بسم الله الرحمن الرحيم

Sudan University of Science and Technology

College of Graduate Studies

College of Computer science and Information Technology

Implementing Integrating Enterprise Systems in Sudan -Using Enterprise Service Bus (ESB)

تطبيق لتكامل انظمة المؤسسة في السودان - باستخدام ناقل الخدمة

This thesis is submitted in partial fulfillment of the academic requirements
for the degree of
Master in Computer Science

By: Ahmed Hamza Abdelmonim.

Supervisor: Dr.Wisal M. Tingari.

February 2016

الاهداء

أَعُوذ بِاللَّهِ مِنْ الشَّيْطَان الرَّجِيم

(وَقَضَىٰ رَبُّكَ أَلَّا تَعْبُدُوا إِلَّا إِيَّاهُ وَبِالْوَالِدَيْنِ إِحْسَانًا) الاية (23) سورة الاسراء

امي وابي ربنا يحفظكم ويجزيكم عنا خير الجزاء

(مِنْ آيَاتِهِ أَنْ خَلَقَ لَكُم مِّنْ أَنفُسِكُمْ أَزْوَاجًا لِّتَسْكُنُوا إِلَيْهَا) الاية (21) سورة الروم

زوجتي و شريكتي

اخواني واصدقائي

Acknowledgements

I would first like to thank my Allah. Special thanks to Dr. Wisal M. Tingari, the door to her office was always open whenever I ran into a trouble spot or had a question about my research or writing. She consistently allowed this research to be my own work, but steered me in the right direction whenever she thought I needed it.
Finally, I must express my very profound gratitude to my wife, my family and my friends, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Abstract

The main objective of this research is to implement an enterprise system integration using an enterprise service bus (ESB) in Sudan.

The research questions are; what framework to be used to complete the integration? And what are the mechanisms that will be used to make sure that the integration process is successful? Finally, how to choose the workflow engine and what are the criteria for the choice. To achieve the research objectives, different approaches to integrate enterprise systems were studied, and enterprise service bus (ESB) was selected. Accordingly the integration environment was set up, and an appropriate workflow engine was demonstrated. Finally, the required adaptors of each integrated system were developed, and the proposed system was tested and implemented.

The research recommends for the use of cluster architecture and load balancing ESB. Moreover, it recommends of use security token service and service locator to maximize the benefits of ESB.

مستخلص

الهدف من البحث تطبيق الية للتكامل بين انظمة المؤسسة في السودان، وذلك من خلال تنفيذ منهجية ناقل الخدمة.

اسئلة البحث هي، ما هو الاطار الذي سيستخدم لاكمال عملية التكامل بين الانظمة وماهي الالية التي سوف يتم بها الاختبار والتأكد من نجاح عملية التكامل وكيف سوف يتم اختيار محرك سير عمل مناسب للتكامل. لتحقيق هدف البحث

تمت دراسة مجموعة من طرق التكامل وتم اختيار منهجية ناقل الخدمة. وعلى ضوء ذلك تم تجهيز بئية التكامل, وتم توضيح محرك سير العمل المختار. واخيرا تم تطوير المحولات التقنية المطلوبة في كل نظام متكامل، واخيرا تم اختبار وتشغيل بيئة التكامل.

توصيات البحث هي استخدام منهجية توزيع الاحمال في بيئة ناقل الخدمات لزيادة الاداء والاعتمادية مع ضرورة استخدام خدمات التأمين ومحدد الخدمات لزيادة مرونة التكامل.

List of Contents

List of Tables

List of Figures

List of Abbreviations

ESB	Enterprise Service Bus
OpenKM	Open Knowledge Management
DMS	Document Management System
EAI	Enterprise Application integration
MIS	Management information system
MOM	Message Oriented Middleware
SOA	Service Oriented Architecture
WSDL	Web Services Description Language
SOAP	Simple Object Access Protocol
HTTP	Hypertext Transfer Protocol
XML	Extensible Markup Language
SMB	Small and Medium Businesses
GUI	Graphical User Interface
B2B	Business-to-Business
CPU	Central Processing Unit
BPMS	Business Process Management System
WFMS	Workflow Management System
TCO	Total Cost of Ownership
OEM	Original Equipment Manufacturer
J2EE	Java 2 Enterprise Edition
CIM	Common Information Model
DICOM	Digital Imaging and Communications in Medicine
WSN	Wireless Sensor Networks
GG	Google Gadgets
NI	National Instruments
DAQ	Data AcQuisition
EIP	Enterprise Integration Patterns
ASF	Apache Software Foundation
API	Application Program Interface
JAX-WS	Java API for XML Web Services
JAX-RS	Java API for RESTful Web Services
OSGI	Open Service Gateway Initiative
JMX	Java Management Extensions
JAR	Java ARchive
KAR	Karaf ARchive
JMS	Java Message Service
SVN	Subversion

JSON	JavaScript Object Notation
STS	Security Token Service
SAML	Security Assertion Markup Language
XPDL	XML Process Definition Language
jPBM	java Business Process Management
LGPL	Lesser General Public License
GPL	General Public License
GWT	Google Web Toolkit
JDK	Java Development Kit
RDBMS	Relational DataBase Management System
ERP	Enterprise resource planning
TAC	Talend Administration Center
SAM	Service Activity Monitoring
IoT	Internet of Things

Chapter One: INTRODUCTION

1.1. Preface

One of the challenges facing the architect is the integration of applications. Implementing enterprise integrations can be complex and daunting, especially for organizations with a legacy information technology environment. The practical approach to integrations should result in maximizing return on investment and achieve a forward-looking, flexible architecture aligned with broader enterprise architecture goals and the emergence of new technologies in the marketplace. An enterprise service bus (ESB) is a standards-based connectivity layer used to integrate distributed systems across functional, enterprise and/or geographic boundaries

It provides a combination of service enablement, messaging, transformation, routing and mediation to address a wide variety of enterprise integration challenges. The ESBs are reliably and securely connected distributed systems and remote locations in a flexible way while reducing the number, size and complexity of application interfaces. ESBs have primarily been the province of only large companies. Gartner estimates that core ESB features are adopted by more than 50% of large organizations (Gartner, 2013), but this pattern is changing. The mid-market is now experiencing faster growth in application integration than large enterprises who have been locked into proprietary, non-standard, "black box" solutions, with little input into the evolution of the ESB products they have come to rely on to integrate their business applications.

1.2. Problem Statement

Any x company might have a set of information systems, such as enterprise resource management system, document management system, email, project management system and test management system. All of these systems are isolated from each other and there is no exchange of any data because they are not integrated. Recently x company decided to integrate workflow management system and document management system to standardize the scattered enterprise processes between their systems. And the main obstacle to implementation is the lack of proper integration between systems.

Although there are several methods to integrate the system, still the question what is an optimal method that could meet the needs of the organization to achieve its goals in an integrated, safe and easy system maintenance and follow-up with the possibility of adding any future systems.

1.3. Research Significance

Lack of interoperability, most of the institutes, companies and corporations in the Sudanese market have many systems, but most of that systems are isolated and there is no exchange of information.

High cost, there are some companies were integrated their systems using non-suitable

methodologies, that caused a rise in the cost of maintenance, follow-up and administration operations.

1.4. Research Objectives

The main objective of this research is to implement enterprise system integration in Sudan. Through:

a. Prepare ESB environment
b. Specify the workflow engine
c. Integrate the workflow engine with DMS
d. Test the new shipped system

1.5. Research Questions

To overcome the previously presented shortages in previous, this research may provide convincible answers to the following research questions:

a. What framework to be used to complete an integration between the two enterprise systems?
b. What are the mechanisms that will be used to make sure that the integration process is successful and that the system is stable?
c. How to choose the workflow engine and what are the choice criteria?

1.6. Research Methodology

To achieve the above-mentioned objective the following methods will be followed:

a. Selecting the ESB integration approach, and define the suitable framework that can be applied to the conducted problem.
b. Set up the integration environment which includes and not limited to the data model format, the communication protocol, and message notation.
c. Selecting the appropriate workflow engine.
d. Develop the required adaptor in each the integrated system.
e. Validate and test the proposed system and its components from the view of transparency and information exchange among the system.

1.7. Research Structure

This research will be structured as follows

Chapter one introduction, while in chapter two we will talk about the theoretical background and related work with discussion to their advantages and disadvantages. In chapter three: will discuss the methodology and the framework for proposed solution and the implementation. Finally in chapter four: draw a conclusion and future work.

Chapter Two: THEORETICAL BACKGROUND

2.1.Introduction

In this chapter previous research has been reviewed to provide a clear thought about the enterprise integration then ESB has been defined in more details, lastly, the proposed ESB and the Workflow management system has been provided.

2.1.1. Key Concepts

a. Enterprise Application Integration (EAI)

According to (Freivald, 2010) EAI (enterprise application integration) refers to the plans, methods, and tools aimed at modernizing, consolidating, and coordinating the computer applications in an enterprise. Typically, an enterprise has existing legacy applications and databases and wants to continue to use them while adding or migrating to a new set of applications that exploit the Internet, e-commerce, extranet, and other new technologies. Enterprise also prefers the business processes and the data are shared without being forced to change their structures. And that what EAI provide to them.

Enterprise Application Integration, or EAI, has existed as a technical term since the early 2000s, but the central problem that it attempts to solve is much older. In a nutshell, EAI is an approach, or more accurately, a general category of approaches, to providing interoperability between the multiple disparate systems that make up a typical enterprise infrastructure.

According to (ALSÈNE, 1994),"Since the early days of computing, organizations have aspired to integrated, enterprise-wide information system architectures. Throughout the years, these aspirations have been reflected in the quest for integrated Management Information System (MIS), enterprise-wide data models, and integrated databases".

According to (Woolf, 2012) enterprise application integration is not an easy task because integration is not just one style or method to be used. The approach used to accomplish integration in a typical organization is related to their applications, developed or bought from third party vendors, operating on different platforms and using diverse technologies inside or even outside the company. Additionally, some applications are not designed to be integrated with other applications but their data is critical to other applications. All of this makes the process of integrating applications complicated and critical to every enterprise.

b. Message Oriented Middleware (MOM)

Message-oriented middleware (MOM) is software or hardware infrastructure supporting sending and receiving messages between distributed systems. It allows software components that have been developed independently and that run on different networked platforms to interact with one another. (Wikipedia, 2013)

c. Service Oriented Architecture (SOA)

Service-oriented architecture (SOA) is a design pattern based on distinct pieces of software providing application functionality as services to other applications via a protocol. This is known as service-orientation. It is independent of any vendor, product or technology. (MSDN, 2004)

d. Web Services

A Web service is a method of communication between two electronic devices over a network. It is a software function provided at a network address over the Web with the service always on as in the concept of utility computing (Anon., n.d.)

The W3C defines a Web service as A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. (W3, 2004)

e. Message Broker

A message broker is an architectural pattern for message validation, message transformation and message routing. It mediates communication amongst applications, minimizing the mutual awareness that applications should have of each other in order to be able to exchange messages, effectively implementing decoupling. (Gregor Hohpe, 2003)

f. Enterprise Service Bus (ESB)

ESB is an "Architectural Pattern", "We describe the enterprise service bus first and foremost as an architectural pattern. In fact, it is possible to construct service buses from a variety of different underlying integration technologies. The architecture pattern remains valid and is a guiding principle to enable the integration and federation of multiple service bus instantiations." (High, 2006)

2.1.2. Integration Strategies

Integration can be complex and expensive. There are many integration software vendors in the marketplace. The first step is to determine the integration strategy that will best achieve the business needs. Integration technologies and concepts have evolved over the last decade, leading to a multitude of architectures and products in the IT market. However, according to (Sachin Chandra, 2009) there are really three broad integration strategies:

i. Point-to-Point integration

In a point-to-point integration approach, each application is integrated directly with the other application via an interface module. While interfaces of this type can be built and implemented relatively quickly and cheaply, the approach has limited consideration for enterprise-wide data integration. As more applications are interconnected with each other, the number of integration modules multiply exponentially. Additionally, those interface modules are directly impacted by underlying application upgrades and data changes.

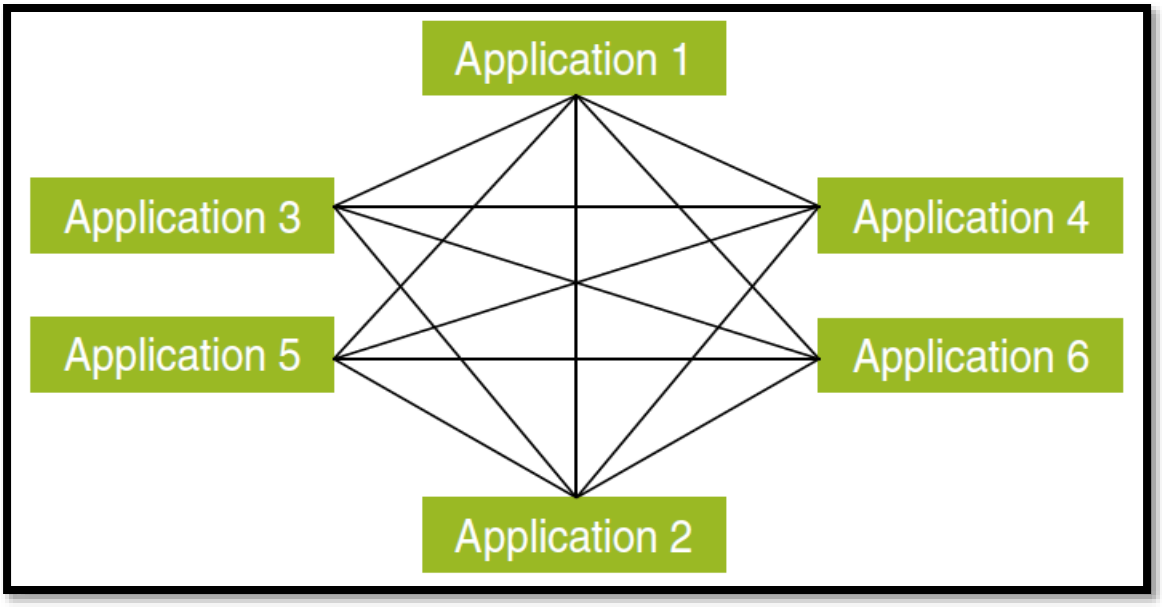

Figure 2.1: Point to Point Approach (Sachin Chandra, 2009)

ii. Spoke-and-hub integration

In a broker approach to EAI, a central integration engine called the broker, resides in the middle of the network, and provides all message transformation, routing, and any other inter-application functionality. All communication between applications must flow through the hub, allowing the hub to maintain data concurrency for the entire network. The broker model allows loose coupling between applications, but like any other architecture model that uses a central engine is that the broker can become a single point of failure for the network.

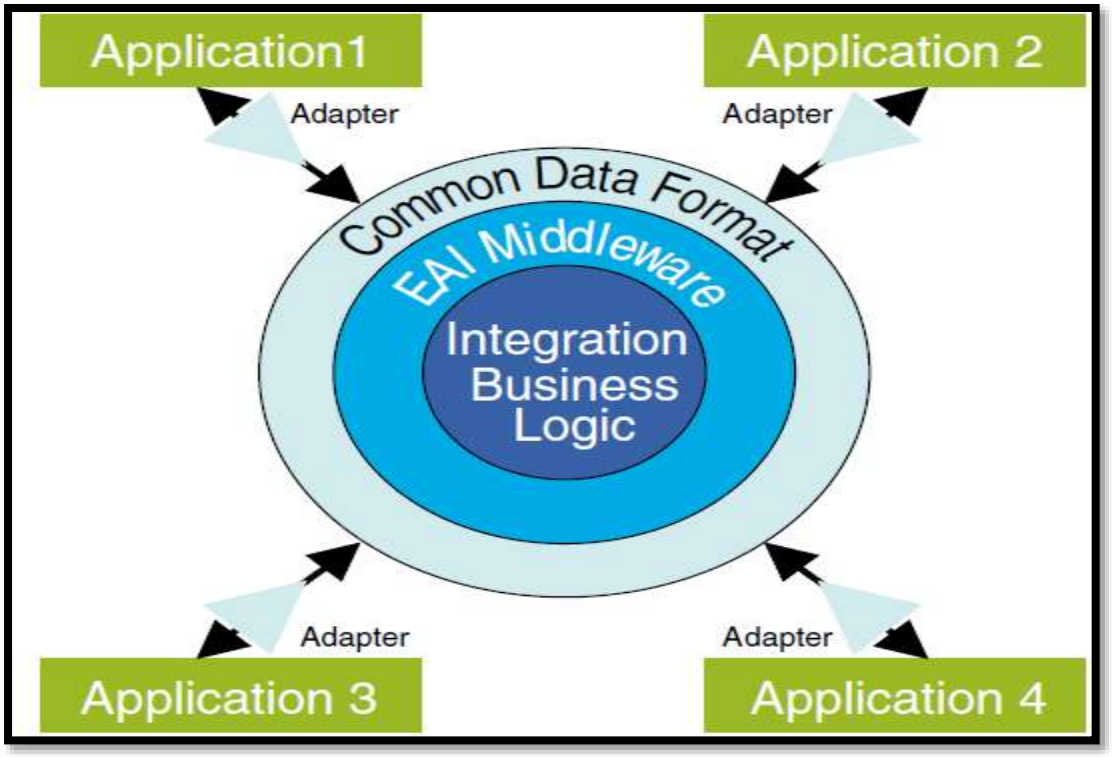

Figure 2.2: EAI Approach (Sachin Chandra, 2009)

iii. Enterprise Service Bus Integration

The ESB Integration strategy is also based on the spoke-and-hub integration topology. With the advent of open, Web-based, and service-oriented business applications, EAI middleware applications have evolved to support Web-based communication standards such as SOAP, XML, HTTP, and other services. The bus architecture sought to lessen the burden of functionality placed on a single component by distributing some of the integration tasks to other parts.

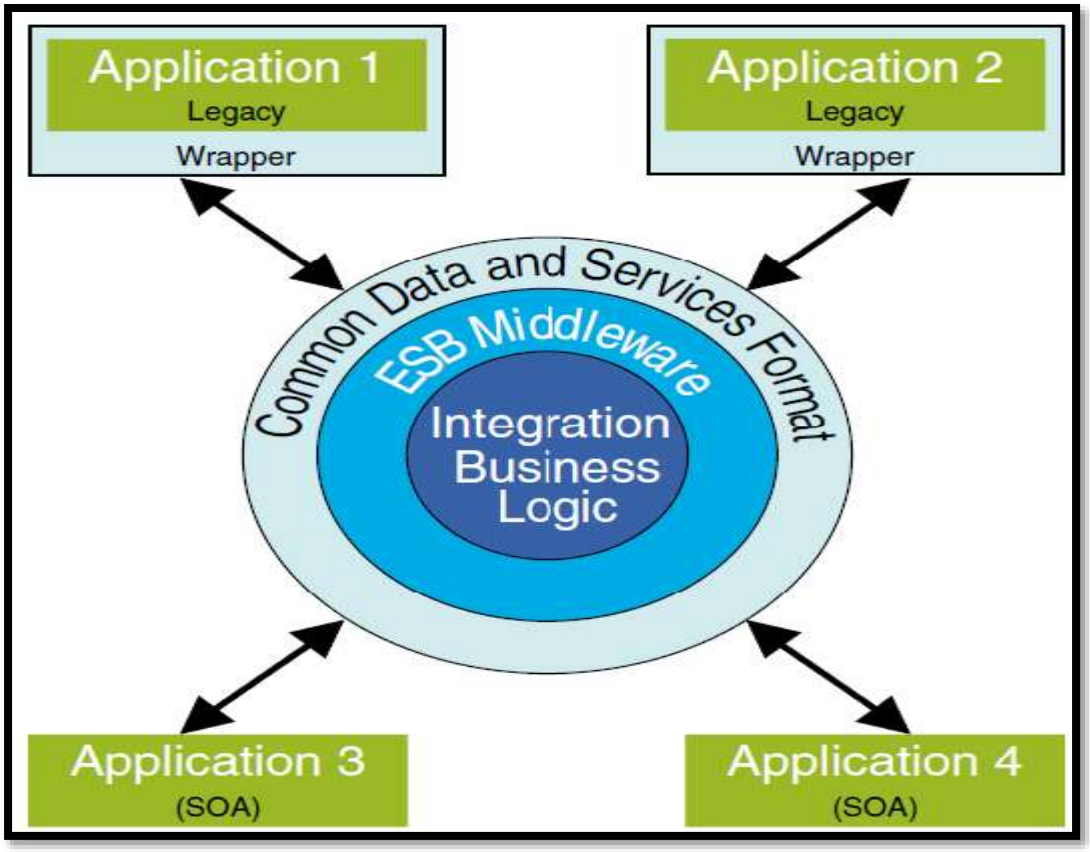

Figure 2.3: ESB Approach (Sachin Chandra, 2009)

2.2. Service Oriented Architecture

In order to define the role of SOA in system integration the related literature in this area was reviewed to illustrate the characteristics of SOA which are beneficial to EAI as a new architectural approach for EAI. It is noticeable that the literature usually talks about SOA and web services because web services are one of the suitable methods to implement SOA architectural model in practice.

The first characteristic of SOA is the definition of service in different literature. (Woolf, 2012) Identified Service as Shared business functions which are well-defined and universally available and responds to requests from "service consumers". According to (High, 2006) SOA is comprised of services that are modularized. These modularized services can then provide coordination to support real-time business processes to function correctly throughout the enterprise. The authors pointed out that SOA is the result of evolution in programming languages and paradigms, distributed computing and business technology.

According to (Woolf, 2012) in SOA integration of a new application is done by using existing remote services provided by other applications, thus calling a service can be regarded as integration between applications. SOA based integration tools usually provide enough simplicity to call an external service the same as a local method.

According to (Papazoglou, et al., 2007), the benefit of using SOA approach is loose coupling allows to break down the integration logic into distinct and easily manageable pieces. Moreover, service orchestration and service choreography are two characteristics that mainly define the interaction protocols coordinating and controlling how services are collaborating.

(Amjad Umara, 2009) Claim that sometimes SOA will not be able to improve the enterprise system regarding complicated issues like:

i. If the target applications are too inflexible and costly to maintain, the integration in-place approach does not work.
ii. Outdated and old functionalities will remain in the system and also the possibility of using a new product more flexible to SOA approach will be ignored.
iii. SOA is producing a great deal of confusion due to its specific array of standards and new products.

According to (Ravi Khadka, 2013) Migration is a multifaceted process that involves technical, organizational and business issues. To manage such a multifaceted process, a central governing body with the suitable governance of the entire migration process is indispensable. Needless to say, a legacy to SOA migration is a complex and challenging process and any failure can threaten the success and fortune of an enterprise.

2.3. Enterprise Service Bus

An enterprise service bus (ESB) is a software architecture for middleware that provides fundamental services for more complex architectures. For example, an ESB incorporates the features required to implement a service-oriented architecture (SOA). In a general sense, an ESB can be thought of as a mechanism that manages access to applications and services (especially legacy versions) to present a single, simple, and consistent interface to end-users via Web- or forms-based client-side front ends. (Rouse, 2012)

According to (IBM, 2009) an ESB enables standards-based integration between loosely-coupled applications and services within and across

a. Services oriented architectures – distributed applications are composed of granular reusable services with well-defined, published and standards-compliant interfaces
b. Message-driven architectures - applications send messages through the ESB to apps
c. Event driven architectures - applications generate and consume messages anonymously

2.3.1. Service Virtualization

SOA Federation solutions focus on taking existing services and ensuring that they meet the requirements of enterprise SOA. To achieve this, the SOA Federation solutions offer a set of core capabilities one of them is Service Virtualization. According to (SOA, 2014) Service Virtualization provides location transparency, service mobility, impedance tolerance and reliable service delivery without requiring a re-platforming of existing platforms or introducing yet another service platform to support the required solution architecture. It can be divided into three sub-categories as follow

a. ESB inter-connects Requestor and Provider
   i. Interactions are decoupled
   ii. Supports key SOA principle – separation of concerns
b. ESB Provides Service Virtualization of
   i. Location and identity
   ii. Interaction pattern and protocol
   iii. Interface
c. ESB also enables Aspect-oriented Connectivity or Mediation
   i. Security
   ii. Log
   iii. Management

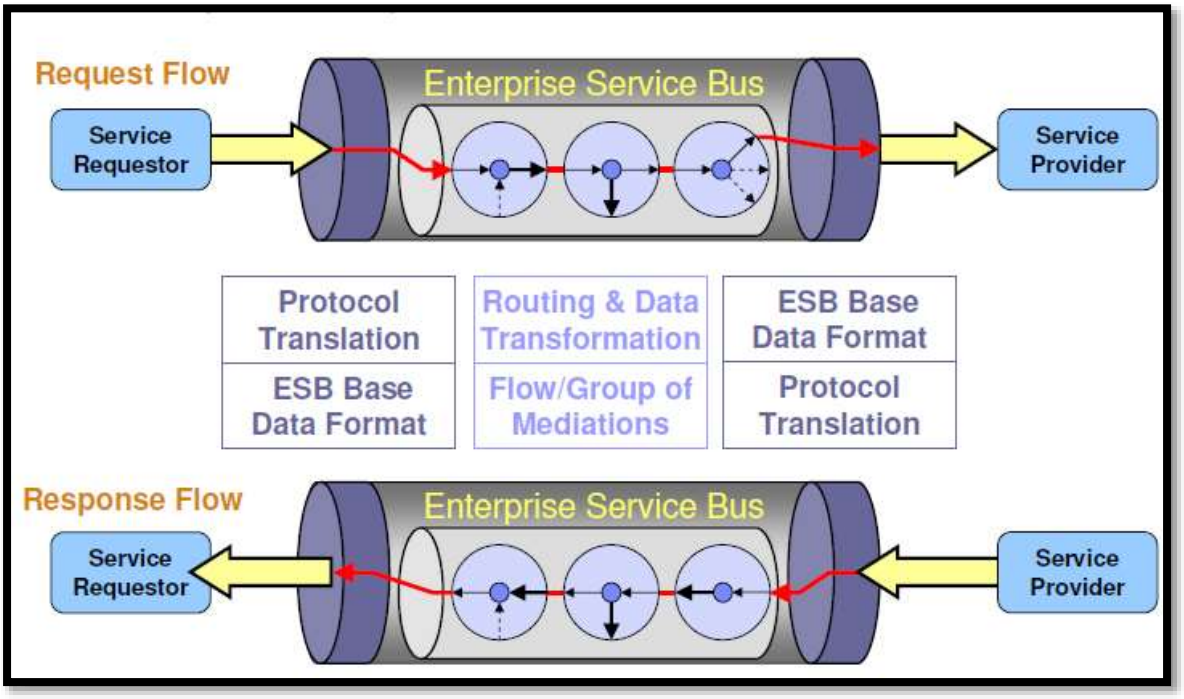

Figure 2.4: ESB Request-Response Flow Review (IBM, 2009)

2.3.2. ESB Routing Rule

In environments where integration is a subject of interest, there exists the necessity to route the messages in an efficient way. This means that a service consumer only receives that piece of information it is interested in, based on the content of the message. In conventional systems, the service provider explicitly specifies the intended message consumers using a unicast or multicast address. This loosely coupling of services also requires some discovery agency, which is capable of connecting the service requestors with the service consumers. Such a discovery of services and binding them does not necessarily happen at design time. It can happen at run-time, dependent on the needs of the service requestor. Content-based routing now comes in play, enabling to read messages and route messages to the right service consumer based on the content of the message. The rule how to interpret the content of the message is called a routing rule. Content-based routing is at the core of systems which integrate between services via the publish-and-subscribe pattern. (yenlo.nl, 2010)

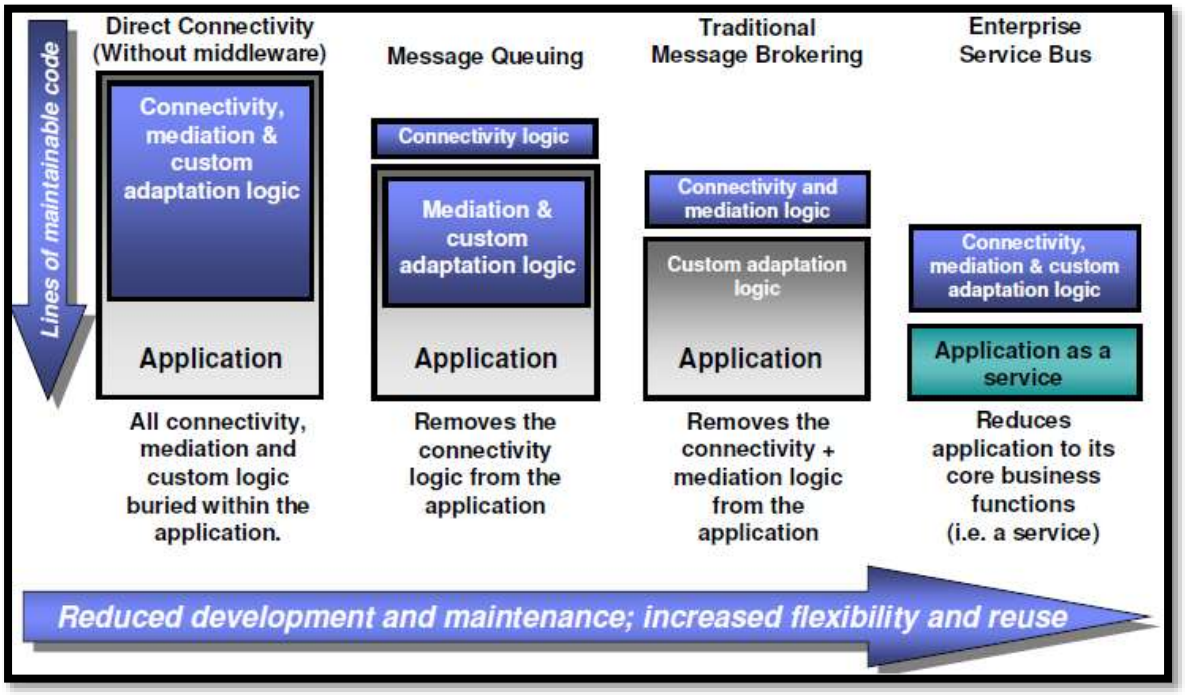

Figure 2.5: ESB Evaluation (IBM, 2009)

2.3.3. ESB's Core Functionalities

There are a number of different ESB products available on the market today, An ESB product should provide a number of core functionalities to be utilized in application integration, below is the summarized list of feature according to (Mason, 2014)

a. Location Transparency: A way of centrally configuring endpoints for messages, so that a consumer application does not require information about a message producer in order to receive messages
b. Transformation: The ability of the ESB to convert messages into a format that is usable by the consumer application.
c. Protocol Conversion: Similar to the transformation requirement, the ESB must be able to accept messages sent in all major protocols, and convert them to the format required by the end consumer.
d. Routing: The ability to determine the appropriate end consumer or consumers based on both pre-configured rules and dynamically created requests.
e. Enhancement: The ability to retrieve missing data in incoming messages, based on the existing message data, and append it to the message before delivery to its final destination.
f. Monitoring / Administration: The goal of ESB is to make integration a simple task. As such, an ESB must provide an easy method of monitoring the performance of the system, the flow of messages through the ESB architecture, and a simple means of managing the system in order to deliver its proposed value to an infrastructure.
g. Security: ESB security involves two main components - making sure the ESB itself handles messages in a fully secure manner and negotiating between the security assurance systems used by each of the systems that will be integrated.

