ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Telecom Engineering (Information Systems (TIS) Stream)

Research Thesis

Data Synchronization Solution Model for Enterprise Applications Integration; The Case of Ethio Telecom.

A Thesis Submitted to Addis Ababa Institute of Technology in Partial Fulfillment of the Degree of Master of Science in Telecom Engineering (Telecom Information System (TIS) Stream).

Werku Melesse Kenaw

Advisor: Mesfin Kifle (Ph.D.)

November 2018 GC

ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

TELECOM ENGINEERING, TELECOM INFORMATION SYSTEMS STREAM

Research thesis

On

Data Synchronization Solution Model for Enterprise Applications Integration; The Case of Ethio Telecom

By**:** Werku Melesse Kenaw

Chairman, School of Graduate Committee	Signature	Date
_____	_____	_____
Advisor	Signature	Date
Dr. Mesfin Kifle (Ph.D.)	_____	_____
Examiners	Signature	Date
Dr. Ephrem Teshale (Ph.D.)	_____	_____
Dr. Surafel Lemma (Ph.D)	_____	_____

ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY (AAIT)
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
TELECOM ENGINEERING TELECOM INFORMATION SYSTEMS STREAM

Werku Melesse Kenaw

Advisor: Dr. Mesfin Kifle (Ph.D.)

This is to certify that the thesis prepared by Werku Melesse Kenaw, in titled 'A Data Synchronization Solution Model for Enterprise Application Integration the Case of ethiotelecom.' which is submitted in partial fulfillment of the requirements for the Degree of Master of Science in Telecom Engineering (Telecom Information Systems (TIS) stream) complies with regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Name	Signature	Date

Advisor:_____

Examiner:_____

Examiner: _____

Abstract

In the rapid development of information technology, many enterprises are challenged by seamless information exchange and resource sharing requirement in a heterogeneous enterprise environment. However, service-oriented middleware technologies can provide a better solution in facilitating seamless resource sharing and information exchange among integrated enterprise applications. It is important to carefully design flexible and scalable integration architecture to create and support a well-performing enterprise system.

In this research, a Data Synchronization Solution Model (DSSM) is designed and implemented to overcome a data synchronization problem in a heterogeneous enterprise environment. The case study and the research idea are initialized based on existing real problems in ethiotelecom working environment. The company faces data synchronization problem between integrated applications. The data in the different application databases have mismatch; for example, for the same customer its service status could be active in one application and barring in another. As a solution, a data synchronization solution model is proposed based on middleware and knowledgebase. The middleware is a message processing part that receives the incoming message, processes and forwarded the message to a destination suitable data format. The knowledgebase is a database for the common data exchanged among integrated applications; which becomes a common data reference for integrated applications.

To implement and validate the data synchronization solution model we simulate a heterogeneous enterprise environment using virtual machines, with different operating systems and databases. The implementation and validation is performed based on the company's SIM card replacement business process. To simulate the message processing, we design Simple Object Access Protocol (SOAP) message with similar message structure in the company. The message processor is designed and implemented using open source WSO2 enterprise integrator tool. The validation result indicates, the practicality of DSSM hypothesis to creating consistency and synchronize data among integrated applications in a heterogeneous environment. The result also shows the features and components of DSSM are functioning per the design specified in the research document. Moreover, unlike the existing system in which the case study is based upon, the new solution is flexible for maintenance and redesign.

Keyword: Data Synchronization, Integration Middleware, EAI, SOA, WSO2.

Acknowledgment

This is the proper place to acknowledge and express my most heartfelt thanks to the people who are involved and provide support for the successful completion of my two years Master study and this final thesis.

First, I would like to acknowledge my advisor, Dr. Mesfin Kifle (Ph.D.) for his constant advice, follow-up and guidance throughout the research and this thesis documentation. I appreciate his dedication to shape and guide me through; this work would not have this shape without his advice and guidance. I would also like to extend my deepest gratitude to Dr. Surafel Lemma (Ph.D.) and Dr. Ephrem Teshale (Ph.D.) for their valuable comment and good advice.

Then I would like to thank ethiotelecom staffs specially Dereje, Tihtena, and Joseph for their document, database, and data related information and support during the thesis process. Also, I would like to thank others, I didn't mention their name but those who provide valuable information for me to understand the company's existing situation better.

At last but not list, I would like to express my deepest gratitude to my beloved family for their extreme support; my wife Elsabeth Zena for her understanding my situation and creating a conducive study environment. She has been handling all the home and kids' management in her own for the past two years so that I can focus only on my study. Also, I would like to thank my kids 'Nati' and 'Laly' for preserving themselves from distractive game and play while I am studying at home.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviations	Description
2G, 3G, 4G…	(First, second, third) Generation
A2A	Application to Application
AAIT	Addis Ababa Institute of Technology
API	Application Programming Interface
APM	Application portfolio management
B2B	Business to Business
BP	Business Process Management
BPM	Business Process Management
BSID	Business Service Identification Number
CBS	Customer Billing System
CDC	Changed Data Capture
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CPM	Customer Profile Management
CRM	Customer Relationship Management
CRUD	Create Read Update Delete
DB	Database
DCOM	Distributed Component Object Model
DSRL	Data Service Resource Language
DSSM	Data Synchronization Solution Model
EAI	Enterprise Application Integration
e-CAF	electronic Customer Accusation Form
EDR	Enterprise Data Replication
EII	Enterprise Information Integration
ER	Entity Relation
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
ETL	Extract Transform Load
eTom	Enhanced Telecom Operations Map
FOS	Free and Open Source
FTP	File Transfer Protocol

