router		
WireTap	Exception Based Routers		
Custom	Template Endpoint		
	Round Robin Message Splitter		
	Custom		

Table 3.32: Patterns and Transformations Mule ESB Supports

Category	Criterion	Points	Short Statement
A Messaging	A.1 Support for EIPs	4	Many patterns supported
	A.2 Routing	4	
	A.3 Message Transformation	4	Arbitrary transformations possible
	A.4 Normalized Message Format	0	No Normalized Message Format used
B Integration	B.1 BPEL support	3	Integration of BPEL engines possible
	B.2 WS-* standards	2	Some important supported
	B.3 JBI support	2	Not built on JBI, but offers adapters
	B.4 Adapters	4	Extensive set of adapters
	B.5 Custom Adapters	4	Fully supported
	B.6 Transport Mechanisms	3	JMS, XMPP, e-mail plus many more
C Runtime	C.1 Dynamic Policy Resolution	0	Not supported
	C.2 Service Registry Support	4	WSDL, metadata, hot-registration, front end
D Security	D.1 Access Control	4	WS-Security, JAAS, SAML, PGP, LDAP
	D.2 Secure Message Transport	4	
E QoS	E.1 Availability and Scalability	0	Only commercial Mule ESB
	E.2 Transactions	0	
F Multi-Tenancy	F.1 Data Isolation	0	No multi-tenancy support
	F.2 Performance Isolation	0	
	F.3 Tenant Based Identity and Access Management	0	
G Extensibility	G.1 Plug-In Mechanism	0	No plug-in mechanism provided
	G.2 Documentation	3	Lacks only detailed topics
	G.3 Community	4	Numerous, supporting, and active
	G.4 Training	4	Available in many different ways
	G.5 Licensing	4	Common Public Attribution License v1.0
Total		**57**	

Table 3.33: Ranking of Mule ESB

Evaluation of Security

With its offering of WS-Security, JAAS, Security Assertion Markup Language (SAML), Pretty Good Privacy (PGP) encryption, and LDAP connectivity, Mule ESB provides the means to support access control and secure message transport and ranks at four points each.

Evaluation of QoS

The open source version of Mule ESB is not able to provide clustering to achieve availability and scalability, and multi-resource transactions. A commercial license for the Mule ESB Enterprise Edition offers these functionalities [cf. Mull]. Because the evaluation in this master's thesis focuses on the open source Mule ESB Community Edition, it ranks at zero points regarding each criterion.

Evaluation of Multi-Tenancy

Mule ESB ranks at zero points in this category for not providing multi-tenancy at all.

Evaluation of Extensibility

Other than the custom transports and custom adapters Mule ESB allows to develop, it is not extensible by plug-ins or similar and therefore ranks at zero points. The documentation is extensive and well written, but lacks some detailed topics [cf. Mulf] resulting in a rating of three points. Mule ESB's community is huge, active and supporting, and provides an astonishing amount of extra adapters and transports at MuleSoft's developer centre MuleForge [cf. Mulc][cf. Mulm]. Therefore, Mule ESB ranks at four points regarding its community. Training is available in a multitude of different ways and ranges from free online video instructions to custom onsite trainings [cf. Mulo] and therefore ranks at four points as well. The license is an open source license namely the Common Public Attribution License v1.0 (CPAL v1.0) [cf. Mulj]. Therefore, Mule ESB ranks at another four points regarding this criterion.

3.4.5 Petals ESB

> "Petals ESB is an open source ESB (Enterprise Service Bus) for large SOA architectures. Our original design, distributed across multiple servers and full compatible with all major industry standards (such as JBI, SCA, BPEL or WSDL) enables users to: reduce maintenance, easily adapt large-scale infrastructure, quickly implement the architecture, thanks to free development tools" [Peta]

OW2 hosts and Petals Link coordinates Petals ESB as an open source ESB available under the GNU Lesser General Public License v2.1 (LGPL v2.1) [cf. Petj]. It scores 63 points in this ranking. This section gives detailed explanation regarding each criterion. Table 3.34 bundles the ranking in a comprehensive overview.

Preliminary Notes

Being Java based Petals ESB relies on a Java runtime and furthermore needs a MySQL database [cf. Petr]. It runs standalone without the possibility to deploy it to an application server.

Evaluation of Messaging

Petals ESB offers limited enterprise application integration patterns with the ability to extend the EIP component by custom development to support further patterns [cf. Petm]. Patterns supported are: Aggregator, Bridge, Dispatcher, Router, DynamicRouter, RoutingSlip, ScatterGather, WireTap, and Splitter. This ranks the product at two points for limited EIP support. It further provides a module based routing component enabling sophisticated routing capabilities [cf. Peto], resulting in a ranking of four points. Message transformation is possible in arbitrary ways with built-in Extensible Stylesheet Language Transformations (XSLT) support [cf. Petk]. The Petals ESB further makes use of a Normalized Message Format [cf. Petl]. This ranks it at four points each for message transformation support and the use of a Normalized Message Format.

Evaluation of Integration

Petals ESB offers orchestration support in several ways. One of them is a BPEL runtime [cf. Petn] that ranks the product at four points. It does not support WS-* standards any further than some WS-Security support for SOAP calls [cf. Petq], but it fully implements the JBI specification and is even certified as such [cf. Peti]. This results in zero points for WS-* standards, but four points for JBI support. Being based on the JBI specification, Petals ESB comes with several adapters [cf. Petf]. Nonetheless, the JBI conformance allows to plug any custom component into Petals ESB as long as it is JBI compliant as well [cf. Petc]. This leads to a ranking of four points each for adapter count and custom adapter support. Regarding the transport mechanisms Petals ESB supports the basic SOAP over HTTP protocol and ranks at one point each for support of JMS and e-mail, but missing XMPP and SIP [cf. Petf]. It further supports File, FTP, SFTP, and databases via JDBC connectors [cf. Petf].

Category	Criterion	Points	Short Statement
A Messaging	A.1 Support for EIPs	2	Limited pattern support
	A.2 Routing	4	Sophisticated routing support
	A.3 Message Transformation	4	Arbitrary transformations possible
	A.4 Normalized Message Format	4	Normalized Message Format used
B Integration	B.1 BPEL support	4	Native BPEL support
	B.2 WS-* standards	0	Only WS-Security for SOAP
	B.3 JBI support	4	JBI support certified
	B.4 Adapters	4	Extensive set of adapters
	B.5 Custom Adapters	4	Fully supported
	B.6 Transport Mechanisms	2	JMS, e-mail plus some more
C Runtime	C.1 Dynamic Policy Resolution	0	Not supported
	C.2 Service Registry Support	2	WSDL, hot-registration
D Security	D.1 Access Control	4	JAAS framework
	D.2 Secure Message Transport	4	Secured internal protocol
E QoS	E.1 Availability and Scalability	4	Native clustering support
	E.2 Transactions	0	No transaction support
F Multi-Tenancy	F.1 Data Isolation	0	No multi-tenancy support
	F.2 Performance Isolation	0	
	F.3 Tenant Based Identity and Access Management	0	
G Extensibility	G.1 Plug-In Mechanism	4	Fully extensible based on JBI
	G.2 Documentation	3	Lacks only detailed topics
	G.3 Community	2	Small, supporting, and active
	G.4 Training	4	Available within Europe
	G.5 Licensing	4	GNU Lesser General Public License v2.1
Total		**63**	

Table 3.34: Ranking of Petals ESB

Evaluation of Runtime

Petals ESB does not offer any native policy resolution at runtime, resulting in zero points regarding this criterion. The integrated service registry can handle WSDL and registering of services at runtime, but offers no front end and is not able to store metadata [cf. Petp], therefore ranking at two points.

Evaluation of Security

The router module in Petals ESB contains an authorization module using JAAS for access authorization [cf. Petb], ranking the product at four points. Secure message transport is provided independent of the transport protocol used, because the Normalized Messages are handled by one standard and secured Petals ESB inter-communication protocol [cf. Petg], ranking the product at another four points.

Evaluation of QoS

Petals ESB offers clustering configurations to achieve higher availability and scalability [cf. Petd]. It is not capable of distributed transaction, but can work with compensations known from BPEL [cf. Pett]. This results in a ranking of four and zero points in this category.

Evaluation of Multi-Tenancy

The Petals ESB does not support multi-tenancy resulting in a ranking of zero points within this category.

Evaluation of Extensibility

Because of the JBI conformance, users can extend Petals ESB in any way according to the JBI specification [cf. Peti], ranking the product at four points. The documentation is well-written, but lacks some detailed topics and is outdated regarding a few aspects [cf. Peth], leading to three points. There is further a small, but active and supportive community [cf. Pete]. Training is offered although only within Europe [cf. Pets]. Therefore, the two criteria lead to a ranking of two and four points. Petals ESB licenses under the GNU Lesser General Public License v2.1 (LGPL v2.1) [cf. Petj]. The product ranks at four points accordingly.

3.4.6 WebSphere ESB

> "WebSphere ESB is a flexible connectivity infrastructure for integrating applications and services. It allows the development of an SOA. [...] WebSphere ESB is ideal if you have other solutions on WebSphere Application Server, WebSphere Portal, or the BPM platform." [IBMa]

IBM develops and distributes the WebSphere Enterprise Service Bus, which is another proprietary product evaluated in this master's thesis. This section shows its ranking within the different categories of criteria where the product ranked at 62 points in total. Table 3.36 bundles the ranking in a comprehensive overview.

Preliminary Notes

IBM WebSphere ESB (further: WebSphere ESB) is Java based and strongly integrated with other IBM products rendering the IBM WebSphere Application Server mandatory with no support for alternatives. It makes use of IBM's own databases, Apache Derby, or Oracle 10g/11g. Other components supported like a JMS provider or a service registry are also IBM centric with no support for other products [cf. IBMm].

Evaluation of Messaging

The WebSphere ESB comes with some of the EIPs that Hohpe and Woolf [cf. HW03] describe, called "Mediation Patterns" by IBM [cf. IBMi]. WebSphere ESB ranks at two points for this criterion. Regarding routing capabilities the product supports the patterns that named authors [cf. IBMn] identify, leading to a ranking of four points. Furthermore, message transformation is extensively supported via the IBM WebSphere Transformation Extender [cf. IBMn], ranking it at three points. Because IBM uses different names for these patterns, they do not map one on one to these Hohpe and Woolf describe, but are still listed in Table 3.35. The product does not make use of a Normalized Message Format, which leads to a ranking of zero points.

Business Object Map	Fan In	Message Logger
Custom Mediation	Fan Out	Policy Resolution
Data Handler	HTTP Header Setter	Service Invoke
Database Lookup	MQ Header Setter	Set Message Type
Endpoint Lookup	SOAP Header Setter	Stop
Event Emitter	Message Element Setter	Type Filter
Fail	Message Filter	XSL Transformation

Table 3.35: Patterns and Transformations WebSphere ESB Supports

Evaluation of Integration

WebSphere ESB on its own does not support BPEL, but can easily integrate with the IBM WebSphere Process Server to offer full BPEL support [cf. IBMc], therefore ranking WebSphere ESB at three points. WS-* standards are said to be supported natively by the product, but because IBM only mentions WS-Security, WS-Policy, and WS-AtomicTransaction [cf. IBMu], the product ranks at zero points for supporting up to three standards. "To date, IBM has abstained from the JBI specification [...]." [IBMh], leading to zero points for JBI support. The product offers a limited set of adapters to connect to flat files, databases via JDBC, PeopleSoft Enterprise, Siebel Business Applications, and SAP Applications [cf. IBMb]. Therefore, it ranks at two points. The possibilities to implement custom adapters are extensive and allow custom adapter development in any way needed [cf. DMC+06], ranking the product at four points. Furthermore, WebSphere ESB comes with a set of transport bindings that includes the basic SOAP over HTTP protocol, but ranks the product at one point for only providing JMS and missing XMPP, e-mail, and SIP [cf. IBMt]. Apart from the protocols evaluated in this master's thesis, WebSphere ESB supports File and databases via JDBC connectors [cf. IBMt].

Evaluation of Runtime

WebSphere ESB fully supports dynamic policy resolution through means of metadata-driven routing [cf. IBMg]. Its service registry can handle WSDL [cf. IBMv], stores metadata via the WebSphere Service Registry [cf. IBMj], supports on the fly registration of services [cf. IBMp] and offers a Web user interface (UI) for accessing the registry [cf. IBMq]. WebSphere ESB ranks accordingly at four points each.

Evaluation of Security

IBM's WebSphere ESB offers product specific access control, ranking it at two points, and native secure message transport [cf. IBMo], ranking it at four points.

Evaluation of QoS

The product fully and natively supports clustering to ensure availability and scalability [cf. IBMd]. Transactions are coordinated by WebSphere Application Server whereas WebSphere ESB only acts as a participant. Nonetheless, transactions are made available independent of other criteria [cf. IBMs]. Therefore, WebSphere ESB ranks at four points regarding each criterion.

Category	Criterion	Points	Short Statement
A Messaging	A.1 Support for EIPs	2	Some patterns supported
	A.2 Routing	4	Full routing capabilities
	A.3 Message Transformation	3	Arbitrary transformations extensible
	A.4 Normalized Message Format	0	No Normalized Message Format used
B Integration	B.1 BPEL support	3	Integration of BPEL engine possible
	B.2 WS-* standards	0	All supported, but only few named
	B.3 JBI support	0	No JBI support
	B.4 Adapters	2	Limited set of adapters
	B.5 Custom Adapters	4	Fully supported
	B.6 Transport Mechanisms	1	JMS plus some more
C Runtime	C.1 Dynamic Policy Resolution	4	Fully supported
	C.2 Service Registry Support	4	WSDL, metadata, hot-registration, front end
D Security	D.1 Access Control	2	Product specific
	D.2 Secure Message Transport	4	Natively supported
E QoS	E.1 Availability and Scalability	4	Native clustering support
	E.2 Transactions	4	Full transaction support extensible
F Multi-Tenancy	F.1 Data Isolation	4	Native multi-tenancy support
	F.2 Performance Isolation	4	
	F.3 Tenant Based Identity and Access Management	4	Extensible by IBM Tivoli Access Manager
G Extensibility	G.1 Plug-In Mechanism	0	No plug-in mechanism provided
	G.2 Documentation	4	Detailed, comprehensive, and up to date
	G.3 Community	1	Small and sporadically active
	G.4 Training	4	Available in many different ways
	G.5 Licensing	0	Proprietary
Total		**62**	

Table 3.36: Ranking of WebSphere ESB

Evaluation of Multi-Tenancy

WebSphere ESB is one of the few products fully supporting multi-tenancy. Therefore, it is able to deliver all three criteria stated within this category: data isolation, performance isolation, and tenant based identity and access management [cf. IBMk][cf. IBMl], the latter by means of the IBM Tivoli Access Manager, and accordingly ranks at four points each.

Evaluation of Extensibility

Apart from custom adapters and custom messaging, routing and transformation patterns, WebSphere ESB offers no extension support, ranking it at zero points. The documentation however is extensive, comprehensive, and up to date [cf. IBMe][cf. IBMf]. Therefore, WebSphere ESB ranks at four points. The community is rather inactive and consists of few members, ranking the product at one point regarding this criterion [cf. Glo]. IBM offers product related training in manifold ways [cf. IBMr], leading to a ranking of four points regarding the training offerings. With the licensing model being proprietary this criterion of the evaluation leads to a ranking of zero points.

3.4.7 WSO2 ESB

> "The ease of downloading and installing, configuring integration through an intuitive graphical interface, help get your project up and going quickly, while the comprehensive set of mediation capabilities and broad set of supported protocols make even complex projects a snap. And the blazing performance and the ability to deploy topologies that give you the best throughput, scaling, and reliability have made the WSO2 ESB a favorite." [WSOa]

WSO2 delivers a full set of enterprise application development focused software with the WSO2 ESB being based on Apache Synapse ESB [cf. WSOq] and available under the Apache License v2.0 [cf. WSOj]. In total the product scores 74 points in this evaluation. This section offers details regarding this ranking. Table 3.38 bundles the ranking in a comprehensive overview.