2.3.4. Enterprise Service Bus Model

The previous literature, in general, provides the following conceptual model to introduce ESBs.

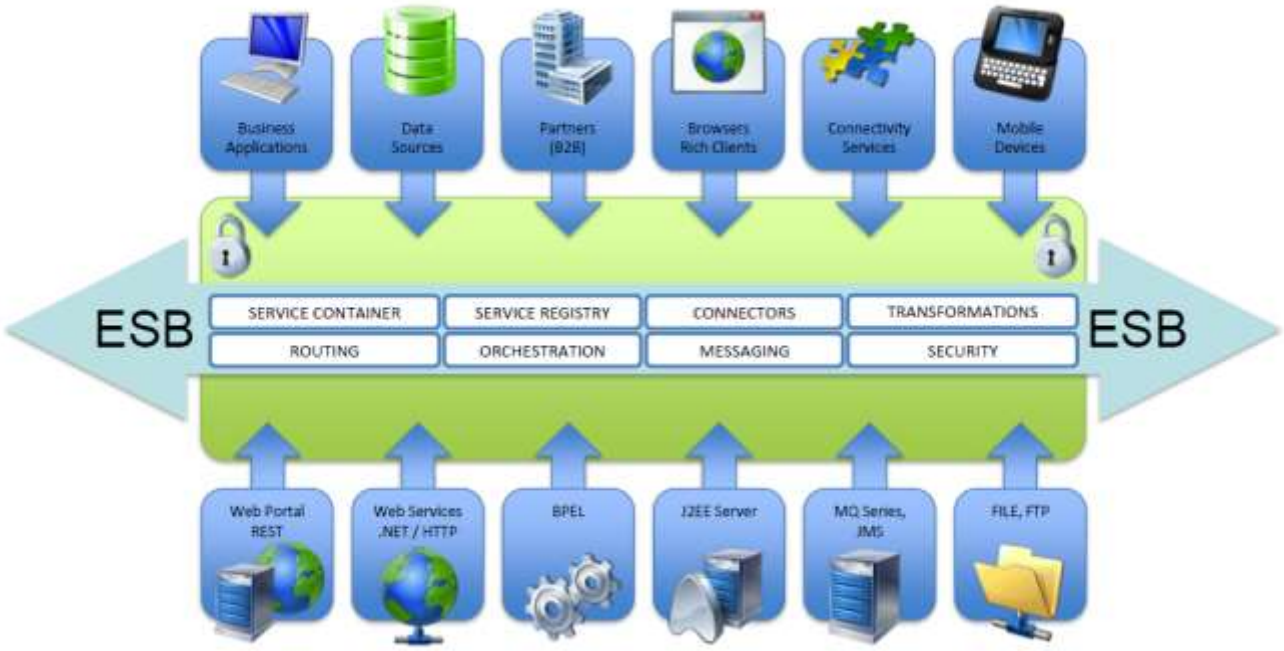

Figure 2.6: Enterprise Service Bus Model (Talend, 2012)

According to the above conceptual model, ESBs play a mediator among different applications based on different platforms and data types. The integration mechanisms provided by an ESB assists the easier way of adding new applications to the integration landscape. Applications communicate with each other through their connection to the ESB, while the complications of implementation the logic behind the integration is mainly dealt by the ESB (Chappell, 2004).

2.4. Related Work

In this section, the literature related to the related work were gathered and reviewed to be able to know the best practices and lessons learned in implementing ESB in different industries.

According to (Dai, 2011) dedicated ESB for power systems has been designed and implemented, the proposed design meeting IEC 61970/61968 standards, and has been developed by Java 2 Enterprise Edition (J2EE) and divided into adaptation layer, service layer and data layer. Common information model (CIM) has been used as the standard of data exchange model and XML has been used to describe the message. The author didn't show a best practice that is followed but he claimed that Objectives of the institution has achieved significantly by used the ESB, but the ESB framework needs more effort in security and log functions.

According to (Chongwen Wang, 2010) DICOM is very complex technology, most of the hospitals used the point-to-point methodology to integrate their DICOM communications. Which eventually lead to high cost of maintenance and expanding process, so to reduce the cost and share the DICOM devices more efficiency. ESB has been designed and implemented to carry the DICOM communication, it has been divided into four modules connector module, message management module, DICOM entity protocol module and service unit module. The author claimed that they are succeeded in implementing communication between DICOM sender and ESB, but according to author the framework needed a lot of work to be more reliable and stable.

According to (Chunmiao, 2012) the purpose of use ESB in Metallurgical production is improve the efficiency and reduce the maintenance cost by providing a model based on ESB to manage the various processes in production. The author designed a new production model that centralized service bus Metallurgical based on the automation and control system. The new model has made great progress in Metallurgical production in china, but compared with international Metallurgical industry there is a considerable gap.

It is not clear why both (Dai, 2011) , (Chunmiao, 2012) and (Chongwen Wang, 2010) designed and implemented their own ESB from scratch, they are not provided any justifies or proof that existing ESB framework did not fit their requirements. But they said there was a significant improvement in their case study after implemented ESB architecture.

According to (Mulik, 2009) their case studies about an enterprise that has distributed

business division. Each one of this division having its own solution that handling its operational functions. Nevertheless, these business units need to integrate with corporate functions and shared services on real-time as well on loosely coupled basis. And the author used the ESB to achieve the enterprise needs. The author chose the Appropriate ESB framework based on clear and reasonable criteria, mainly the ESB vendor reputation and pricing license (TCO). The lessons learned from this research could be as follows:

a. Be prepared to have some vendor lock-in unless there is an unusually high level of commitment to standards within an organization.
b. Provision for proportionately higher coordination efforts in such projects.

According to (U. Raza, 2012) a novel approach has been presented for monitoring a typical plastics industry environment based on three major technologies: Wireless Sensor Networks (WSN), Service Oriented Architecture (SOA) and Google Gadgets (GG). This is applied to a heterogeneous network of WSN nodes and National Instruments (NI) high-speed data acquisition (DAQ) devices. Although SOA mostly used to make software resource run as service, according to the author SOA architecture has been used to enable hardware resources like WSN to be accessed as a service. So to achieve their goal, a proposed system to monitor micro injection molding has been designed and implemented. ESB has been implemented as one of three core modules -WSO2 used as ESB- and the ESB responsible for thousands of WSN that integrated with ESB and GG and DAQ. According to the author, the proposed system has proven to be flexible and easy to use. In future, the author aims to monitor the complete Micro Injection Molding. The author describes systems integration implementation beyond traditional forms of integration, where they used the concepts of SOA, ESB, IoT to link between factory systems and wireless sensor network systems of mechanical devices, which lead them to develop monitor and control system with an acceptable cost.

Chapter Three: METHODOLOGY

3.1. Preface

To achieve the research objective; implementing an enterprise system integration in Sudan, the ESB integration approach was selected, and a suitable framework was defined to be applied to the conducted problem. The following were the phases adopted:

- Setting the criteria for selecting an appropriate mechanism to integrate an enterprise system.
- Selecting the appropriate workflow engine.
- Preparing the integration environment which includes data model format, communication protocol, and message notation.
- Developing the required adaptor in each integrated system.
- Validating and testing the proposed system and its components; based on the view of transparency and information exchange among the system.

3.2. The Proposed Enterprise Service Bus

3.2.1 Criteria

A. According to (Wähner, 2013) it is hardly possible to create a good and useful matrix because the products offer different functionalities and concepts. Besides, the feature list also changes virtually every day in the IT world.
Therefore, and based on the literature review, the author suggests the following criteria:

i. **Usability**: How complicated is the installation? How many tools are needed? Is the development environment intuitive?
ii. **Maintainability**: How do you administer the product? Is there a GUI for monitoring services?
iii. **Community**: Are there active public forums or mailing lists? Are numerous articles, tutorials, articles, and videos available? Is the product supported by several companies?
iv. **Enterprise Support**: What support options are offered ("business hours", "24/7" hotline vs. Email vs. on-site support, etc.)? Can the required service level agreements be guaranteed? Is support offered in your preferred language?
v. **Functionality**: Are all the required functionalities offered?
vi. **Flexibility**: Can you customize functionalities of the product to fit my needs?
vii. **Expandability**: Is it possible to expand the product? is the product and its interfaces based on standards?

viii. **Connectors**: Are adapters for all required technologies available? Are there adapters for B2B products such as SAP or Salesforce? How easily can I build your own adapter?

ix. **Cost**: What is the full cost (total cost of ownership) of the product - including maintenance, all required ancillary products, connectors, etc.)?

x. **Licensing**: What licensing or subscription model is used? What happens when requirements change (more computers, more CPUs, switching to virtual machines, etc.)? Are upgrades for free? Are downgrades possible, too? Are the costs "foreseeable" at all, is the price list even understandable?

According to his claim there no winning ESB product and it is advisable to pre-define your own needs, and then to evaluate which products are best suited.

B. While (Gartner, 2013) define key technical characteristics that slightly differ from the previous criteria.

Technical characteristics

i. **Communications**: Vendor offerings must implement an interoperability layer that supports interactions among application and system components via a variety of. Vendor offerings must also enable a broad array of interaction styles — such as request/reply, conversational, publish and subscribe, and asynchronous messaging. Finally, vendor offerings should provide support for the idempotent delivery of messages — that is, the ability to (1) guarantee the delivery of each message, (2) to deliver each message only once and (3) to deliver messages in the order sent by the source program(s).

ii. **Data Transformation**: Vendor offerings should support the translation of data from the format, structure and semantics native to the source application to that required by the target applications.

iii. **Orchestration**: Vendors should provide technology that hosts the execution of process logic spanning interactions with multiple back-end services or applications with the aim of implementing composite services or automated system-to-system processes.

iv. **Application Connectivity**: Vendors should provide an array of adapters or wrappers — that is, a technology that combines design tools and runtime software to implement programs that act as "glue," bridging protocol differences and connecting to databases, as well as most popular packaged applications and SaaS offerings.

v. **Development Environment**: Each vendor must provide a software application that provides comprehensive facilities to enable integration staff to efficiently design, implement, test and deploy integration interfaces and service interfaces.

vi. **B2B Interactions**: Vendors should provide connection provisioning capabilities for B2B protocols. Vendors should also support Web services-based connections with external business partners.

vii. **Governance**: Governance is the assignment of decision rights to ensure desirable behavior. Vendor support is expected for managing the lifecycle of integration solutions during design time and to manage qualities of service at runtime. Expected functionality includes a registry/repository, policy definition and management, and API management.

viii. **Security**: Vendors should enable implementation of effective security support to enable capabilities such as authentication of endpoints, authorization of service or interface access, message/document encryption/decryption…etc.

ix. **Administration and Monitoring**: Vendors should provide technology that enables visibility into and effective management of the solutions that are created through the integration of programs and services.

According to (A., 2010) the support for open standards has to be one of the main characteristics to be considered. Other important characteristics to take into account are the followings: the implementation support, the ease of use and the GUI support. Obviously, there are no standard criteria that can be used in ESB evaluation, so the decision maker needs to do his homework and define what the criteria that suit their needs.

"Democratizing" is introducing a democratic system or democratic principles to make something accessible to everyone. Open source software democratizes the ESB by making it accessible to a much broader group of developers and organizations. By addressing the primary challenges developers face—access to low-cost, powerful development tools that are stable, easy-to-use and fully supported. According to (Thompson, 2010), "With several commercially supported alternatives available, open-source ESBs have moved from a developer-initiated experiment to a viable choice for mainstream organizations." Small and medium businesses (SMBs), as well as departmental users, can now gain the productivity, efficiency and cost advantages of application integration that until now were only exploitable by larger enterprises. Enterprise-class integration is now accessible to a greater number of organizations that can not only participate in the economic benefits of ESBs but can ensure an ongoing voice in how their ESB software evolves to support the needs of the ESB user community over time.

3.2.2 Motivation for Choosing Talend ESB

According to (Asankha C. Perera, 2013) the Talend ESB SE 5.3.1 performed slightly better than the Mule CE 3.4.0 ESB and encountered only 3 HTTP level errors for 11,138,400 requests. While The WSO2 ESB 4.7.0 suffered a severe response corruption defect for payloads larger than 16KB, and a failure of the XSLT test cases.

According to (Gartner, 2013) Talend ESB pursues an open-core, commercial open-source model providing mission-critical features, support and maintenance via subscriptions. The vendor offers BPMS functionality via a partnership with Bonitasoft. And although a relative newcomer to the ESB suite market, Talend is the first vendor to offer a platform that integrates a suite for application integration with data integration and BPM technologies through a common repository/environment.

According to (Gartner, 2013) Talend has many strengths that listed as follows

a. Strengths
   i. Talend Open Studio for ESB is a robust suite founded on the broadly adopted Apache CXF, Camel, Karaf and ActiveMQ open-source offerings, to which its engineers are active contributors.

ii. Talend uses a graphical approach to implementing Apache Camel Enterprise Integration Patterns, which includes an all-in-one feature for testing the implementation of these enterprise integration patterns.
iii. Talend's go-to-market approach of five platform offerings (Big Data, Data Management, Data Services, Enterprise Integration and Master Data Management), with all the platforms integrated via a single repository, is unique.

b. Cautions
i. Talend is a relatively young company, founded in 2005 and dual-headquartered in Los Altos, California, and Suresnes, France. It is methodically expanding into its established markets (that is, the U.S. and EMEA) and is opening up its Asia/Pacific efforts with offices in Tokyo and Beijing. However, its products do not have a worldwide installed base comparable with the leading integration vendors.
ii. B2B support is limited to the most common B2B file formats. However, these can be configured into the product.
iii. Talend looks to Apache projects and its R&D staff to provide adapters, some of which are also contributed by community members. While Talend provides adapters to SAP, Microsoft CRM and salesforce.com, it only offers a limited set of application integration adapters for widely deployed commercial packaged applications, such as PeopleSoft and Siebel.

3.3 The Proposed Workflow System

All of the research that has been done related to choosing workflow management system (WFMS), leading us to one question "What functionality and capabilities should a WFMS provide for it to fit the enterprise requirement?"
They claim that there are no WFMS that can be suitable for all enterprises, instead, the enterprise should define the criteria and evaluate the WFMS available in the market. According to (Boucher, 2012), the criteria that have been chosen are

a. Ability to fully integrate with your company's line-of-business application
b. Ability to create and configure simple or complex processes
c. Management dashboard for performance metrics
d. Allow for multiple users interface deployment options
e. Reasonable TCO
f. OEM agreement support

The WFMSs that has been evaluated are Bizagi, Bonita and Activiti. The following table illustrates the evaluated result

Table 3.1: Workflow Criteria Matrix

Criteria / WFMS	Bizagi	Bonita	Activiti
Ability to fully integrate with your company's line-of-business application	√	√	√
Ability to create and configure simple or complex processes	√	√	√
Management dashboard for performance metrics	√	√	×
Allow for multiple users interface deployment options	√	√	√
Reasonable TCO	×	√	×
OEM agreement support	×	√	×

So Bonita WFMS has been chosen to be the workflow engine to the company and to be integrated with DMS system or any other system that will implement in the company.

3.4 Implementation

In the following section of implementation, Preparing the integration environment has been set up which includes data model format, communication protocol, and message notation. Then the required adaptor in each integrated system has been developed. Finally validating and testing the proposed system and its components; based on the view of transparency and information exchange among the system.

3.4.1 Introduction to Talend ESB solutions

The Enterprise Service Bus (ESB) has always been the cornerstone of every vendor's Service Oriented Architecture (SOA) strategy. (Talend, 2014) Talend ESB is a considerable improvement on previous ESBs in that it:

a. has relatively small footprint
b. uses proven open source technologies
c. enables easy integration of existing applications and infrastructures

This chapter gives a high-level overview of Talend ESB solutions, their components, and features. It also describes the integration process between Bonita workflow management system and OpenKM document management system using Talend ESB.

3.4.2 Talend ESB Features

According to (Gartner, 2013) Talend ESB is a versatile and flexible ESB that allows organizations to address diverse integration challenges. It supports a broad set of standard transports and protocols, as well as enterprise integration patterns (EIPs).

Leveraging Apache CXF, Apache Camel and Apache ActiveMQ open source integration projects, Talend ESB makes enterprise-class integration accessible by delivering a cost-effective and easy-to-use way to integrate and expand systems and applications.

Apache CXF is an open source services framework, Apache CXF helps companies build and develop services using frontend programming APIs like JAX-WS and JAX-RS. (CXF, 2016)

Apache Camel is an open source integration framework that lets developer leverage EIPs to implement routing, transformation and mediation rules. (Camel, 2015)

a. Web Services Support

Talend ESB helps the developer to create new web services or to service-enable existing applications and interfaces for use with the web. Talend ESB leverages the features of Apache CXF for developing and deploying web Services and REST applications.
According to (CXF, 2016) Apache CXF supports all important web services standards and fully complies to the Java API for XML web Services (JAX-WS) specification.
Talend ESB supports the creation of SOAP and REST web services and offers WS-* functionality including support for WS- Addressing, WS-Reliable Messaging, and WS-Security over both HTTP and JMS transports. (Talend, 2014)
In addition, the web services stack in Talend ESB distributions goes well beyond Apache CXF, with support for:

i. OSGi containers
ii. Graphical Data Service Development using the Talend Studio
iii. Advanced Service Governance using Deployment Time Policies
iv. Central deployment and configuration options via Web User Interfaces and JMX-based APIs
v. Management and monitoring of services

b. Standard OSGi Runtime

According to (Talend, 2015) the standard runtime in Talend ESB is an OSGi container. The OSGi implementation shipped with Talend ESB is Apache Karaf using Eclipse Equinox as OSGi Runtime, providing a lightweight container into which various components and applications can be deployed.

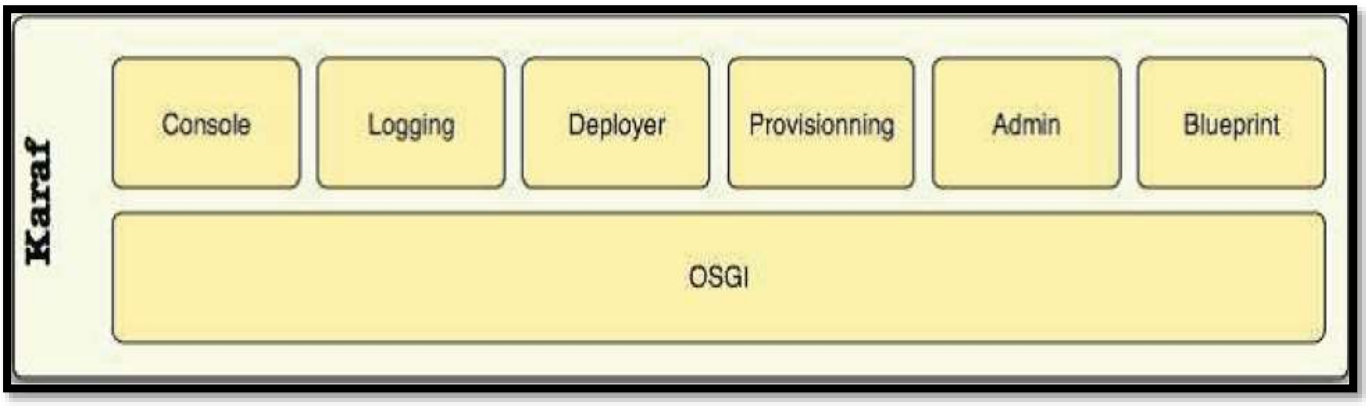

Figure 3.1: Overview of Karaf Components (Talend, 2014)

Karaf supports the following features:

i. Hot deployment: Karaf monitors any file inside the [home]/deploy directory. So if a file is copied to this directory, it is automatically installed inside the runtime; subsequently, this can be updated or deleted, and Karaf will act correspondingly.
ii. Dynamic configuration: Services are usually configured through a standard OSGi service, using property files, which are monitored; changes are propagated to the service.
iii. Logging: using a centralized logging back end supported by Log4J.
iv. Managing instances: Karaf provides simple console commands for managing multiple instances.

c. Messaging

Talend ESB embeds Apache ActiveMQ message broker to support a number of different messaging options. ActiveMQ is written in Java and implements the JMS specification.

The job of the message broker is to transport events between distributed applications, guaranteeing that they reach their intended recipients. The broker, therefore, must be highly available, performant, and scalable for this goal, which Apache ActiveMQ provides an easy to use way in Talend ESB, as it is embedded in Talend Runtime. (Talend, 2014)

d. Talend Studio

The Talend studio provides a graphical development tool with:

i. an **Integration** perspective
ii. a **Mediation** perspective
iii. a **Java** perspective (Enterprise and Platform studios including m2eclipse Plugin)
iv. a **soapUI** perspective (Enterprise and Platform studios only)

These are discussed in more detail in the rest of this section.

i. Integration Perspective

The Integration perspective is a graphical tool within the Talend Studio which allows the developer to use the extensive list of data adapters and components to build ESB data services and export them for standalone or for deployment in a local Talend Runtime container.

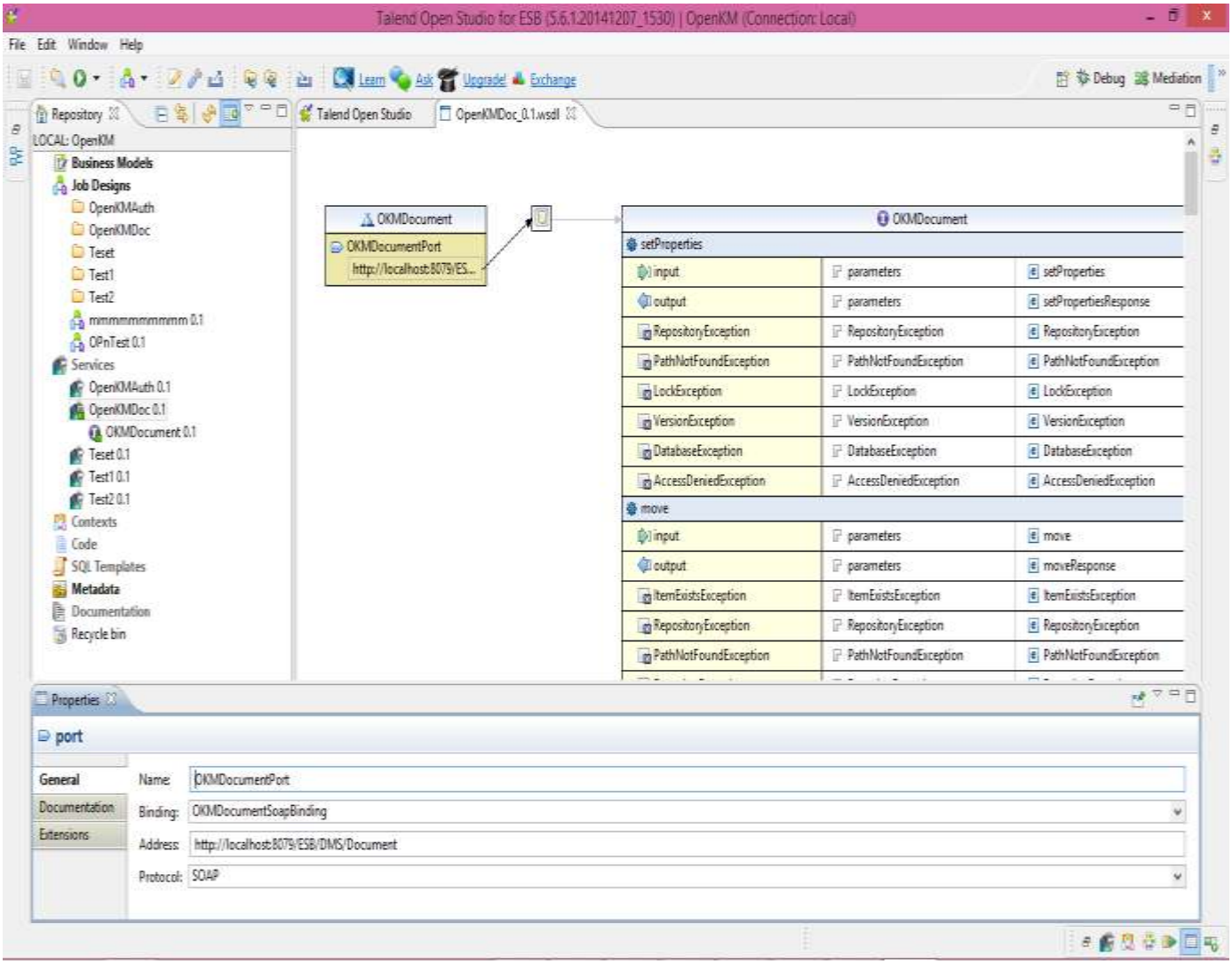

Figure 3.2: The Integration Perspective with a Service Design

A Service in the 'Services' node is a Web-Service defined by a WSDL. The WSDL can be just imported, created from scratch in the tooling using the embedded graphical WSDL editor or an existing WSDL can be imported and then edited within the studio. In this case, the service is based on this WSDL information and each service operation can then be implemented in the job design node.