HLD	High-level Design
HR	Human Resource
IBM	International Business Machines Company
ICD	Integration Control Document
ID	Identification
IP	Internet Protocol
IPCC	Internet Protocol Call Center
IS	Information Systems
JBI	Java Business Integration
JDK	Java Development Kit
JMS	Java Message Service
JSON	JavaScript Object Notation
KB	Knowledgebase
LDAP	Lightweight Directory Access Protocol
LLD	Low-Level Design
MDM	Master Data Management
MOM	Message Oriented Middleware
MQ	Message Queue
NE	Network Element
NGBSS	Next Generation Business Support Systems
ODBC	Open Database Connectivity
OS	Operating System
OSS	Operation Support System
P2P	Point to Point
RDBMS	Relational Database Management System
REST	Representational State Transfer
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RQ	Research Question
RS	Receiving Sequence
SIM	Subscriber Identity Module
SLA	Service Level Agreement
SME	Small and Medium Enterprise
SOA	Service Oriented Architecture Integration

SOAI	Service Oriented Architecture Integration
SOAIF	Service Oriented Architecture Integration Framework
SOAP	Simple Object Access Protocol
SOAPUI	Simple Object Access Protocol-based application User Interface
SOHO	Small Office Home Office
SQL	Structured Query Language
TEP	Telecom Expansion Project
UDDI	Universal Description Discovery and Integration
URL	Uniform Resource Locator
VAS	Value Added Services
WSDL	Web Service Definition Language
WSO2	Web Service the Oxygen Company
WSO2ei	Webservice Oxygen enterprise Integrator
XML	Extensible Markup Language
XPath	Path expression to access XML
XQuery	a query in XML format with vendor-specific extension for other data format.

Chapter One: Introduction

An Enterprise tends to be a heterogeneous environment. They are composed of different systems, each with their own data structures and requirements. As companies grow, in addition to existing legacy applications new operating systems, new applications, and new databases introduced to handle the increasing volume of business activities in the company; over the years heterogeneity in terms of enterprise applications, platforms, data sources, and language occurs.

Enterprise applications are business applications used in an enterprise to provide a variety of services and to automate enterprises business activities of different departments within an organization [51] [52]. The most common enterprise applications are software packaged products that support enterprise-wide business process and used in more than one functional unit of the enterprise. These applications also deal with data-stores/ database systems to provide persistent storage. Typical examples of enterprise applications can be Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Customer Profile Management (CPM), Business Process Management (BPM), finance and human resource management systems can be used as an example. Enterprise applications may perform on their own, but they usually need to integrate seamlessly with other entries applications for data and business logic sharing in the organization.

A variety of techniques and technologies are involved to integrate and synchronize data in integrated enterprise applications. Most commonly used techniques are data consolidation which is a technique that captures data from multiple source systems and integrates it into a single persistent data store, data federation technique which provides a single virtual view of one or more data source, and data propagation technique which is a technique that copies data from one location to another in real time, changed data capture technique which is based on detecting changes in a database, extract the changed data, and send the changed data to another application[53][54]. A variety of technologies commonly used to integrate and synchronize enterprise data are Enterprise Application Integration technologies (EAI) which is used to share data and process in the enterprise, Enterprise Information Integration (EII) technologies which provides a virtual business view of dispersed enterprise data, Service Oriented Architecture technologies (SOA, web service) which manages enterprise applications and data as a loosely coupled service, Enterprise Data Replication (EDR) technologies which is mainly based on techniques and technologies used to replicate the enterprise data, Extract Transform Load (ETL) technologies extract data from source systems, transforms based on the business requirements, and loads the results into a target destination; and Middleware technologies which is based on intermediary application to integrate and synchronized enterprise data.

Enterprise Application Integration (EAI) is a solution, which is a combination of processes, software, hardware, and standards resulting in the seamless integration of two or more enterprise systems allowing them to operate as one [1]. EAI assists in unrestricted sharing of business processes and data between connected applications and/or data sources in the enterprise without making major changes to the applications or data structures. The EAI has numerous benefits; as it lets different applications exchange data and business functions. What is needed in these applications' integration is, a way for an application to seamlessly and accurately communicate with other integrated applications in a heterogeneous environment. Sometimes the enterprise applications integration is challenged by the heterogeneous enterprise data environment. To overcome these EAI challenges one should understand the main EAI components and its heterogeneous environment.

Service-oriented architecture is a collection of loosely coupled reusable services component that could be discovered through standards-based interfaces. The loosely coupled components are the core characteristics of SOA that leads to agility and flexibility. SOA allows integration of existing systems, applications and users into a flexible architecture that can easily accommodate changing needs. It provides a promising way to address problems related to the integrations of heterogeneous applications. In heterogeneous enterprise environment enterprises require a seamless data integration and synchronization among integrated systems.

Data synchronization is the process of establishing consistency among different data sources; that is from a data source to target data-storage and vice versa, including the continuous harmonization of the data over time [9]. Data synchronization refers to the idea of maintaining data integrity or keeping multiple copies of a dataset in coherence with one another. Especially when multi-vendor applications are involved, maintaining consistency among integrated applications become too difficult and may depend on the enterprise's multi-vendor management capability to harmonize the enterprise data. The same is true in ethiotelecom in which, data synchronization problem has been a critical problem for the past four years. The core problem is the heterogeneity of exchanged data between integrated multi-vendor applications. Due to the vendor's source-code privacy policy, the company experts couldn't investigate the middleware and other integrated applications working principle. Managing multi-vendors to solve the problem took more than four years but the problem still exists.