Preliminary Notes

Because the WSO2 ESB uses Apache Synapse ESB as its base, it is a Java based product rendering a Java runtime mandatory. The product can run standalone [cf. WSOp] and in combination with Java application servers like IBM's WebSphere Application Server [cf. WSOu].

Evaluation of Messaging

WSO2 ESB "supports most of the Enterprise Integration Patterns (EIP)" [WSOh] and offers advanced routing capabilities [cf. WSOh], but uses a different naming convention. Sophisticated message transformations are provided under the term "message mediation" [cf. WSOk]. Table 3.37 shows the patterns supported and available for custom development of sequences by combining these patterns. Furthermore, the product parses incoming messages into an internal format and vice versa, thus making use of a Normalized Message Format [cf. WSOm]. This ranks WSO2 ESB at four points for each of the four criteria within this category.

Core	Filter	Transform	Advanced	Extension
Send	Filter	XSLT	Cache	Class
Log	Out	XQuery	Clone	POJOCommand
Property	In	Header	Iterate	Script
Sequence	Switch	Fault	Aggregate	Spring
Event	Router		Callout	
Drop	Validate		Transaction	
Enrich			Throttle	
			RMSequence	
			DBReport	
			DBLookup	
			Rule	
			Entitlement	
			OAuth	

Table 3.37: Patterns and Transformations WSO2 ESB Supports

Evaluation of Integration

WSO2 ESB is not able to handle BPEL, but can integrate with the WSO2 Business Process Server for BPEL support [cf. WSOc], ranking the product at three points. Users can extend it to support some WS-* standards, namely WS-Addressing, WS-Security, WS-ReliableMessaging, WS-Eventing, and WS-Policy [cf. WSOh]. The product does not offer JBI support [cf. WSOi]. This results in two points for up to six WS-* standards supported, but zero points for the missing JBI support. The product does not come with any adapters to other applications, but in return offers extensive possibilities to develop custom adapters [cf. WSOf], resulting in zero points for adapter count and four points for custom adapter development support. In addition it interacts with numerous transport protocols [cf. WSOh], including the basic SOAP over HTTP protocol. It further supports JMS and e-mail, but does not support XMPP or SIP, leading to a ranking of two points regarding the transport protocols. Apart from the protocols evaluated, WSO2 ESB supports Microsoft .NET WCF, virtual file systems (e.g., SFTP, File, zip/tar/gz, Web-Based Distributed Authoring and Versioning (WebDAV), Common Internet File System (CIFS), etc.), AMQP, Financial Information Exchange (FIX), and Hessian binary protocol for Web services [cf. WSOh].

Category	Criterion	Points	Short Statement
A Messaging	A.1 Support for EIPs	4	Many patterns supported
	A.2 Routing	4	
	A.3 Message Transformation	4	Arbitrary transformations possible
	A.4 Normalized Message Format	4	Normalized Message Format used
B Integration	B.1 BPEL support	3	Integration of BPEL engine possible
	B.2 WS-* standards	2	Some important extensible
	B.3 JBI support	0	No JBI support
	B.4 Adapters	0	No adapters provided
	B.5 Custom Adapters	4	Fully supported
	B.6 Transport Mechanisms	2	JMS, e-mail plus some more
C Runtime	C.1 Dynamic Policy Resolution	2	Support extensible
	C.2 Service Registry Support	4	WSDL, metadata, hot-registration, front end
D Security	D.1 Access Control	3	LDAP native, XACML extensible
	D.2 Secure Message Transport	4	Natively supported
E QoS	E.1 Availability and Scalability	4	Native clustering support
	E.2 Transactions	2	As participant of JTA provider
F Multi-Tenancy	F.1 Data Isolation	4	Native multi-tenancy support
	F.2 Performance Isolation	4	
	F.3 Tenant Based Identity and Access Management	4	Approach: "super-tenant"
G Extensibility	G.1 Plug-In Mechanism	0	No plug-in mechanism provided
	G.2 Documentation	4	Extensive, comprehensive, and up to date
	G.3 Community	4	Numerous, supporting, and active
	G.4 Training	4	Available in many different ways
	G.5 Licensing	4	Apache License v2.0
Total		**74**	

Table 3.38: Ranking of WSO2 ESB

Evaluation of Runtime

WSO2 ESB does not support dynamic policy resolution, but it can achieve such functionality to a certain extent through integration with the WSO2 Business Rules Server [cf. WSOd], ranking the product at two points. The integrated service registry supports WSDL and service registration at runtime, stores service metadata, and offers a front end [cf. WSOa][cf. WSOo]. Therefore, WSO2 ESB ranks at four points for its service registry.

Evaluation of Security

Access control in WSO2 ESB comes in two parts. First there is a simple user management able to connect to an existing LDAP source [cf. WSOt], second the WSO2 Identity Server can extend the product for Extensible Access Control Markup Language (XACML) support offering fine grained resource based authorization [cf. WSOb]. These parts lead to a ranking of three points for access control. WSO2 ESB is able to use mediators to add security to message flows, thus enabling secure message transport [cf. WSOn] and ranking at four points.

Evaluation of QoS

WSO2 ESB offers clustering support in several configurations [cf. HLAA08], therefore ranking at four points. It further supports acting as a transaction participant along resources that are able to handle the Java Transaction API (JTA), but needs a JTA provider for this purpose [cf. WSOs]. This downside ranks the product at two points for transaction support.

Evaluation of Multi-Tenancy

Because WSO2 ESB is fully tenant aware it is capable of data isolation and performance isolation [cf. WSOl]. It further provides management functionality to enable tenant based identity and access management [cf. WSOl]. To achieve this, a "super-tenant" governs each environment by authorizing and supervising the tenants attributed. All in all WSO2 ESB ranks at four points regarding each criterion for full multi-tenancy support.

Evaluation of Extensibility

Apart from the custom adapters and transport mechanisms called "mediators" in WSO2 ESB the product offers no further plug-in mechanism, ranking it at zero points. The documentation that is delivered alongside, is extensive, comprehensive, and up to date [cf. WSOg]. The community is active and supporting and is even running and maintaining a developer portal of considerable size [cf. WSOe]. Training is available regarding several different aspects and ranges from technical training to project planning training [cf. WSOr]. As mentioned, the license is an open source license namely the Apache License v2.0 [cf. WSOj]. This leads to a ranking of four points for each of the latter four criteria.

3.4.8 Evaluation Roundup

Following the detailed evaluation of each product, Table 3.40 shows the ranking in a matrix of all products and all criteria. For more intuitive reading, the points of the ranking are mapped to symbols according to Table 3.39. To fit the table on one page, products are written in their short notation.

Count	Symbol
-	n/a
0	- -
1	-
2	o
3	+
4	++

Table 3.39: Mapping of Counted Values to Symbols

3.5 Product Selection

WSO2 ESB is the product scoring highest in the ranking, but detailed research shows shortcomings that render the product unsuitable for this master's thesis:

- *Missing technical details on multi-tenancy*: Several documents are available that describe WSO2's multi-tenancy on a high level, but none of them gives the technical details this master's thesis needs to achieve its goals [cf. Fre10][cf. WSO10].

- *Different multi-tenancy approach*: This master's thesis aims at achieving native multi-tenancy within the ESB as described in Section 2.3. But Fremantle [cf. Fre10, p. 35] gives the impression that WSO2 maintains one ESB container for each tenant with a tenant router outside the ESB, leading to as many containers as tenants the system serves.

Nonetheless, all documents available regarding WSO2 and its multi-tenancy contain interesting concepts and architectures to multi-tenancy in ESBs and are further used as inspiration for some of the approaches that this master's thesis takes. Because of the named shortcomings of WSO2, this work focuses on the product ranking second in the evaluation (Section 3.4) Apache ServiceMix. It does not offer any multi-tenancy yet, which gives the opportunity to design and implement a multi-tenancy approach from the beginning.

Category	Criterion	ServiceMix	BizTalk	JBoss	Mule	Petals	WebSphere	WSO2
A Messaging	A.1 Support for EIPs	++	o	o	++	o	o	++
	A.2 Routing	++	o	++	++	++	++	++
	A.3 Message Transformation	+	o	+	++	++	+	++
	A.4 Normalized Message Format	++	--	++	--	++	--	++
B Integration	B.1 BPEL support	+	o	+	+	++	+	+
	B.2 WS-* standards	++	++	o	o	--	--	o
	B.3 JBI support	+	--	--	o	++	--	--
	B.4 Adapters	++	++	--	++	++	o	--
	B.5 Custom Adapters	++	++	o	++	++	++	++
	B.6 Transport Mechanisms	++	-	o	+	o	-	o
C Runtime	C.1 Dynamic Policy Resolution	--	++	o	--	--	++	o
	C.2 Service Registry Support	--	++	+	++	o	++	++
D Security	D.1 Access Control	++	o	o	++	++	o	+
	D.2 Secure Message Transport	++	++	o	++	++	++	++
E QoS	E.1 Availability and Scalability	++	o	++	--	++	++	++
	E.2 Transactions	o	++	o	--	--	++	o
F Multi-Tenancy	F.1 Data Isolation	--	--	--	--	--	++	++
	F.2 Performance Isolation	--	--	--	--	--	++	++
	F.3 Tenant Based IAM	--	--	--	--	--	++	++
G Extensibility	G.1 Plug-In Mechanism	++	--	--	--	++	--	--
	G.2 Documentation	+	++	++	+	+	++	++
	G.3 Community	++	++	++	++	o	-	++
	G.4 Training	--	++	++	++	++	++	++
	G.5 Licensing	++	--	++	++	++	--	++
Total		**66**	**53**	**53**	**57**	**63**	**62**	**74**

Table 3.40: Comparison of ESB Products

3.6 Product Description

This section describes Apache ServiceMix regarding its existing structure and components.

3.6.1 Architectural Overview

Apache ServiceMix is built on the JBI specification [cf. Thep][cf. Jav]. This specification is designed to provide an approach for building components used in SOAs. Figure 3.1 shows the JBI based architecture of Apache ServiceMix. Notable components are the Normalized Message Router, the Binding Components, the Service Engines, and the JMX based Management Application. The following subsections explain each of these in further detail.

3.6.2 Normalized Message and Normalized Message Router

The central component of a JBI based ESB is the Normalized Message Router. Each incoming message is transformed into a Normalized Message, which is then routed through the ESB. This ensures one standard transmission format and prevents that each service provider and consumer needs to be aware of the message format expected by other providers or consumers. Therefore, the Normalized Message is a way to prevent the need for n:m message transformations. Figure 3.2 shows the structure of a Normalized Message in Apache ServiceMix.

It consists of Message Properties, which carry data like the security context of the message, correlation identifications for subsequent requests, timestamps, or requestor data. The Message Payload is the content of the original incoming message encoded in XML, whereas further and mainly binary information like arbitrary files can be stored in the Attachments of the Normalized Message.

3.6.3 Binding Components

To transform the original message, which could for example be a SOAP message, a JMS message, or a simple e-mail, to a Normalized Message or vice versa, Binding Components are used. These are shown in the lower part of the architectural overview shown in Figure 3.1. For each specific protocol that the ESB supports, there is one Binding Component capable of transforming incoming messages to Normalized Messages or outgoing Normalized Messages to messages of the specific communication protocol. Listing 3.1 shows a SOAP message, which would be transformed into a Normalized Message through the relevant Binding Component by adding the SOAP header information to the Message Properties of the Normalized Message while the SOAP body would become the Message Payload of the Normalized Message.

Because the JBI specification is open and standardized, further Binding Components can be implemented to cover even more protocols than the ones already served by Binding Components delivered with Apache ServiceMix.

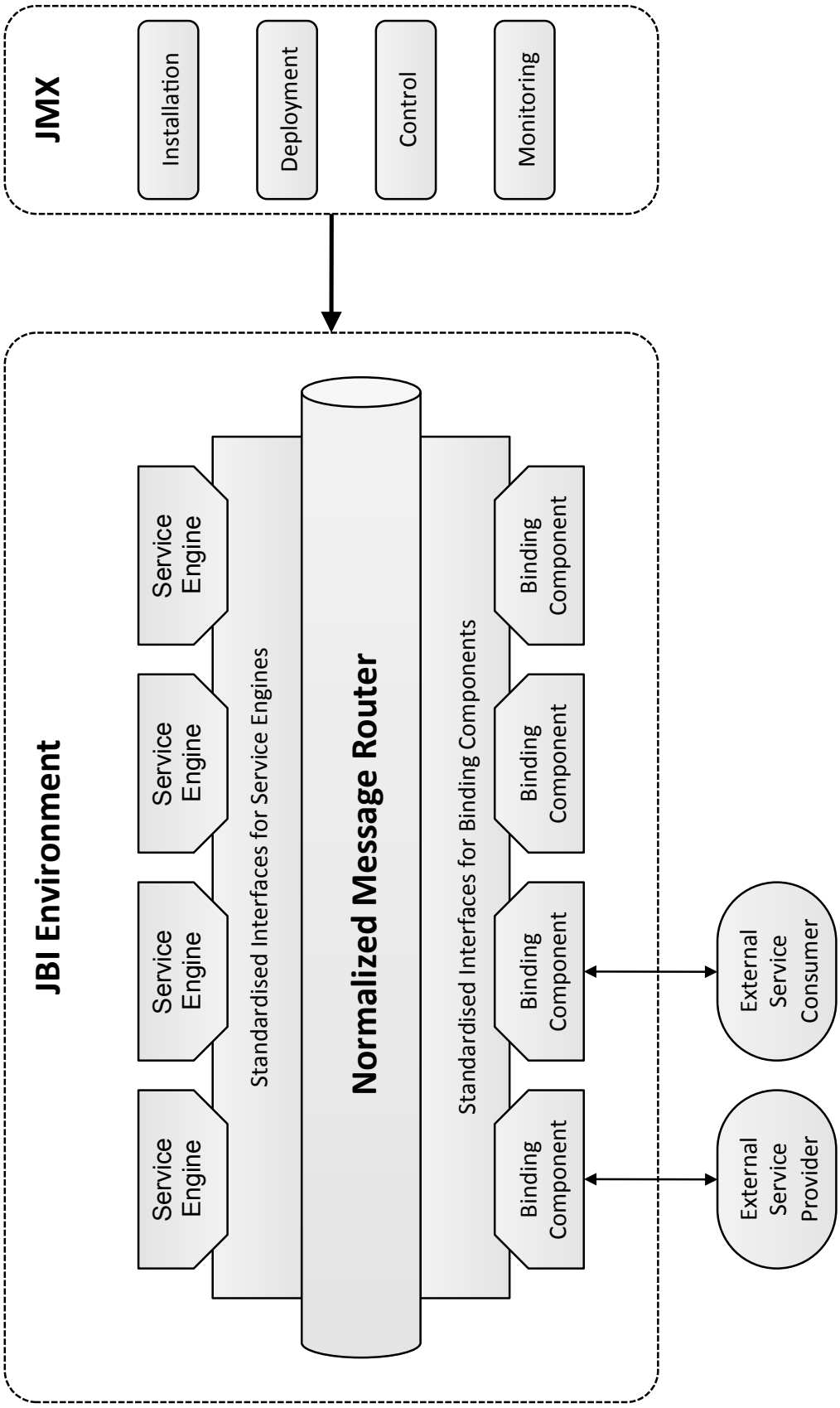

Figure 3.1: Architecture of Apache ServiceMix [Thep]

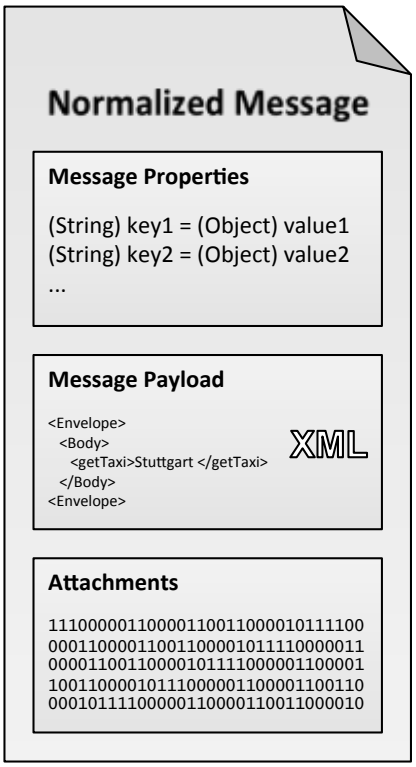

Figure 3.2: Normalized Message in Apache ServiceMix [Thep]

```
<soapenv:envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:typ="http://iaas.uni-stuttgart.de/taxiRequest/types">
   <soapenv:header>
      <typ:requestId>8bccdf62-5ddf-4777-ab86-1ebcdcca5142</typ:requestId>
   </soapenv:header>
   <soapenv:body>
      <typ:taxiRequest>
         <typ:from>Universitaetsstr. 38, 70569 Stuttgart</typ:from>
         <typ:to>Flughafenstr. 43, 70629 Stuttgart</typ:to>
      </typ:taxiRequest>
   </soapenv:body>
</soapenv:envelope>
```

Listing 3.1: Example Basic SOAP Request

3.6.4 Service Engines

The upper part of the architectural overview in Figure 3.1 shows the Service Engines. These route, transform, or otherwise handle Normalized Messages within the ESB. Via the integration of Apache Camel there are for example Service Engines capable of providing nearly all of the enterprise integration patterns introduced by Hohpe and Woolf [cf. HW03]. Further Service Engines that come with Apache ServiceMix provide functionality for caching, scheduling, workflows, scripting, mediation, or message validation.