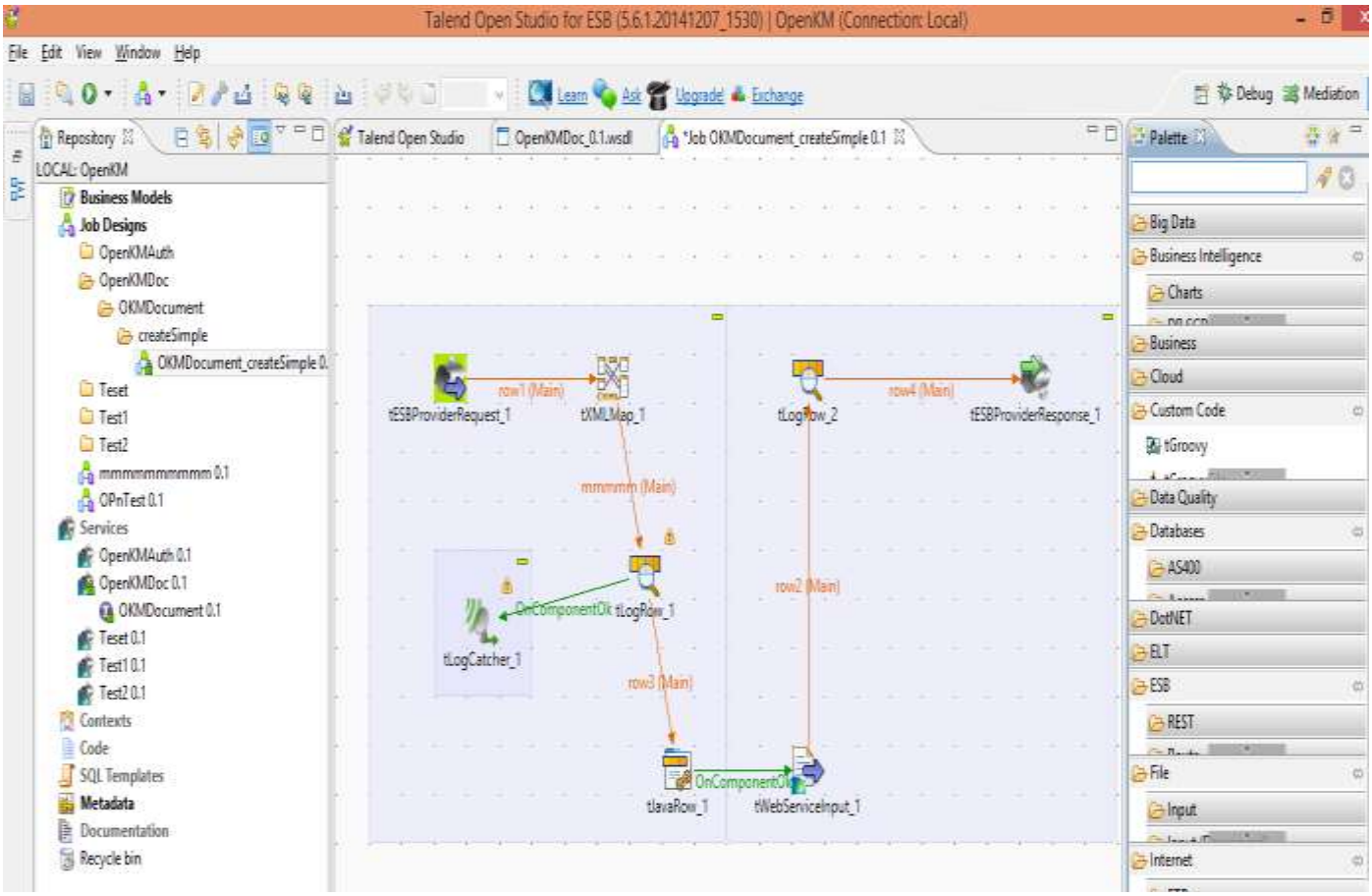

Figure 3.3: The Integration Perspective with a Job Design

A data service Job is a graphical design, of one or more components connected together, that allows the developer to set up and run data service operations. Jobs address all of the different sources and targets that you need for data integration processes and combine it with Web services. Additionally, in enterprise and platform studios, the developer can use the shared repository feature to work in larger teams, and share resources. It has the facility of team collaboration - team members can store and share their business models, integration and service jobs, integration services and metadata in an industry-standard source manager (SVN).

ii. Mediation Perspective

This section first deals with Apache Camel, and then the Mediation perspective which is a graphical interface to this functionality.

i. Apache Camel

The Mediation functionality of Talend ESB is based on the Apache Camel project. The core of the Camel framework is a routing engine. It allows the developer to define routing rules that accept and send messages through components. There are no restrictions on the message format - for example,

Java objects, XML, JSON, plain text and so on, can be used. These routing rules are based on the book (Woolf, 2012); et al. Thus, Apache Camel is a framework allowing developers to assemble Endpoints / Processors into workflows (Routes) to achieve higher level functionality (Wikipedia, 2014). It facilitates application integration by leveraging Enterprise Integration Patterns (EIPs) to essentially assemble scalable services, and make message-based system integration simpler to design and implement.

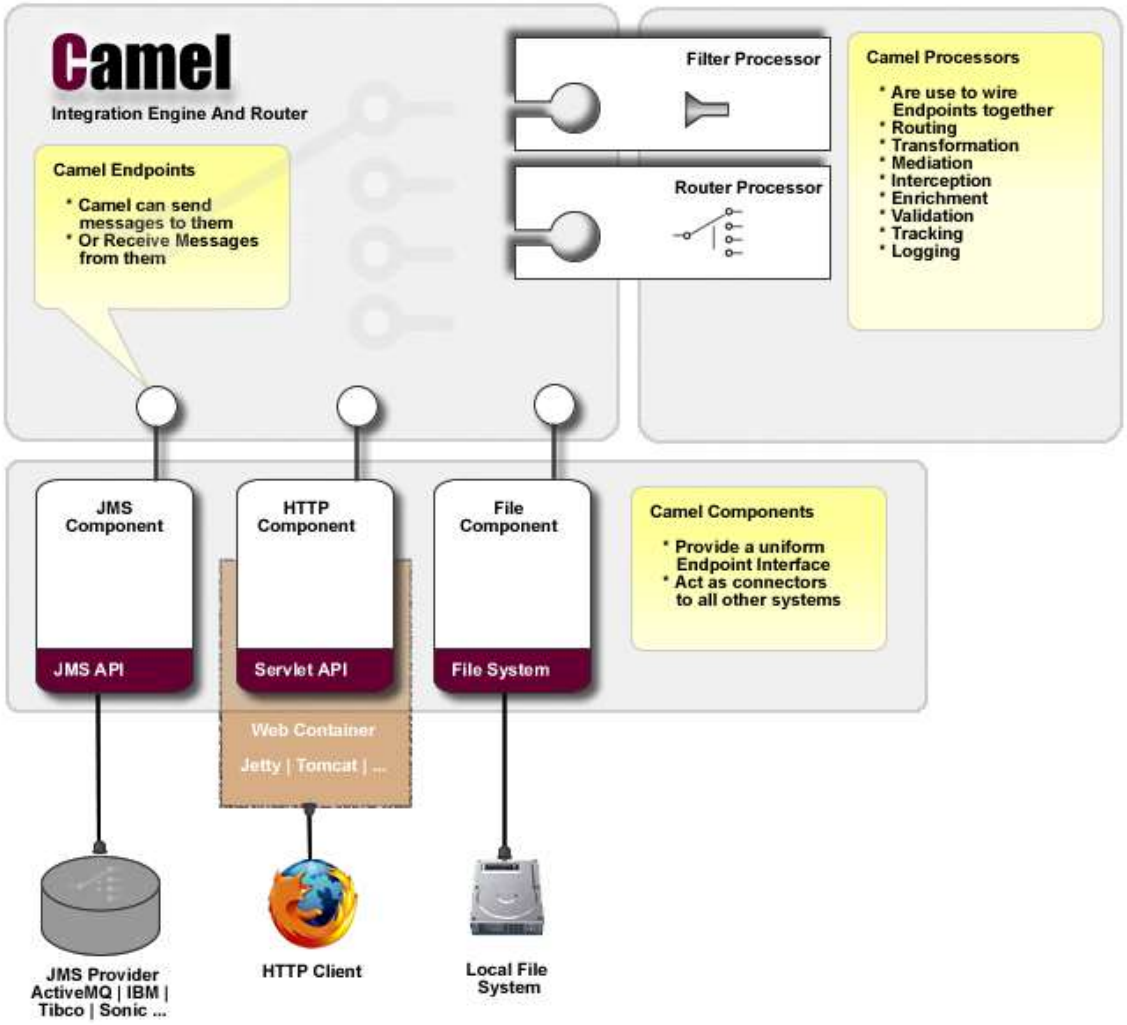

Figure 3.4: Apache Camel Architecture (Camel, 2013)

ii. Mediation perspective

On top of Apache Camel, and integrated with the Talend Studio, the Route Builder (**Mediation** perspective) is a GUI that allows a developer to build these Routes in a visual way.

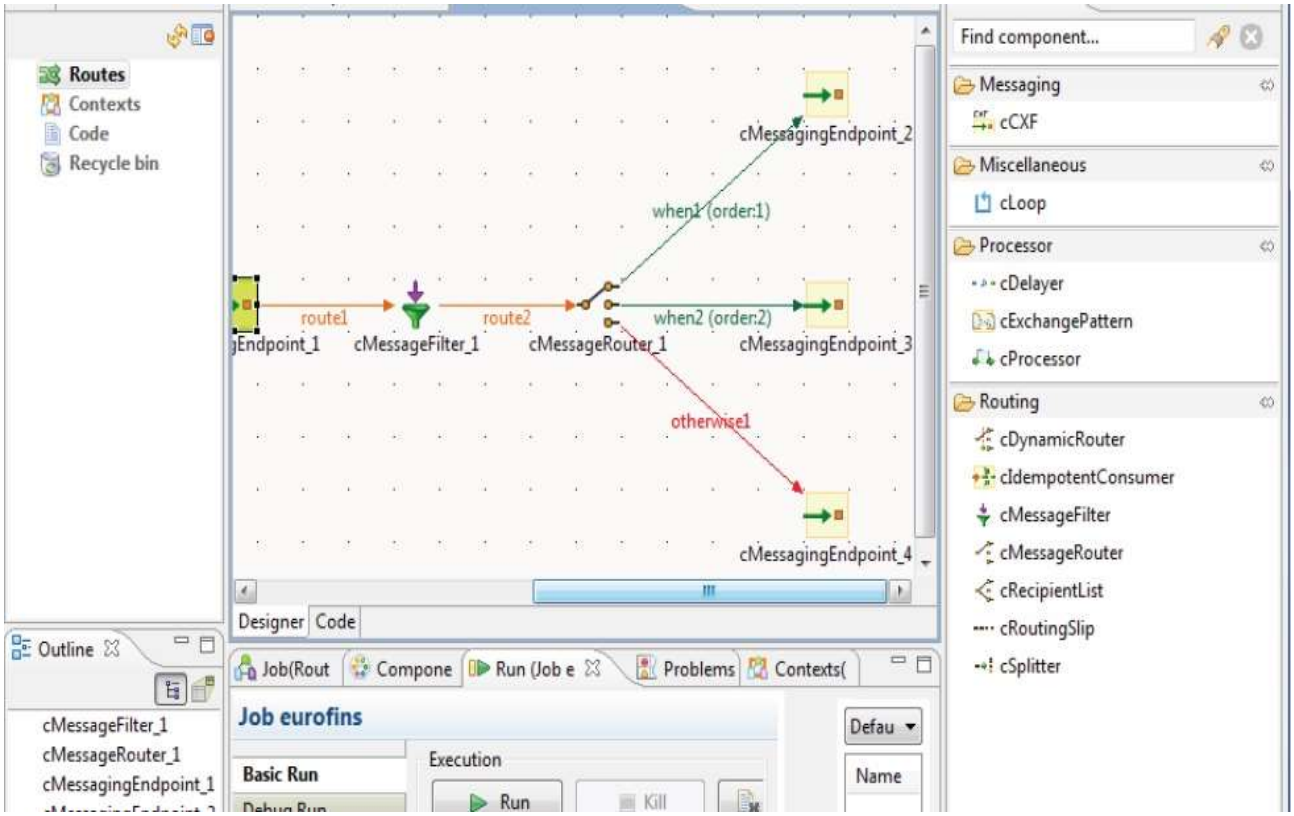

Figure 3.5: The Mediation Perspective

3.4.3. Talend ESB Products and Architecture

Talend provides ESB functionality in four different packages described in the following sections (Talend, 2015)

a. Talend ESB Standard Edition (SE)

Talend ESB standard edition is a standards-based connectivity layer used to integrate distributed systems across functional, enterprise, and geographic boundaries. Capabilities include messaging, web services, intelligent routing, and data transformation. It is available under the open source apache license.

b. Talend Open Studio for ESB

Talend open studio for ESB – an eclipse-based tooling environment for modeling, configuring, deploying and managing data services – includes Talend ESB standard edition.

c. Talend Enterprise ESB

Enterprise ESB is designed for application teams that need to manage development projects across teams and operate their integrated production environments across

their enterprise in a coherent manner. As such, Talend ESB includes all of the functionality of Talend open studio for ESB and extends it with team collaboration, enterprise management and other capabilities.

d. Talend Platforms

Talend platforms extend Talend enterprise ESB with advanced clustering, business process management, application integration, extended data mapping and data management features allowing firms to increase business productivity, deliver projects faster, and lower operating costs.

3.4.4 Introduction to Bonita Business Process Management

Bonita BPM is an open-source business process management and workflow suite created in 2001. It was started in France national institute for research in computer science and then had incubated several years inside of the French computer science company Groupe Bull. Since 2009, the development of Bonita is supported by a company dedicated to this activity: **Bonitasoft**. (Wikipedia, 2013)

Bonita BPM has three major components (BonitaSoft, 2014):

a. **Bonita Studio:** allows the user to graphically modify business processes following the BPMN standard.
b. **Bonita BPM Engine:** The BPM engine is a JAVA API that allows the developer to interact programmatically with his processes. It is available under LGPL. It relies on Hibernate.
c. **Bonita Portal:** is a portal that allows each end-user to manage in a webmail-like interface all the tasks in which he or she is involved.

Bonita BPM is open source and can be downloaded under GPL.

3.4.5 Introduction to OpenKM Document Management System

OpenKM is a free libre document management system that provides a web interface for managing arbitrary files. OpenKM is developed using Java technology based on Java EE standards and the tomcat server. (wikipedia, 2015)

3.4.6 Case Study

a. Analysis

i. System Problem

In most of the institutions that want to use the document management system, they need to address the business process, in spite of all the features offered by OpenKM system in the following document management, it still suffers from a defect and

failure in business process management. Some institutions have the BPM and some do not have, in all cases, the Foundation would like to work integration between document management system, BPM and the rest of the Enterprise Systems.

So-hoc basis appeared to use the ESB to achieve the goals of integration between the old and new systems, taking into account the approved criteria such as expansion, maintenance, performance, and security.

ii. Business Model

The research subject company owns several information systems, constituting document management system, enterprise resource management system, e-mail system and project management system, and was able to link some of the information systems. But the rate of change in business requirements led to that there must be a workflow system in the company, the problem is that the current enterprise architecture that was used in integrated the systems was peer-to-peer, and the possibility of accepting the expansion, change and maintenance in this architecture is low and costing the company a lot of money.

The needs of the institution are that there must be a mechanism for integration with an ability that all the services separate from each other, low systems dependency. With the possibility of change some service without affect by the rest of the integrated systems. The possibility of expansion and the addition of new services at the lowest possible costs.

So the institution's desire to not only to integrate systems but accompany future changes likes the increasing in information use and direction of most of the institutions to the cloud computing and internet services, which requires the existence of the possibility of future integration with external services also.

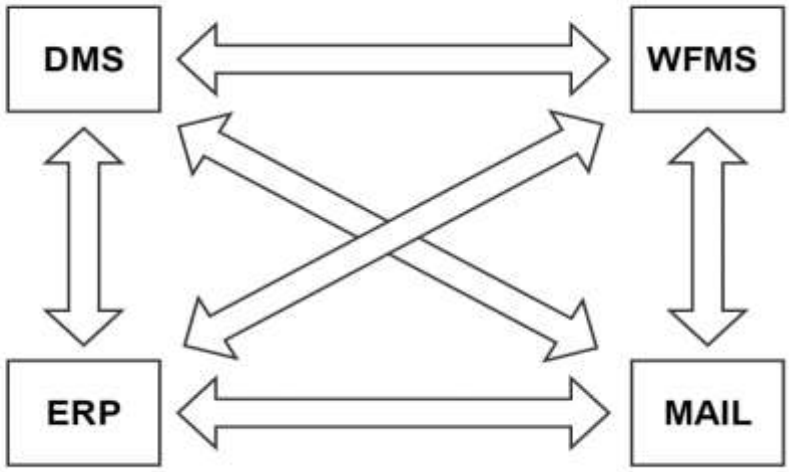

Figure 3.6: The Current Business Model

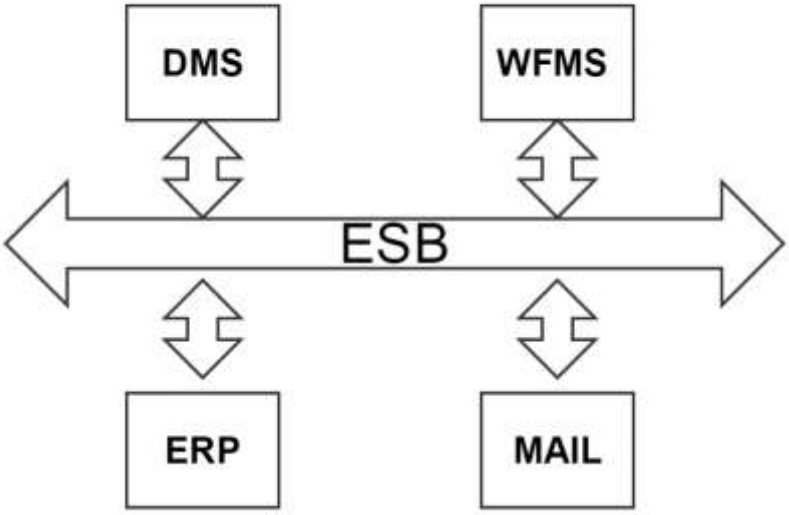

Figure 3.7: The New Business Model

iii. Consumer Activity Diagram

The action sequence will begin when an employee uses the system to request new service. The activities will be as follows:

a. Workflow system will connect to the enterprise data bus and ask for permission to use the document management service.
b. Workflow management system will send attachments to the Enterprise service bus.
c. The manager will ask the workflow system to retrieve the document from the Enterprise service bus.
d. The manager will update the document and sent it back to the Enterprise service bus.

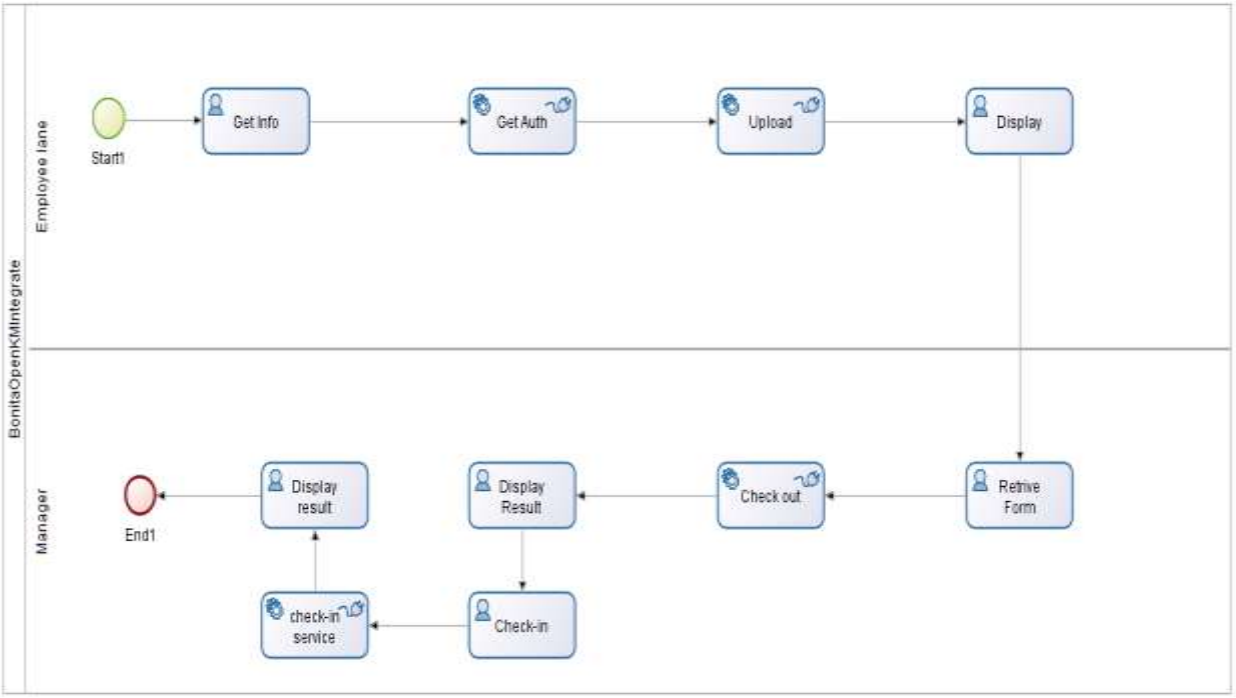

Figure 3.8: The Consumer Activity

3.4.7 System Design and Implementation

There are a number of parts been involved in developing this service, which has been implemented by reused existing functionalities and components, and it include created a DMS service provider, created a WFMS consumer and export the service to a Talend runtime container. Lastly service activity monitor has been used to be sure of service flow.

a. Talend Enterprise Service Bus

Powered by the leading Apache open source integration projects, Talend ESB is a standards-based connectivity layer used to integrate distributed systems across functional, enterprise, and geographic boundaries. Capabilities include messaging, web services, intelligent routing, and data transformation. Its modular architecture allows it to be easily expanded to suit most enterprise requirements.

i. Service Design

The Integration perspective of Talend studio combines data integration with Web services and enables the graphical design of a Service which includes a WSDL file and one or more data service Jobs that addresses all of the different sources and targets required to publish the Web service. The WSDL editor makes it possible to create and edit WSDL files graphically, automating most of the tasks involved with these processes.

A Service has been designed in the Integration perspective of Talend studio, using following steps

a. Import existing WSDL files from OpenKM DMS for structured viewing.
b. Associate Services with data service Jobs.
c. Create and add items to the repository for reuse and sharing purposes (in other projects or Services or with other users).

The Integration perspective of Talend studio enables the developer to create a Service from an existing WSDL file or to create a new WSDL file from scratch using the WSDL editor. When created a service for the first time the following dialog box displays to help developer define the main properties of the new Service.

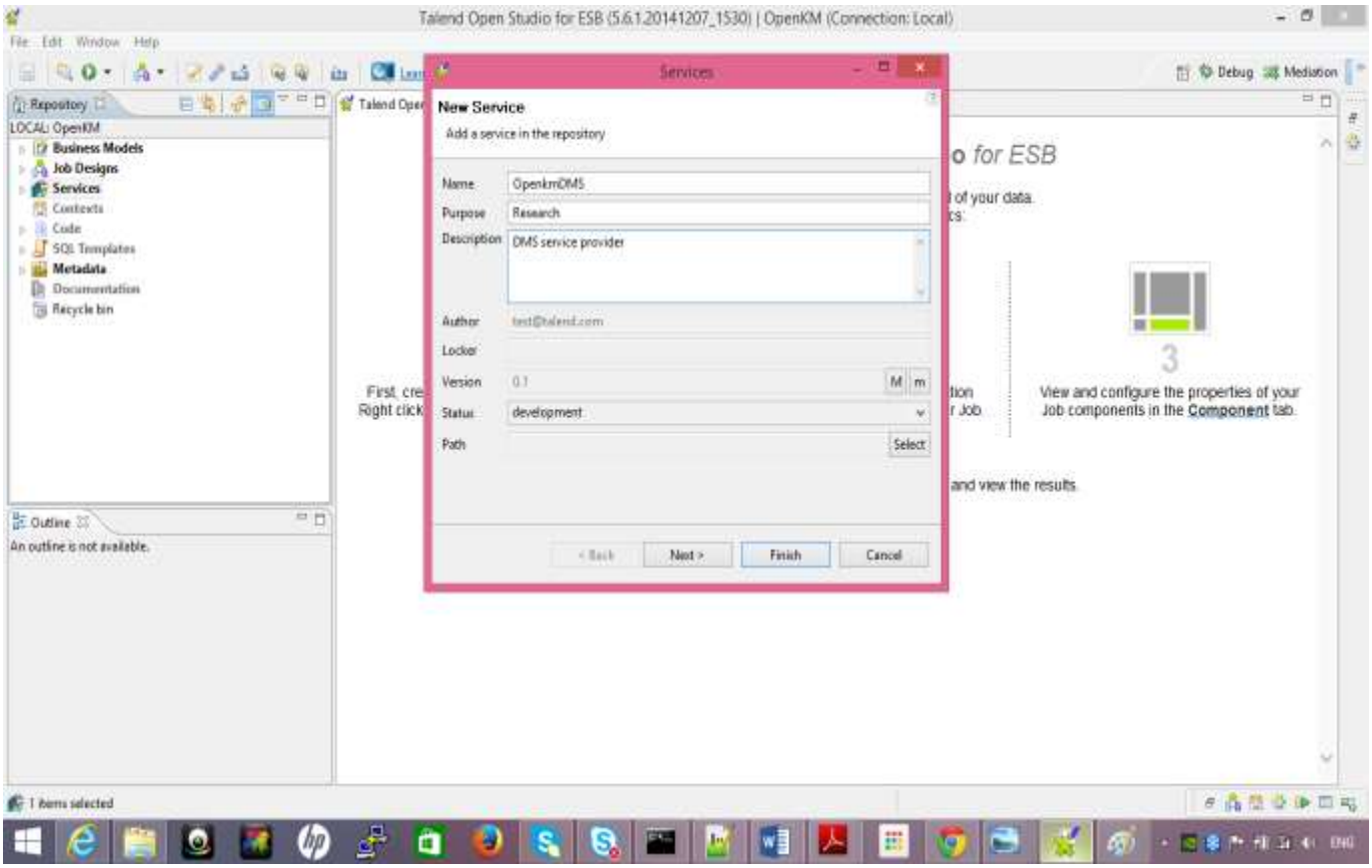

Figure 3.9: Service Creation Process

Table 3.2: Service Properties

Field	Description
Name	The name of the new Service.
Purpose	Service purpose or any useful information regarding the Service use.
Description	Service description.

Author	A read-only field that shows by default the current user login.
Locker	A read-only field that shows by default the login of the user who owns the lock on the current Service.
Version	Read-only field
Status	List to select from the status of the Services you are creating.
Path	List to select from the folder in which the Service will be created.

In the next step [**Assign WSDL**] has been selected, and then [**Import existing WSDL**] has been selected, when finished a screen has appeared as shown in figure 3.10.