Many researchers have conducted research on enterprise data synchronization. Mihaela Iridon [14], conducted research in 2009 to produce an Enterprise Data Integration and Synchronization Solution that allows multiple enterprise domains to share data using various system models and

design artifacts. Zhitong Su and Xiaoli Rou [15] in 2010 researched the application of ESB for data synchronization. They use SOA web-service XML technology to produce a data synchronization solution based on ESB. Yubiao Wang, Xibin Wang and Xiru Rao [11] in 2012 analyzes the problem of heterogeneous database and synchronization mechanism in enterprise data integration and as a solution, they produce a heterogeneous database synchronization updates based on web-service that synchronizes updates to the source and central databases. SONG Dachuan and JING Shaohong [10] in 2016 could produce a data synchronization solution based on SQL server 2012 replication technologies which is a reliable and low-cost solution to integrate and synchronize cement enterprises branch office data to headquarter in real time. JianJun Lv and XiangYun Zheng in 2009 researched various data synchronization techniques and design a data synchronization model based on middleware and rule-based to implement data synchronize among various databases [9].

However, these research works could not be used for ethiotelecom environment due to different limitations of the research; for example, [9] focused on creating consistency on databases and data stores; but does not mention the problems created by the integrated applications data mismatch. Other related work [57] is implemented based on a peer-to-peer model which has a known limitation of high complexity and scalability in large enterprises environment. Unlike the ethiotelecom environment, [10] the other research work is conducted in a homogenous enterprise environment. Therefore, a new Data Synchronization Solution Model (DSSM) based on ethiotelecom existing enterprise environment is proposed and the research process to produce the proposed solution model is documented in this research document.

1.1 Background

In this research, ethiotelecom's enterprise applications working environment is used as an initial idea source and reference. ethiotelecom is a government-owned integrated telecommunications services provider in Ethiopia, which provides message, data, voice, and telecom infrastructure services. It is one of the "Big-5" state-owned public enterprises in Ethiopia [23] and the largest mobile operator in Africa with more than 67.5 million mobile subscribers in one country [37].

Telecom Enhancement Project phase one (TEP1) in ethiotelecom has been implemented for NGBSS systems in 2014. In this project-contract, large-scale enterprise applications and its integration to applications within NGBSS and other legacy and third-party systems is included.

As per [4] the ethiotelecom's Application Portfolio Management (APM) in October 2016, there are around 180 applications and are categorized into business applications, support applications,

corporate applications, and office tools. Business applications are used to provide services to the customers, corporate applications automate the company's office tasks, and support and office tools are used for supporting and monitoring the company's business and corporate applications. Regardless of the technology, language and platform difference, these applications are required to integrate and work in a seamless manner per the company's business needs to integrate.

Applications can reside within the enterprise boundaries which are called intra-organizational applications or outside of the enterprises boundary in other company's boundary connected over network or internet which is inter-organization applications. When integration looked from these two perspectives, there are two types of integration, the first is Application-to-Application (A2A) integration or generally called Enterprise Application Integration (EAI) which is the integration of applications and business processes within an enterprise. And the second is Business-to-Business (B2B) integration which is an integration of two or more enterprises' systems and their business processes as shown in Figure1 below. In ethiotelecom, both A2A and B2B integrations types are applied within the organization and across in some business partners like Bank.

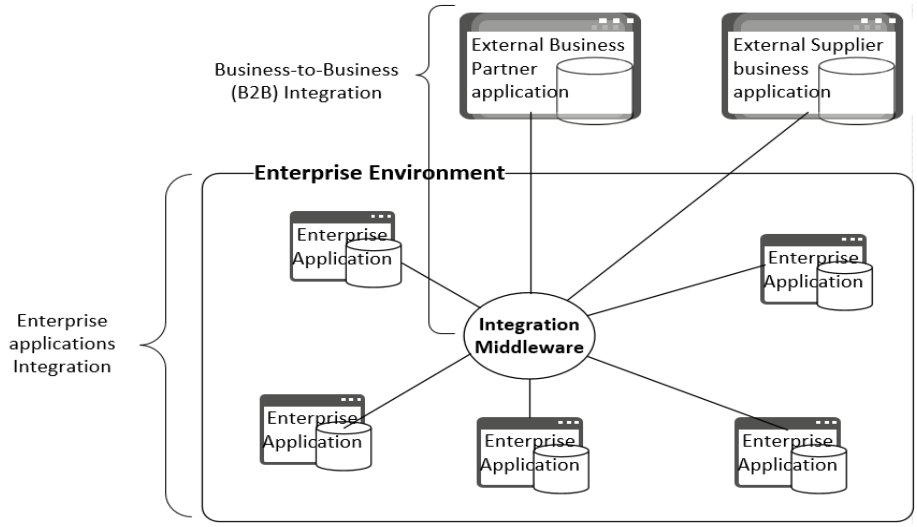


Figure 1. Types of Application Integration (Based on concepts from [3])

though we couldn't find any recorded document about when enterprise applications integration is started in ethiotelecom, it's known that enterprise applications have been working integrated before the Telecom Expansion Project phase one (TEP1) implemented in 2014 [21]. While implementing TEP1 projects for IS division, the vendor who took the responsibility of implementing NGBSS systems and its integration with other third-party applications following eTom standard, and the application integration is based on SOA design principles (SOA integration framework). In its implementation, the vendor uses IBM WebSphere ESB as a

middleware for applications integration within NGBBSS solution and its integration with other third-party and legacy applications. The middleware used is a significant and fundamental business integration framework which performs business process orchestration, data transformation, and transportation among various applications.

1.2 Motivation

After Telecom Enhancement Project phase one (TEP1) is implemented in ethiotelecom in 2014 enterprise application's data synchronization problem has been a loud problem in the company; it affects the company's operational activity heavily. To keep the company's business continuity, other workarounds have been tried out to minimize the problems, but the problems still exist.

According to the seminar report on 'assessment of EAI Problems in ethiotelecom' [2], challenges and pitfalls are identified; such as, data synchronization and data inconsistency between integrated systems, multi-vendor system interoperability problem, quality of integration, missed functionality between integrated systems, and multi-vendor management problems are mentioned in the report. Among these enterprise application integration issues, data synchronization is widely seen in different business units. It exists in business applications, corporate applications, and support applications integration. The main causes of data synchronization problem is the data inconsistency between integrated applications and their datastores, problem in the integration middleware and its implementation, the migrated data quality and problem in some business process are identified in the seminar report.