3.6.5 JMX Based Management Application

The right hand side of Figure 3.1 shows JMX as part of the JBI specification targeted at lifecycle management and administration of the whole platform. Any standard JMX based management application can connect to Apache ServiceMix and can be used to perform these tasks.

4 Specification

Based on the analysis of the state of the art in Chapter 3, this part of the work points out shortcomings of the existing products in Section 4.1 and identifies requirements that a multi-tenant aware ESB product has to meet in Section 4.2. Section 4.3 illustrates requirements that are important for this specific master's thesis whereas Subsection 4.3.3 establishes use cases that base on these requirements.

4.1 Shortcomings of Existing Products

The evaluation in Section 3.4 shows several shortcomings of existing products:

- *Missing Multi-tenancy*: The evaluation roundup in Subsection 3.4.8 shows that only two of the products in the evaluation offer multi-tenancy support.
- *Multi-tenancy Realization*: Section 2.3 shows two approaches to achieve multi-tenancy. Native multi-tenancy is thereby more suitable for a Cloud computing scenario than multiple instances multi-tenancy, because it supports a higher customer count. This results in a shortcoming if products miss support for native multi-tenancy.
- *Weak Clustering Support*: The evaluation roundup in Subsection 3.4.8 shows that some products miss or offer only limited support for clustering configurations. This implies limitations to the scalability and availability of applications and resources, which are a fundamental requirement of Cloud computing settings. Because of the Cloud computing focus of this master's thesis, this is another shortcoming.
- *Lack of Integration*: The evaluation roundup in Subsection 3.4.8 further shows that some products offer weak support of WS-* standards and adapters to other systems. The resulting problems for the integration of the products are another shortcoming.

4.2 Multi-Tenant ESB Requirements

Based on the shortcomings this master's thesis identifies in Section 4.1 it determines the following requirements a multi-tenant aware ESB needs to cover.

- *Clustering Support for Scalability*: The product has to offer clustering functionality to scale appropriately and distribute well in a Cloud computing infrastructure.

- *Support of WS-* standards*: To leverage integration and functionality, the ESB has to support as many WS-* standards as possible.
- *Adapters*: The more adapters a product offers, the easier is the integration with other systems, software, and services.
- *Transport Mechanisms*: To connect to as many other systems as possible, the support for different transport mechanisms is crucial.
- *Multi-tenancy Approach*: To fit into the Cloud computing approach, the ESB has to support the native multi-tenancy approach.
- *Tenant Awareness*: The product has to be aware that it handles different tenants and has to be able to identify these.
- *Tenant Isolation*: The ESB has to ensure isolation to prevent tenants from gaining access to other tenant's data or computing power.
- *Service Registry*: To manage the services usable by the ESB and its service requestors the product has to offer an advanced service registry.
- *Multi-Tenant Service Registry*: The service registry has to be able to process tenant metadata associated with services.
- *Tenant Management*: The product has to offer tenant management functionality, that is to allow association of new tenants with existing services or control the access rights of tenants.
- *Tenant Based Correlation*: The ESB has to be able to correlate service responses correctly to the right service requests of the right tenant.

4.3 Requirements in This Thesis

Apache ServiceMix is not multi-tenant aware yet. It is missing many components needed to enable multi-tenancy. Some are requirements that this master's thesis depends on to enable multi-tenancy, but are out of the scope of this master's thesis. These components are detailed in Subsections 4.3.1 and 4.3.2 with references to theses that are currently working on these topics. Whenever this master's thesis needs these functionalities, it uses mock ups that can be replaced by the implementation of the relevant services or components later on.

4.3.1 Tenant Aware Service Registry

A service registry needs to be implemented that includes tenant management functionality. It needs to be able to associate data with services, that state, which services serve which tenants. It further needs to provide user specific management functionality, because some tenants may wish to serve their users different services.

Because Apache ServiceMix does not yet come with a service registry that can handle service metadata and offers a front end for administration, another diploma thesis is being worked on to provide this functionality [cf. Muh12].

4.3.2 Tenant Registry

The tenant registry stores all data that is directly related to tenants. This includes the Tenant Context that components need for processing tenant specific messages. Section 5.2 introduces the Tenant Context. Further data stored in this registry could be different roles of personnel associated with each tenant, contact data, or tenant related service preferences.

As with the tenant aware service registry, a diploma thesis is being worked on to provide this functionality [cf. Muh12].

The requirements this master's thesis focuses on, are the native multi-tenancy approach, tenant awareness of Apache ServiceMix, and tenant based correlation. These are detailed into functional and non-functional requirements. Subsection 4.3.3 explains the functional requirements whereas Subsection 4.3.4 focuses on the non-functional requirements.

4.3.3 Functional Requirements

Tenant Aware Normalized Messages

This master's thesis needs to extend the Normalized Message Format used in Apache ServiceMix by means to embed tenant data into each Normalized Message. This enables the ESB to distinguish requests from different tenants and for example route them according to the specific tenant's preferences as stored in the tenant registry. It allows the emission of information enabling metering, accounting, and billing for each tenant. Further, it provides the foundation for ensuring tenant isolation and enforcing security principles among tenants.

Tenant Aware Adapters

This work needs to extend the adapters, called Binding Components in Apache ServiceMix as shown in Figure 3.1 and explained in the architectural overview in Subsection 3.6.1. A Binding Component needs to be able to identify the tenant sending a request and needs to include this data into the internal Normalized Messaging Format of the ESB. This allows the multi-tenancy approach to meet the requirement of the *Tenant Aware Normalized Message*.

Tenant Aware Routing Components

The routing components as one specific type of Service Engines in Apache ServiceMix route incoming service requests to a service provider that either the service requestor specifies or the component chooses based on the service provider's ability to handle the request. To enable multi-tenancy in an ESB, the routing components need to be able to identify the tenant that sent the original request, look up the services associated with this tenant from the tenant aware service registry and route the requests accordingly for each tenant. The routing components have to be able to check the permissions of each tenant. These permissions can be configured in the tenant and service registry, and retrieved from these by the routing components. A specific service may for example be available only to a specific set of tenants. The routing components have to avoid routing that does not match these constraints.

Tenant Based Correlation

Apart from being able to identify the initiating tenant in incoming service requests, the ESB needs to correlate the responses to the specific requests in long running and asynchronous interactions. This resembles the correlation handling that long running workflows already do. The scenario to handle has several concurrent requests to a pool of services that reply in an asynchronous manner. The ESB is not able to identify correctly, which reply belongs to which original request if the replies arrived at the ESB in another order than the original requests. To cope with this situation, Hohpe and Woolf suggest to assign each incoming message a unique identifier, called Correlation Identifier [HW03, p. 163. ff.]. The processing service will then copy this identifier and assign it to its reply. This enables the ESB to map the reply to the correct request.

Use Cases

This work establishes three use cases that base on the requirements the previous chapter illustrates:

- Table 4.1: *Use Case 1 - Send Tenant Specific Service Request To ESB*
- Table 4.2: *Use Case 2 - Receive Tenant Specific Service Request From ESB*
- Table 4.3: *Use Case 3 - Receive Tenant Specific Service Response From ESB*

Name	*Send Tenant Specific Service Request To ESB*
Goal	The tenant of an incoming message has to be correctly identified and encoded within the Normalized Message to show tenant awareness of the adapter and the Normalized Message Format.
Actor	A tenant sends a request to the ESB using a predefined endpoint.
Pre-Condition	The ESB is up and running and the adapter for the communication protocol of the incoming message is installed. Moreover the corresponding configuration for the adapter is deployed.
Post-Condition	A Normalized Message is delivered to the Normalized Message Router. ESB and adapter are waiting for service requests.
Post-Condition in Special Case	A notification is sent to the tenant. ESB and adapter are waiting for service requests.
Normal Case	The adapter identifies the requesting tenant, constructs a Normalized Message with the tenant data encoded within and forwards this Normalized Message to the ESB's Normalized Message Router.
Special Case	ESB is unable to identify the requesting tenant and notifies the tenant and/or administrative staff thereof. Nonetheless, it has to be able to accept further service requests afterwards.

Table 4.1: Use Case 1: Send Tenant Specific Service Request to ESB

Name	*Receive Tenant Specific Service Request From ESB*
Goal	A Normalized Message with encoded tenant data has to be correctly routed to the service provider specified for this tenant to show tenant awareness of the routing component within the ESB.
Actor	A routing component routes the service request to the service provider.
Pre-Condition	Successful completion of Use Case 1 as described in Table 4.1.
Post-Condition	The message is routed and delivered to the correct tenant specific service provider.
Post-Condition in Special Case	A notification is sent to the tenant. ESB and adapter are waiting for service requests.
Normal Case	The routing component reads the tenant data from the Normalized Message, looks up the service provider associated with this tenant and routes the message accordingly.
Special Case	The routing component is not able to read the tenant data. The routing component is not able to route the request accordingly, because the service provider is not available to this specific tenant.

Table 4.2: Use Case 2: Receive Tenant Specific Service Request From ESB

Name	*Receive Tenant Specific Service Response From ESB*
Goal	A tenant has to receive a service response to its specific service request to show tenant correlation within the ESB.
Actor	The ESB forwards a service response to the correct tenant.
Pre-Condition	Successful completion of Use Case 1 and Use Case 2 as described in Table 4.1 and Table 4.2. The service provider has delivered a tenant specific response to the ESB.
Post-Condition	The service response is forwarded to the correct tenant who sent the service request.
Post-Condition in Special Case	A notification is sent to the ESB administrator. ESB is waiting for service requests and responses.
Normal Case	The ESB reads the tenant correlation data encoded in the service response and forwards the message to the correct service requesting tenant.
Special Case	The ESB is not able to read the tenant data. The ESB is not able to correctly identify the service requesting tenant.

Table 4.3: Use Case 3: Receive Tenant Specific Service Response From ESB

4.3.4 Non-Functional Requirements

The non-functional requirements of this master's thesis involve requirements regarding the categories extensibility, reusability, and compatibility. The following subsections list these requirements in the specific category and explain them in detail.

Extensibility

The load balancers being essential in large elastic systems need to be tenant aware. For each incoming request, the load balancers need knowledge about where the relevant services targeted at processing each tenant's request reside, and where to forward the requests to. The detailed specification of this aspect is out of the scope of this master's thesis, but it needs to keep it in mind to be able to extend the concept in future works.

Reusability

Apache ServiceMix is an open source project that evolves over time. Therefore, the multi-tenancy approach needs to be ready to work with different versions of Apache ServiceMix. This avoids the need to change the code of all the components that are made multi-tenant aware by this master's thesis with every new version of Apache ServiceMix.

Compatibility

An ESB needs to serve a multitude of systems, some of which may be years or even decades old legacy systems. Because those are often not multi-tenant aware and cannot be modified anymore to enable multi-tenancy awareness, Apache ServiceMix needs to be able to handle message exchanges with these components. This means that Apache ServiceMix needs to handle multi-tenancy in a transparent way with the ability to handle messages without any Tenant Context in the way it did before this master's thesis modifies it to enable multi-tenancy. Thus, backward compatibility for non-tenant aware communication has to be ensured.

5 Design

This chapter refines the requirements that Section 4.3 elaborates, into a design that provides the basis for the implementation in Chapter 6. It will start with an architectural overview in Section 5.1 that shows, which parts of Apache ServiceMix the implementation changes. Section 5.2 introduces a Tenant Context that allows the correlation of messages to a specific tenant whereas Section 5.3 shows how this work changes the Normalized Message Format of Apache ServiceMix to incorporate this Tenant Context. Section 5.4 on adapters, called Binding Components in Apache ServiceMix, enunciates how these have to be changed to get the tenant information from the original protocol into the internal Normalized Messaging Format that Apache ServiceMix uses. Section 5.5 introduces the Tenant Router Enterprise Integration Pattern whereas Section 5.6 describes Correlation Identifiers as a way to correlate asynchronous responses of long running services that target a specific tenant to this tenant's earlier request.

5.1 Architecture

Figure 3.1 in the product description in Section 3.6 introduced the architecture of Apache ServiceMix. It presented the main components Normalized Message Router, Binding Components, and Service Engines. Figure 5.1 highlights the components that this master's thesis extends with a bold border. Regarding the Binding Components, these are the SOAP over HTTP Binding Component, the JMS Binding Component, and the e-mail Binding Component. It further uses the Service Engines Content Enricher and Content Based Router, which together form the new Enterprise Integration Pattern Tenant Router that Section 5.5 introduces. Both Service Engines base on the Apache Camel Service Engine that comes with Apache ServiceMix. It provides integration with the full set of EIPs that Apache Camel supports.

Apart from the overall architecture, this master's thesis extends the internal messaging format of Apache ServiceMix to include tenant related data into the Normalized Messages. Figure 5.2 highlights the part of the Normalized Messaging Format that this work changes with a bold border.

The combination of both parts with the motivating scenario, that Section 1.3 introduced, leads to a model of the communication of components that this master's thesis builds its implementation on as Figure 5.3 shows. This illustration already incorporates the process that the motivating scenario defines by numbering the actions that take place within this architecture. It further shows the roles that are involved within this process.