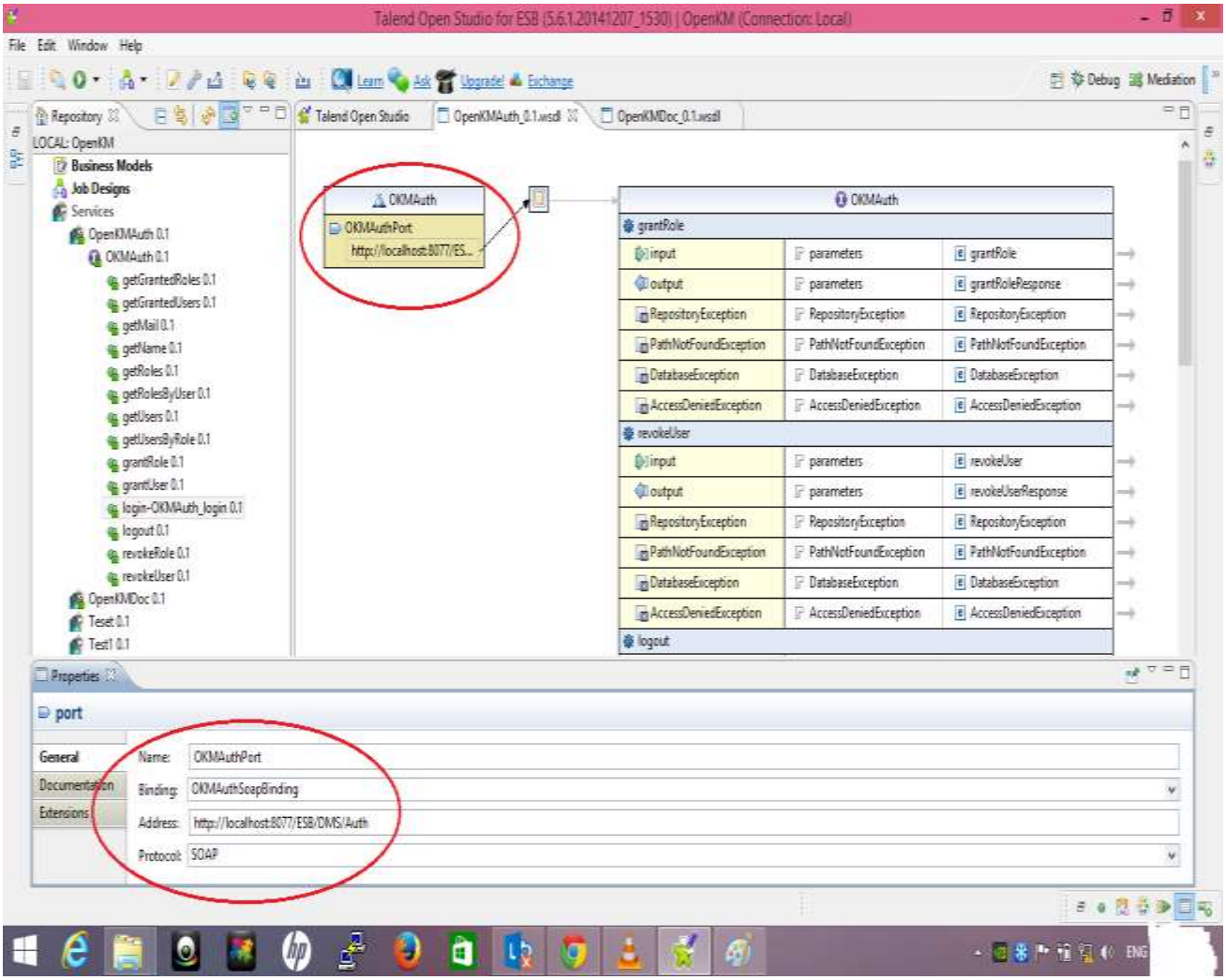

Figure 3.10: Service Schema

The figure 3.10 gives a basic WSDL skeleton which contains:

a. Service used to aggregate a set of related ports which specify addresses for bindings, thus defining a single communication endpoint. In this research two services have been created OkmAuthentication and OkmDocument, OkmAuthentication service provide to the consumer the right authentication and make sure that every request from any consumer is not a threat for the service provider. While OkmDocument service provide to consumer service like upload, edit, retrieve documents.
b. A binding specifies the concrete protocol and data format specifications for the operations and messages defined by a particular port type. ESB has many core principles and the most important of the them are:
    - ESB interconnects requester and provider which mean Interactions are decoupled and separation of concerns.
    - ESB provides Service virtualization of Location and identity, Interaction pattern and protocol, and Interface Binding operation uses to be part of fulfilling the previous principle see figure 3.11, the service has been published in the address [http://localhost:8077/ESB/DMS/] so every consumer will request service from that address. While the DMS is actually in a cloud environment and the ESB runtime installed on a local machine, although that the WFMS consumer will communicate with the ESB on a local machine. All ESB services have been published as web services using SOAP protocol.
c. Port type, a set of abstract operations that each refers to an input message and output messages.

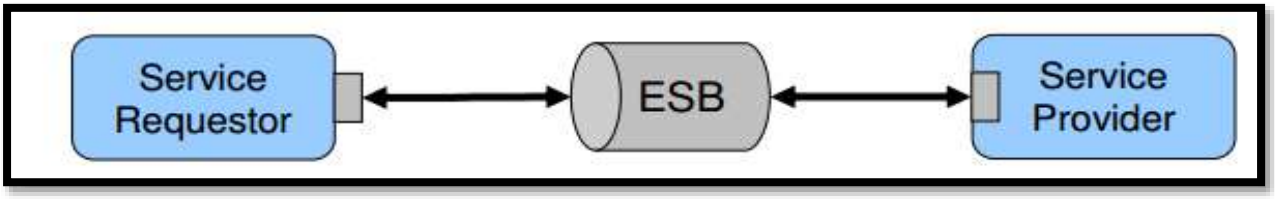

Figure 3.11: Service Virtualization

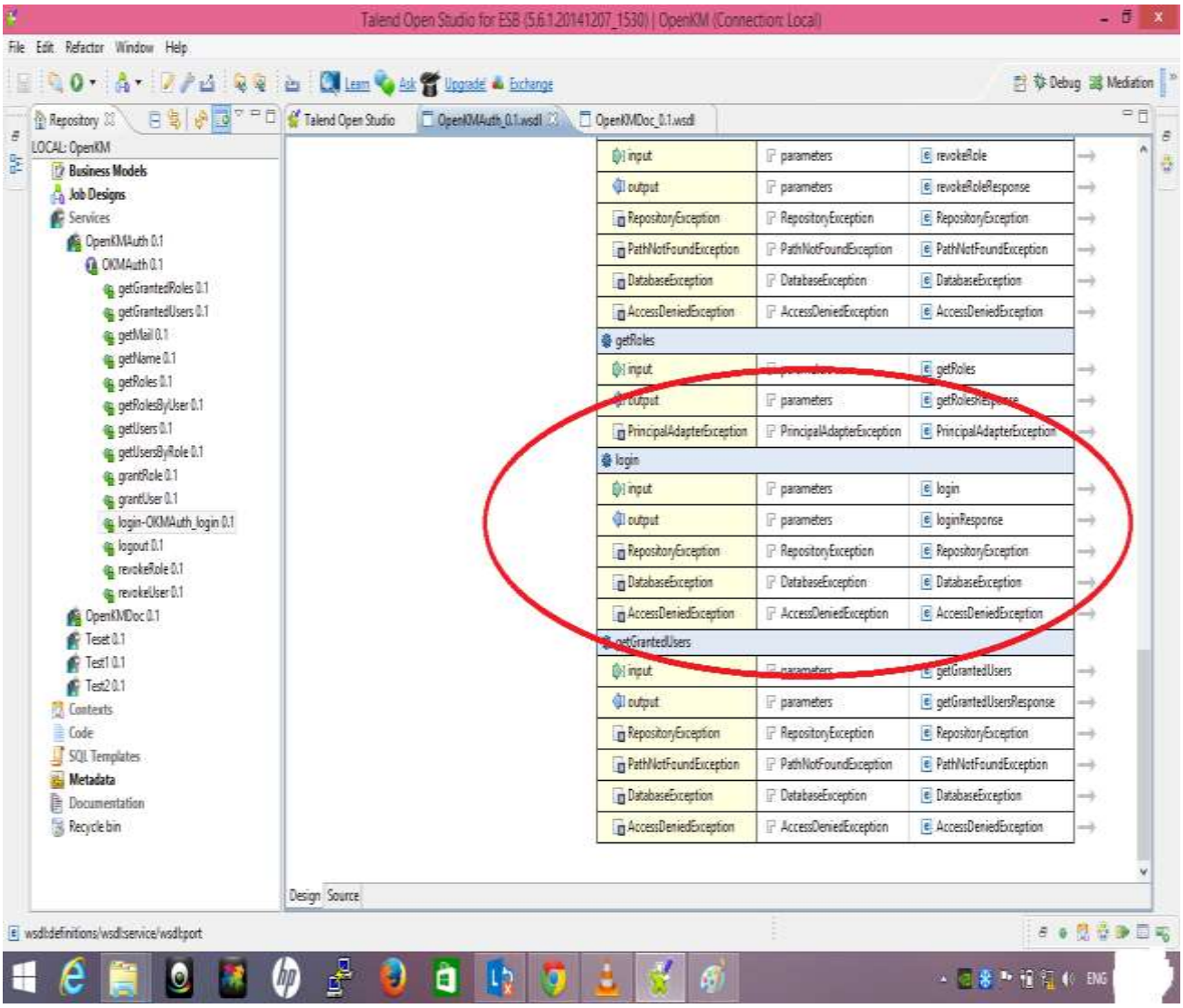

Figure 3.12: Service Port type

The properties view that located on the lower part of the designing editor of Talend studio, displays a list of attributes and editable attribute values of a selected WSDL object and contains the following tabs to edit:

a. The general tab displays a list of object attributes.
b. Documentation tab, specifies the information developer want the user to read.
c. Extensions tab used to add extension components.

After the WSDL file has been created, some operations in the WSDL file has been chosen to associate with a data service provider Job to implement the Web service.

b. Job Design

A job design or data service Jobs is the runnable layer of a business model. It is a graphical

design, of one or more components connected together, that allows the developer to set up and run data flow management processes. A job design translates business needs into code, routines, and programs, in other words, it technically implements company/organization data flow.

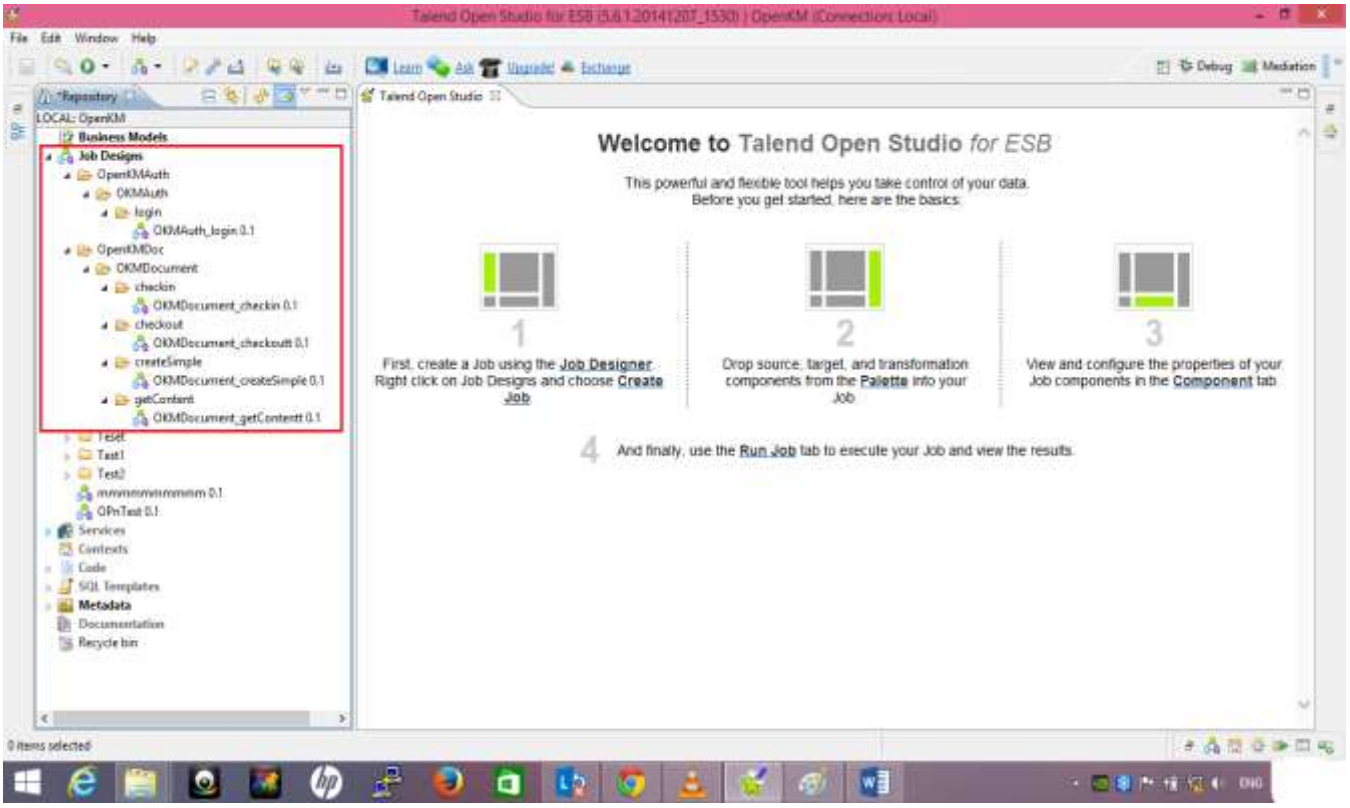

Figure 3.13: Job Design Repository

i. Authentication

Authentication job is associated with login operation in OkmAuth service, any login process between the DMS provider and any consumer must be through this job, the job accepts two parameters username and password, and return security token that the consumer must use it in any request later.

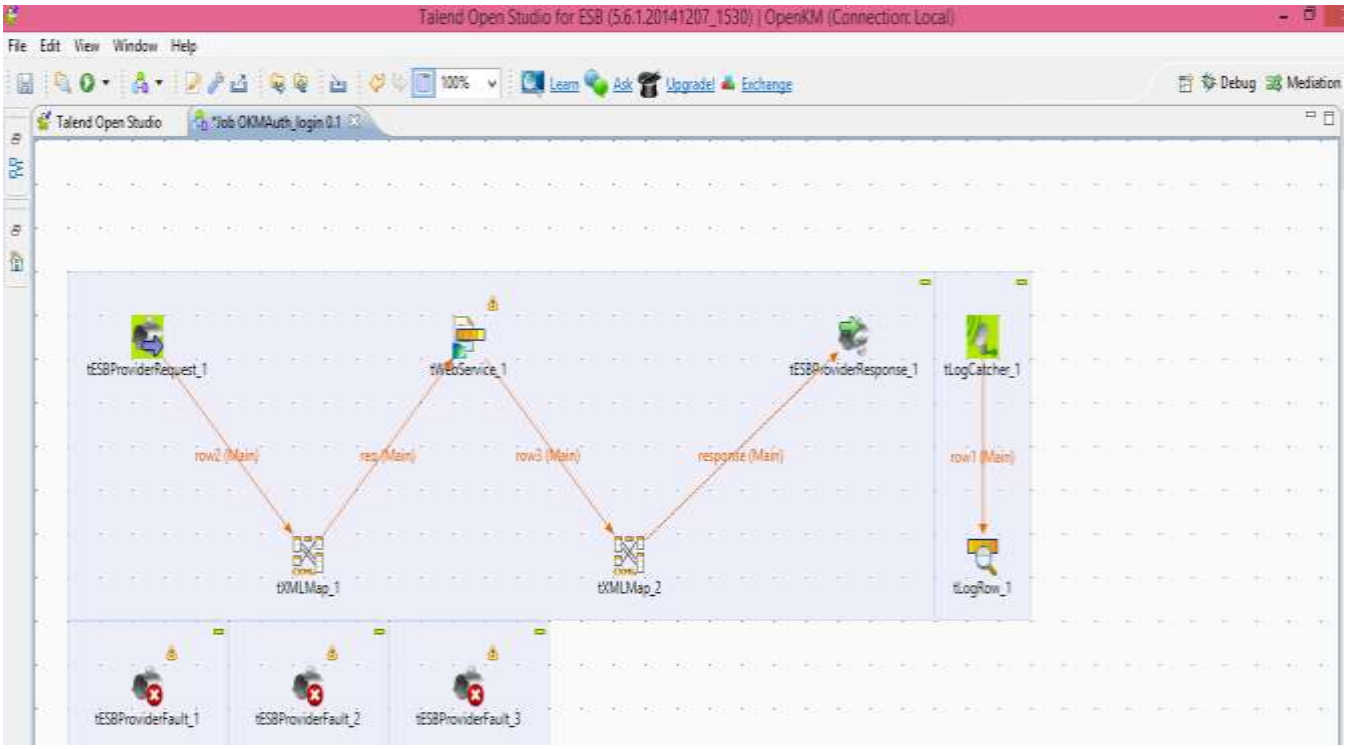

Figure 3.14: OkmAuth_login Job

Authentication job is about receiving the username and password from consumer and then pass to integrated document system in ESB, after that it must return back to the consumer a security token or fault message. The authentication job contains the following components and has been listed in Table 3.2, Table 3.3, Table 3.4, Table 3.5, Table 3.6, Table 3.7 and Table 3.8.

Table 3.3: tESBProviderRequest Properties

Component family	ESB/Web Services	
Function	Wraps Talend job as web service.	
Purpose	Waits for a request message from a consumer and passes it to the next component.	
Usage	This component covers the possibility that a job can be wrapped as a service, with the ability to input a request to a service into a Job and return the Job result as a service response. The tESBProviderResponse component can both deliver the payload of a SOAP message and also access the HTTP and SOAP headers of a service. The tESBProviderRequest component should be used with the tESBProviderResponse component to provide a Job result as a	

	response, in case of a request-response communication style.

Table 3.4: tXMLMap Properties

Component family	Processing/XML	
Function	tXMLMap is an advanced component fine-tuned for transforming and routing XML data flow (data of the Document type), especially when processing numerous XML data sources, with or without flat data to be joined.	
Purpose	tXMLMap transforms and routes data from single or multiple sources to single or multiple destinations.	
Usage	Possible uses are from a simple reorganization of fields to the most complex jobs of data multiplexing or de-multiplexing transformation, concatenation, inversion, filtering and so on. It is used as an intermediate component.	

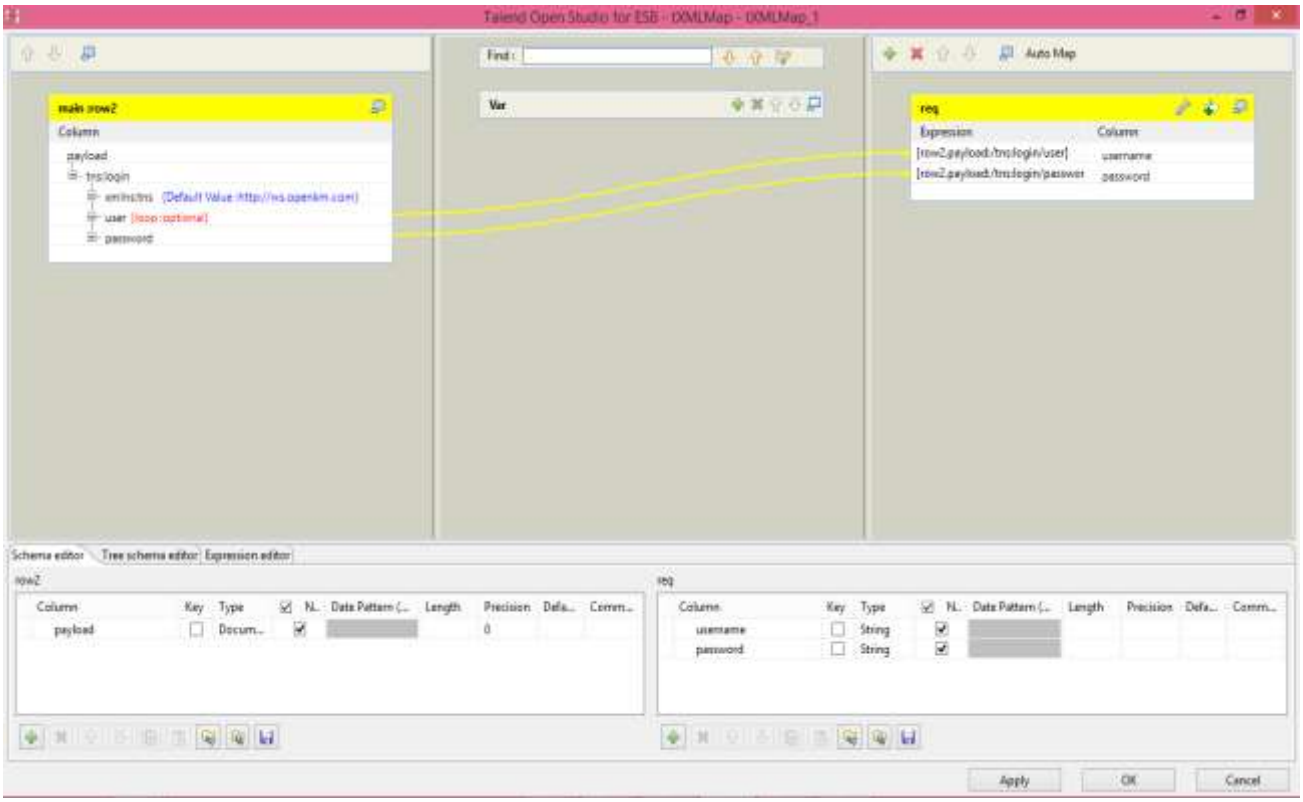

Figure 3.15: OkmAuth_login_tXMLMap

tXMLMap component has been used to convert the data that comes from tESBRequestProvider, in the login job it has been used to extract the username and password from XML file request that sends by the consumer and then converts to text base variable that can be passed to the next tWebService component.

Table 3.5: tWebService Properties

Component family	Internet	
Function	tWebservice calls the defined method from the invoked Web service and returns the class as defined, based on the given parameters.	
Purpose	This component calls a method via a Web service in order to retrieve the values of the parameters defined in the component editor.	
Usage	This component can be used as an input or as an intermediate component. It must be linked to an output component.	

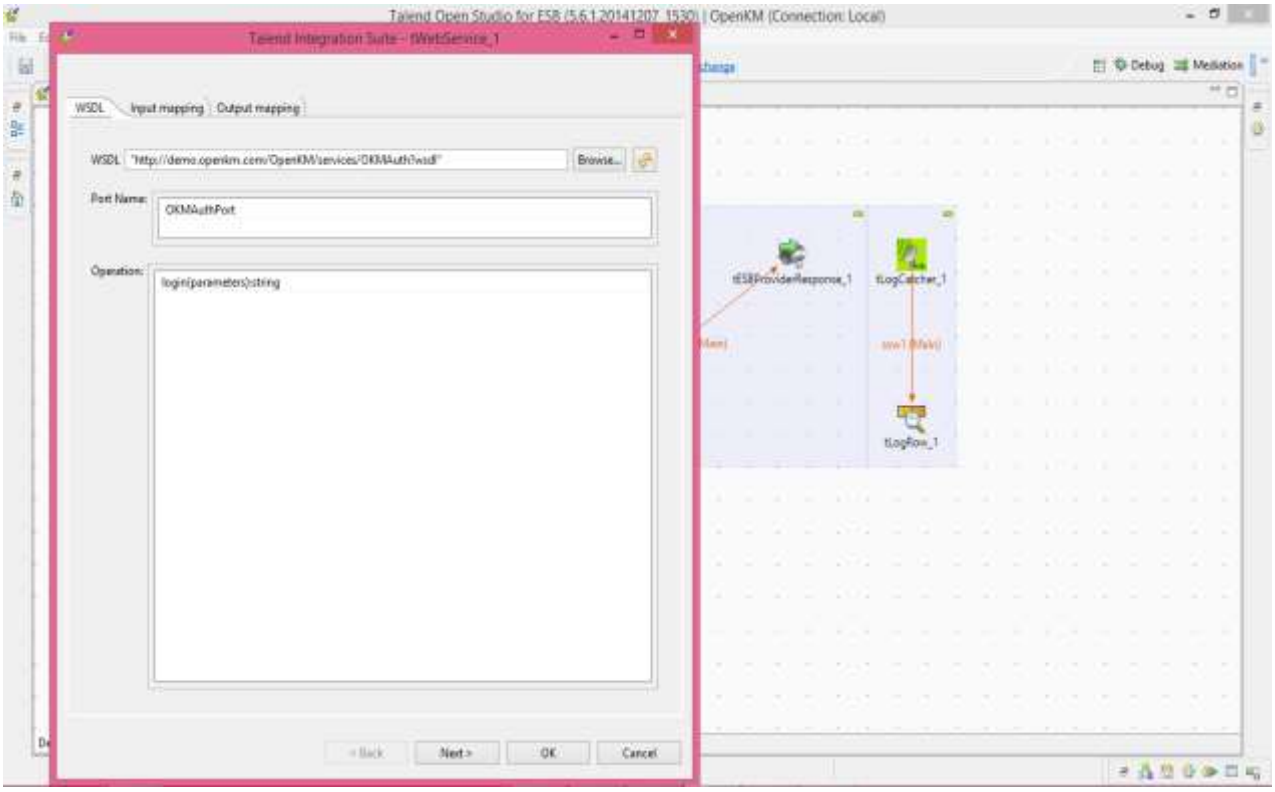

Figure 3.16: tWebService WSDL Configuration

In figure 3.16 the WSDL field web service address has been entered

[http://demo.openkm.com/OpenKM/services/OKMAuth?wsdl], after that from port name list a portType has been selected. Finally from operation list the login (parameters): string service has been chosen.