The data inconsistency among integrated applications affects the company's telecom service delivery and contributed a bad reputation in the customers' experience. For example, a customer just paid his bill, his subscription status would be changed to 'active' following his bill payment; but if the two systems couldn't synchronize, his status may remain suspended in the billing system. Therefore, due to the status mismatch between Customer Relationship Management (CRM) and Convergent Billing System (CBS), the customer will not get the telecom service he supposed to get. Then based on the customers complain usually the service status would be activated manually by operational staffs.

Data mismatch, data type, syntax and format mismatch between integrated applications are also other issues that inhibit data synchronization between integrated applications.

These real data synchronization problem scenarios that suffer and costs the company community a lot for a long time is the motivation behind this research.

1.3 Statement of the Problem

The enterprise applications developed in different projects with different vendors are heterogonous and scattered. This heterogeneity in data type, format, technology, and language difference creates data inconsistency among scattered application databases. This intern inhibits data synchronization among business and mission-critical enterprise applications.

The data inconsistency between integrated applications impacts other enterprise systems data exchange. While reconciling reports, the reports generated from different integrated systems sometimes found inconsistent; on the other hand, Customers are not getting the service per the Service Level Agreement (SLA) on the citizens charter, the company is losing revenue, and staff are suffering to trace the data synchronization problem in integrated enterprise applications boundary.

Mihaela Iridon[14], in 2009 produced an Enterprise Data Integration and Synchronization Solution that allows multiple enterprise domains to share data using various system models and design artifacts. In 2009 JianJun Lv and XiangYun Zheng [9] also researched various data synchronization techniques and design a data synchronization model based on middleware and rule-based to implement data synchronize among various databases. In 2010, Zhitong Su and Xiaoli Rou [15] researched the application of ESB for data synchronization using SOA web-service XML technology.

in 2012 Yubiao Wang, Xibin Wang and Xiru Rao [11] analyzed the problem of heterogeneous database and synchronization mechanism in enterprise data integration and produce a heterogeneous database synchronization updates based on web-service that could synchronize updates to the source and central databases. And in 2016 SONG Dachuan and JING Shaohong [10] based on SQL server 2012 replication technologies produce a data synchronization solution that could integrate and synchronize cement enterprises branch office data to headquarter in real time.

However, data synchronization solutions provided by other researchers in the reviewed literature have limitation to be used as a solution for the above-mentioned problems in ethiotelecom case. For example, reviewed literature [10] focused on creating consistency on databases and data stores; but doesn't not mention/not consider the problems created by the integrated applications data mismatch. Other related work [57] is implemented based on a peer-to-peer model which has a known limitation of high complexity and scalability in large enterprises integration

environment. Unlike the ethiotelecom environment, the other research work [9] is conducted in a homogenous enterprise environment. None of the above-mentioned literature could be used as a solution set for ethiotelecom's specific data synchronization problems. Therefore, this data synchronization problem should get a solution which is less complex and feasible. This research would try to address specifically the following research question (RQ).

RQ. Regardless of the heterogeneous enterprise environment, what data synchronization solution can be provided to create data consistency among integrated enterprise applications; so that, the data can be synchronized seamlessly keeping existing enterprise applications in the environment.

Thus, this research is proposed to fill the above-mentioned gaps and to produce a data synchronization solution model for integrated enterprise applications for the case of ethiotelecom.

1.4 Research Objective

1.4.1 General Objective

The general objective of this research is to provide a service-oriented data synchronization solution model, keeping the existing integration environment in the enterprise.

1.4.2 Specific Objective

The specific objectives of the research are:

- To understand the enterprise applications and their integration environment by reviewing system implementation documents and system manuals.
- To understand state-of-the-art around enterprise application integration with a particular focus on data synchronization and data inconsistency problems.
- To assess approaches associated with EAI data synchronization problem resolution.
- To design a data synchronization solution model to be implemented as an additional data synchronization layer in existing EAI architecture.
- To define modular loosely-coupled services that could be encapsulated within the data synchronization solution model.
- To construct a structured Knowledgebase repository for exchange data in the integration.
- To evaluate/validate the proposed solution model practicality by simulating a heterogynous enterprise environment.

1.5 Research Method

This research is conducted in three phases. In the first phase, different literatures, company documents, and related works are reviewed to understand and clarify concepts in the boundaries of the research. Key informant in-depth interview is also conducted, and the collected data is analyzed. To gather supportive information from the industry, event and error logs are collected and analyzed. In the second phase, an appropriate solution model is designed based on SOA design principles [Section 2.6.1]. In the third phase, the designed architectural model is checked for conceptual validity. To validate the new solution model a showcase prototype is developed by simulating a heterogeneous enterprise environment; and the solution is validated for its practicality.

In addition to the computer and hardware, the below tabulated applications and tools are used to simulate a heterogenous enterprise environment and to design, implement and validate the proposed solution model in this research.

Table 1. Tools used for the research.