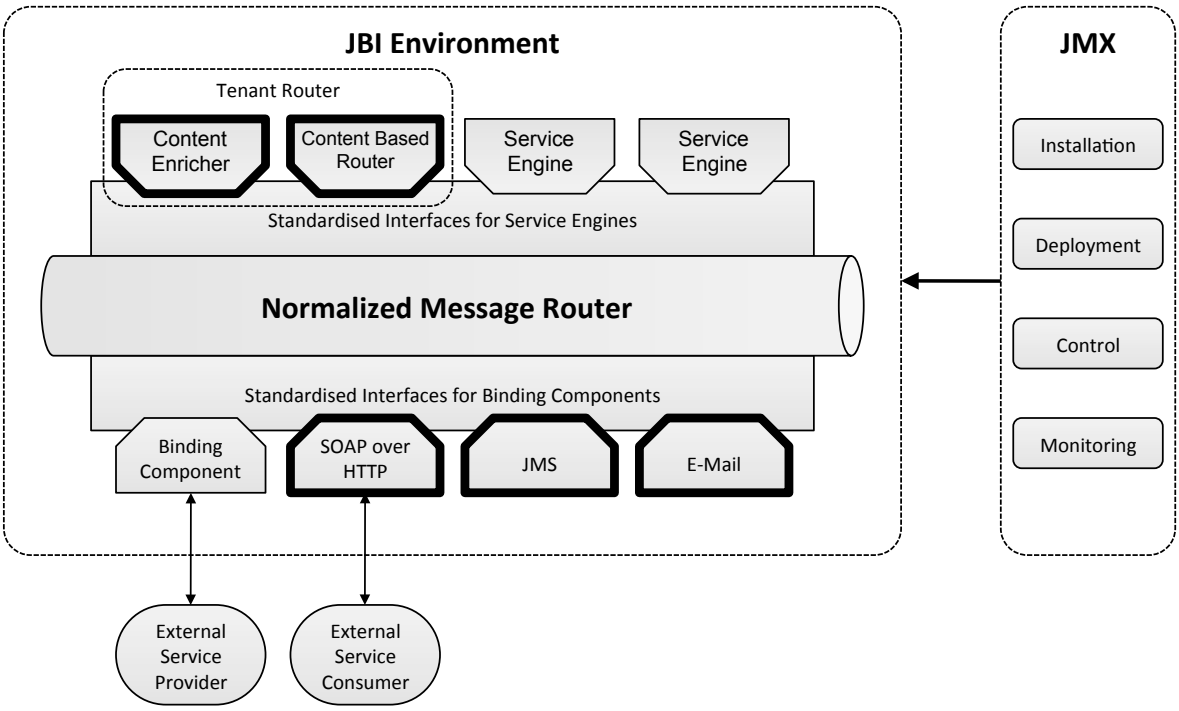

Figure 5.1: Modifications to the Architecture of Apache ServiceMix

Figure 5.2: Modifications to the Normalized Message in Apache ServiceMix

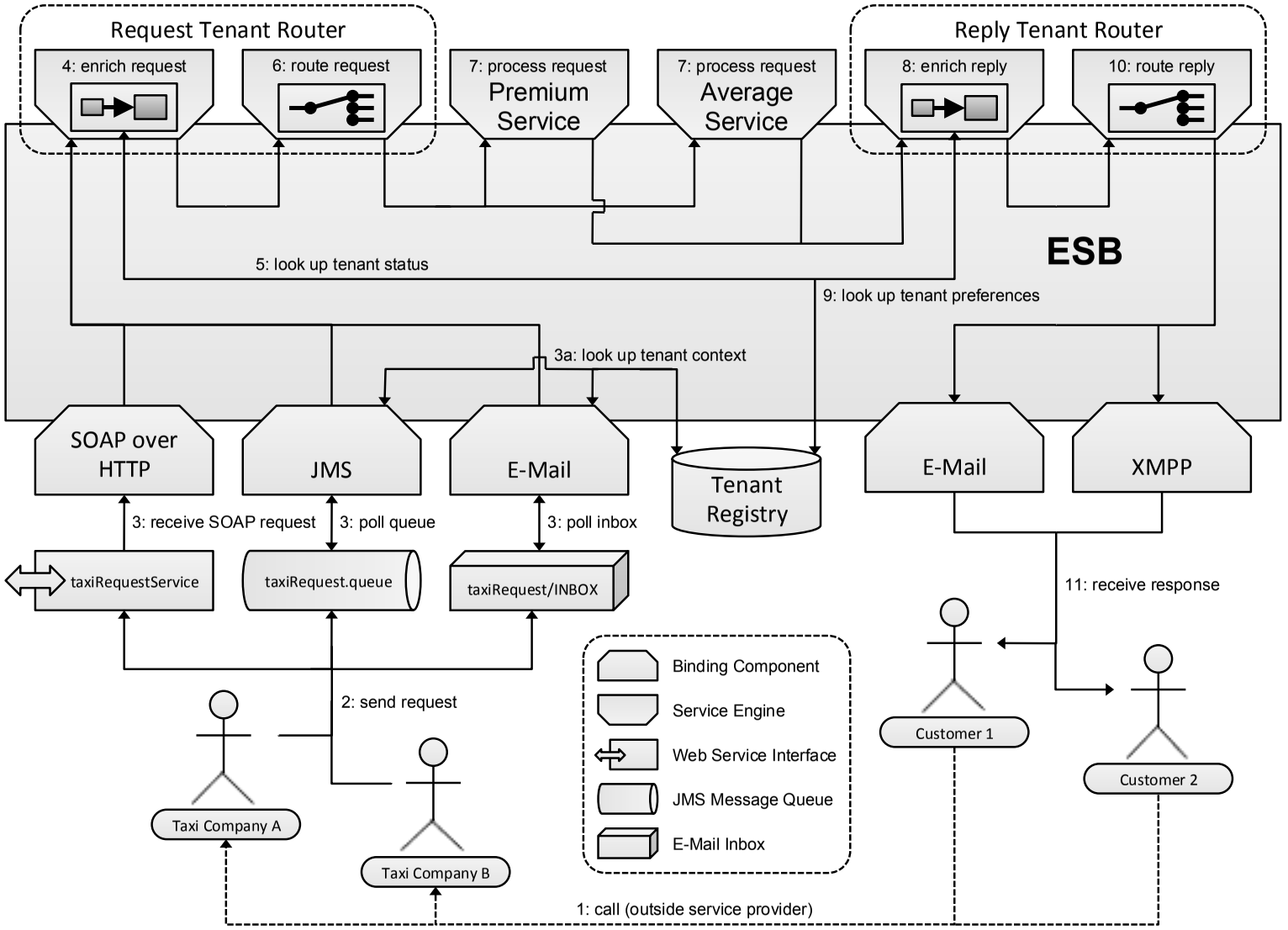

Figure 5.3: Communication of Components in the Implementation

5.2 Tenant Context

To enable multi-tenancy, the ESB itself and any provider, consumer, routing module, or other component needs to know, which tenant it is serving or processing for. Any message sent to, handled by, and sent from the ESB must therefore carry a Tenant Context that uniquely identifies the tenant and user the message belongs to. Listing 5.1 shows a XML Schema (XSD) representation of the Tenant Context this master's thesis elaborates.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="UUIDType">
    <xsd:simpleType>
      <xsd:restriction base="ID">
        <xsd:pattern
          value="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>

  <xsd:group name="tenantUserId">
    <xsd:sequence>
      <xsd:element name="tenantId" ref="UUIDType" />
      <xsd:element name="userId" ref="UUIDType" />
    </xsd:sequence>
  </xsd:group>

  <xsd:element name="tenantContext">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:choice>
          <xsd:element name="tenantContextKey" ref="UUIDType" />
          <xsd:group ref="tenantUserID" />
        </xsd:choice>
        <xsd:element name="optionalEntry" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:element name="key" type="xsd:string" />
            <xsd:element name="value" type="xsd:anyType" />
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Listing 5.1: XSD Representation of Tenant Context

For providing unique IDs, this master's thesis uses Universally Unique Identifiers (UUID). RFC 4122 states that it assumes UUIDs to be globally unique without the need of coordination by a centralized instance [cf. Net05]. This renders them useful in highly distributed SOA and Cloud computing approaches like the one covered in this master's thesis. The Java class `java.util.UUID` of the Java API can generate UUIDs. The method call `UUID.randomUUID().toString()` will return a string representation of an UUID. This master's thesis will use this Java class and method whenever UUIDs are needed within the implementation. Lines 3 through 10 in Listing 5.1 show the representation of an UUID as a XSD type.

The Tenant Context consists of the mandatory and the optional part. Mandatory is a unique key associated with each Tenant Context or a combination of a unique tenant ID and user ID. The combination of tenant ID and user ID enables the ESB, its components and services to identify the tenant and user even in cases where there are several tenants or users with the same name. Lines 12 through 17 in Listing 5.1 show the definition of a XSD group that represents this combination.

Lines 26 through 31 in Listing 5.1 show the optional part, which can carry any key value pairs, for example the names of tenant and user or further tenant or user specific data. Because the XSD defines the value part of the optional entries as a `xsd:anyType` it can contain any data type, that applications built around the services using this XSD, need. These could even be further `xsd:complexType` structures. Thus, the Tenant Context is extensible.

The unique Tenant Context key, that Line 23 in Listing 5.1 shows, aims at enabling multi-tenancy in protocols that are not extensible by the full Tenant Context in all its structure. Within these protocols the single Tenant Context key uniquely references the full Tenant Context. It enables the Binding Component for this specific protocol to look up the full Tenant Context from the tenant registry by the unique key and embed it into the message it dispatches to the ESB although the incoming message did not carry the full Tenant Context. The unique key is important with regards to later extensibility of this concept. Given for example dynamic policy based service discovery, the key enables Binding Components and other application parts to look up policies associated with this specific tenant and dynamically discover services that match the specific policies.

Depending on the communication protocol used, this offers two options to include the Tenant Context within each message:

- Include full Tenant Context. Listing 5.2 shows an example instance of this approach based on the XSD in Listing 5.1.

- Include only the unique key of the Tenant Context. Listing 5.3 shows an example instance of this approach based on the XSD in Listing 5.1.

```
<tenantContext>
  <tenantId>16c20253-8605-4b67-9001-935c50c8b707</tenantId>
  <userId>dc0b71dd-4c99-4efb-964b-bc30efd552cc</userId>
  <optionalEntry>
    <key>tenantName</key>
    <value>Example Inc.</value>
  </optionalEntry>
  <optionalEntry>
    <key>userEmailAddress</key>
    <value>user@example.org</value>
  </optionalEntry>
</tenantContext>
```

Listing 5.2: Example Full Tenant Context

```
<tenantContext>
  <tenantContextKey>2c4d0df8-f1f3-4501-b25f-1ec17fdee8b8</tenantContextKey>
</tenantContext>
```

Listing 5.3: Example Simple Tenant Context

The Tenant Context is only needed in case multi-tenancy is desired. Because this master's thesis pursues the goal to enable multi-tenancy within Apache ServiceMix, it will from here on assume that multi-tenancy communication is desired and a Tenant Context is needed.

5.3 Normalized Message Format

This work needs to extend the Normalized Message Format that Apache ServiceMix uses to store the Tenant Context within each message. Figure 5.4 shows the structure of a Normalized Message. This master's thesis realizes the tenant awareness by adding the Tenant Context shown in Listing 5.1 to the Message Properties of the Normalized Message Format.

This approach allows to embed the Tenant Context within the Normalized Message without interfering with any of the tenant's application parts. Because the Message Properties are transparent to application components, they do not have to handle or understand them. Embedding the Tenant Context within the Message Properties however allows Binding Components or Service Engines to make use of them for any tenant related action like routing or mediation within the ESB. Extending the Message Payload by the Tenant Context would not be acceptable, because it contains the data used and processed by the tenant's application components itself whereas the tenants must not be aware of the multi-tenancy within the ESB. The Attachments of the Normalized Message are also application related content and may be binary data, which renders them unsuitable for embedding Tenant Context as well.

The goal is to make use of the Tenant Context transparently without the tenant itself knowing or noticing anything of the ESB's tenant awareness.

Figure 5.4: Extending the Normalized Message Format

5.4 Binding Components

To get the Tenant Context from the messages of the original transport protocol into the Normalized Message Format, this master's thesis adjusts the relevant Binding Components. Based on the motivating scenario defined in Section 1.3, this work extends the Binding Components for the transport protocols SOAP over HTTP, JMS, and e-mail.

5.5 Tenant Router

Fehling [Feh09, p. 67] introduces a pattern derived from the Content Enricher [HW03, p. 336 ff.] and Content Based Router [HW03, p. 230 ff.] called Tenant Router. Figure 5.5 shows this EAI pattern.

Because the focus of this master's thesis is the routing of requests to different services for different tenants, the pattern is slightly modified to the one shown in Figure 5.6.

First the Tenant Router adds tenant specific data to each incoming request, one part being data about the actual service that handles the request of the specific tenant according to the service registry or data about the tenant itself according to the tenant registry. Second the Tenant Router routes the message to the right service provider with the help of the data now embedded in the message.

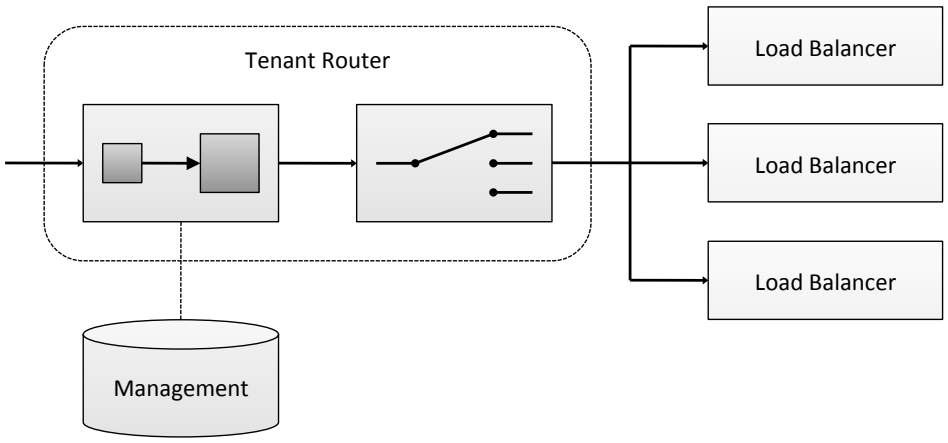

Figure 5.5: Original Tenant Router EAI Pattern [Feh09, p. 67]

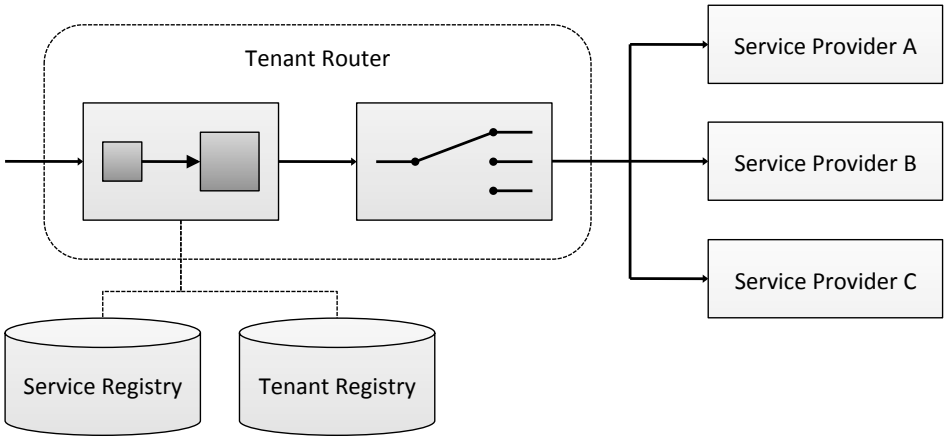

Figure 5.6: Tenant Router EAI Pattern in This Master's Thesis

This pattern is fully compatible to the one Fehling [Feh09, p. 67] describes. It can make use of load balancing capabilites by combining the patterns and adding load balancers for each of the service providers.

Within Apache ServiceMix there is a Service Engine that allows it to connect to Apache Camel and its multitude of Enterprise Integration Patterns [cf. Theg]. These also contain the patterns Content Enricher and Content Based Router that this master's thesis needs to combine and modify to form the Tenant Router.