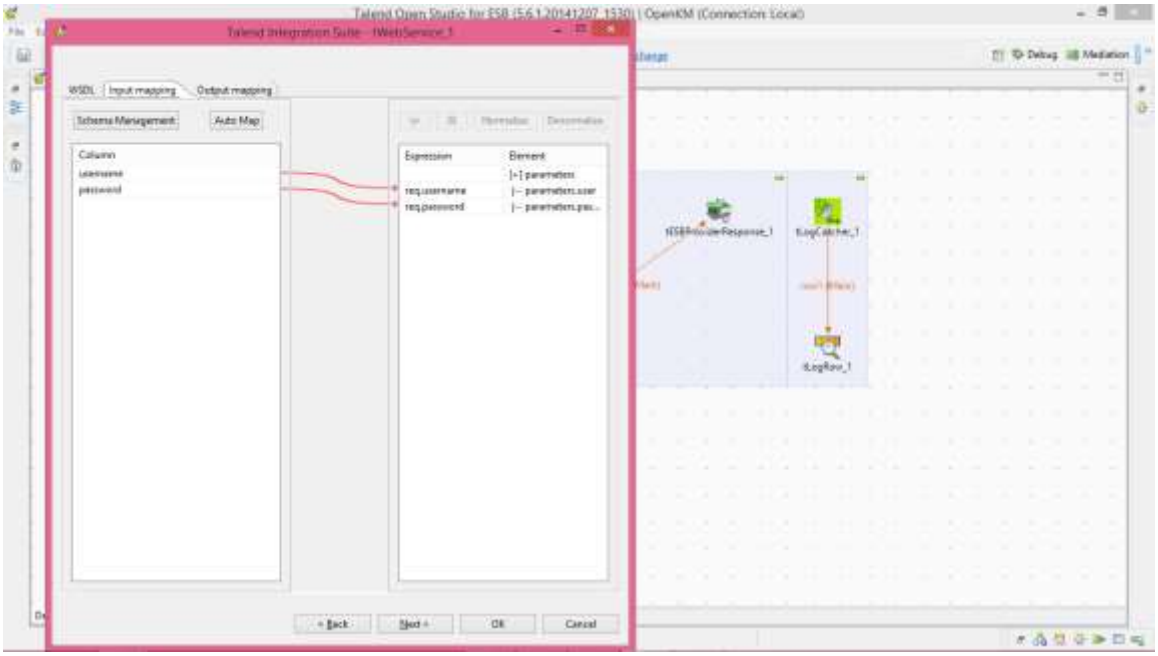

Figure 3.17: tWebService Input Mapping

In figure 3.17 a connection between the input schema – **username and password** - and the input parameter - **parm: username and parm: password** - of the defined Web service have been created

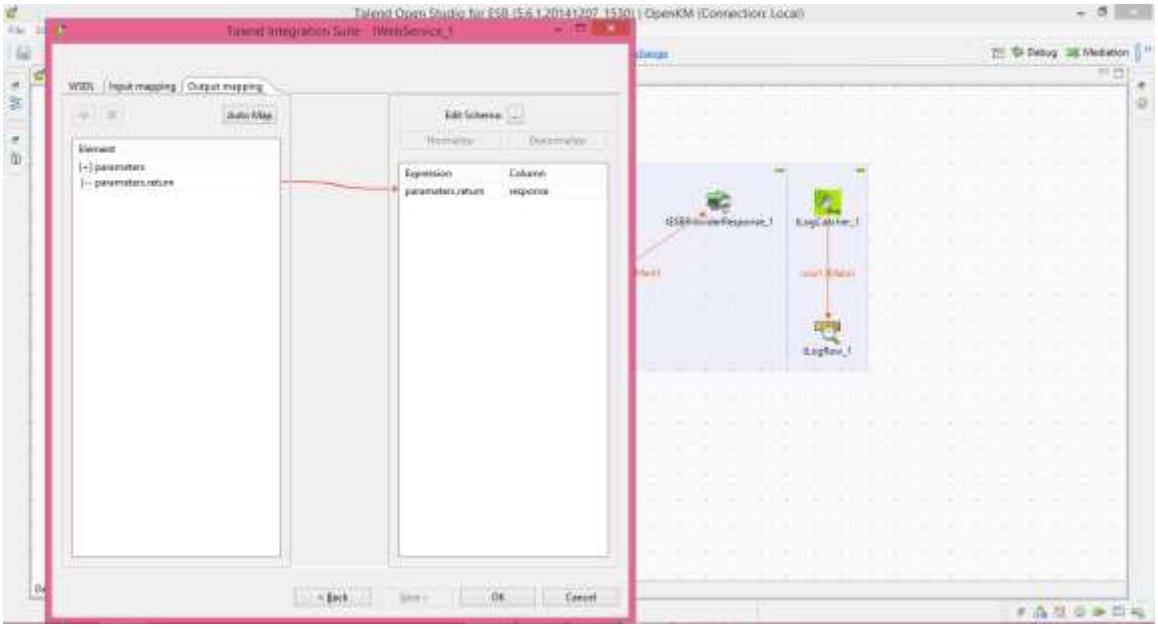

Figure 3.18: tWebService Output Mapping

In figure 3.18 a connection between the service call result - **return** - and the output schema - **parm: response** - of the defined Web service has been created.

Table 3.6: tESBProviderResponse Properties

Component family	ESB/Web Services	
Function	Serves a Talend Job cycle result as a response message.	
Purpose	Acts as a service provider response builder at the end of each Talend Job cycle.	
Usage	The tESBProviderResponse component only is used with the tESBProviderRequest component to provide a Job result as a response for a web service provider, in the case of a request-response communication style.	

While tESBProviderRequest acting like the listener to consumer request, the tESBProviderResponse handle the response to that request, using tXMLMap and built-in schema, to send the response in a suitable XML data.

Table 3.7: tLogCatcher Properties

Component family	Logs & Errors	
Function	Fetches set fields and messages from Java Exception, tDie and/or tWarn and passes them on to the next component.	
Purpose	Operates as a log function triggered by one of the three: Java exception, tDie or tWarn, to collect and transfer log data.	
Usage	This component is the start component of a secondary Job which automatically triggers at the end of the main Job	

tLogCatcher and tLogRow are very useful in the development and debug mode because Talend components are not that easy to implemented. While in the Talend wiki and development reference the examples are made to look simple, unfortunately, it is not. So it was important to use those components to solve the

problems that faced the research implementation in Talend open studio.

Table 3.8: tLogRow Properties

Component family	Logs & Errors	
Function	Displays data or results in the Run console.	
Purpose	tLogRow is used to monitor data processed.	
Usage	This component can be used as an intermediate step in a data flow or as an end object in the Job flowchart.	

Table 3.9: tESBProviderFault Properties

Component family	ESB/Web Services	
Function	Serves a Talend job cycle result as a fault message of the web service in case of a request-response communication style.	
Purpose	Acts as Fault message of the Web Service response at the end of a Talend Job cycle.	
Usage	This component only is used with the tESBProviderRequest component.	

While tlogCatcher and tLogRow catch the exceptions inside the job, the tESBProviderFault main job is to throw the fault message to other consumers.

ii. Upload Document

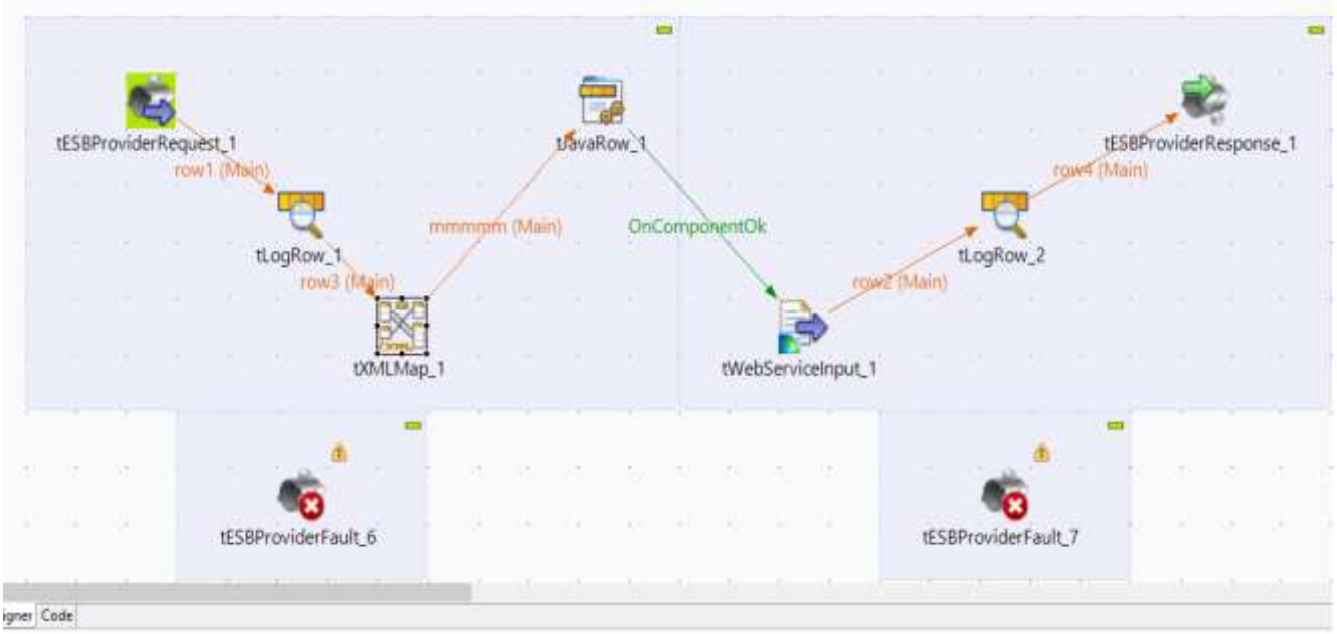

Figure 3.19: Upload Document Job

Upload document job is about receiving the content from consumer and then pass to integrated document system in ESB, it has the most components that have been mentioned early in previous authentication job, in addition to the tJavaRow component.

Table 3.10: tJavaRow Properties

Component Family	Custom Code	
Function	tJavaRow allows the developer to enter customized code which can integrate into a Talend program. With tJavaRow, the developer can enter the Java code to be applied to each row of the flow.	
Purpose	tJavaRow allows the developer to broaden the functionality of Talend Jobs, using the Java language.	
Usage	This component is used as an intermediary between two other components. It must be linked to both an input and an output component.	

Although the Talend ESB said that, there no need to write a code. Unfortunately, it is not true. Will talk about it in next chapter.

iii. Edit document, Download document and Update document, all of them have the same job design, the difference are:

a. Everyone has unique service that associated with it, which mean difference in schema and data flow.
b. Everyone has unique java code inside the tJavaRow component.
c. Everyone has a slight difference in tXmlMap component configuration.

c. Publishing Wrapped Service

Talend studio provides the ability to deploy a job as a web service in order for the job to be called by other applications via the internet. With Talend studio, any job can be exported and exposed as a web service. But before export job the Talend runtime has been started and has been checked using command terminal.

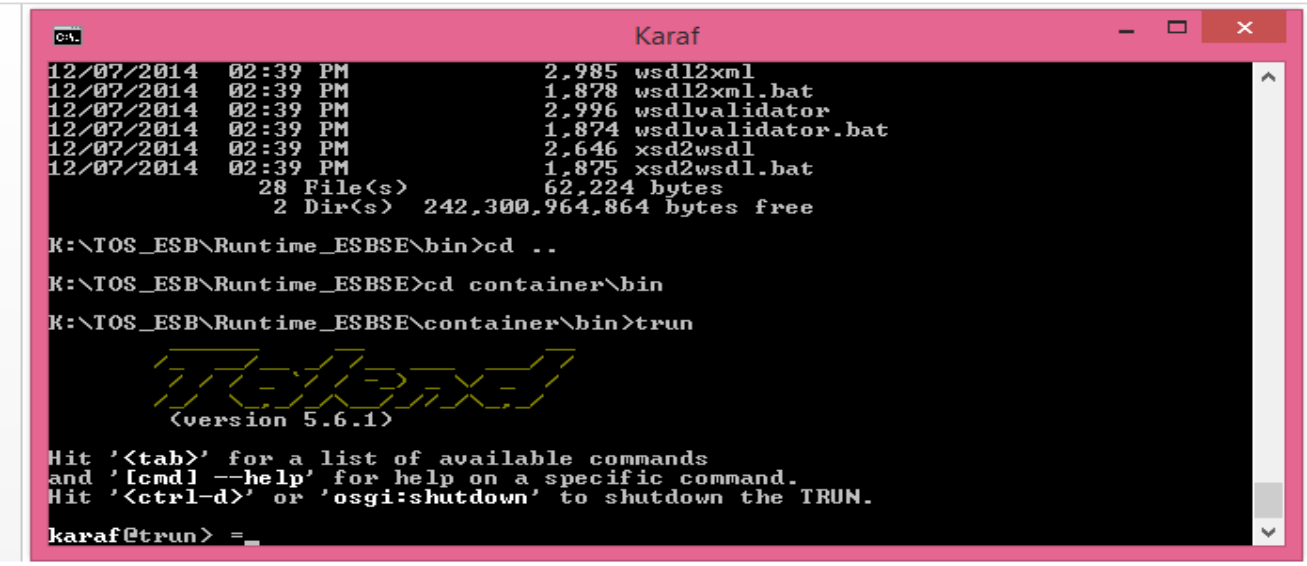

Figure 3.20: Talend ESB Runtime

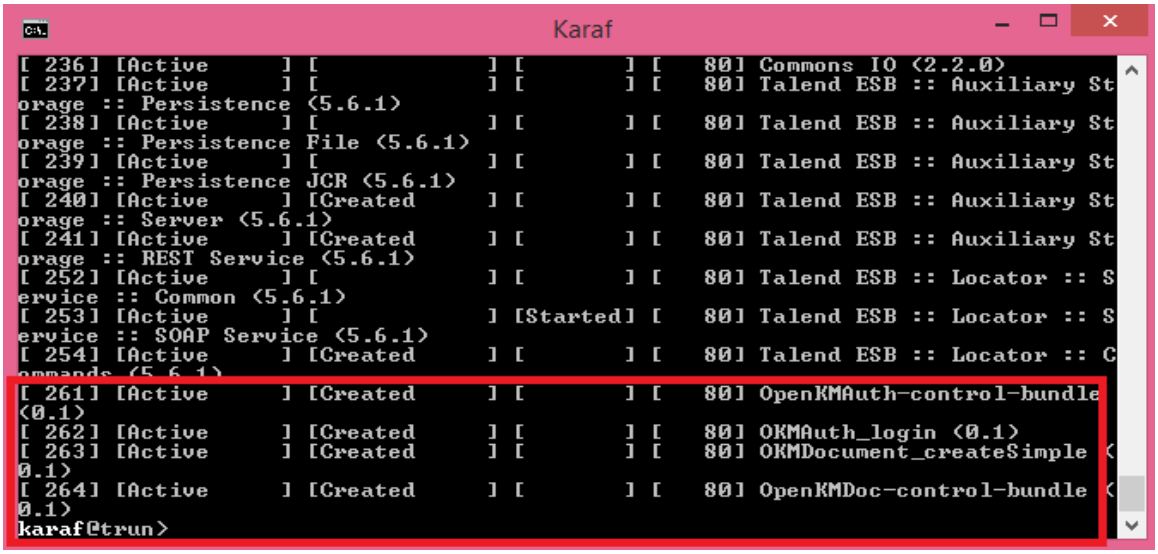

Figure 3.21: Apache Karaf Service List

Talend runtime is based on apache Karaf project, so to check the exported service has installed correctly the command [list] has been used in Karaf command line, and as mentioned in figure 3.21 the two service Authentication and OkmDocument has been installed, activated and ready to use by another system.

3.4.8 Bonita Workflow Management System

Bonita is a workflow management system tool that has been used in the research to implement the idea of workflow, holiday request has been used as an example of workflow case study.

a. Business Model

   The company has many workflow cases and most of them have documents included, the growth of using workflow led to increasing in a number of documents which needed to be stored in the central repository. Eventually, the need for another system to manage the attached documents was necessary, but the company needs to be sure the new system will be able to manage all documents that come from other sources, so the integrate between company systems simply must be SOA based principles.

b. Connectors

   A connector is implemented in Bonita BPM in two parts, the definition, and the implementation. This enables developers to change the implementation without changing the definition. Several implementations can be created for a single definition.
   A connector is used in Bonita to integrate with another system, it has a many support type of connections but in this research, it has been used a web service connector.

   i. Authentication connector

      A connector is used to get authentication from DMS service provider. It use SOAP protocol as shown in figures from 3.22 to 3.28.

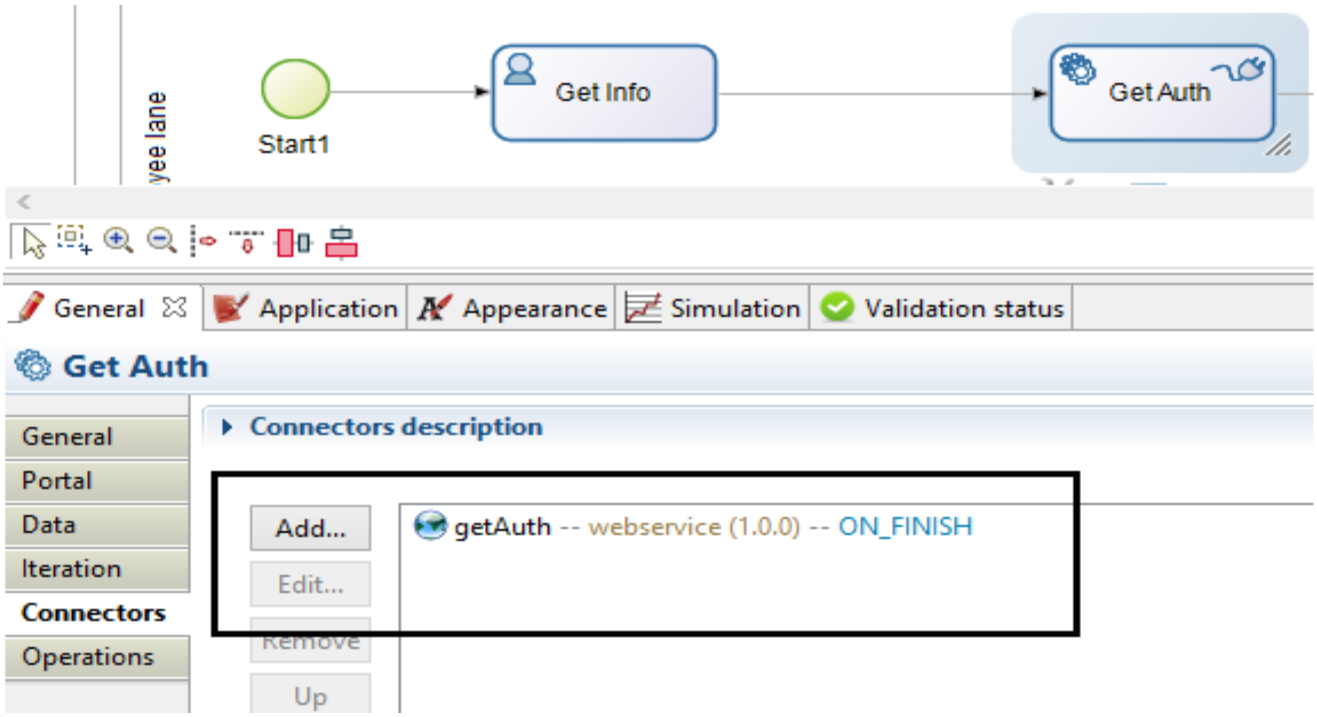

Figure 3.22: Authentication Connector

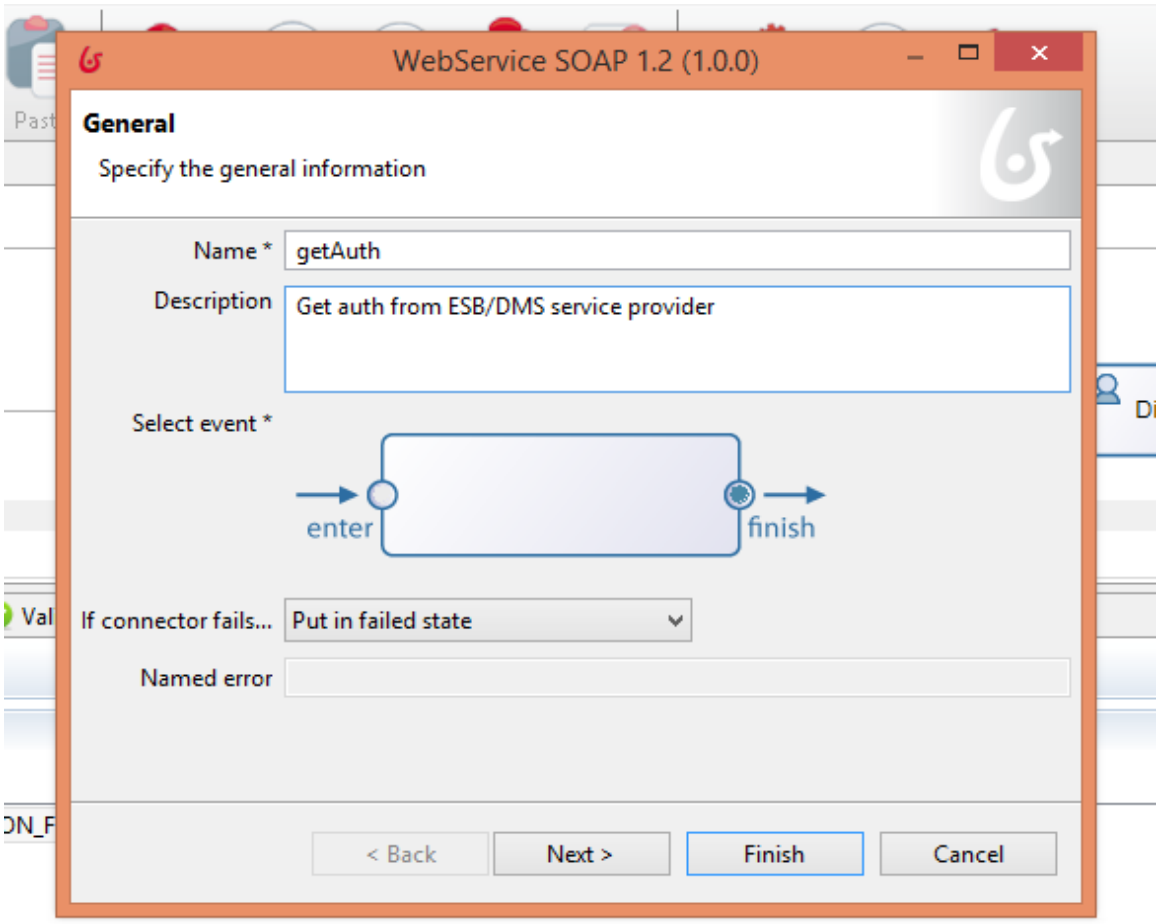

Figure 3.23: Connector General Information

In this stage general information has been entered, like connector name, description, event state and what happen when connector fail as shown in figure 3.23.

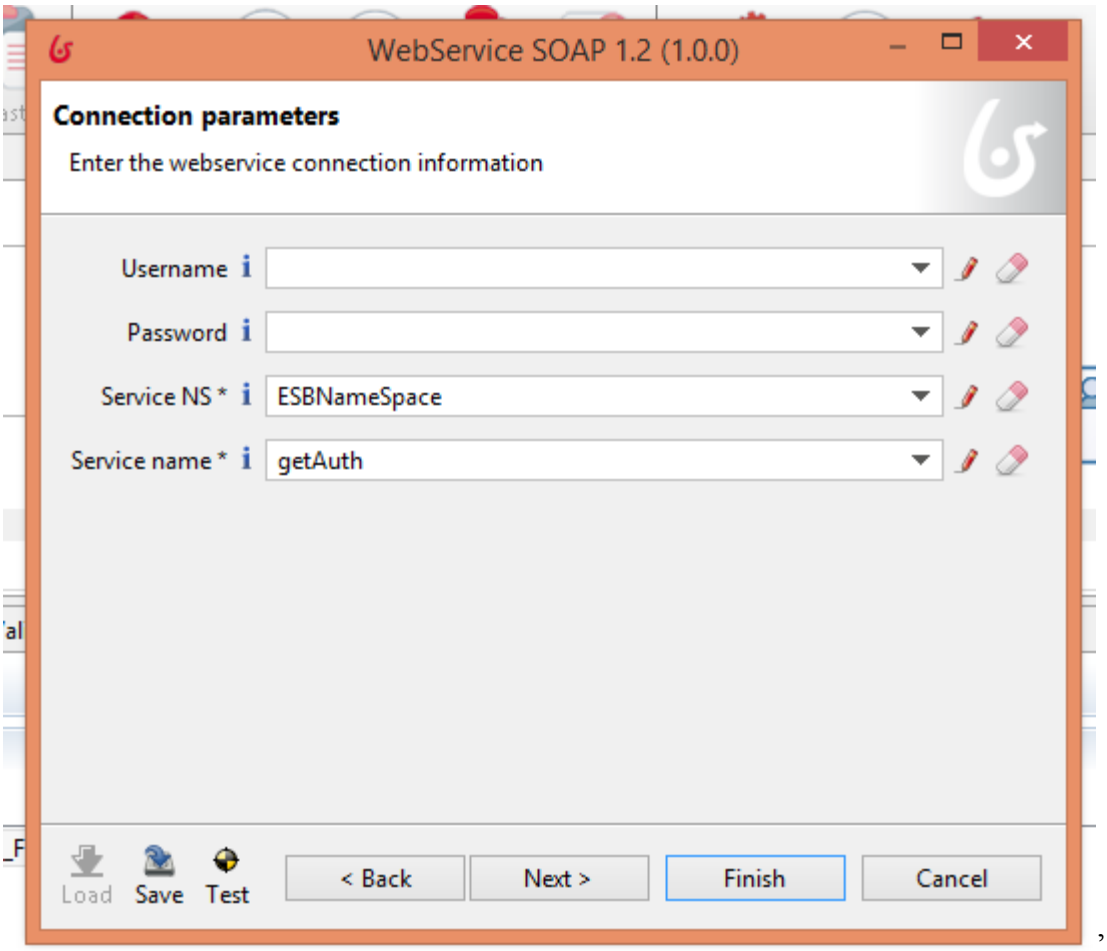

Figure 3.24: Connector Parameters

In this stage connector parameters have been entered, like service namespace and service name as shown in figure 3.24, there are other options which are username and password and it may be used in case of service provider asking for this information.

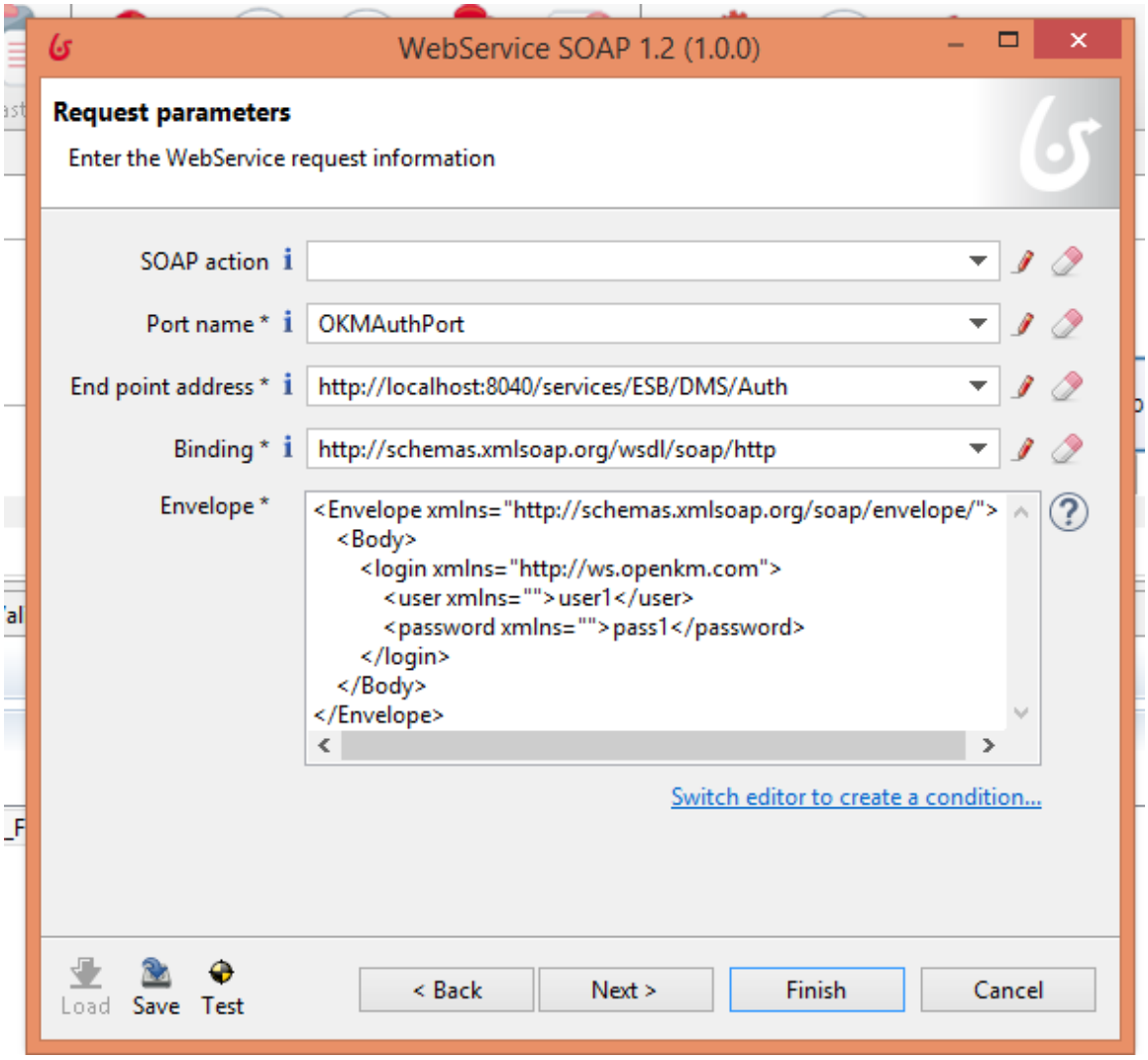

Figure 3.25: Authentication Connector Request Parameters

Well, this is most important and sensitive stage in a connector implementation, most of the information entered here have been extracted from service provider WSDL, and one wrong character can make a connector fail. The first parameter is the port name and it comes directly from the WSDL file, secondly, the endpoint address in which the connector will call the service and as mentioned in figure 3.25 the endpoint is localhost although the OpenKM DMS is hosted in the cloud, but thanks to ESB who make the services isolated from each other. Binding is different from SOAP1.1 to SOAP1.2, here it has been used SOAP1.1 binding. Lastly, the connector needs to have envelope body to be able to send the XML request.