Tools	Type and Version	Purpose
Operating Systems	Windows Server 2012R2, Windows 10,	To simulate a heterogeneous platform.
Databases	Oracle 11g, MySQL and SQL	Are datastores for different applications participated in the integration.
JDBC drivers	JDBC Oracle, JDBC MySQL, JDBC SQL driver	To integrate the databases to the data service of the DSSM solution.
Web browsers	Google Chrome, Mozilla Firefox, internet explorer.	To access the WSO2 management console.
Other applications	WSO2 enterprise integrator (WSO2ei)	For Data Synchronization Solution Model (DSSM) message processing and Data service implementation.
	SOAPUI	To create request message and response message from the WSDL file and to create a mock service.
	Eclipse 6.2.0	To create services WSDL file.
	Oracle JDK 1.8	JDK 1.8 or above is a platform to run WSO2 carbon-based applications and to run apache ant and apache maven.
	Apache Ant 3.0	Prerequisite to install WSO2ei
	Apache maven 1.7.0	Prerequisite to install WSO2ei
	Microsoft Visio 2013	For drawing different diagrams
	Edraw Max 9.0	For drawing different diagrams

1.6 Scope and Limitations of the Research

1.6.1 Scope of the Research

The Scope of this research is defined based on scenarios, requirements, and boundaries described by Subscriber Identity Module (SIM) card replacement business process lifecycle in ethiotelecom [Section 4.2]. It is focused on the scenarios in which the SIM card sales and activation process lifecycle encompassed applications such as, e-CAF, CRM and CBS and based on the English language only.

This research does not consider the data synchronization problem due to quality of the migrated old data in the applications database in the company; which is caused by previous data migration process. Rather, this problem can be solved through a well-planned data cleaning process.

1.6.2 Limitation of the Research

The research focuses on the practicality of the proposed solution; it doesn't deeply consider the security and performance concern of the proposed solution. it is conducted considering enterprise applications that use only English language and the use of other local languages is not considered.

1.7 Significance of the Research

In ethiotelecom due to the vendor's privacy policy in a multi-vendor environment, the experts couldn't make any change for any known problems; rather they would be engaged calling and managing multiple vendors for a meeting to solve a particular problem. Due to vendors pushing tasks not to take responsibility, vendor management is too difficult and time-consuming. This research tries to provide a data synchronization middleware solution by keeping the existing applications, infrastructure and even the vendor privacy policy in the company. The successful implementation of this research would show the professional community the possibility of providing solution for a particular problem even with all the constraints like the vendor's privacy policy or restrictions to modify existing enterprise environment.

In addition to synchronizing the enterprise data in the integration environment, once the data synchronization layer is established adding modules like security policy, business rules, dynamic recommendation systems, and other loosely-coupled services would be easy. Insertion of these additional services can be done with little or no modification to existing enterprise applications.

1.8 Document Organization

The DSSM research documentation is organized hierarchically in a way that discusses from the general concept of the research and its background information to a specific implementation and validation scenarios. The rest of the Chapters in this research document is organized as follows.

Chapter Two discuss the theoretical background and detail concepts of the research based on reviewed literature. In this Chapter, the fundamental concepts of enterprise applications, Enterprise Applications Integration (EAI), enterprise applications data synchronization, Service Oriented Architecture (SOA) and related concepts are discussed in detail. In Chapter Three other researchers related works is discussed, the methodology used, solution provided, research validation and limitations of the related works is discussed.

Chapter Four discusses the situational analysis of the company in which this research is based upon. Studying the company's existing situation is very important and basic activity to propose a solution based on the findings in the existing situation. Therefore, existing situational analysis of the company is discussed in Chapter four. Whereas in Chapter Five, the design and design considerations of the proposed DSSM is explained together with its architecture and implementation design.

In Chapter Six, the Implementation and Validation of the proposed DSSM is explained. Section 6.2 describes the proposed solution implementation and configuration. Section 6.3 discusses the validation of the proposed solution. Chapter Seven is the last Chapter of the research document. It contains the validation result, discussion, and interpretation of the DSSM; it also explains the Limitation of the research and a concluding remark about the DSSM research; at last Recommendation and Future work is also discussed.

Chapter Two: Literature Review

In this Chapter, the fundamental and building block concepts of this research work are discussed. It is organized into five main topics that discuss enterprise applications, enterprise application integration, Service Oriented Architecture, Enterprise applications data synchronization, and integration components. The subtopics and sub-subtopics of this Chapter also discuss granularly deeper details about their corresponding topic. These concepts are important, and millstone notes for the construction of the proposed solution which would be discussed in detail in Chapter Four.

2.1 Enterprise Applications

Enterprises are typically comprised of hundreds of applications (custom-built, acquired from a third-party, legacy system applications, or a combination of these) [10]. These applications are developed by different people using different programing language, that operates in different operating system platforms and uses different technologies. These Enterprise applications are software packaged applications that support the integration of business process and data across the boundaries of business functions at both intra-organizational and inter-organizational levels [8] [9] [10]. Due to the heterogeneity of the programming language, the diversity of the applications platform and incompatible application interfaces, enterprise applications cannot communicate each another in terms of their data sharing and/or business logic.

Spreading business functions across multiple applications provides Enterprise with the flexibility to select the best of bread applications that best suits the business' needs [10]. But to be effective organizations require a methodology that enables these applications to integrate and work together. To share data and business logic these independently working enterprise applications need to coordinate and integrate in a loosely coupled way; so that to address the constantly evolving business needs in the organization and to achieve their business goal.

2.2 Data Integration and Synchronization

a wide variety of techniques and technologies are involved to integrate and synchronize enterprise data in EAI environment. however, most commonly used techniques are data consolidation, data federation, and data propagation technique [53] [54]. These three techniques in-turn uses changed data capture and data transformation techniques during data integration and synchronization. The common technologies used to integrate and synchronize enterprise data are EAI, EII, SOA, EDR, ETL, and Middleware technologies.

2.2.1 Data Integration Techniques

There are a variety of data integration and synchronization techniques in EAI environment. But the following three are the main techniques used in most data integration environment; these are data consolidation, data federation, and data propagation techniques.

Data consolidation: Data Consolidation is a technique that captures data from multiple data source and integrates it into a single persistent and consistent data store. In this technique, there is usually a latency or delay between the time updates occur in data source systems and time the updates appear in the target system. Depending on business needs, this latency may be a few seconds, several hours, or many days. Data with zero latency is known as real-time data, but this is difficult to achieve using data consolidation.