5.6 Correlation Identifiers

Subsection 4.3.3 suggests the use of Correlation Identifiers [cf. HW03, p. 163. ff.] to correlate requests and responses in long running service calls. Apache ServiceMix allows the use of Correlation Identifiers in the headers of message exchanges [cf. Ther]. Through its integration with Apache Camel, the latter adds Correlation Identifiers automatically as soon as the

Splitter, Multicast, Recipient List, or Wire Tap Enterprise Integration Patterns are used [cf. Thef]. Nonetheless, any component can use Correlation Identifiers in Apache Camel any time it relies on them.

Because these Correlation Identifiers work for each exchange, no further correlation mechanisms are needed for tenant based correlation. The exchange correlation is a subset of the tenant correlation and therefore already finer grained than the tenant correlation itself. Each request that issued a message exchange belongs to only one specific tenant, which leads to unambiguous correlation of the response to the correct request and therefore tenant and user if the message exchange itself is correctly correlated.

6 Implementation and Evaluation

This part of the work illustrates the implementation of the requirements Section 4.3 defines and the design in Chapter 5 refines. It explains in detail, which components of Apache ServiceMix this master's thesis modifies and how it accomplishes these modifications. The explanations include code clippings and references to programming languages, classes, and methods this master's thesis uses for its implementation.

Section 6.1 explains the changes to the Binding Components file, SOAP over HTTP, JMS, and e-mail, whereas Section 6.2 illustrates the changes to the Enterprise Integration Patterns Content Enricher and Content Based Router, which Apache ServiceMix embeds as Service Engines. Section 6.3 concludes this chapter with an evaluation of this implementation that bases on the motivating scenario Section 1.3 introduced.

6.1 Binding Components

First this master's thesis extends the file Binding Component by the ability to read the Tenant Context from the Message Properties of the Normalized Message and write them to disk. This functionality is for debugging purposes and to verify that the other protocol Binding Components correctly embed the Tenant Context into the Message Properties of the Normalized Message.

Section 5.4 identifies, based on the motivating scenario described in Section 1.3, three transport protocols apart from the file Binding Component that need change to enable multi-tenancy within these components. The subsections in this chapter explain the implementation of the changes to the Apache ServiceMix Binding Components regarding SOAP over HTTP, JMS, and e-mail.

The changes to these Binding Components show the two options to achieve multi-tenancy communication via a single Tenant Context Key or a full Tenant Context, that Section 5.2 introduced. SOAP on the one hand can embed the full Tenant Context whereas e-mail or JMS on the other hand require the Tenant Context Key.

6.1.1 File

The file Binding Component `servicemix-file` provides the functionality to either read files from disk and send them to the Normalized Message Router as Normalized Messages or receive Normalized Messages from the Normalizes Message Router and write them to file on disk. This master's thesis uses the latter functionality.

The goal is to extract the otherwise transparent Message Properties from the Normalized Message and write them to file alongside the message content to make them visible. This allows to check whether the other Binding Components extended in this master's thesis correctly embed the Tenant Context into the Normalized Message.

To avoid modification of the core of the Binding Component itself and increase portability, Apache ServiceMix allows to implement custom marshalers that are used to marshal the communication from the original protocol to Normalized Messages and vice versa. In case of the file Binding Component, the custom marshaler needs to extend the abstract Java class `org.apache.servicemix.components.util.FileMarshaler`. This class contains a method to write the content of the Normalized Message received from the Normalized Message Router to the path the configuration file specifies, named `writeMessage(MessageExchange exchange, NormalizedMessage message, OutputStream out, String path)`. To get the Message Properties, the method call `message.getPropertyNames()` provides a list that carries all the names of the Message Properties associated with the Normalized Message. This list can be iterated to retrieve the relevant property values via the method `message.getProperty(String propertyName)`. In case of SOAP over HTTP the default marshaler of the HTTP Binding Component within Apache ServiceMix embeds only one property, which itself is again a `Map<String name, Object value>` that can be iterated. It contains all the headers the SOAP message originally contained plus some further data on the HTTP request and the requestor.

Listing 6.1 shows the Spring bean configuration file `xbean.xml` for the bean `file:sender`. Lines 8 and 11 are the configuration options used to refer to a custom marshaler.

```
1  <beans>
2     <!-- Namespaces omitted to improve readability -->
3
4     <file:sender service="fileSender"
5        endpoint="fileSenderEndpoint"
6        autoCreateDirectory="true"
7        directory="file:target/files"
8        marshaler="#marshaler">
9     </file:sender>
10
11    <bean id="marshaler" class="de.unistuttgart.iaas.taxiRequest.file.marshaler" />
12 </beans>
```

Listing 6.1: Configuration of File Service Unit via `xbean.xml`

6.1.2 SOAP Over HTTP

Listing 6.2 shows an example transport request via SOAP a taxi provider receives via a Web service interface.

```
<soapenv:envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:typ="http://iaas.uni-stuttgart.de/taxiRequest/types">
   <soapenv:body>
      <typ:taxiRequest>
         <typ:from>Universitaetsstr. 38, 70569 Stuttgart</typ:from>
         <typ:to>Flughafenstr. 43, 70629 Stuttgart</typ:to>
      </typ:taxiRequest>
   </soapenv:body>
</soapenv:envelope>
```

Listing 6.2: Example Transport Request via SOAP Over HTTP

By default such requests do not allow to identify the requesting tenant at all. To enable multi-tenancy in SOAP requests within Apache ServiceMix, this master's thesis uses the ability to include header data with SOAP requests. Listing 6.3 shows the SOAP request enriched with the Tenant Context that Section 5.2 introduced. Lines 4 through 7 show the Tenant Context within the SOAP header.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:typ="http://iaas.uni-stuttgart.de/taxiRequest/types">
   <soapenv:header>
      <typ:tenantContext>
         <typ:tenantId>16c20253-8605-4b67-9001-935c50c8b707</typ:tenantId>
         <typ:userId>dc0b71dd-4c99-4efb-964b-bc30efd552cc</typ:userId>
      </typ:tenantContext>
   </soapenv:header>
   <soapenv:body>
      <typ:taxiRequest>
         <typ:from>Universitaetsstr. 38, 70569 Stuttgart</typ:from>
         <typ:to>Flughafenstr. 43, 70629 Stuttgart</typ:to>
      </typ:taxiRequest>
   </soapenv:body>
</soapenv:envelope>
```

Listing 6.3: Example Multi-Tenant Transport Request via SOAP Over HTTP

As with the file Binding Component, Apache ServiceMix allows the use of custom marshalers with the `servicemix-http` component. The default marshaler `org.apache.servicemix.http.endpoints.HttpSoapConsumerMarshaler` already embeds SOAP headers found in an incoming message into the Message Properties of the Normalized Message dispatched to the

Normalized Message Router. However, it creates a `Map<String name, Object value>` with all data from the HTTP request and embeds the SOAP header into one entry as a Document Object Model (DOM) representation of the original header. This entry can be used by processing services to read data that was originally embedded in the SOAP header. However, because it is inefficient for the routing and handling of the Normalized Message to retrieve the full Tenant Context from this map, look up the needed key from the DOM representation and read the relevant value at each routing step, the custom marshaler implemented in this master's thesis embeds at least the combination of tenant and user ID into the Message Properties of the Normalized Message. Of the optional entries, it embeds the tenant and user name into the Message Properties of the Normalized Message. If additional optional entries of the Tenant Context are needed for routing, the custom marshaler needs to be extended.

To keep backward compatibility, the changed marshaler checks if tenant data are present in the incoming message. If not, it handles the message the way Apache ServiceMix handled messages before this master's thesis extended it by multi-tenancy awareness.

To embed the tenant and user ID or any other data into the Normalized Message, a custom marshaler for the SOAP over HTTP Binding Component extends the class `org.apache.servicemix.http.endpoints.HttpSoapConsumerMarshaler`. The Message Properties of the Normalized Message are set within the method `createExchange(HttpServletRequest request, ComponentContext context)`. Calling the specific method of the parent class via the method call `super.createExchange(request, context)` creates a Message Exchange. The method call `exchange.getMessage("in")` on the retrieved Message Exchange accesses the incoming Normalized Message. The Message Properties are then set by the method `message.setProperty(String name, Object value)` on the Normalized Message.

Listing 6.4 shows the Spring bean configuration file `xbean.xml` for the bean `http:soap-consumer`. Lines 11 through 13 are optional and configure a custom marshaler.

```
1  <beans>
2     <!-- Namespaces omitted to improve readability -->
3
4     <http:soap-consumer service="getTransportRequestService"
5        endpoint="getTransportRequest"
6        targetService="customTargetService"
7        targetEndpoint="customTargetEndpoint"
8        wsdl="classpath:getTransportRequestService.wsdl"
9        locationURI="http://exampleTaxi.org:8192/getTransportRequestService/" >
10
11       <http:marshaler>
12          <bean class="de.unistuttgart.iaas.taxiRequest.http.marshaler" />
13       </http:marshaler>
14
15    </http:soap-consumer>
16 </beans>
```

Listing 6.4: Configuration of SOAP Over HTTP Service Unit via `xbean.xml`

6.1.3 JMS

The Apache ServiceMix Binding Component `servicemix-jms` allows connections to JMS destinations. As with the file and HTTP Binding Components, `servicemix-jms` endpoints are configurable to allow the use of custom marshalers, which need to extend the abstract Java class `org.apache.servicemix.jms.endpoints.AbstractJmsMarshaler`. This class provides the method `createExchange(JmsContext jmsContext, ComponentContext jbiContext)` to create a new message exchange. Within this method `exchange.getMessage("in")` retrieves the incoming normalized message. The method call `normalizedMessage.setProperty(String name, Object value)` sets a property of the Normalized Message as specified in the method call. Listing 6.5 shows the Spring bean configuration file `xbean.xml` for the bean `jms:consumer`. Lines 9 and 16/17 are optional and configure a custom marshaler.

```
1  <beans>
2     <!-- Namespaces omitted to improve readability -->
3
4     <jms:consumer service="getTransportRequestService"
5        endpoint="getTransportRequest"
6        targetService="customTargetService"
7        targetEndpoint="customTargetEndpoint"
8        destinationName="transportRequest.queue"
9        marshaler="#marshaler"
10       connectionFactory="#connectionFactory" >
11    </jms:consumer>
12
13    <amq:connectionFactory id="connectionFactory"
14          brokerURL="tcp://localhost:61616" />
15
16    <bean id="marshaler"
17          class="de.unistuttgart.iaas.taxiRequest.jms.marshaler" />
18 </beans>
```

Listing 6.5: Configuration of JMS Service Unit via `xbean.xml`

JMS itself allows message senders to embed custom headers within a JMS message. These are called properties and the method call `JMSMessage.setStringProperty(String name, String value)` sets them on the JMS message. The JMS API comes with methods to embed the most common Java types as headers via the respective method calls. To set a value of the data type Double as a JMS message property, use the method call `JMSMessage.setDoubleProperty(String name, Double value)` for example.

The default JMS marshaler that comes with Apache ServiceMix embeds the JMS message properties as Message Properties into the Normalized Message dispatched to the Normalized Message Router. This master's thesis embeds the unique Tenant Context Key as a JMS message property, because of the inability of the JMS message properties to reflect the nested structure

of the full Tenant Context. The marshaler of the JMS Binding Component reads the JMS message property that carries the Tenant Context Key, looks up the full Tenant Context from the tenant registry, and embeds it into the Message Properties of the Normalized Message.

For backward compatibility, the changed marshaler first checks if a Tenant Context is present in the incoming message. If not, it handles the message the way the default marshaler of Apache ServiceMix handled any message before this master's thesis extended it by multi-tenancy awareness.

If the payload of the incoming JMS message is a SOAP message itself, the implementation of a custom marshaler for the JMS Binding Component can extract the Tenant Context from the SOAP header and embed it as Message Properties into the Normalized Message as described in the first part of this chapter. However, this master's thesis uses JMS message properties. The goal is to clearly distinguish between message headers or properties on the one hand and message content or payload on the other hand. In this case, the payload consists of the SOAP message including its headers, whereas the header is represented by the JMS message properties.

6.1.4 E-Mail

Apache ServiceMix comes with the Binding Component `servicemix-mail`, which allows to connect to e-mail accounts via the protocols IMAP and POP to receive e-mails or SMTP to send e-mails. Again, Apache ServiceMix allows the implementation and use of custom marshalers to marshal the communication from or to the Normalized Message Router via e-mail. A custom marshaler extends the abstract Java class `org.apache.servicemix.mail.marshaler.AbstractMailMarshaler` and implements the two methods shown in Listing 6.6. To embed the Tenant Context into Normalized Messages that are marshaled from incoming e-mail messages, lines 1 and 2 show the stub of the method to modify.

```
1 convertMailToJBI(MessageExchange exchange, NormalizedMessage nmsg,
2       MimeMessage mailMsg)
3
4 convertJBIToMail(MimeMessage mimeMessage, MessageExchange exchange,
5       NormalizedMessage nmsg, String configuredSender, String configuredReceiver)
```

Listing 6.6: Abstract Methods of The E-Mail Marshaler to Implement

Because e-mail allows the use of key value pairs as custom headers, this approach tries to embed the tenant information into these headers. However, this would result in an overflowing number of non-obviously related headers, because a single header lacks the ability to represent the nested structure of the Tenant Context. This would especially be the case if the Tenant Context contains several optional entries. Therefore, this master's thesis embeds only the `tenantContextKey` into the e-mail header and, on arrival of an e-mail, looks up the full Tenant Context from the tenant registry with this key. As mentioned in Subsection 4.3.3,

Apache ServiceMix does not yet come with such a registry, but Muhler [Muh12] is working on implementing this functionality. This master's thesis uses a simple method that returns the Tenant Context for predefined tenants. This method can be extended by the relevant service calls to the tenant registry as soon as this additional component exists.

The default e-mail marshaler that comes with the `servicemix-mail` Binding Component of Apache ServiceMix already embeds each e-mail header into the Message Properties of the Normalized Message dispatched to the Normalized Message Router. Within the method `convertMailToJBI` shown in lines 1 and 2 in Listing 6.6, the method call `nmsg.getProperty( "tenantContextKey")` retrieves the Tenant Context Key embedded in the e-mail header. Because this key uniquely identifies a specific Tenant Context, a request to the tenant registry retrieves the full Tenant Context, which the custom marshaler can embed into the Message Properties of the Normalized Message through the method call `nmsg.setProperty(String name, Object value)` before it dispatches the message to the Normalized Message Router.

For backward compatibility the changed marshaler first checks if tenant data are present in the incoming e-mail. If not, it handles the e-mail the way the default marshaler of Apache ServiceMix handled any e-mail before this master's thesis extended it by multi-tenancy awareness.

Listing 6.7 shows the Spring bean configuration file `xbean.xml` for the bean `mail:poller`. Lines 11 through 13 are optional and show the configuration of a custom marshaler.