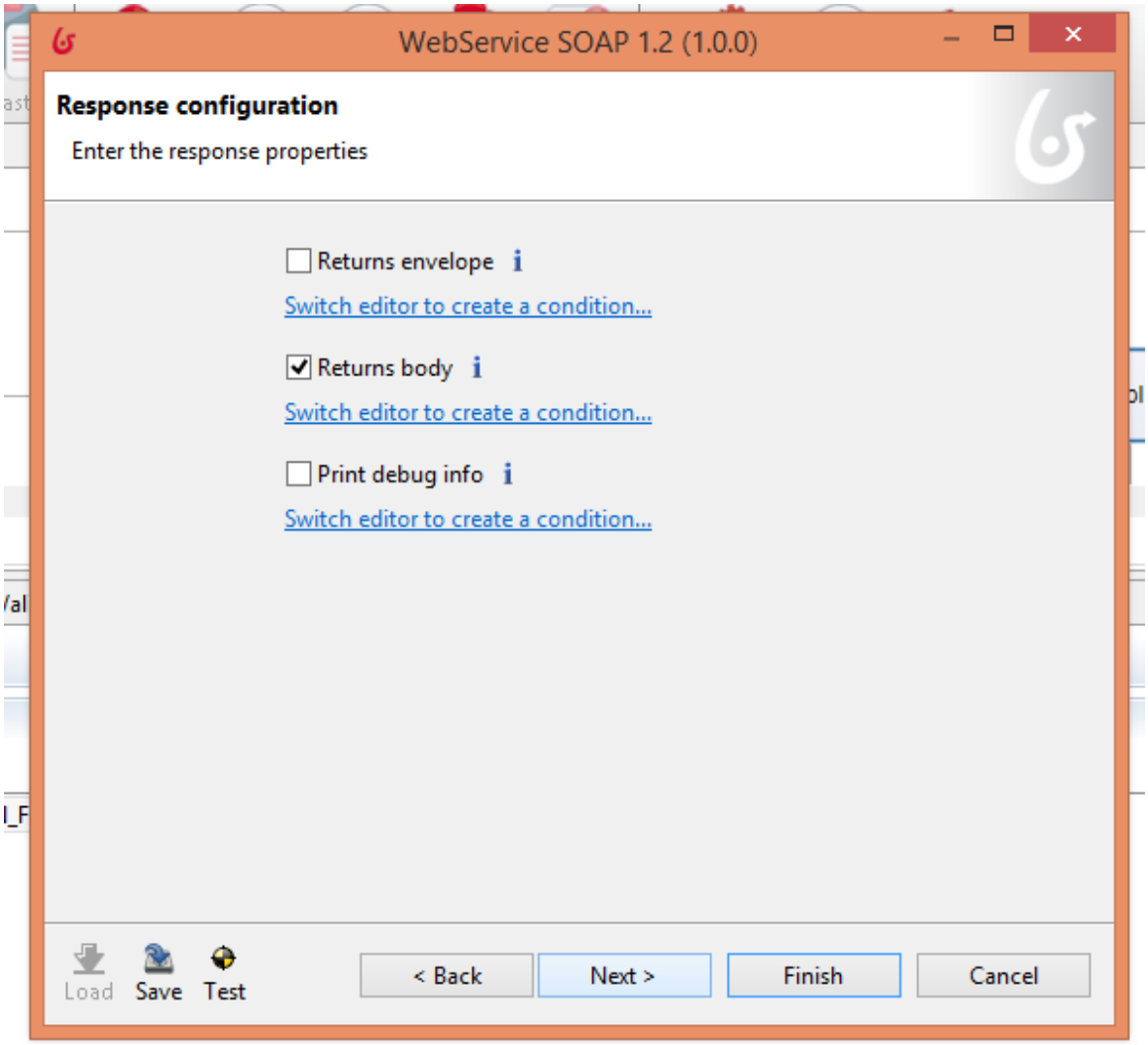

Figure 3.26: Connector Response Configuration

In connector response configuration stage it is just [**return body**] has been checked to make it return an only response body as shown in figure 3.26, in development normally the envelope to debug the response but when has been a switch to production mode it was better to disable this feature.

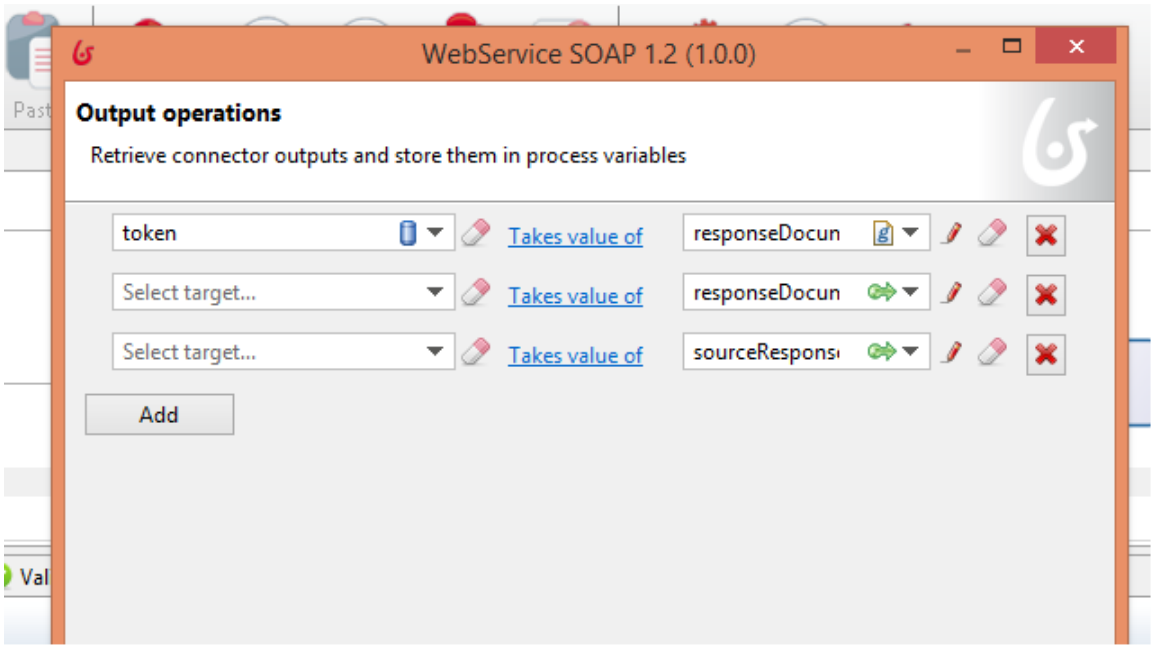

Figure 3.27: Connector Output Operations

In connector output stage, the connector output has been retrieved and stored in a specific variable as shown in figure 3.27.

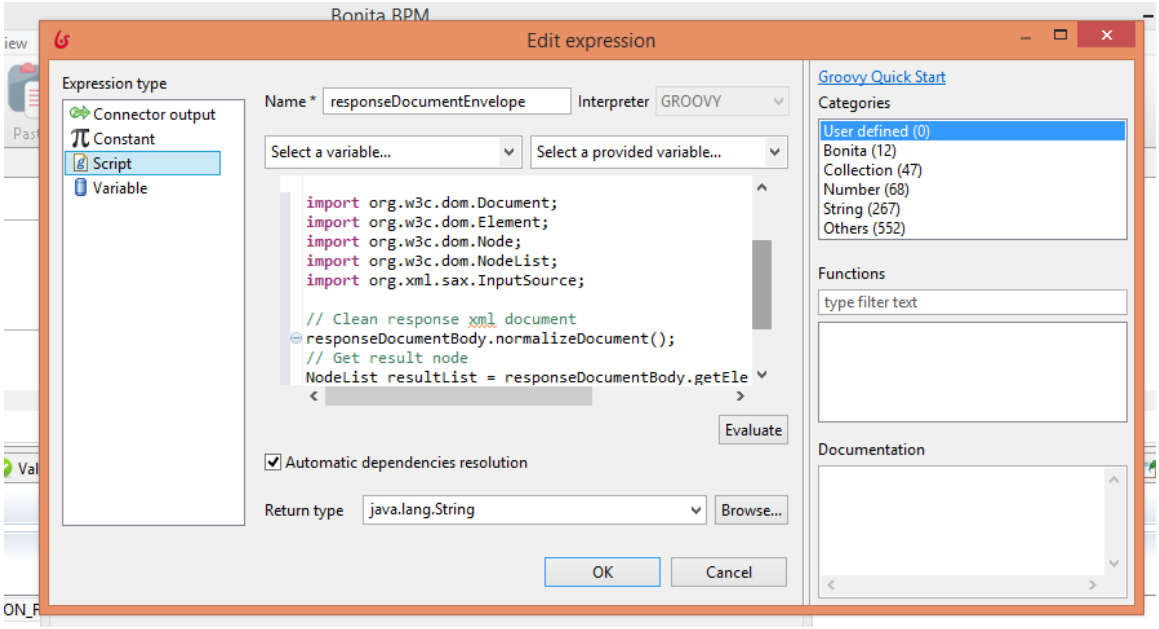

Figure 3.28: Connector Output Expression

In connector output expression stage, the expression code has been used to manipulate the stored data because the retrieved data has been returned in XML

structure and then data has been extracted and converted to a type that Bonita tool can understand. Figure 3.28 shows the expression.

ii. Upload Document connector

This connector is responsible for preparing the file, convert its content to a byte stream and then send it to the service provider as shown in figures 3.29 and 3.30.

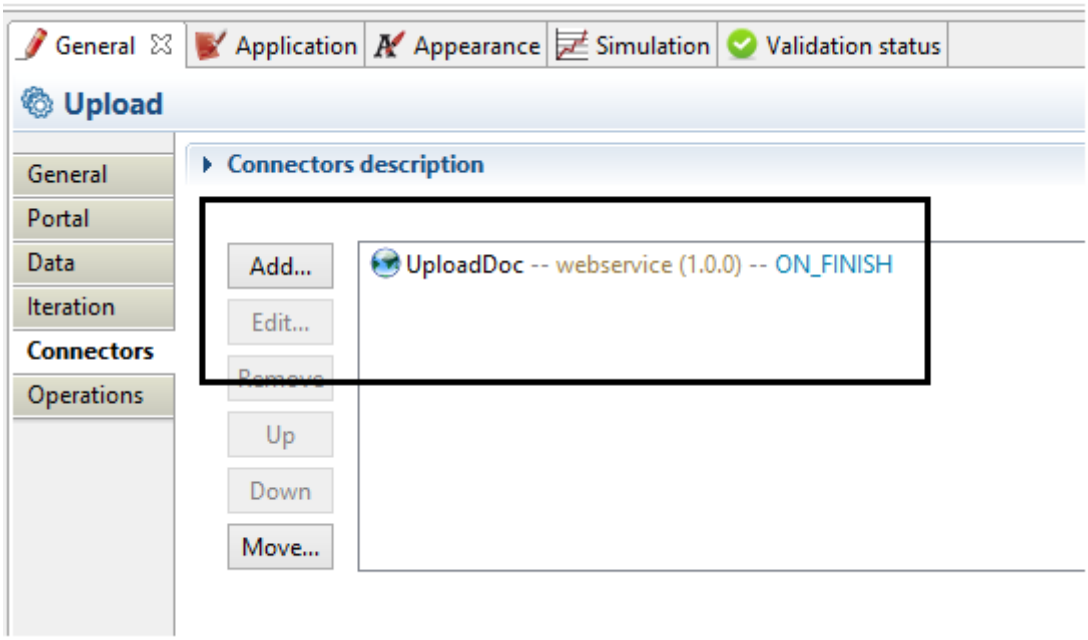

Figure 3.29: Upload Document Definition

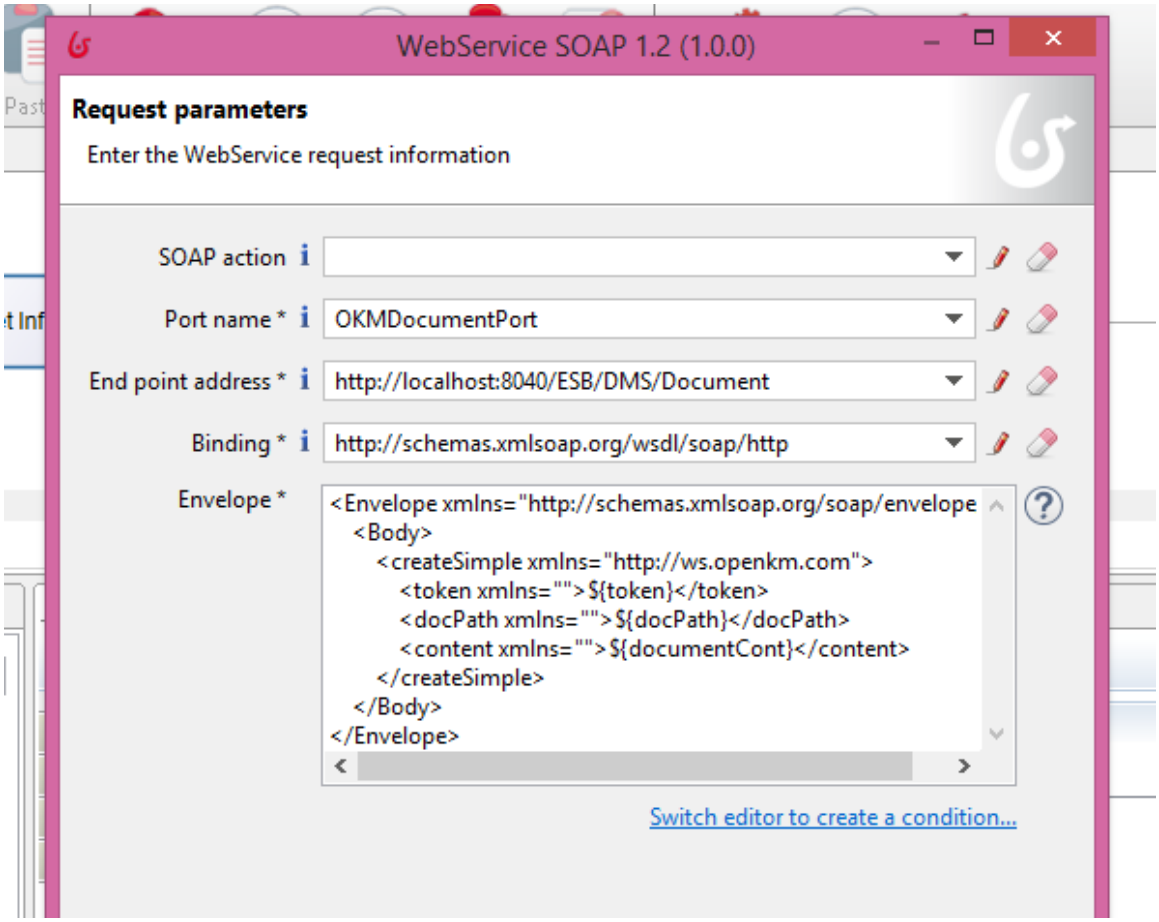

Figure 3.30: Upload Connector Request Parameter

There is a few similarity between connectors implementation but they differ in request parameter configuration because every connector has a unique envelope structure. in this envelope, there are three variables that must be sent to endpoint address which is the security token which uses to authenticate the request, document path where the document must be stored in DMS system and document content which using expression code later to convert from file to Unicode 64 byte stream.

iii. Edit Document connectors

This connector is responsible for download and check-out the file, as shown in figures 3.31 to 3.33

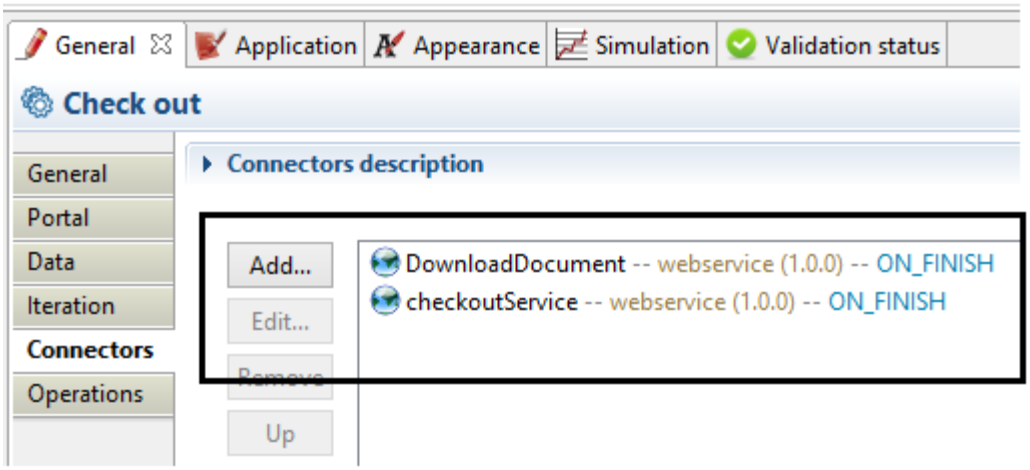

Figure 3.31: Edit Document Connector Definition

Edit document activity has two connectors as shown in figure 3.31 because it contains two of processes, first the download document process and the second is change the state of the document in DMS to update status.

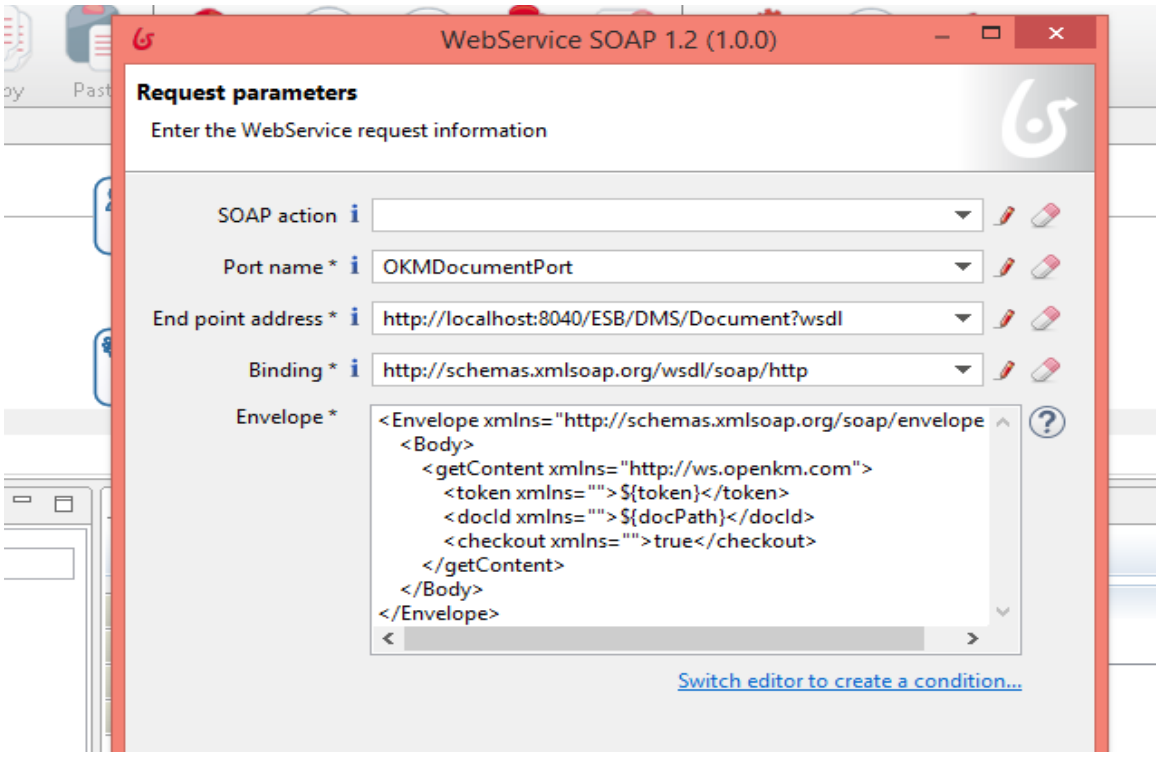

Figure 3.32: Download Document Connector Request Parameter

Figure 3.32 shows three variables that must be sent to endpoint address which are security token, document path which used by DMS to get the document, and Boolean flag which has been set a true value to tell the DMS

that the change of state will be after the download operation.

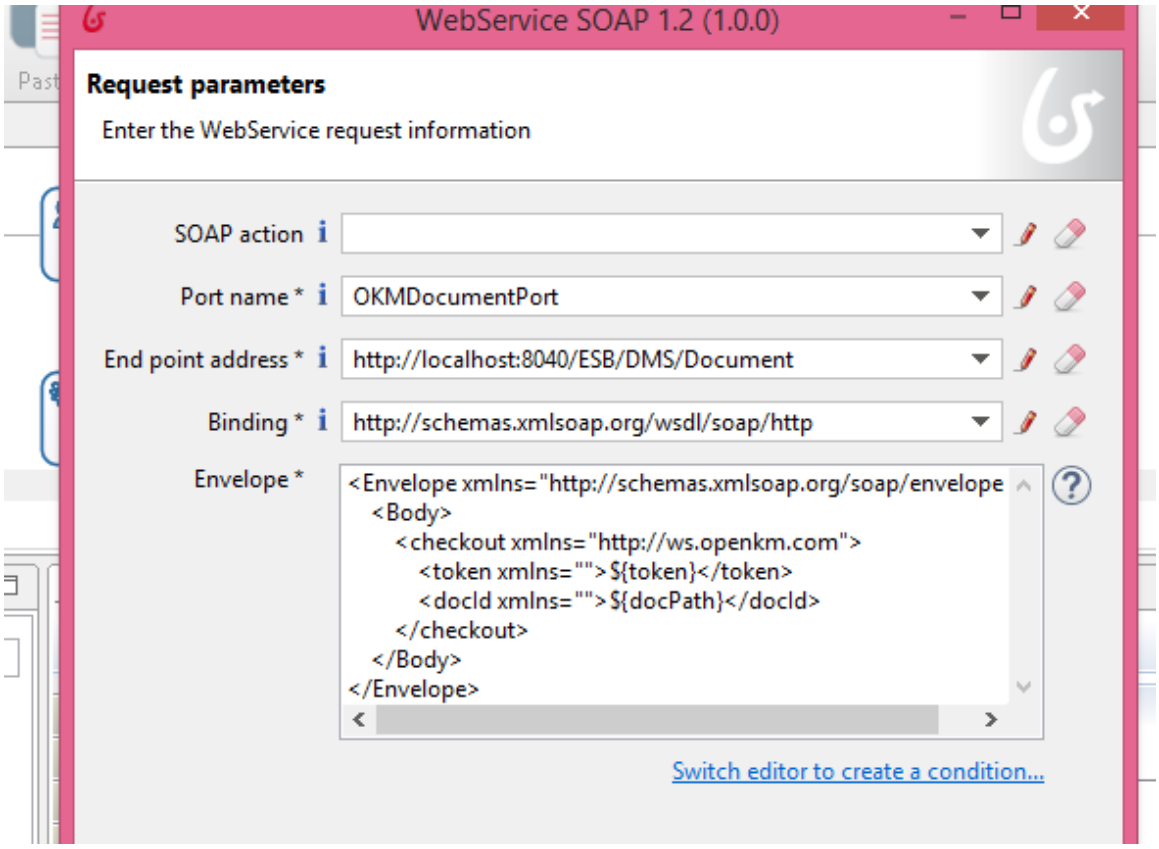

Figure 3.33: Check-out Document Connector Request Parameter

In this connector, there are two variables that have been sent to DMS through ESB, security token and the document path as shown in figure 3.33.

iv. Update Document connector

This connector is responsible for update the document, and it has four variables that must be sent to DMS through ESB which are security token, the document path, document content, and comment. Groovy expression code has been used to manipulate the file content and convert it to Unicode 64 to be understandable by DMS system. As shown in figures 3.34 and 3.35.