Target data stores that contain high-latency data are built using batch data integration applications that pull data from the source systems at scheduled time intervals. Low-latency target data stores, on the other hand, are updated by online data integration applications that continuously capture and push data changes to the target store from data source systems. This push approach requires the data consolidation application to identify the data changes to be captured for data consolidation. Changed Data Capture (CDC) technique is usually used to do this.

The advantage of data consolidation is, it allows large volumes of data to be transformed as it flows from source systems to target data store so that it could consistently synchronized with the target system. The disadvantages are the computing resources required to support the data consolidation process and the amount of disk space required to support the target data store.

Data federation: Data Federation provides a single virtual view of one or more data sources. When a business application issues a query against this virtual view, a data federation engine retrieves data from the appropriate data source, integrate it to consistently match the virtual view and query defined, and sends the results to the requesting business application. Data federation always pulls data from source systems on an on-demand basis. Any required data transformation is done as the data is retrieved from the source data files.

The main advantages of a federated approach are, it provides access to current data and removes the need to consolidate source data into another data store. Data federation, however, is not well suited for retrieving and reconciling large amounts of data or for applications where there are significant data quality problems in the source data. Another consideration is the potential performance impact and overhead of accessing multiple data sources at runtime.

Data propagation: Data Propagation applications copy data from one location to another. These applications are event-driven and usually, operate in real time and push data to the target location. Updates to a source system may be propagated synchronously or asynchronously to the target system. In any case, propagation synchronizes by guarantying the delivery to the target system. This guarantee delivery is a key distinguishing feature of data propagation. Most synchronous data propagation technologies support two-way data exchange between source and target data. Enterprise application integration and enterprise data replication are examples of technologies that support data propagation.

The advantage of data propagation is, it can be used for real-time data or near-real-time data movement. Other benefits include guaranteed delivery and two-way data propagation and synchronization.

Hybrid approach: data integration techniques used by integration applications depend on both technology and business requirements. It is common in data integration application to use hybrid data integration techniques. For example, if a customer updates his or her name and address during a Web store transaction, this change could be sent to the consolidated data store and then propagated to other source systems such as a retail store customer database so that all databases in the integration keep synchronized.

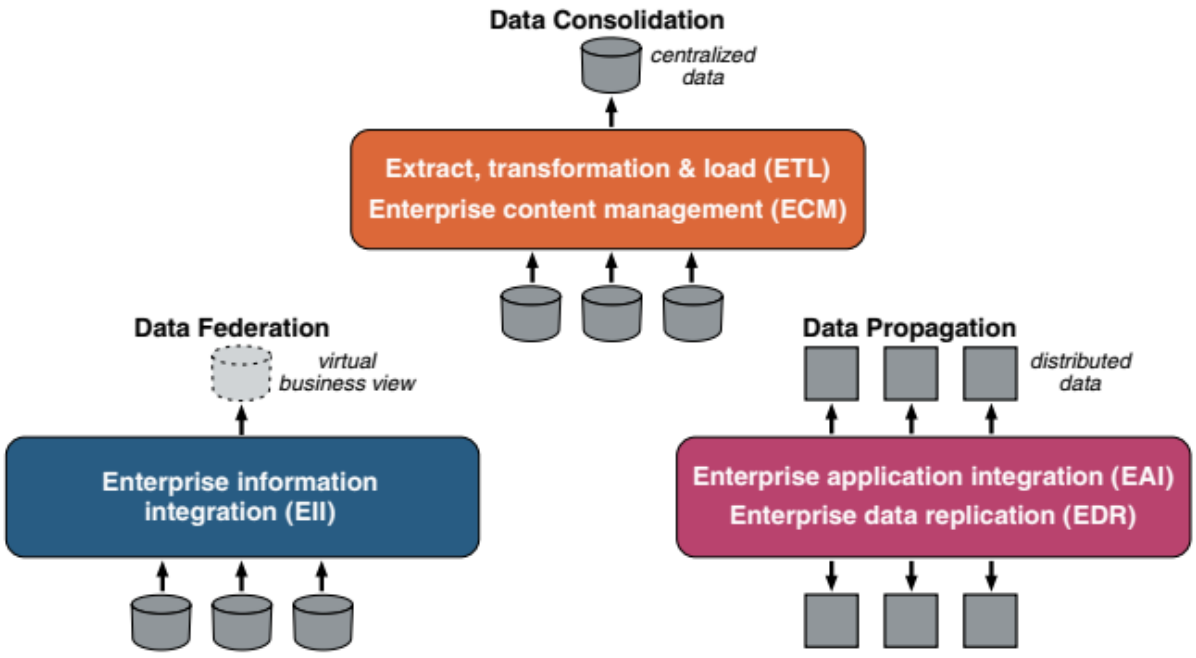


Figure 2. Data integration techniques: Consolidation, Federation, and Propagation [53]

2.2.2 Data Integration Technologies

A variety of Data integration and synchronization technologies are available for implementing data integration and synchronization techniques introduced in Section 1.1 above. Some of the common technologies include EAI technologies, EII Technologies, SOA technologies, EDR technologies, ETL technologies and Middleware technologies are widely used.

Extract Transform Lod (ETL)

As the name implies, ETL technology extracts data from source systems, transforms it to satisfy business requirements, and loads the results into a target destination. Data can be extracted in schedule-driven pull mode or event-driven push mode. Both modes can take advantage of changed data capture. Pull mode operation supports data consolidation and is typically done in batch. Push mode operation synchronizes the data online by propagating data changes to the target data store.

Enterprise Information Integration (EII)

EII provides a virtual business view of dispersed data. This view can be used for demand-driven query access to operational business transaction data. EII supports a data federation approach to data integration. The objective of EII is to provide applications view of the dispersed data as if it resided in a single database. EII can shields applications from the complexities of retrieving data from multiple locations/sources, where the data may differ in semantics and formats and it may employ different data interfaces. EII products have evolved from two different technologies relational DBMS and XML. The trend of the industry, however, is toward products supporting both SQL (ODBC and JDBC) and XML (XQuery and XPath) data interfaces.