```
1  <beans>
2     <!-- Namespaces omitted to improve readability -->
3
4     <mail:poller service="getTransportRequestService"
5        endpoint="getTransportRequest"
6        targetService="customTargetService"
7        targetEndpoint="customTargetEndpoint"
8        connection="imaps://user@mailProvider/INBOX?password=secret"
9        deleteProcessedMessages="false" processOnlyUnseenMessages="true" >
10
11       <property name="marshaler">
12          <bean class="de.unistuttgart.iaas.taxiRequest.mail.marshaler" />
13       </property>
14
15    </mail:poller>
16
17 </beans>
```

Listing 6.7: Configuration of E-Mail Service Unit via `xbean.xml`

As with the JMS Binding Component, the custom marshaler could extract tenant related data from the message content itself. However, this master's thesis does not. The goal is to clearly distinguishes headers or properties of messages on the one hand and content or payload of messages on the other hand.

6.2 Service Engines

As the Tenant Router proposed in Section 5.5 relies on message enrichment and message routing, these tasks have to support multi-tenancy. Because enrichment and routing components are a subset of the Service Engines in Apache ServiceMix, the sections in this chapter explain the implementation of the specific Service Engines.

The Service Engines in this work are Enterprise Integration Patterns [cf. HW03] and are used within Apache ServiceMix through its ability to interact with Apache Camel. To implement several Enterprise Integration Patterns, the component `servicemix-camel` of Apache ServiceMix can be either used once or separately for each pattern. This master's thesis uses a separate `servicemix-camel` Service Engine for each pattern, because this provides each implementation its own endpoint, allowing other components to reuse each pattern implementation separately rather than all patterns combined as one endpoint. The file `camel-context.xml` contains the configuration for each component. Within this file the patterns are either used via the Spring Expression Language [cf. Theh], or the configuration file contains a reference to a Java Bean, through which the patterns are used via the Bean Language [cf. Thed]. This master's thesis uses the Bean Language, because it bases on Java Beans and therefore favors extensibility, because Java code that already exists can easily be embedded. This is important with regards to the upcoming tenant and service registry or for later extension by dynamic policy based service discovery.

To maintain the goal of backward compatibility, the Service Engines of this implementation contain handling assignments for messages without a Tenant Context. These are simply handled the way Apache ServiceMix would have handled them before this master's thesis made any changes to it.

6.2.1 Content Enricher

To use the Content Enricher pattern, the Bean Language allows to associate a `Processor` with an endpoint that receives messages. The `Processor` is a Java method that can retrieve the incoming message from the message exchange, enrich it in any way possible within Java and pass it on to the next endpoint. Listing 6.8 shows an example Content Enricher that the evaluation scenario uses to enrich the service reply with the tenant's contact preferences.

```
1 public class MyRouteBuilder extends RouteBuilder {
2
3    public void configure() {
4
5       from("jbi:endpoint:http://iaas.uni-stuttgart.de/taxiRequest/
6                replyContentEnricher/replyContentEnricherEndpoint").
7          process(new Processor() {
8             public void process(Exchange exchange) {
9                Message in = exchange.getIn();
```

```
10                if (in.getHeader("tenantContextKey") != null) {
11                   in.setHeader("tenantContactPreference",
12                         getTenantContactPreference(
13                         in.getHeader("tenantContextKey")); } } } ).
14       to("jbi:endpoint:http://iaas.uni-stuttgart.de/taxiRequest/
15                contentBasedReplyRouter/contentBasedReplyRouterEndpoint");
16
17    }
18 }
```

Listing 6.8: Example Content Enricher in Bean Language

Lines 5 and 6 in Listing 6.8 show the endpoint the Content Enricher establishes to receive messages whereas lines 14 and 15 in Listing 6.8 show the endpoint the Content Enricher forwards the message to after it successfully enriched the message. Because the Tenant Router is a unit of Content Enricher and Content Based Router, the target endpoint of the Content Enricher shown in Listing 6.8 is the relevant Content Based Router, which the next subsection describes. Line 7 in Listing 6.8 shows the association of a `Processor` with the message transfer that the Java method `process(Exchange exchange)` implements as shown in lines 8 through 13. In this case this master's thesis uses an inline method, because of its simplicity and for demonstration purposes. For more complex processing methods, the Bean Language can reference any Java Bean as a processor [cf. Thee]. Line 10 in Listing 6.8 shows the check whether a Tenant Context can be found in the message. If not, no additional headers are set and the message is forwarded without any modification to keep backward compatibility with non tenant aware messages. At the time of writing, the method call `getTenantContactPreference(String tenantContextKey)` in line 12 and 13 in Listing 6.8 leads to a mock-up service and will be used to query the future tenant registry described in Subsection 4.3.2.

6.2.2 Content Based Router

A `choice()` construct in the Apache Camel Bean Language represents the Content Based Router pattern. Listing 6.9 shows the Content Based Router implementation used in the evaluation scenario of this master's thesis. It routes the reply to the customer and uses the protocol that the contact preference of the tenant specifies. The address is still the one of the customer, the tenant's preferences just influence the way the taxi service provider contacts the customer.

```
1 public class MyRouteBuilder extends RouteBuilder {
2
3    public void configure() {
4
5       from("jbi:endpoint:http://iaas.uni-stuttgart.de/taxiRequest/
6                contentBasedReplyRouter/contentBasedReplyRouterEndpoint").
7          choice().
```