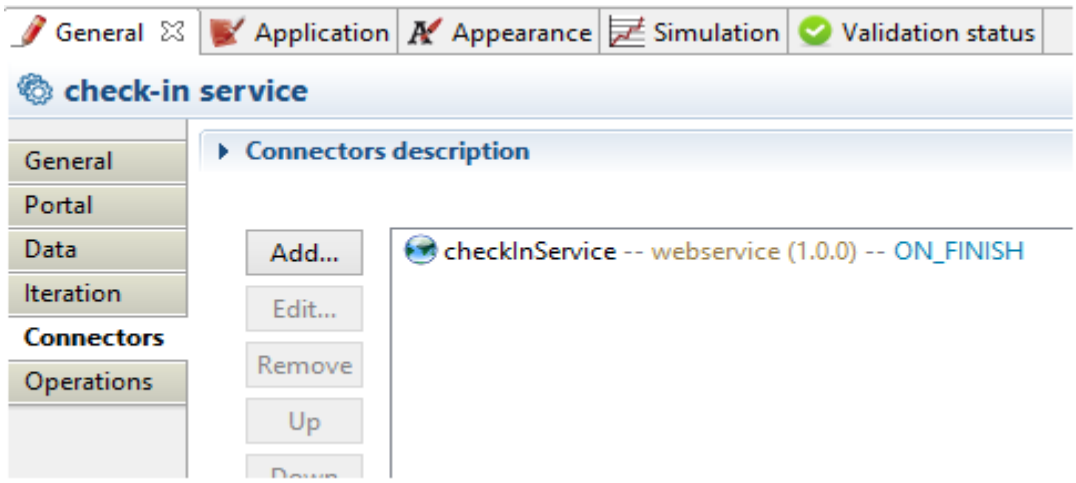

Figure 3.34: Check-in Document Connector Definition

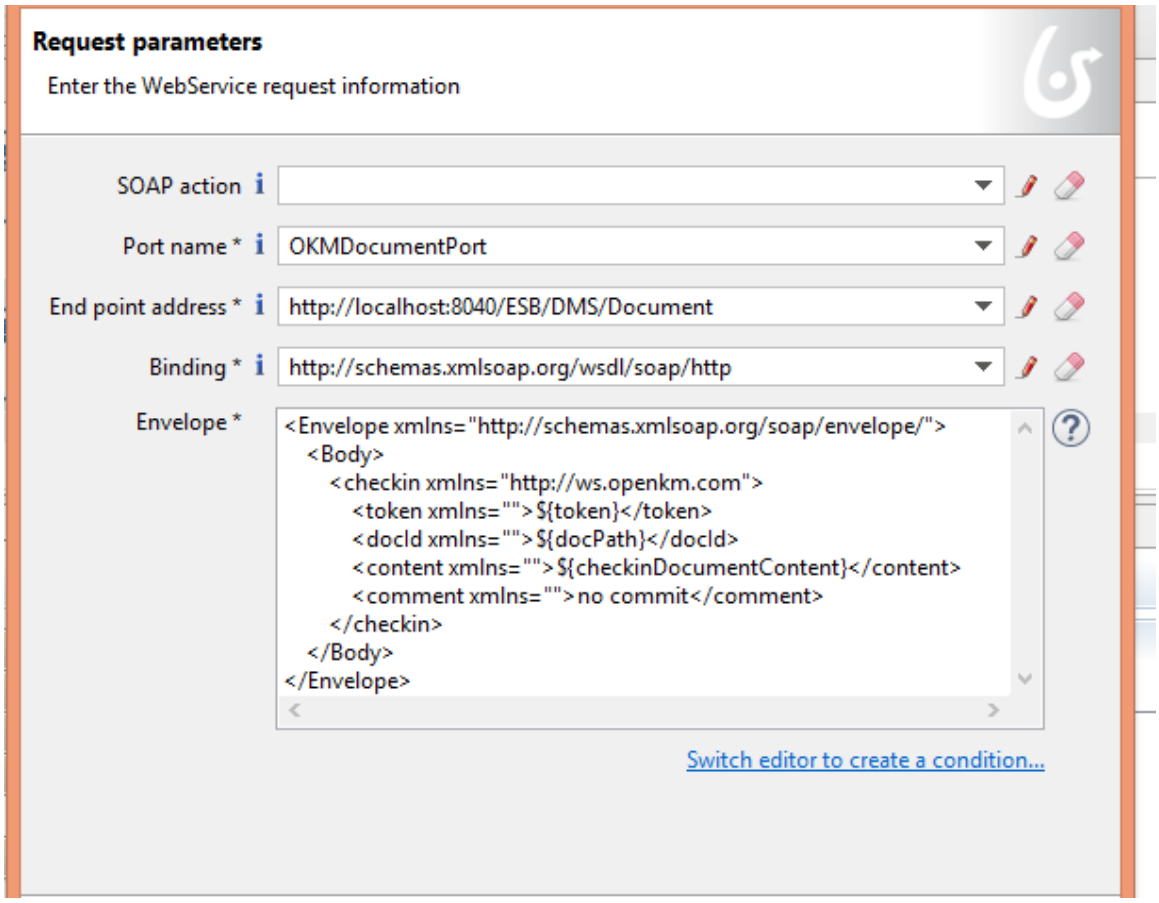

Figure 3.35: Check-in Document Connector Request Parameter

3.4.9 OpenKM DMS

OpenKM is document management system that has been used in this research as

DMS service provider and has been integrated with ESB. The service was hosted in cloud environment with the domain name http://demo.openkm.com/OpenKM/

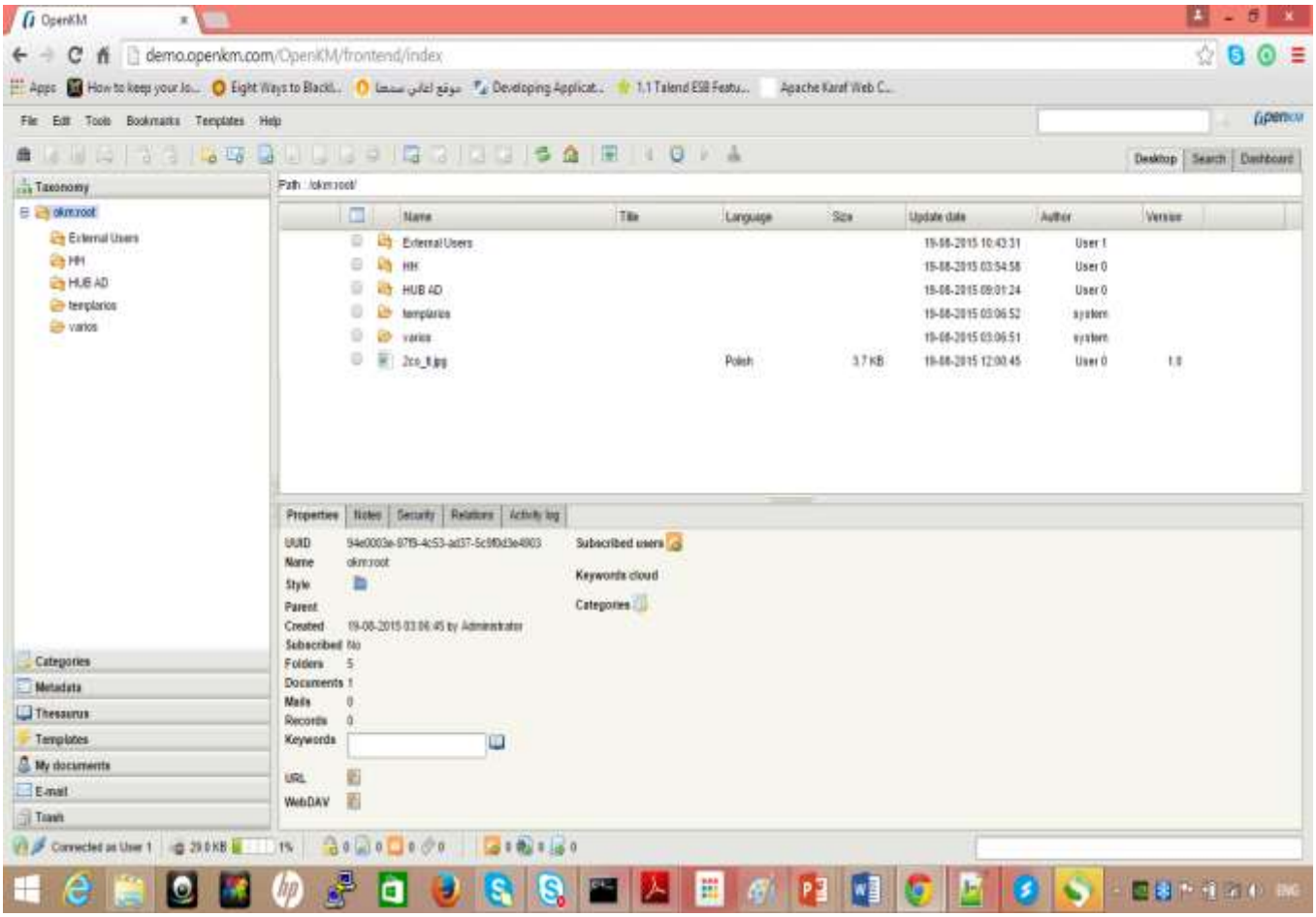

Figure 3.36: OpenKM Main Page

3.4.10 System Testing

This tool has been used to make sure that all service provided by ESB are stable and reliable

a. Introduction to Soap UI

SoapUI is a free and open source cross-platform Functional Testing solution. With an easy-to-use graphical interface and enterprise-class features, SoapUI allows the developer to easily and rapidly create and execute automated functional, regression, compliance, and load tests. In a single test environment, SoapUI provides complete test coverage and supports all the standard protocols and technologies. (SoapUI, 2015)

b. Service Test

SoapUI tool has been used to test four services

i. Login

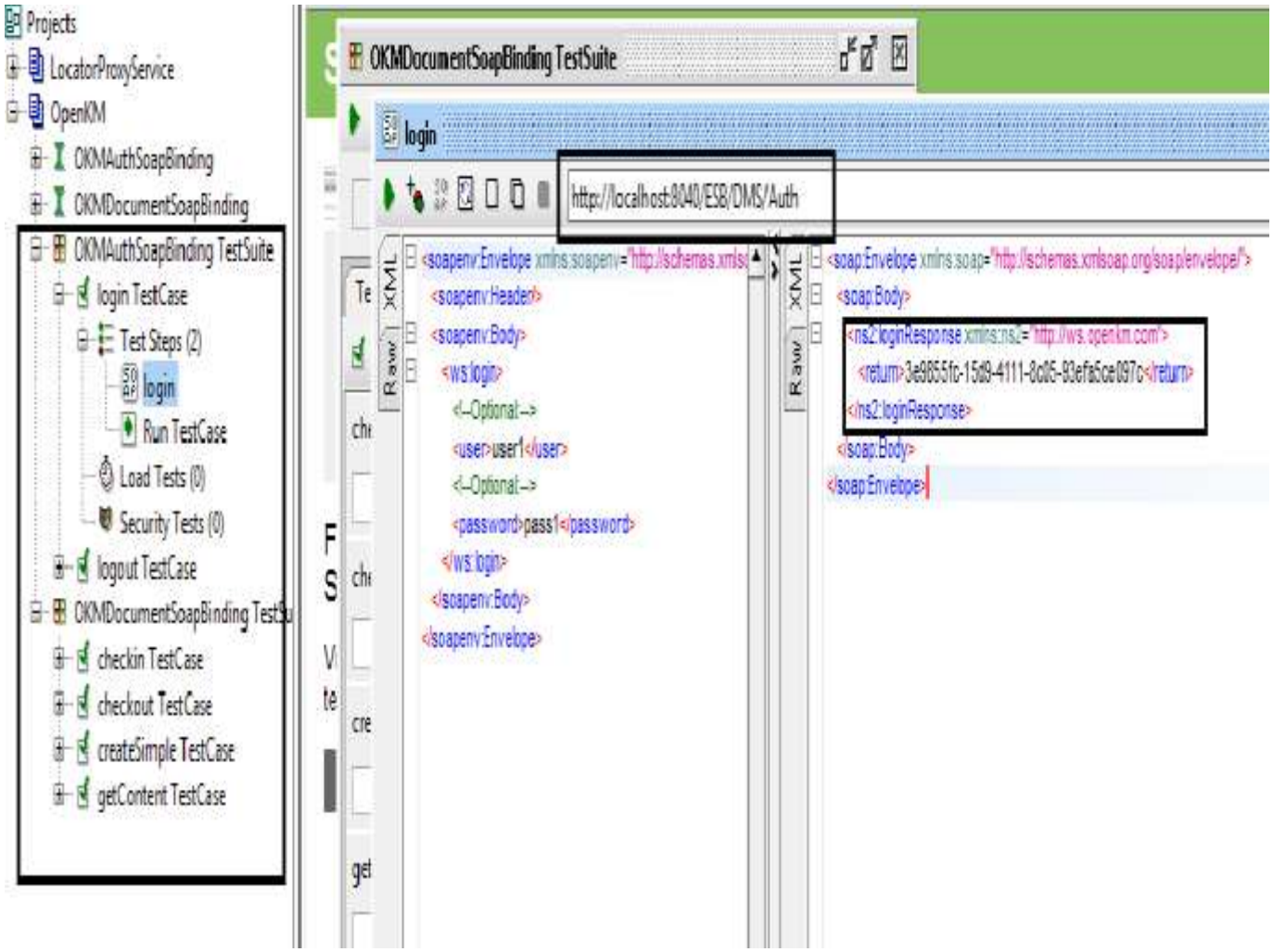

Figure 3.37: Login Test Case

The left side of the screen is requested panel and the right side is the response panel from ESB. Using SoapUI the login request has been sent to local ESB and the ESB sent a response to contain valid security token. This token has been used in all other test cases.

ii. Upload document

In upload test case, three variable has been sent using SoapUI, security token from login test case, document path, and document content as shown in figure 3.38. It has been used the website http://www.motobit.com/util/base64-decoder-encoder.asp to convert the file to base64.

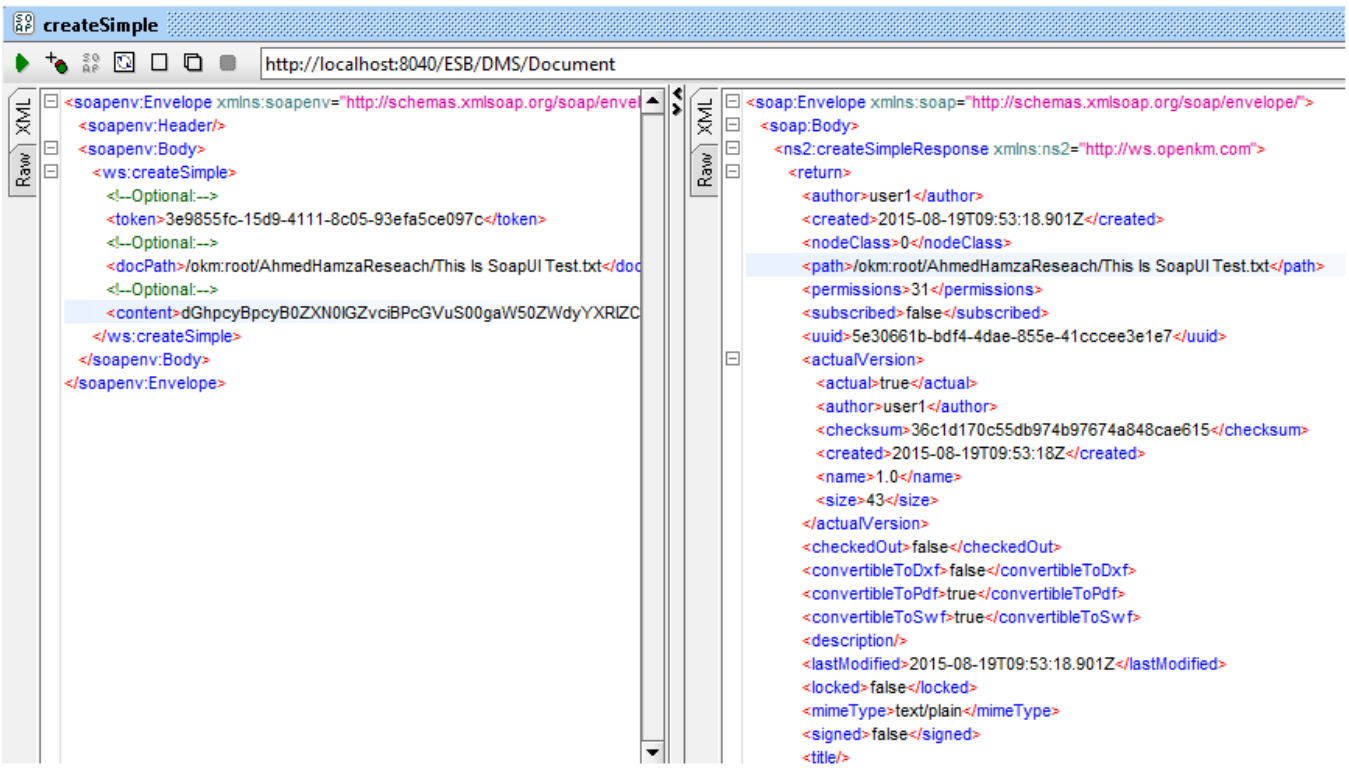

Figure 3.38: Upload Test Case

iii. Download document

- Get file content

In download test case, two variables have been sent using SoapUI, security token from login test case, document path. The response is long string based on Unicode 64 as shown in figure 3.39.

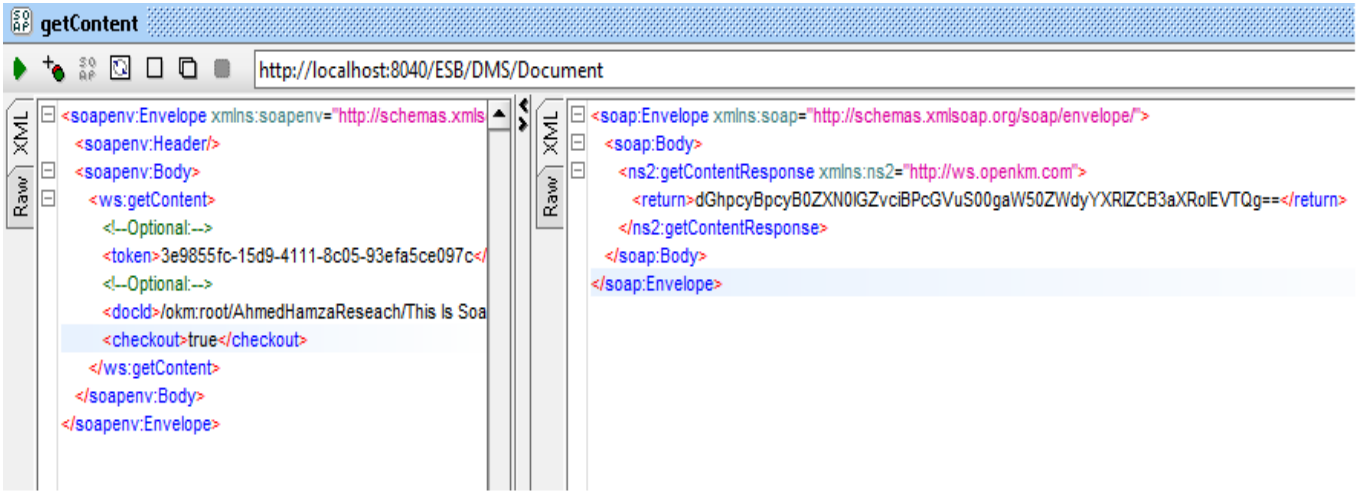

Figure 3.39: Download Test Case

- Check-out operation

In a check-out test case, two variables have been sent using SoapUI, security token from login test case, document path as shown in figure 3.40. The response is empty envelope but when reviewed the DMS itself the check-out flag has been enabled as shown in figure 3.41, arrow shape has been used to clarify the check-out operation.

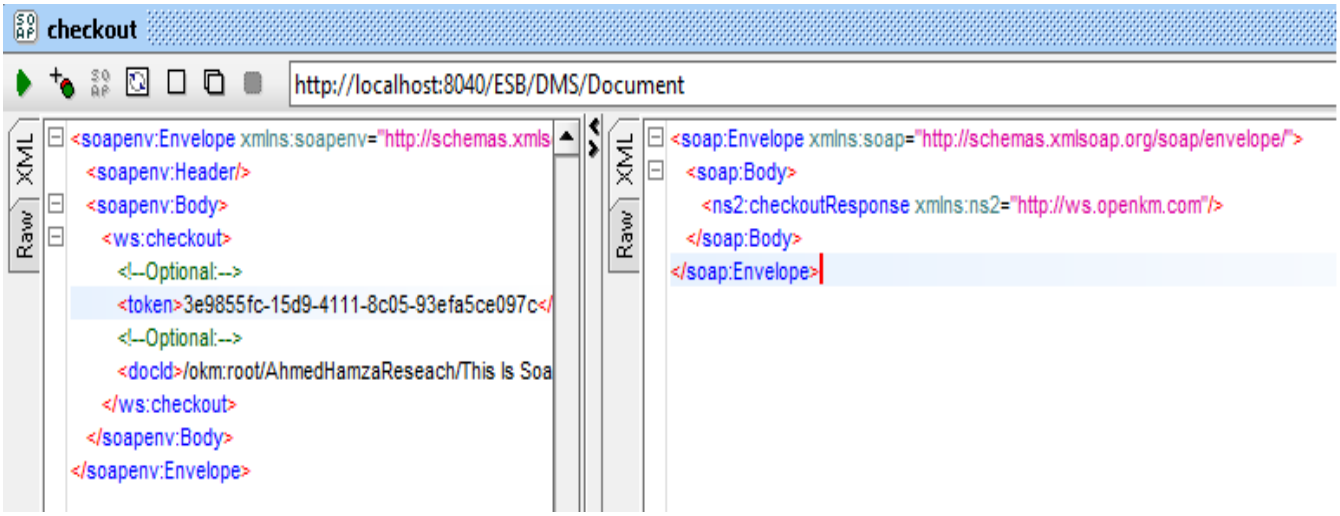

Figure 3.40: Check-out Test Case

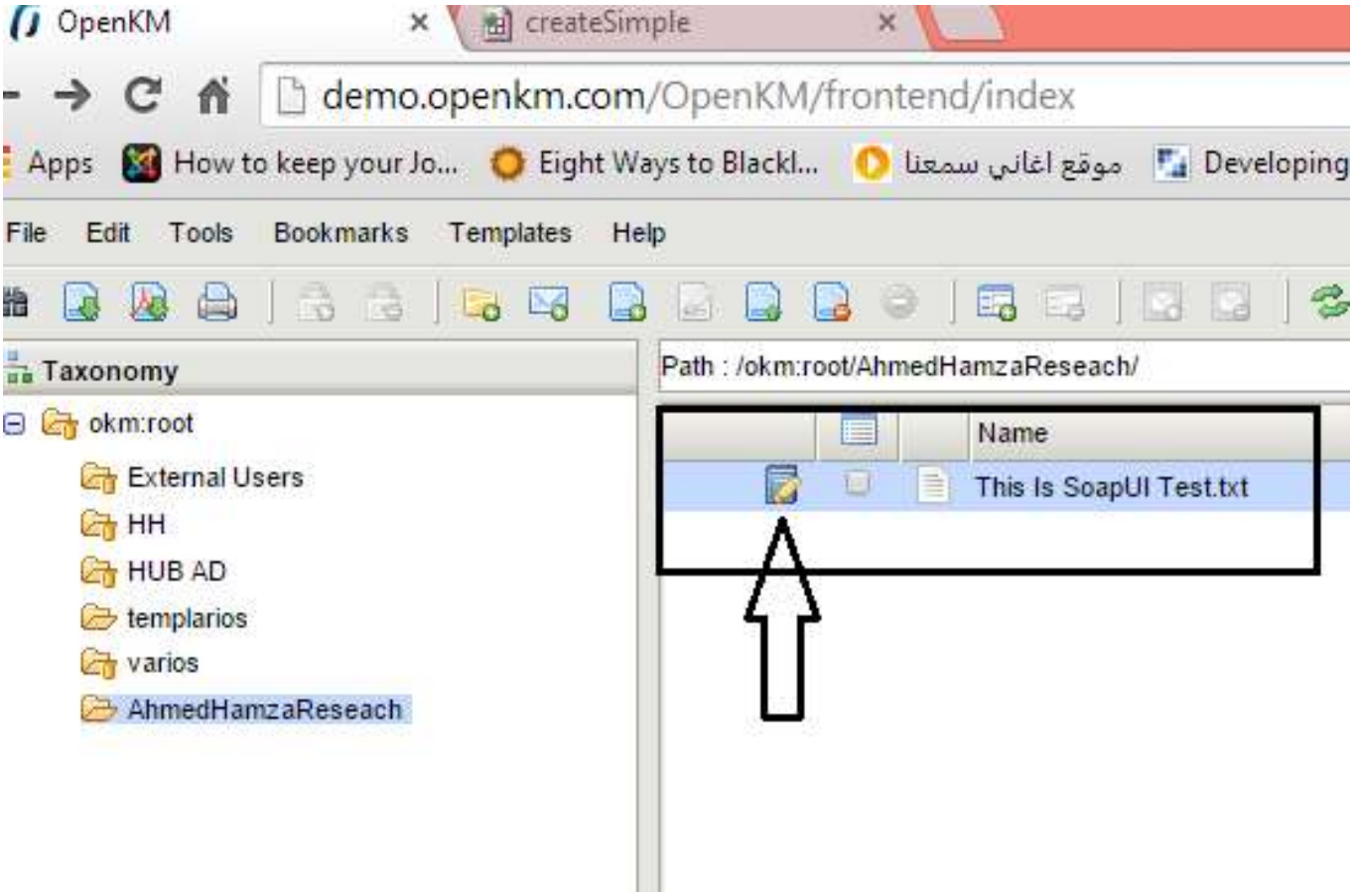

Figure 3.41: Document after Check-out

iv. Update document

In update test case, three variables have been sent using SoapUI, security token from login test case, document path, and document content as shown in figure 3.42. The response is envelope contain the document properties like author, created date and size. Figure 3.43 shows the document version has been increased to 1.1, arrow shape has been used to clarify the update operation.

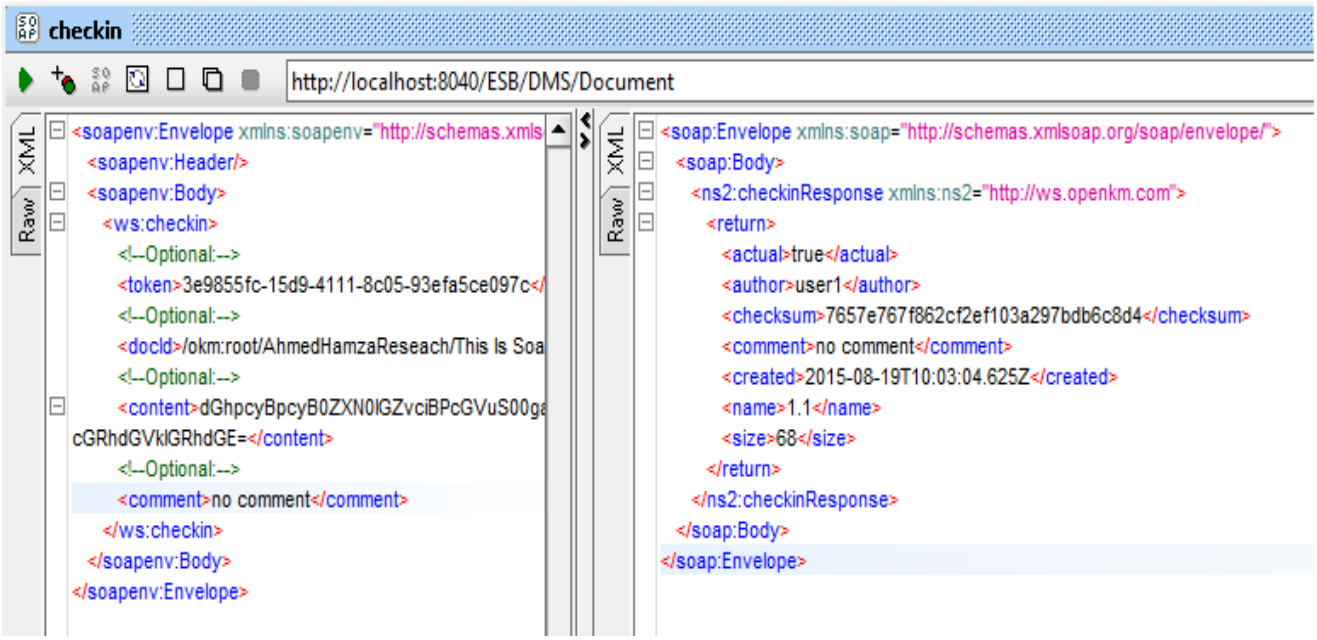

Figure 3.42: Update Document Test Case

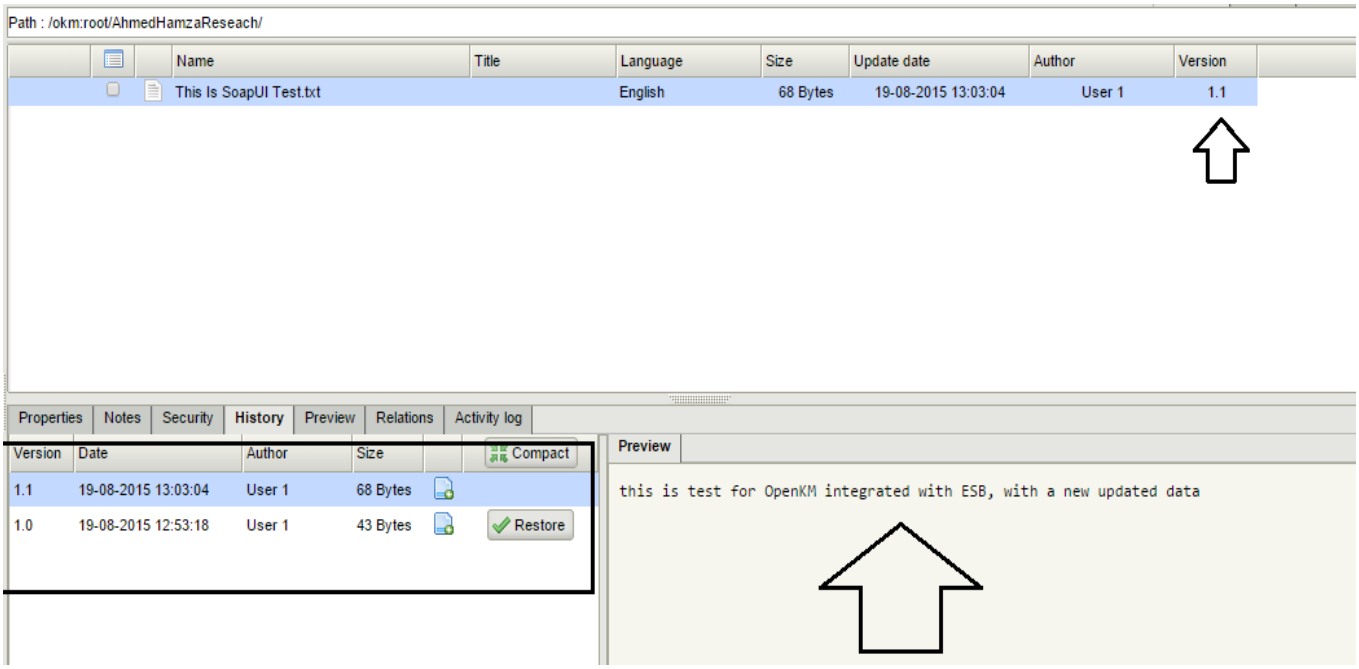

Figure 3.43: Document after Check-in

All test cases have been run successfully

3.5 System Deployment

When Bonita server has been started and the flow began, the only endpoint address is localhost

and ESB take the responsibility of making services connected, based on SOA principles as shown in figure 3.44.