Enterprise Data Replication (EDR)

Data replication is the process of generating, reproducing, and maintaining a defined set of data in one or more locations [53] [58]. it supports both the data propagation and changed data capture techniques to data integration. EDR is used for data integration, backup and recovery, data mirroring and workload balancing scenarios. EDR tools vary in their capabilities. Replication tools often employ database triggers and recovery logs to capture source data changes and propagate it to one or more remote databases so that the databases have consistent data. All major relational DBMS vendors provide data synchronization and replication capabilities and Several others also allow data to be transformed as it flows between databases. Also, companies offering CDC solutions often employ data replication. Some EDR products, however, support two-way data synchronous propagation between multiple databases.

In terms of communication and timeliness, the replication process can be either synchronous or asynchronous. The relationship between databases within the replication process can be a master-slave relation, which is only one primary copy of the data can be accessed, and the rest of the synchronized copies are only readable. On the other hand, in multi-master replication systems, more than one copy of the replication can be updated, and updates on any of the copies are synchronized to all replications.

2.3 Enterprise Application Integration

Enterprise Application Integration (EAI) is a strategic approach to bind many information systems together both at the service and information levels [57]. EAI plays an important role in linking enterprise applications to share data and business logic. It also helps to centralize business policies and rules, to encourage vendor independence among participated applications in the integration. EAI is not a product or a specific integration framework, but a combination of processes, software, standards, and hardware that allow for the end-to-end integration of several enterprise applications and enable them to appear as a single system [42].

2.3.1 Enterprise Application Integration Layers

All integration platforms exhibit layered structures (Section 2.4.5); For managing complexity and assignment of functionalities, use of conceptual layers appear as a common principle [24]. In the traditional EAI layered architecture, there is no global consciences or global definition about EAI layers, some literature defines the EAI architectural layers as application layer, middleware layer and data layer [3][57]; another literature define it as service consumer layer, Integration layer, and service Provider layer [25]. Some other literature defines it as a four-layer architecture as User interface layer, application layer, integration layer and data/resource layer [8]. But in this research to better consistently describe EAI the authors prefer to consider EAI as a three-layered architecture as the Application layer, Middleware layer, and Data layer. Figure 3 below shows the traditional EAI three-layer architecture.

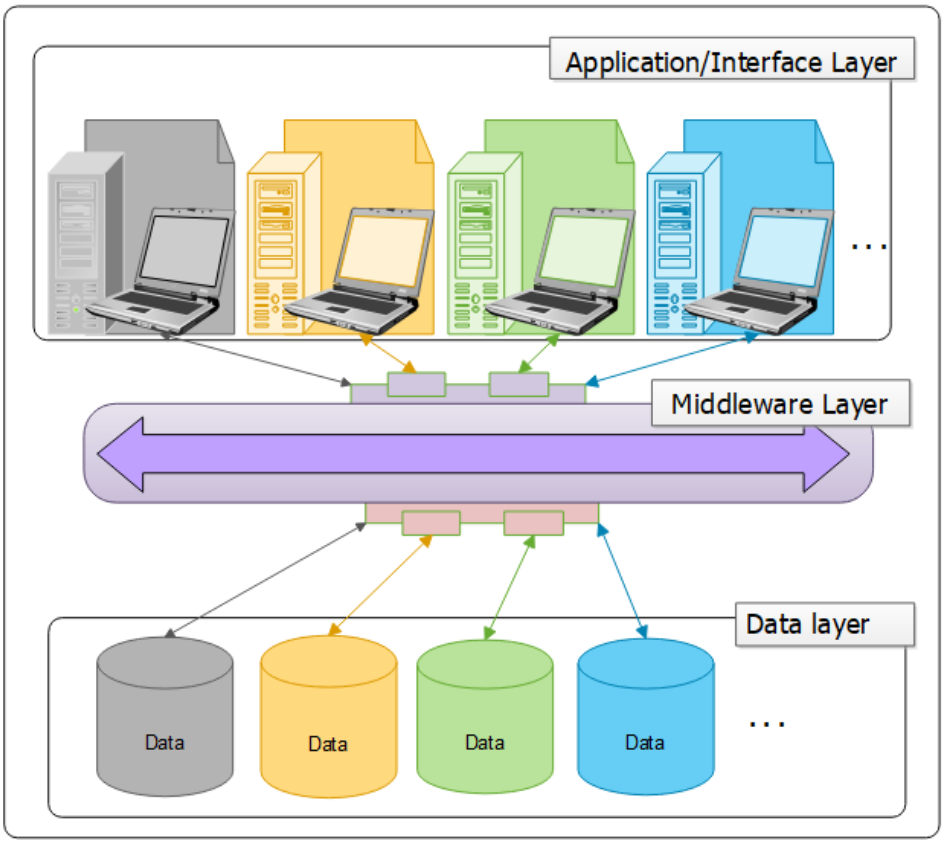


Figure 3. Existing integration layered architecture (based on [3]).

Application layer refers to a variety of enterprise applications that communicate each other through middleware. The middle layer provides access to the data for all applications' request participated in the integration. Data layer refers to a variety of resources and data stored in the databases

2.3.2 Levels of Integration

There are three or four integration levels, according to David S. Linthicum [49] there are three levels of integration; Data level, object level, and process level. Whereas H. Barki and A. Pinsonnault [48] identifies four main levels of integration; data level, application level, method level, and user interface level.

Data level integration: In data-level integration, the data store is the primary point of integration. It is a low-level integration in which knowledge of the application logic not required. The applications share only data, but they don't need to know each other and no need to change application logic.