```
8             when(header("tenantContactPreference").isEqualTo("mail")).
9                to("direct:mail").
10            when(header("tenantContactPreference").isEqualTo("xmpp")).
11               to("direct:xmpp").
12            otherwise().
13               to("direct:nonTenantAwareMessageEndpoint");
14
15       from("direct:mail").
16          recipientList(simple("smtps://mailsender@smtp.gmail.com:465?
17                   password=secret&to=${header.customerMailContact}"));
18
19       from("direct:xmpp").
20          recipientList(simple("xmpp://talk.google.com:5222/
21                   ${header.customerXmppContact}?serviceName=gmail.com&
22                   user=xmppsender@googlemail.com&password=secret"));
23
24    }
25 }
```

Listing 6.9: Example Content Based Router in Bean Language

The Bean Language allows many kinds of expressions to evaluate the choices [cf. Thed]. In this case the method call `header("tenantContactPreference").isEqualTo("mail")` in lines 8 and 10 checks whether the Message Property `tenantContactPreference` of the Normalized Message carries the value `mail` or `xmpp`. This is the header that was previously set by the Content Enricher that Subsection 6.2.1 introduced.

Line 5 in Listing 6.9 shows the endpoint the Content Based Router is listening at to receive messages. The `choice()` construct follows the endpoint, evaluates the Message Property `tenantContactPreference` and routes to the relevant intermediary endpoint or to an endpoint that handles non tenant aware messages. This is important to keep backward compatibility with non tenant aware messages. Lines 15 to 22 in Listing 6.9 show the intermediary endpoints, which are needed, because Apache Camel is not able to evaluate dynamic expressions in the static `to("someEndpoint")` clause. As this is the only valid clause allowed within the choice construct, the Content Based Router uses intermediary endpoints to dynamically construct the reply endpoint based on the customer contact data that the Message Properties of the Normalized Message carry.

6.3 Evaluation

Section 1.3 introduced a motivating scenario to review the approach taken in this master's thesis. It sketched an example process within the taxi scenario that a multi-tenant aware ESB should be able to successfully execute. This section returns to this motivating scenario and uses it for the evaluation of the implemented approach. The specification in Chapter 4 refined the abstract requirements the motivating scenario defined and narrowed them down to precise functional and non-functional requirements a multi-tenant aware ESB needs to

meet. It further established use cases to clarify how an actual process would reflect these requirements. In short this leads to the following evaluation criteria a multi-tenant aware ESB needs to be able to fulfill:

- The messages of the internal messaging format need to relate to a tenant.
- The adapters have to identify tenants and relate tenant data to internal messages.
- The ESB needs to be able to route or handle messages based on tenant specific demands.
- The ESB has to correctly map responses from long running services to the tenant that originally sent out the request.
- The multi-tenancy approach must not obstruct load balancing functionality.
- The implementation should be portable through different versions of the used product.
- The ESB has to serve requestors and providers that are not multi-tenant aware yet.

The implementation in Chapter 6 showed how this work changed the specific components and implemented the evaluation process within Apache ServiceMix. It modified the marshalers of the adapters, called Binding Components in Apache ServiceMix, to embed the tenant data into the Message Properties of the Normalized Messaging Format. This meets the first two criteria, whereas the use of the marshalers avoids changes to the Binding Components and makes the implementation portable across different versions of Apache ServiceMix. The marshalers are further able to distinguish among messages that carry tenant data and messages that do not. This allows non-multi-tenant aware requestors and service providers to still use Apache ServiceMix. The routing and processing components, a subset of the Service Engines in Apache ServiceMix, are now able to handle messages in a multi-tenant aware manner and to correlate responses of long running services to the correct requests via correlation IDs. The implementation took care not to obstruct load balancing for example by choosing a Tenant Router pattern that can be easily extended by further routing patterns that enable load balancing.

This master's thesis built a demo that bases on the process the motivating scenario in Section 1.3 defines. It shows the correct handling of all the criteria above within this example process. Figure 6.1 shows a screenshot of the combination of a SOAP over HTTP request and customer information by e-mail, whereas Figure 6.2 shows a screenshot of the combination of a JMS request and customer information by XMPP.

The upper part of Figure 6.1 shows the SOAP over HTTP request that is sent to the ESB, the lower part shows the customer information by e-mail. The taxi company with the `tenantID` `16c20253-8605-4b67-9001-935c50c8b707` is a premium tenant and therefore its customers are handled with priority. This is reflected by the arrival of a taxi within seven minutes as stated in the e-mail response. This taxi company configured the processing service to inform its customers by e-mail. The header `customerMailContact` of the SOAP request contains the e-mail address of the taxi customer. This is the recipient of the e-mail shown in the lower part of Figure 6.1 that informs the customer about the taxi arrival time.

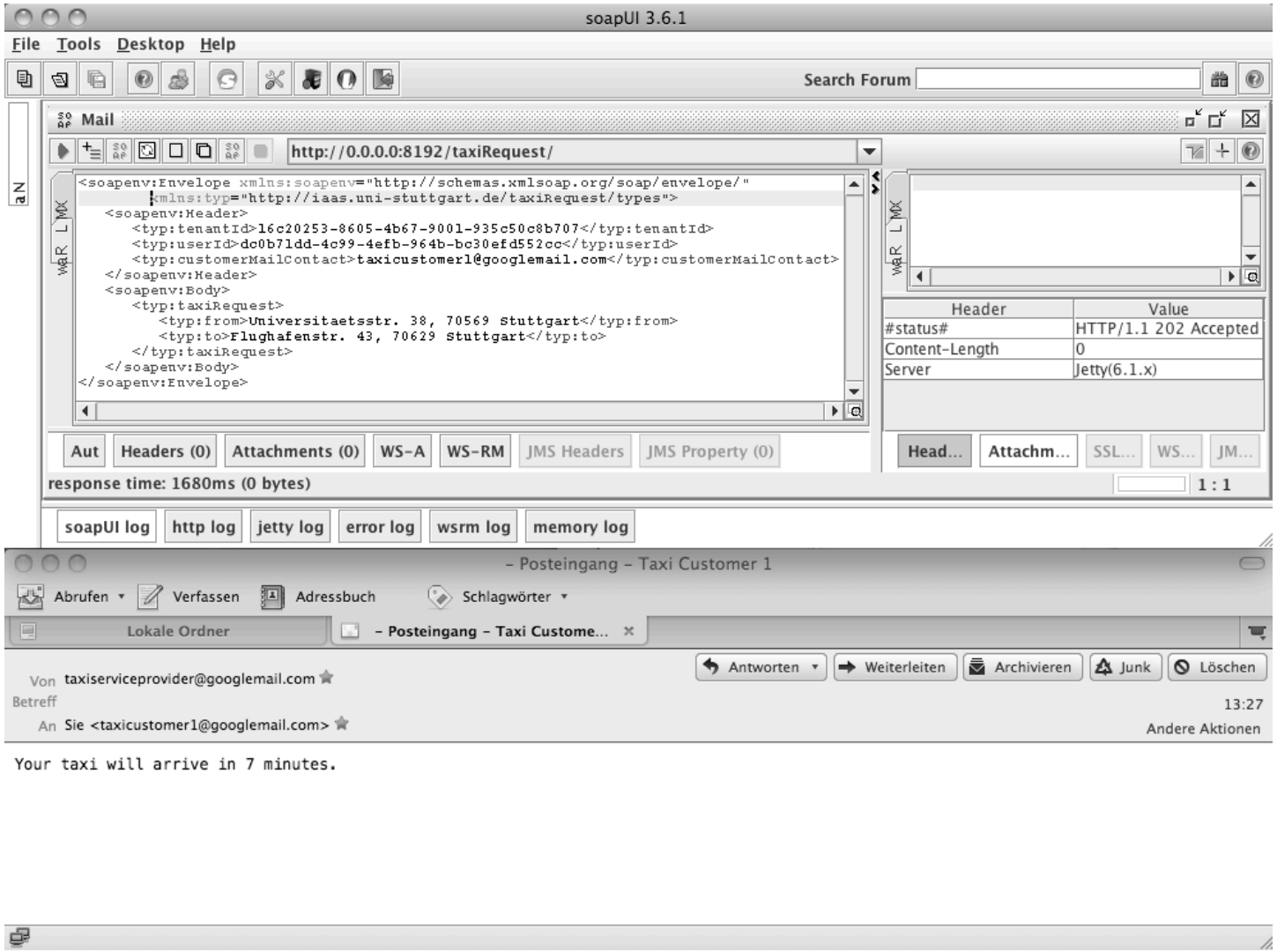

Figure 6.1: Screenshot of The Demo With a SOAP over HTTP Request

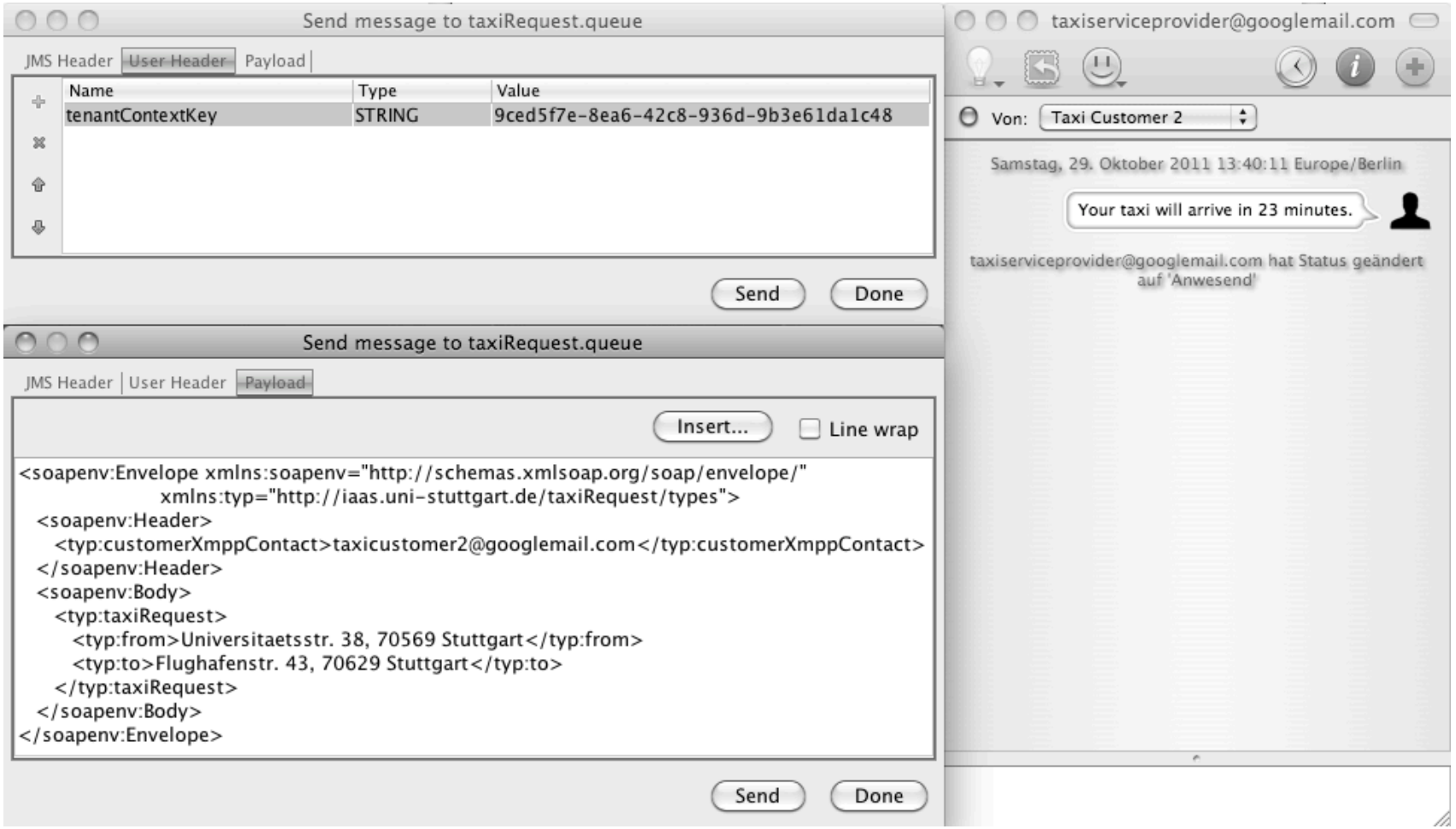

Figure 6.2: Screenshot of The Demo With a JMS Request

The left-hand side of Figure 6.2 shows the JMS request that is sent to the ESB, the right-hand side shows the customer information by instant message over XMPP. The upper part of the JMS request shows the JMS message headers, the lower part shows the message content. The `tenantContextKey 9ced5f7e-8ea6-42c8-936d-9b3e61da1c48` refers to a Tenant Context with the `tenantID 4309595b-93ef-46fd-bcf5-7f32c80b211b`. The JMS marshaler that this master's thesis implemented uses the `tenantContextKey` to look up the `tenantID` and `userID` from the tenant registry upon arrival of the message at the ESB. The specific tenant is an average tenant and therefore its customers are handled without priority. This is reflected by the arrival of a taxi within twenty-three minutes as stated in the instant message response. This taxi company configured the processing service to inform its customers by instant message. The header `customerXmppContact` of the SOAP request in the message content contains the XMPP address of the taxi customer. This is the recipient of the instant message on the right-hand side of Figure 6.2, which informs the customer about the taxi arrival time.

7 Conclusion and Future Work

This master's thesis extended the open source ESB Apache ServiceMix for multi-tenancy support with respect to communication. The goal was to differentiate among tenants within such a setting and enable different handling and routing for each tenant, while keeping backward compatibility for non multi-tenant aware communication.

To achieve the goal, Chapter 3 of this work conducted an extensive analysis of the state of the art of ESBs, elaborated criteria and the operationalization thereof to rate existing ESB products, and chose the open source ESB Apache ServiceMix as the product to evaluate the concept of this master's thesis by implementation and application to a concrete scenario. Chapters 4 and 5 introduced this concept, which includes the Tenant Context and modifications to the components of an ESB that are responsible for the connection to other systems and the routing and handling of requests. Chapter 6 showed the implementation of this concept in the open source ESB Apache ServiceMix. The evaluation in that chapter demonstrated that this concept can achieve multi-tenancy within an ESB and maintain backward compatibility for service providers and requestors that are not multi-tenant aware.

To further extend and enhance this concept, another thesis is currently being worked on to provide Apache ServiceMix with advanced service and tenant registry features [cf. Muh12].

Fehling [Feh09] and Fehling et. al. [FLM10] give valuable directions on how to distribute and load balance tenants in multi-tenancy Cloud computing applications. These can be combined with the concept of this master's thesis to gain a multi-tenant aware ESB that can be load balanced and distributed in a large scale environment.

Mietzner et. al. [MvLW+09] deliver a framework on how to select services that serve a tenant's requests based on policies rather than on defined routing rules. In combination with the concept of this master's thesis, an ESB can be enabled to choose services that provide specific functionality by tenant specific policies. This would provide tenants with more flexible and configurable request routing.

Further, dynamic policy-based service selection would allow the ESB to assign each request a service based on WS-Policy assertions [cf. W3C07] the requestor may specify for each request. The project Web Service Policy Environment (WeSPE) [cf. WeS] shows how WS-Policies can be created and managed, attached to services, and used in an ESB to identify matching services.

Mietzner et. al. [MUTL09] provide multi-tenancy patterns for service providers. These can be adopted by the service providers that publish their services to the multi-tenant ESB to enable these by multi-tenancy awareness.

Bibliography

[AFG+09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, Electrical Engineering and Computer Sciences, University of California at Berkeley, February 2009. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[Ahl10] Mikael Ahlberg. Enterprise Service Buses: A Comparison Regarding Reliable Message Transfer. Master's thesis, KTH Computer Science and Communication, Stockholm, Sweden, 2010.

[AKK10] Irina Astrova, Arne Koschel, and Tobias Kruessmann. Comparison of enterprise service buses based on their support of high availability. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 2495–2496, New York, NY, USA, 2010. ACM.

[Bay08] Thomas Bayer. OpenESB and ServiceMix in Comparison, September 2008. http://www.predic8.com/openesb-servicemix-comparison.htm.

[BGK+11] Michael Behrendt, Bernard Glasner, Petra Kopp, Robert Dieckmann, Gerd Breiter, Stefan Pappe, Heather Kreger, and Ali Arsanjani. IBM Cloud Computing Reference Architecture 2.0, February 2011. https://www.opengroup.org/cloudcomputing/uploads/40/23840/CCRA.IBMSubmission.02282011.doc.

[CC06] Frederick Chong and Gianpaolo Carraro. Architecture Strategies for Catching the Long Tail, April 2006. http://msdn.microsoft.com/en-us/library/aa479069.aspx.

[CCW06] Frederick Chong, Gianpaolo Carraro, and Roger Wolter. Multi-Tenant Data Architecture, June 2006. http://msdn.microsoft.com/en-us/library/aa479086.aspx.

[Cha04] David Chappell. *Enterprise Service Bus*. O'Reilly, Beijing, Cambridge, 2004.

[Cob04] Michael Cobban. What is BPEL and why is it so important to my business?, 2004. http://www.softcare.com/whitepapers/wp_whatis_bpel.php.

[Dav09] Jeff Davis. *Open Source SOA*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2009.

[DMC+06] Saida Davies, William Moore, Albert Chung, Qian Li Jin, Gregorio Patino, and Lakshminarayanan Sreenivasan. *WebSphere Adapter Development*. IBM Redbooks, June 2006.

[Feh09] Christoph Alexander Fehling. Provisioning of Software as a Service Applications in the Cloud. Master's thesis, Institute of Architecture of Application Systems, University of Stuttgart, 2009.

[FLM10] Christoph Fehling, Frank Leymann, and Ralph Mietzner. A Framework for Optimized Distribution of Tenants in Cloud Applications. In *Proceedings of the 2010 IEEE International Conference on Cloud Computing (CLOUD 2010)*, 2010.

[Fou] XMPP Standards Foundation. About XMPP. `http://xmpp.org/about-xmpp/`.

[Fre10] Paul Fremantle. Building Cloud Native Software. Presentation, December 2010. `http://www.slideshare.net/pizak/building-cloud-native-software`.

[Glo] Global WebSphere Community. IBM WebSphere ESB - Community. `http://www.websphereusergroup.org/`.

[Got07] Derek Gottfrid. Self-Service, Prorated Supercomputing Fun!, November 2007. `http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/`.

[GSH+07] Chang Jie Guo, Wei Sun, Ying Huang, Zhi Hu Wang, and Bo Gao. A Framework for Native Multi-Tenancy Application Development and Management. In *E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007. CEC/EEE 2007. The 9th IEEE International Conference on*, pages 551 –558, July 2007.

[Gup08] Rakesh Gupta. Enterprise Service Bus Capabilities Comparison, April 2008. `http://www.ppc.com/documents/enterprisebus.pdf`.

[HLAA08] Eric Hubert, Ruwan Linton, Asanka Abeysinghe, and Afkham Azeez. WSO2 ESB/Apache Synapse Clustering Guide, 2008. `http://wso2.org/files/esb_clustering_guide.pdf`.

[HT04] Gregor Hohpe and Hsue-Shen Tham. Enterprise Integration Patterns with BizTalk Server 2004, July 2004. `http://eaipatterns.com/docs/integrationpatterns_biztalk.pdf`.

[HW03] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[IBMa] IBM. IBM WebSphere ESB. `http://www-01.ibm.com/software/integration/wsesb/about/?S_CMP=wspace`.

[IBMb] IBM. IBM WebSphere ESB - Adapters. `http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.wesb.doc/doc/covw_sca_adapters.html`.

[IBMc] IBM. IBM WebSphere ESB - BPEL. http://www.ibm.com/developerworks/websphere/library/techarticles/0608_kagan/0608_kagan.html.

[IBMd] IBM. IBM WebSphere ESB - Clustering. http://www.ibm.com/developerworks/websphere/library/techarticles/0812_howes/0812_howes.html.

[IBMe] IBM. IBM WebSphere ESB - Documentation I. http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.wesb.doc/info/welcome.html.

[IBMf] IBM. IBM WebSphere ESB - Documentation II. http://www-947.ibm.com/support/entry/portal/All_documentation_links/Software/WebSphere/WebSphere_Enterprise_Service_Bus.

[IBMg] IBM. IBM WebSphere ESB - Dynamic Policy Resolution. http://www.ibm.com/developerworks/websphere/techjournal/0810_tost/0810_tost.html.

[IBMh] IBM. IBM WebSphere ESB - JBI. http://www.ibm.com/developerworks/websphere/library/techarticles/0712_grund/0712_grund.html#N10442.

[IBMi] IBM. IBM WebSphere ESB - Mediation Primitives. http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.websphere.wbpm.scenarios.esb1.doc/concepts/cwesb_scenariooverview.html.

[IBMj] IBM. IBM WebSphere ESB - Metadata. http://www-01.ibm.com/software/integration/wsrr/library/faqs.html.

[IBMk] IBM. IBM WebSphere ESB - Multi-Tenancy. http://www.ibm.com/developerworks/webservices/tutorials/ws-multitenantpart7/index.html.

[IBMl] IBM. IBM WebSphere ESB - Multi-Tenancy Performance Isolation. http://www.ibm.com/developerworks/webservices/library/ws-multitenant/.

[IBMm] IBM. IBM WebSphere ESB - Prerequisites. http://www-01.ibm.com/support/docview.wss?rs=2307&uid=swg27016262.

[IBMn] IBM. IBM WebSphere ESB - Routing. http://www.ibm.com/developerworks/websphere/techjournal/0707_reinitz/0707_reinitz.html.

[IBMo] IBM. IBM WebSphere ESB - Security. http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.websphere.wesb.doc/doc/tsec_secapps.html.

[IBMp] IBM. IBM WebSphere ESB - Service Registry. http://www-01.ibm.com/software/integration/wsrr/library/faqs.html.

[IBMq] IBM. IBM WebSphere ESB - Service Registry Web UI. http://www.ibm.com/developerworks/websphere/library/techarticles/1012_stevens/1012_stevens.html.

[IBMr] IBM. IBM WebSphere ESB - Training. http://www-01.ibm.com/software/websphere/education/.

[IBMs] IBM. IBM WebSphere ESB - Transactions. http://www.ibm.com/developerworks/websphere/library/techarticles/1012_screen/1012_screen.html?ca=drs-.

[IBMt] IBM. IBM WebSphere ESB - Transport Mechanisms. http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wesb610.doc/concepts/cwesb_wesb.html.

[IBMu] IBM. IBM WebSphere ESB - WS-* standards. http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.websphere.wesb610.doc/concepts/cwesb_wesb.html.

[IBMv] IBM. IBM WebSphere ESB - WSDL. http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.websphere.wesb.doc/taghubs/wsdl.html.

[IBM98] IBM Global Services. Improving Systems Availability, 1998. http://www.dis.uniroma1.it/~irl/docs/availabilitytutorial.pdf.

[Jav] Java Community Process. JSR-208 Java Business Integration . http://jcp.org/aboutJava/communityprocess/final/jsr208/index.html.