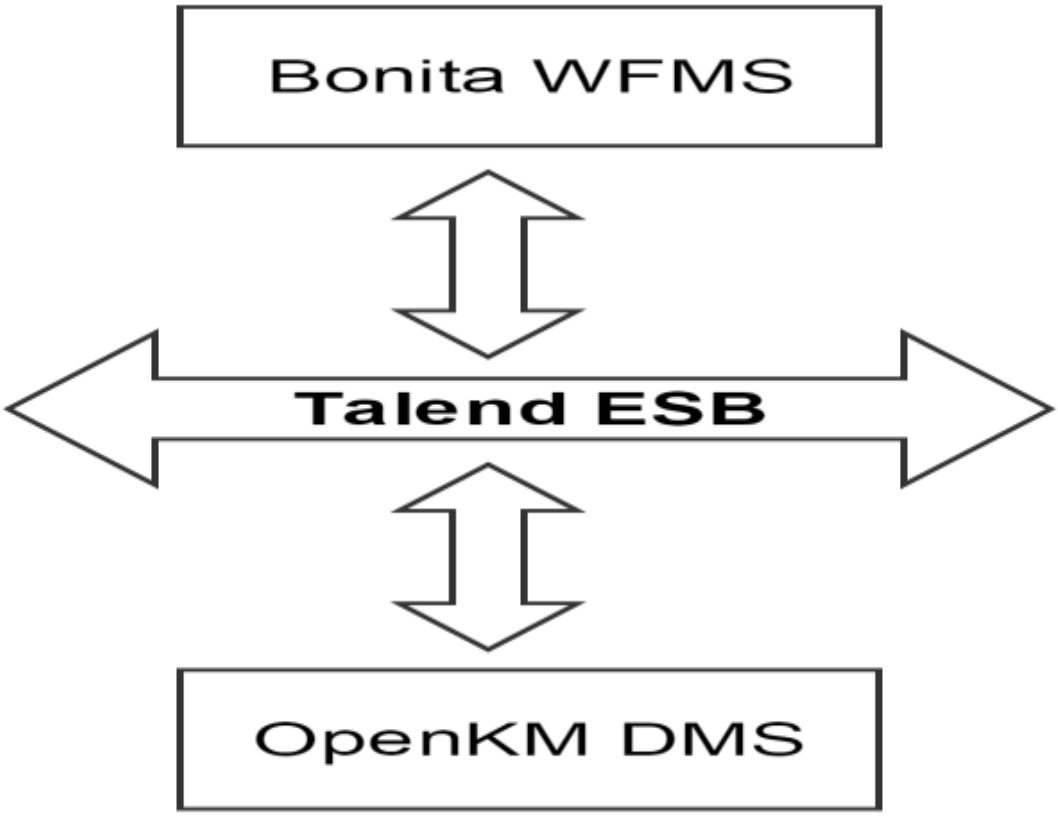

Figure 3.44: Deployment Model

Figure 3.45 shows the Bonita process begun and then in figure 3.46 shows the Document that has been uploaded in OpenKM DMS. After the document uploaded Bonita process downloaded it again to implement check-out operation as shown in figure 3.47 and the check-out flag has been enabled as shown in figure 3.48, finally, the document after been modified it has been updated in OpenKM DMS and the check-in operation has been executed as shown in figure 3.49 and 3.50.

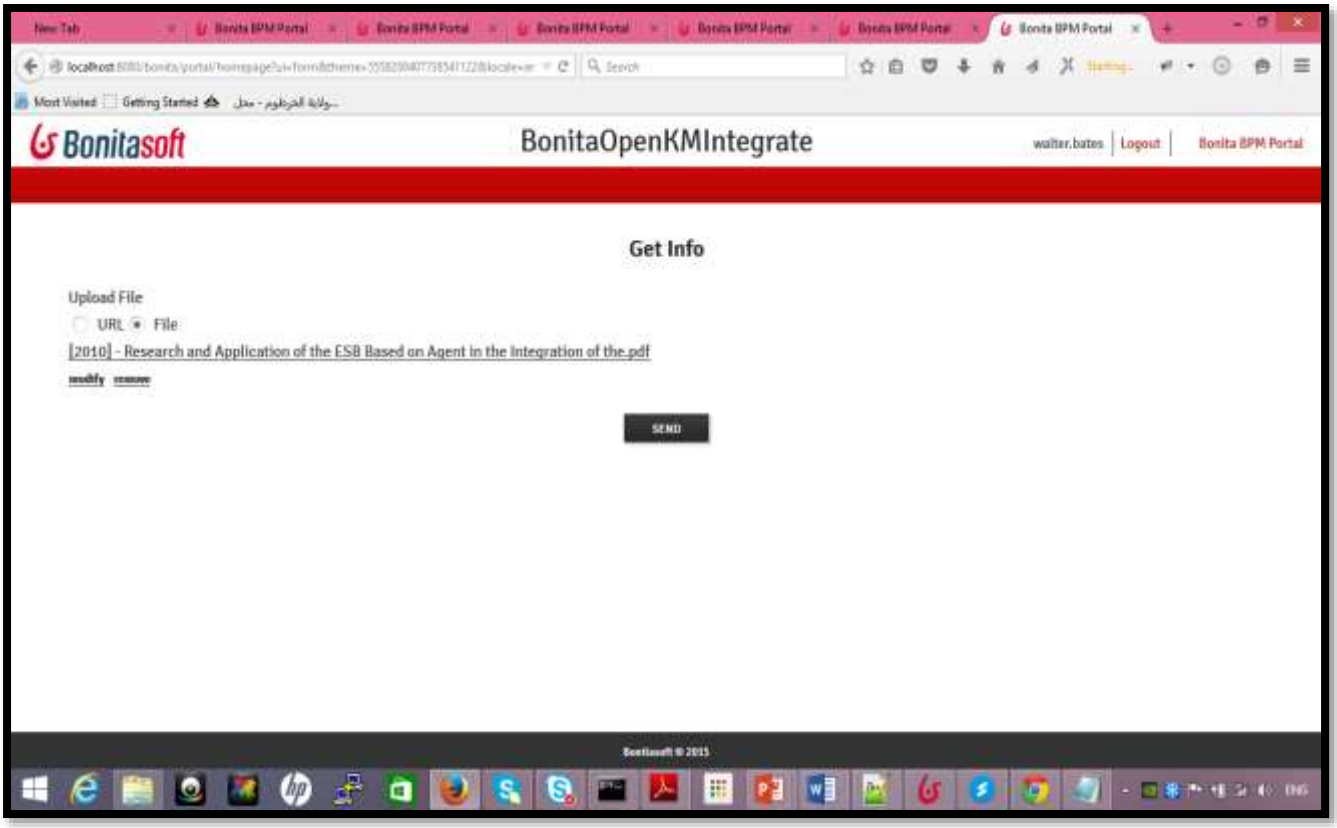

Figure 3.45: Bonita Upload Stage

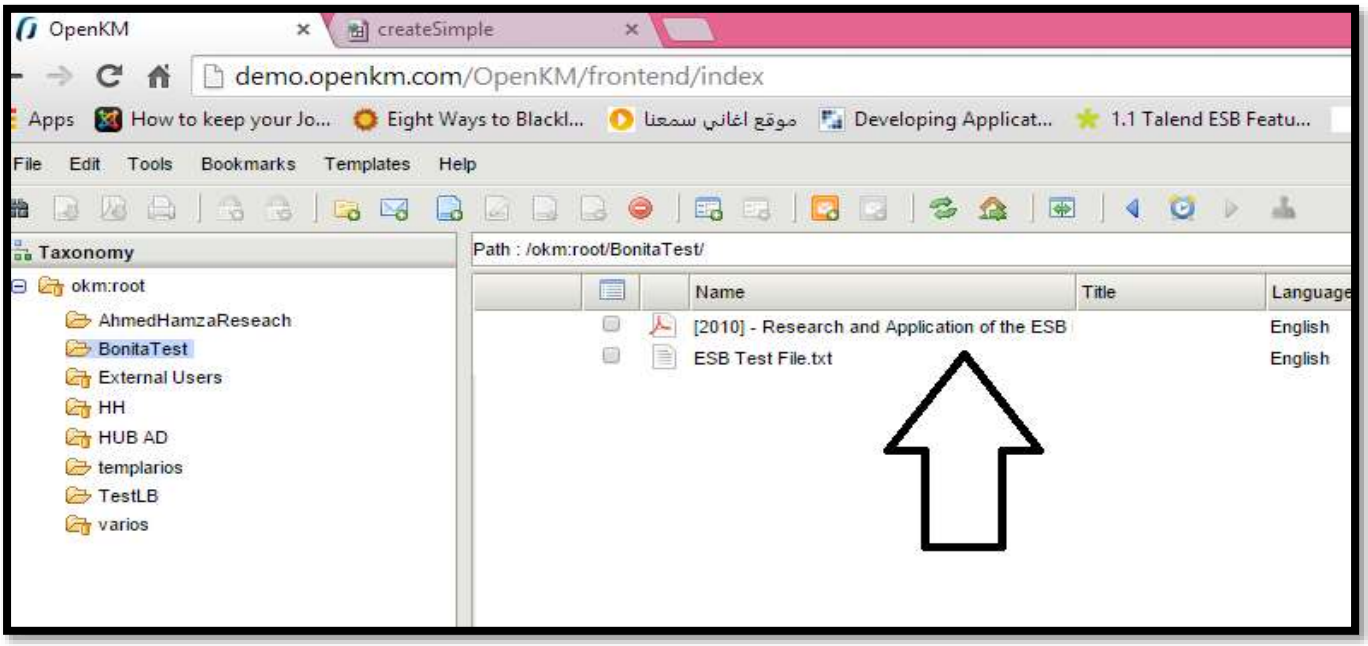

Figure 3.46: OpenKM Taxonomy

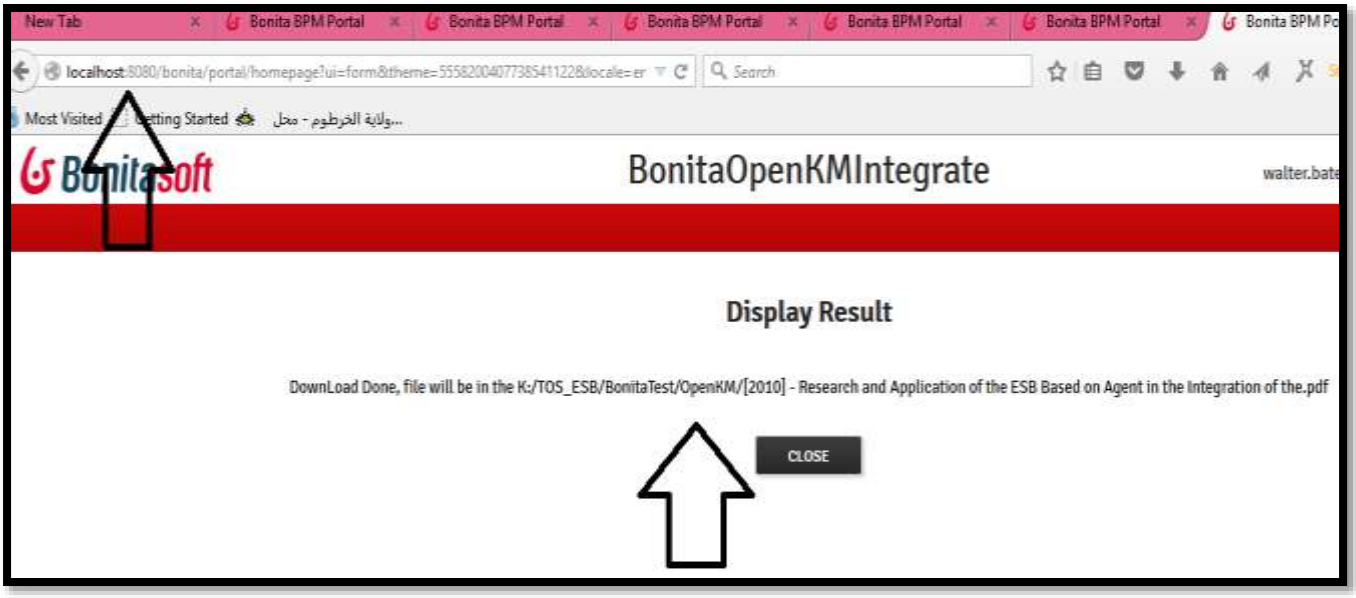

Figure 3.47: Bonita Download Stage

Figure 3.48: OpenKM Check-out Status

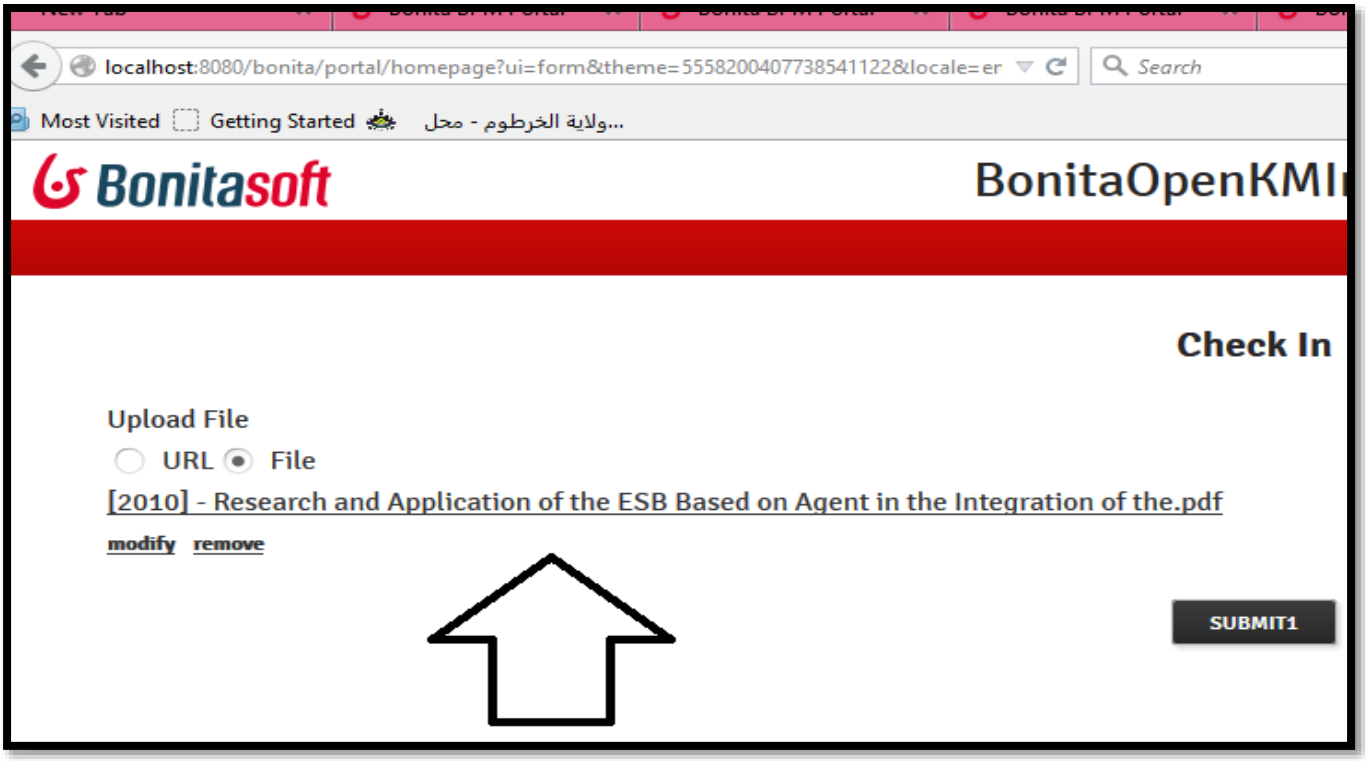

Figure 3.49: Bonita Update Stage

Document has been ready to be updated.

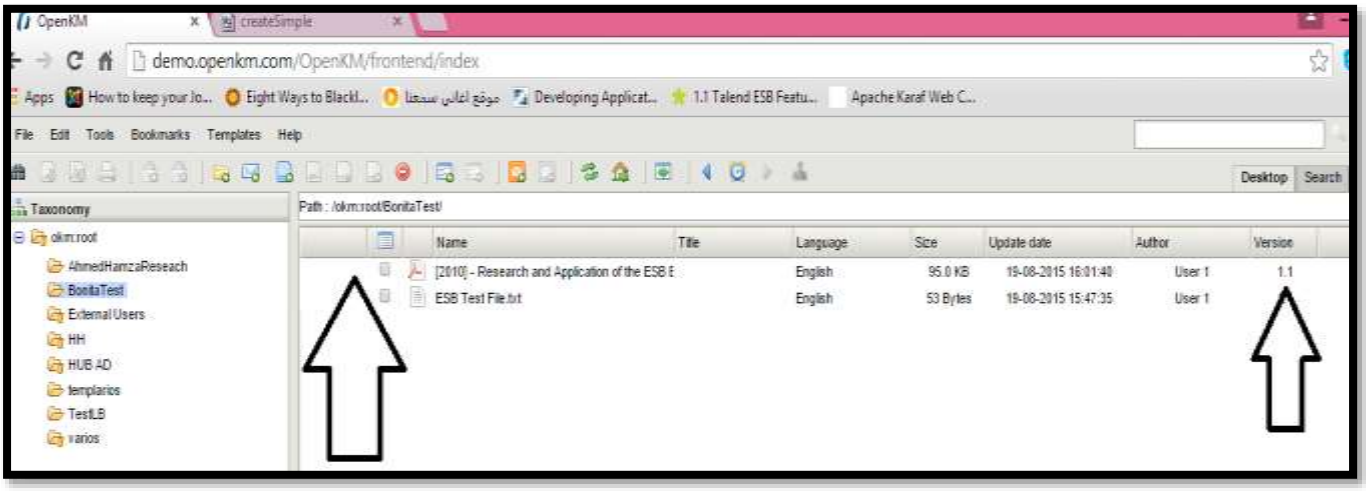

Figure 3.50: OpenKM Version View

Document has been updated and version increased by 0.1

Chapter Four: CONCLUSION AND RECOMMENDATIONS

4.1. Conclusion

The research studied different methodologies to integrate enterprise systems. This is to achieve successful and stable integration processes.

4.2. The Result

a. Three open sources were examined: OpenKM, Bonita and Talend ESB. OpenKM and Bonita were found to be not free of charge, since they highly charge their costumers for essential team training and for buying a startup package for the production phase. Moreover, they only respond to queries of subscribed costumers previously paid for their subscription.
b. Talend ESB was chosen among the others because of its advantage of providing ability for writing codes during product customization.
c. Finally, the research ended by implementing and successfully testing ESB in a pioneer company after setting up the integration environment and using an appropriate search engine.

4.3. Recommendations

Based on the results the researcher recommends for the following studies and usages:

a. Use the service locator.
b. Use STS to increase security.
c. Upgrade to Talend platform or Talend enterprise to work with advanced components that are not available in the standard edition like TAC component and SAM server.
d. Use cluster container and load balancing in ESB runtime.

REFERENCES

A., G.-J. F. J. &. M.-C. M., 2010. *Evaluating Open Source Enterprise Service Bus,* s.l.: s.n.

ALSÈNE, E., 1994. Computerized integration and the organization of work in enterprises. *International Labour Review*, pp. 657-677.

Amjad Umara, A. Z., 2009. Reengineering for service oriented architectures: A strategic decision model for integration versus migration. *Journal of Systems and Software,* 82(3), pp. 448-462.

Anon., n.d. [Online].

Asankha C. Perera, R. L., 2013. *ESB Performance.* [Online]
Available at: http://esbperformance.org/display/comparison/ESB+Performance
[Accessed January 2015].

BonitaSoft, 2014. *development.* [Online]
Available at: http://documentation.bonitasoft.com/product-bos-sp/development
[Accessed January 2016].

Boucher, K., 2012. *Selecting a Workflow Management System for Your Company.* [Online]
Available at: http://blog.lansa.com/application-modernization/workflow/selecting-workflow-management-system-company
[Accessed March 2015].

Camel, A., 2013. *Architecture.* [Online]
Available at: http://camel.apache.org/architecture.html
[Accessed March 2016].

Camel, A., 2015. *Apahe camel.* [Online]
Available at: http://camel.apache.org/documentation.html
[Accessed 2015].

Chappell, D., 2004. *Enterprise Service Bus.* 1st ed. s.l.:s.n.

Chongwen Wang, Y. H., 2010. *DICOM Communication Mechanism and Engineering Project,* s.l.: s.n.

Chunmiao, X., 2012. *The Study of Metallurgical Automation Based on Integrated Model of the Enterprise Bus ,* s.l.: s.n.

CXF, A., 2016. *CXF Overview.* [Online]
Available at: https://cxf.apache.org/
[Accessed 2016].

Dai, P., 2011. *Design and Implementation of ESB Based on SOA in Power System,* s.l.: s.n.

Freivald, J., 2010. EAI – Enterprise Application Integration.

Gartner, 2013. *Magic Quadrant for On-Premises Application, Integration Suites,* s.l.: s.n.

Gregor Hohpe, W., 2003. *Hub and Spoke [or] Zen and the Art of Message Broker Maintenance.* [Online]
Available at: http://www.enterpriseintegrationpatterns.com/ramblings/03_hubandspoke.html
[Accessed June 2015].

High, R., 2006. SOA. *SOA Foundation Architecture Whitepaper.*

IBM, 2009. *The ESB Architectural Pattern,* s.l.: s.n.

Mason, R., 2014. *enterprise-application-integration.* [Online]
Available at: http://www.mulesoft.com/resources/esb/enterprise-application-integration-eai-and-esb
[Accessed January 2016].

MSDN, 2004. *Chapter 1: Service Oriented Architecture (SOA).* [Online]
Available at: https://msdn.microsoft.com/en-us/library/bb833022.aspx
[Accessed November 2015].

Mulik, S., 2009. *Using Enterprise Service Bus (ESB) for connecting Corporate Functions and Shared Services with Business Divisions in a large Enterprise,* s.l.: s.n.

Papazoglou, M. T. U. T. T. P., Dustdar, S. & Leymann, F., 2007. *Service-Oriented Computing: State of the Art and Research Challenges,* s.l.: s.n.

Ravi Khadka, A. S. S. J. J. H., 2013. *Migrating a Large Scale Legacy Application to SOA: Challenges and Lessons Learned,* s.l.: s.n.

Rouse, M., 2012. *enterprise-service-bus.* [Online]
Available at: http://searchsoa.techtarget.com/definition/enterprise-service-bus
[Accessed 2015].

Sachin Chandra, R. J., 2009. *A Practical Approach to Enterprise Integration,* s.l.: Defense AT&L.

SOA, 2014. *enterprise_service_bus.* [Online]
Available at: http://www.soa.com/solutions/enterprise_service_bus
[Accessed January 2016].

SoapUI, 2015. *what-is-soapui.* [Online]
Available at: http://www.soapui.org/about-soapui/what-is-soapui-.html
[Accessed 2015].

Talend, 2012. *ESB Model.* [Online]
Available at: www.talend.com/esb
[Accessed 2015].

Talend, 2014. *Talend ESB Getting Started Guide 5.6.1.* 2nd ed. s.l.:Talend Inc..

Talend, 2014. *Talend ESB Infrastructure Services ,* s.l.: s.n.

Talend, 2015. *Talend ESB Service.* 1nd ed. s.l.:Talend Inc.

Talend, 2015. *TalendOpenStudioComponentsReferenceGuide56EN.* [Online]
Available at:
https://help.talend.com/display/TalendOpenStudioComponentsReferenceGuide56EN/Home
[Accessed january 2016].

Thompson, J. G., 2010. *Predicts 2011: Application Integration,* s.l.: s.n.

U. Raza, B. W. F. H., 2012. *AN ENTERPRISE SERVICE BUS (ESB) AND GOOGLE GADGETS BASED MICRO-INJECTION MOULDING PROCESS MONITORING SYSTEM ,* s.l.: s.n.

W3, 2004. *Web Services Glossary. W3C..* [Online]
Available at: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice
[Accessed 2015].

Wähner, K., 2013. *ESB Integration.* [Online]
Available at: http://www.infoq.com/articles/ESB-Integration
[Accessed 2016].

Wikipedia, 2013. *Bonita_BPM.* [Online]
Available at: https://en.wikipedia.org/wiki/Bonita_BPM
[Accessed January 2016].

Wikipedia, 2013. *Message_oriented_middleware.* [Online]
Available at: http://en.wikipedia.org/wiki/Message_oriented_middleware
[Accessed 2015].

Wikipedia, 2014. *Apache Camel.* [Online]
Available at: http://en.wikipedia.org/wiki/Apache_Camel
[Accessed 2015].

wikipedia, 2015. *OpenKM.* [Online]
Available at: https://en.wikipedia.org/wiki/OpenKM
[Accessed 2016].

Woolf, H. &., 2012. *Enterprise Integration Patterns.* s.l.:s.n.

yenlo.nl, 2010. *what-are-routing-rules-in-an-enterprise-service-bus-environment.* [Online]
Available at: http://www.yenlo.nl/nl/what-are-routing-rules-in-an-enterprise-service-bus-environment/
[Accessed 2015].

APPINDEX A: Bonita Code

Upload document code

```
def byte[] filecontent =
apiAccessor.getProcessAPI().getDocumentContent(docu11.getContentStorageId());
//filecontent.encodeBase64();


byte[] decodedBytes = filecontent.encodeBase64().toString().decodeBase64();
return filecontent.encodeBase64().toString();
```

Download document code

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import org.bonitasoft.engine.api.APIAccessor;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

// Clean response xml document
responseDocumentBody.normalizeDocument();
// Get result node
NodeList resultList = responseDocumentBody.getElementsByTagName("return");
Element resultElement = (Element) resultList.item(0);
String msg =  "None";
//byte[] bFile = resultElement.getTextContent().bytes;
byte[] decodedBytes = resultElement.getTextContent().decodeBase64();
try {

//convert array of bytes into file
FileOutputStream fileOuputStream =
                new
FileOutputStream("K:/TOS_ESB/BonitaTest/OpenKM/"+docu11.getContentFileName());
fileOuputStream.write(decodedBytes);
fileOuputStream.close();
msg =  "DownLoad Done, file will be in the "+
"K:/TOS_ESB/BonitaTest/OpenKM/"+docu11.getContentFileName();
}catch(Exception e){
msg = e.getMessage();
//     e.printStackTrace();
}
return msg;
```

```
return "/okm:root/BonitaTest/"+docu11.getContentFileName();
```

APPINDEX B: ESB Code

tJavaRow code inside check-out job

```
//Code generated according to input schema and output schema

globalMap.put("file_token", input_row.token);
globalMap.put("file_docId", input_row.docId);
```

tJavaRow code inside upload document

```
String s = new String(input_row.content);
globalMap.put("file_token", input_row.token);
globalMap.put("file_docPath", input_row.docPath);
globalMap.put("file_content",s);
```