[JBoa] JBoss by Red Hat. JBossESB - Training. http://www.jboss.com/services/training/.

[JBob] JBoss Community. JBossESB. http://www.jboss.org/jbossesb.

[JBoc] JBoss Community. JBossESB - BPEL. http://community.jboss.org/wiki/JBossESBjBPMIntegration.

[JBod] JBoss Community. JBossESB - Clustering. http://docs.jboss.org/jbossesb/docs/4.9/manuals/html/Programmers_Guide/index.html#d0e1843.

[JBoe] JBoss Community. JBossESB - Community. http://community.jboss.org/en/jbossesb.

[JBof] JBoss Community. JBossESB - Custom Adapters. http://docs.jboss.org/jbossesb/docs/4.9/manuals/html/Programmers_Guide/index.html#chap-SOA_ESB_Programmers_Guide-Connectors_and_Adapters.

[JBog] JBoss Community. JBossESB - Documentation. http://www.jboss.org/jbossesb/docs.

[JBoh] JBoss Community. JBossESB - Drools Integration. http://docs.jboss.org/jbossesb/docs/4.9/manuals/html/Services_Guide/index.html#d0e1159.

[JBoi] JBoss Community. JBossESB - EAI patterns. http://community.jboss.org/wiki/JBossESBEIP.

[JBoj] JBoss Community. JBossESB - Features. http://www.jboss.org/jbossesb/features.html.

[JBok] JBoss Community. JBossESB - JBossWS Integration. http://docs.jboss.org/jbossesb/docs/4.9/manuals/html/Programmers_Guide/index.html#d0e3849.

[JBol] JBoss Community. JBossESB - License. http://www.jboss.org/jbossesb/downloads.

[JBom] JBoss Community. JBossESB - Message Transformation. http://community.jboss.org/wiki/MessageTransformation.

[JBon] JBoss Community. JBossESB - Normalized Message Format. http://docs.jboss.org/jbossesb/docs/4.9/manuals/html/Getting_Started_Guide/index.html#sect-esbaware-unaware.

[JBoo] JBoss Community. JBossESB - Routing. http://docs.jboss.org/jbossesb/docs/4.9/manuals/html/Services_Guide/index.html#chap-content_based_routing.

[JBop] JBoss Community. JBossESB - Service Registry. http://docs.jboss.org/jbossesb/docs/4.2.1GA/manuals/html/services/Registry.html.

[JBoq] JBoss Community. JBossESB - What is an ESB. http://www.jboss.org/jbossesb/resources/WhatIsAnESB.

[JBor] JBoss Community. JBossESB - WS-* standards. http://www.jboss.org/jbossesb/resources/product_overview/standards.html.

[JBos] JBoss Community. JBossWS - Overview. http://www.jboss.org/jbossws.

[Jos07] Nicolai M. Josuttis. *SOA in practice*. O'Reilly Media, Inc., 2007.

[Ker05] Sean Michael Kerner. Enterprise Service Bus Effort Under Apache Incubation, August 2005. http://www.internetnews.com/dev-news/print.php/3528941.

[Log] LogiCoy. Open ESB. http://www.logicoy.com/openesb.html.

[Mal06] Paolo Malinverno, editor. *Service-Oriented Architecture Craves Governance*. Gartner, Inc., January 2006.

[MG09] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing, 2009. http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf.

[Mica] Microsoft. Microsoft BizTalk Server. http://www.microsoft.com/biztalk/en/us/overview.aspx.

[Micb] Microsoft. Microsoft BizTalk Server - Access Control. http://msdn.microsoft.com/en-us/library/aa561838(v=BTS.10).aspx.

[Micc] Microsoft. Microsoft BizTalk Server - Adapters. http://www.microsoft.com/germany/biztalk/interop/adapter.mspx.

[Micd] Microsoft. Microsoft BizTalk Server - BPEL. http://msdn.microsoft.com/en-us/library/aa559758(v=bts.70).aspx.

[Mice] Microsoft. Microsoft BizTalk Server - Clustering. http://msdn.microsoft.com/en-us/library/aa560059(v=bts.70).aspx.

[Micf] Microsoft. Microsoft BizTalk Server - Custom Adapters. http://msdn.microsoft.com/en-us/library/ee268560(v=bts.10).aspx.

[Micg] Microsoft. Microsoft BizTalk Server - Documentation. http://msdn.microsoft.com/en-US/library/dd547397(v=BTS.10).aspx.

[Mich] Microsoft. Microsoft BizTalk Server - Dynamic Policy Resolution. http://207.46.16.252/en-us/library/ee250045(BTS.10).aspx.

[Mici] Microsoft. Microsoft BizTalk Server - Message Transformation. http://msdn.microsoft.com/en-us/library/ee236691(v=BTS.10).aspx.

[Micj] Microsoft. Microsoft BizTalk Server - Normalized Message Format. http://msdn.microsoft.com/en-us/library/aa561650(v=bts.70).aspx.

[Mick] Microsoft. Microsoft BizTalk Server - Routing. http://msdn.microsoft.com/en-us/library/ee236697(v=BTS.10).aspx.

[Micl] Microsoft. Microsoft BizTalk Server - Secure Message Transport. http://msdn.microsoft.com/en-us/library/aa577456(v=BTS.10).aspx.

[Micm] Microsoft. Microsoft BizTalk Server - Training. http://www.microsoft.com/biztalk/en/us/learning-resources.aspx.

[Micn] Microsoft. Microsoft BizTalk Server - Transactions. http://msdn.microsoft.com/en-us/library/ee265616(v=bts.10).aspx.

[Mico] Microsoft. Microsoft BizTalk Server - WS-* standards. http://msdn.microsoft.com/en-us/library/aa475433(v=bts.10).aspx.

[Muh12] Dominik Muhler. Extending an Open Source Enterprise Service Bus for Multi-Tenancy Support Focusing on Administration and Management. Master's thesis, Institut für Architektur von Anwendungssystemen, Universität Stuttgart, 2012.

[Mula] MuleSoft. Mule ESB Community - Application Connectors. http://www.mulesoft.org/muleforge/applications.

[Mulb] MuleSoft. Mule ESB Community - BPEL. http://www.mulesoft.org/documentation/display/MULE2USER/BPM+Transport.

[Mulc] MuleSoft. Mule ESB Community - Community. http://www.mulesoft.org/community-support.

[Muld] MuleSoft. Mule ESB Community - Connectivity and Transports. http://www.mulesoft.org/available-transports.

[Mule] MuleSoft. Mule ESB Community - Custom Connectors. http://www.mulesoft.org/documentation/display/MULE2USER/Creating+Transports.

[Mulf] MuleSoft. Mule ESB Community - Documentation. http://www.mulesoft.org/mule-documentation.

[Mulg] MuleSoft. Mule ESB Community - EAI Patterns. http://www.mulesoft.org/documentation/display/MULE3USER/Using+Message+Routers.

[Mulh] MuleSoft. Mule ESB Community - Embedding Mule in JBI. http://www.mulesoft.org/documentation/display/JBI/Embedding+Mule+in+JBI.

[Muli] MuleSoft. Mule ESB Community - Galaxy Features and Architecture. http://www.mulesoft.org/documentation/display/GALAXY/Galaxy+Features+and+Architecture.

[Mulj] MuleSoft. Mule ESB Community - License. http://www.mulesoft.org/documentation/display/MULE/License.

[Mulk] MuleSoft. Mule ESB Community - Message Transformation. http://www.mulesoft.org/documentation/display/MULE3USER/Using+Transformers.

[Mull] MuleSoft. Mule ESB Community - Mule ESB Community vs. Mule ESB Enterprise. http://www.mulesoft.org/mule-community-vs-mule-enterprise.

[Mulm] MuleSoft. Mule ESB Community - MuleForge. http://www.mulesoft.org/muleforge.

[Muln] MuleSoft. Mule ESB Community - Platforms and Technologies Compatible with Mule ESB. http://www.mulesoft.org/documentation/display/MULE3REFERENCE/Platforms+and+Technologies+Compatible+with+Mule+ESB.

[Mulo] MuleSoft. Mule ESB Community - Training. http://training.mulesoft.com/.

[Mulp] MuleSoft. Mule ESB Community - What is Mule ESB? http://www.mulesoft.org/what-mule-esb.

[MUTL09] Ralph Mietzner, Tobias Unger, Robert Titze, and Frank Leymann. Combining Different Multi-Tenancy Patterns in Service-Oriented Applications. In IEEE, editor, *Proceedings of the 13th IEEE Enterprise Distributed Object Conference*, 2009.

[MvLW+09] Ralph Mietzner, Tammo van Lessen, Alexander Wiese, Matthias Wieland, Dimka Karastoyanova, and Frank Leymann. Virtualizing Services and Resources with ProBus: The WS-Policy-Aware Service and Resource Bus. In *Proceedings of the 7th International Conference on Web Services (ICWS) 2009*, 2009.

[Net05] Network Working Group. A Universally Unique IDentifier (UUID) URN Namespace, July 2005. http://tools.ietf.org/html/rfc4122.

[Ora] Oracle. Java Message Service (JMS). http://www.oracle.com/technetwork/java/index-jsp-142945.html.

[Par66] D. F. Parkhill. *The Challenge of the Computer Utility*. Addison-Wesley Publishing Company, 1966.

[Peta] Petals ESB. Petals ESB. http://petals.ow2.org/.

[Petb] Petals ESB. Petals ESB - Access control. http://doc.petalslink.com/display/petalsesb30/Petals+Technical+Overview#PetalsTechnicalOverview-RouterModule.

[Petc] Petals ESB. Petals ESB - Artifacts. http://doc.petalslink.com/display/petalsesb/Petals+Artifacts.

[Petd] Petals ESB. Petals ESB - Clustering. http://doc.petalslink.com/display/petalsesb30/Topology+Configuration.

[Pete] Petals ESB. Petals ESB - Community. http://petals.ow2.org/community.html.

[Petf] Petals ESB. Petals ESB - Components. http://petals.ow2.org/download-jbi-components.html.

[Petg] Petals ESB. Petals ESB - Distributed Environment. http://doc.petalslink.com/display/petalsesb30/Petals+Technical+Overview#PetalsTechnicalOverview-DistributedEnvironment.

[Peth] Petals ESB. Petals ESB - Documentation. http://doc.petalslink.com/display/welcome/Getting+started.

[Peti] Petals ESB. Petals ESB - JBI. http://doc.petalslink.com/display/petalsesb/Java+Business+Integration.

[Petj] Petals ESB. Petals ESB - License. http://petals.ow2.org/license.html.

[Petk] Petals ESB. Petals ESB - Message Transformation. http://doc.petalslink.com/display/petalscomponents/Petals-SE-XSLT+2.3.

[Petl] Petals ESB. Petals ESB - Normalized Message Format. http://doc.petalslink.com/display/petalsesb/Java+Business+Integration#JavaBusinessIntegration-Messages.

[Petm] Petals ESB. Petals ESB - Orchestration. http://doc.petalslink.com/display/petalscomponents/Petals-SE-EIP.

[Petn] Petals ESB. Petals ESB - Product Datasheet. http://www.petalslink.com/en/products/petals-esb.

[Peto] Petals ESB. Petals ESB - Routing. http://doc.petalslink.com/display/petalsesb30/Petals+Technical+Overview#PetalsTechnicalOverview-RouterModule.

[Petp] Petals ESB. Petals ESB - Service Registry. http://doc.petalslink.com/display/petalsesb30/Petals+Technical+Overview#PetalsTechnicalOverview-TechnicalRegistry.

[Petq] Petals ESB. Petals ESB - SOAP. http://doc.petalslink.com/display/petalscomponents/Petals-BC-SOAP+4.1.x.

[Petr] Petals ESB. Petals ESB - Supported Platforms. http://doc.petalslink.com/display/petalsesb30/Supported+Platforms.

[Pets] Petals ESB. Petals ESB - Training. http://petals.ow2.org/training.html.

[Pett] Petals ESB. Petals ESB - Transactions. http://forum.petalslink.com/Tansactions-in-Petals-ESB-td2686711.html.

[RCL09] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A Taxonomy and Survey of Cloud Computing Systems. In *Fifth International Joint Conference on INC, IMS and IDC*, 2009.

[RD08] Tijs Rademakers and Jos Dirksen. *Open-Source ESBs in Action*. Manning Publications Co., Greenwich, CT, USA, 2008.

[Sch10] Dr. Holger Schmidt. Open Source Enterprise Service Bus (ESB): Ein Vergleich aktueller Produkte. Technical report, Ancud IT-Beratung GmbH, March 2010.

[Thea] The Apache Software Foundation. Apache Camel. http://camel.apache.org/.

[Theb] The Apache Software Foundation. Apache Camel - Adapters. http://camel.apache.org/components.html.

[Thec] The Apache Software Foundation. Apache Camel - Bean Integration. http://camel.apache.org/bean-integration.html.

[Thed] The Apache Software Foundation. Apache Camel - Bean Language. http://camel.apache.org/bean-language.html.

[Thee] The Apache Software Foundation. Apache Camel - Content Enricher. http://camel.apache.org/content-enricher.html.

[Thef] The Apache Software Foundation. Apache Camel - Correlation Identifier. http://camel.apache.org/correlation-identifier.html.

[Theg] The Apache Software Foundation. Apache Camel - Enterprise Integration Patterns. http://camel.apache.org/enterprise-integration-patterns.html.

[Theh] The Apache Software Foundation. Apache Camel - Spring Expression Language. http://camel.apache.org/spel.html.

[Thei] The Apache Software Foundation. Apache CXF - Project Status. http://cxf.apache.org/project-status.html.

[Thej] The Apache Software Foundation. Apache ServiceMix - Adapters. http://servicemix.apache.org/components-list.html.

[Thek] The Apache Software Foundation. Apache ServiceMix - BPEL. http://servicemix.apache.org/bpel.html.

[Thel] The Apache Software Foundation. Apache ServiceMix - Clustering. http://servicemix.apache.org/clustering.html.

[Them] The Apache Software Foundation. Apache ServiceMix - Community. http://servicemix.apache.org/community.html.

[Then] The Apache Software Foundation. Apache ServiceMix - Documentation. http://servicemix.apache.org/documentation.html.

[Theo] The Apache Software Foundation. Apache ServiceMix - Home. http://servicemix.apache.org/home.html.

[Thep] The Apache Software Foundation. Apache ServiceMix - JBI. http://servicemix.apache.org/5-jbi.html.

[Theq] The Apache Software Foundation. Apache ServiceMix - License. http://servicemix.apache.org/license.html.

[Ther] The Apache Software Foundation. Apache ServiceMix - Message Exchange Headers. http://servicemix.apache.org/common-headers.html.

[Thes] The Apache Software Foundation. Apache ServiceMix - NMR. http://servicemix.apache.org/5-jbi.html#5.JBI-NormalizedMessageRouter.

[Thet] The Apache Software Foundation. Apache ServiceMix - Secure Message Transport. http://servicemix.apache.org/security.html.

[Theu] The Apache Software Foundation. Apache ServiceMix - Security. http://servicemix.apache.org/nmr/5-security.html.

[Thev] The Apache Software Foundation. Apache ServiceMix - Transactions. http://servicemix.apache.org/transactions.html.

[Thew] The Apache Software Foundation. Apache ServiceMix - WAR Deployment. http://servicemix.apache.org/war-deployment.html.

[Thex] The Apache Software Foundation. Apache ServiceMix - WSDL. http://servicemix.apache.org/servicemix-http.html.

[They] The Internet Engineering Task Force (IETF). RFC 5322 - Internet Message Format. http://tools.ietf.org/html/rfc5322.

[TNB06] Jan Trautvetter, Aydin Necati, and Felix Billau. Vergleich von kommerziellen Implementierungen eines Enterprise Service Bus. Technical report, Institut für die Architektur von Anwendungssystemen, Universität Stuttgart, May 2006.

[W3C07] W3C Web Services Policy Working Group. Web Services Policy 1.5 - Framework, September 2007. http://www.w3.org/TR/ws-policy/.

[WCL+05] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, and Donald F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Prentice Hall PTR, March 2005.

[WeS] WeSPE. Web Service Policy Environment. http://www.iaas.uni-stuttgart.de/forschung/projects/wespe/.

[Woo06] Robert Woolley. Enterprise Service Bus (ESB) Product Evaluation Comparisons. Technical report, Utah Department of Technology Services, October 2006.

[WSOa] WSO2. WSO2 ESB. http://wso2.com/products/enterprise-service-bus/.

[WSOb] WSO2. WSO2 ESB - Access Control. http://wso2.org/library/articles/2010/10/using-xacml-fine-grained-authorization-wso2-platform.

[WSOc] WSO2. WSO2 ESB - Business Process Server. http://wso2.com/products/business-process-server/.

[WSOd] WSO2. WSO2 ESB - Business Rules Server. http://wso2.com/products/business-rules-server/.

[WSOe] WSO2. WSO2 ESB - Community. http://wso2.org/.

[WSOf] WSO2. WSO2 ESB - Custom Adapters. http://wso2.org/library/articles/2010/06/wso2-esb-business-service-adapter-part-1-integration-paypal-ws-api.

[WSOg] WSO2. WSO2 ESB - Documentation. http://wso2.org/project/esb/documentation.

[WSOh] WSO2. WSO2 ESB - Features. http://wso2.org/projects/esb/java/features.

[WSOi] WSO2. WSO2 ESB - JBI. http://wso2.org/forum/thread/3253.

[WSOj] WSO2. WSO2 ESB - License. http://wso2.org/licenses.

[WSOk] WSO2. WSO2 ESB - Message Transformation. http://wso2.org/project/esb/java/3.0.1/docs/mediator_guide.html.

[WSOl] WSO2. WSO2 ESB - Multi-Tenancy. http://soa-platform.blogspot.com/2010/09/architecture-for-multitenant-enabling.html.

[WSOm] WSO2. WSO2 ESB - Normalized Message Format. http://thiliniishaka.blogspot.com/2010/08/wso2-enterprise-service-bus-esb.html.

[WSOn] WSO2. WSO2 ESB - Secure Message Transport. http://wso2.org/project/esb/java/3.0.1/docs/.

[WSOo] WSO2. WSO2 ESB - Service Registry. http://wso2.org/project/esb/java/1.7.1/docs/release_notes.html.

[WSOp] WSO2. WSO2 ESB - Standalone. http://wso2.org/project/esb/java/3.0.1/docs/admin_guide.html#Standalone.

[WSOq] WSO2. WSO2 ESB - Synapse. http://wso2.com/wp-content/themes/wso2ng/images/wso2_esb_product_data_sheet.pdf.

[WSOr] WSO2. WSO2 ESB - Training. http://wso2.com/support/.

[WSOs] WSO2. WSO2 ESB - Transactions. http://wso2.org/project/esb/java/3.0.1/docs/transactions.html.

[WSOt] WSO2. WSO2 ESB - User Management. http://wso2.org/project/esb/java/3.0.1/docs/admin_guide.html#UserMan.

[WSOu] WSO2. WSO2 ESB - WSO2 ESB on IBM WebSphere Application Server. http://wso2.org/library/knowledge-base/install-wso2-esb-ibm-websphere-application-server.

[WSO10] WSO2. Cloud Native: Essential Characteristics of Platform-as-a-Service Offerings. White Paper, December 2010.

All links were last followed on October 30, 2011

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Ich erkläre außerdem, dass an mich an keiner wissenschaftlichen Hochschule bereits ein Thema zur Bearbeitung als Masterarbeit oder als vergleichbare Arbeit in einem gleichwertigen Studiengang vergeben worden ist.

Stuttgart, den 02. November 2011

(Stefan Essl)