Application-level integration: In application level integration, applications' functionality is used by another application through the interfaces exposed. The interfaces may allow sharing of business processes, data or both.

Business process/method level: Method-level Integration allows business logic sharing among integrated applications by centralizing methods that provide common business logic that could be accessed by other applications as well.

User interface level: User interface level integration is sometimes referred as presentation level integration. It is used to integrate applications through their interfaces. Two main approaches commonly used at this level are screen scraping and content aggregation. Screen scraping is usually used for mainframe applications that could not be accessed at data or method level; whereas content aggregation is used for collecting content from various applications and presenting it through the common portal interface.

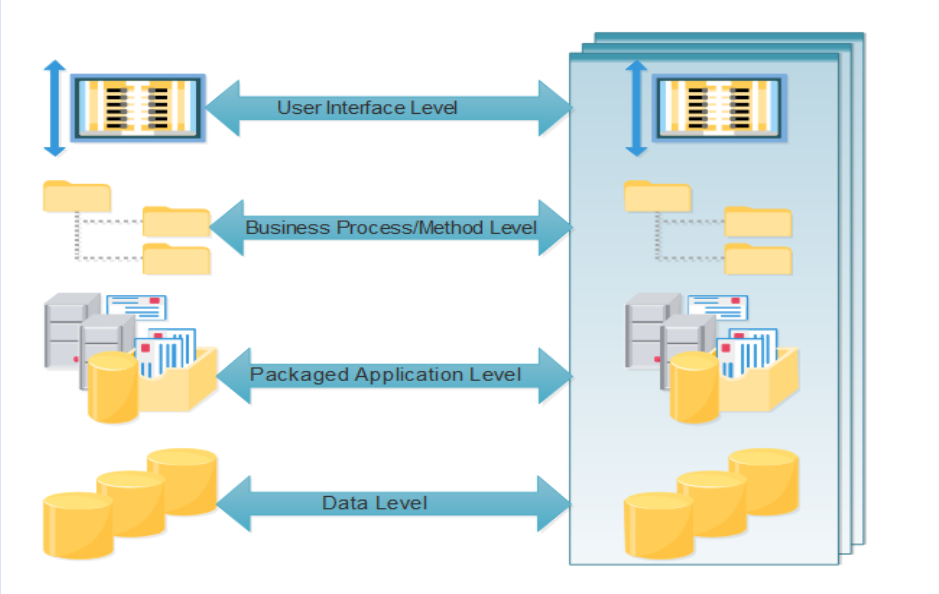


Figure 4. Levels of Integration (based on [47])

2.3.3 Enterprise Application Integration Architecture

Based on its design architecture there are Four common EAI architectures; Point-to-Point (P2P) architecture, Hub-Spoke architecture, Bus architecture and Service Oriented Architecture (SOA).

Point-to-point (P2P) Architecture: in this architecture integration is the simplest way of integrating two independent applications. In this architecture to be fully integrated each application is required to directly connect to all participating applications. As the number of

applications to be integrated increase complexity of integration also increases. That is for n number of applications there would be n (n-1)/2 numbers of connections would be required. To communicate in point-to-point architecture for each application a connector which is responsible for data transformation would be implemented [50]. In point to point architecture, the communication is fast. It could be better and easily build for small infrastructure. It is also suitable for integrating a few applications in small organizations. But integrated applications are tightly coupled. As the number of applications required to integrate into a point to point application increase complexity of integration and maintenance costs also increase.

Hub and Spoke Architecture: a centralized broker called Hub and adapters commonly called Spokes are the building blocks of the architecture. The Spokes connects the applications to the central Hub and could converts application data to the format that the central Hub can understand and vice versa. The role of the Hub in this architecture is brokering, transforming, translating and routing messages and message contents depending on the need to the format that the receiving application can understand. Hub and spoke architecture lack standards and the centralized broker model can be considered the single point of failure [41].

Bus architecture (Pipeline architecture): in this architecture, independent systems along the value-added chain are integrated through a message bus. The bus architecture capability gives applications a local access to the bus interfaces. It is highly scalable lightweight and reliable integration architecture; it can be designed with minimum coding and with minimum or no modification to the integrated applications. The integrated applications and other components can be hosted in the network. Bus architecture solves the scalability issues of the point-to-point and the hub-spoke architecture; bus architecture can be horizontally scaled as a centralized messaging. The bus system is responsible for message distribution. The transformation and routing function rules are stored in the hub central repository [15]. Depending on the middleware product, business functions and rules also be represented in the middleware used.

Service-oriented architecture (SOA): is an architectural approach in which software resources and well-defined services in an enterprise are available and discoverable through network. In SOA business processes and applications' data are orchestrated as independent service calls in which the services are orchestrated through the Enterprise Service Bus (ESB) (Section 2.4.5.1). An ESB is one of the central integrations and communication components for service calls [25] [26]. It comprises of essential information about IT related elements and their interrelations within the company business process and data.

2.3.4 Application Integration Technology

The common objective in EAI is business process and data sharing between integrated applications. To share data and business process between integrated application Middleware technologies plays a major role in minimizing the pain of integrating enterprise applications. To exchange data, applications need to understand the format and structure of the messages being communicated. In a simple format, applications communicate through Applications Programming Interfaces (APIs); as integration scenarios go beyond simple API calls' more technical features would be needed. Integration technologies, such as middleware, are needed to handle technical aspects like message transactions handling, asynchronous communication, dynamic binding, loose coupling, and messaging patterns [3] [49]. The first generation of middleware technologies that were employed to address EAI problem was technologies like Remote Procedure Call (RPC) and Message Queuing (MQ). These technologies are still in use today, but they address only parts of the EAI problem domain. As Middleware technology advances more recent technologies like Java Message Service (JMS), Remote Method Invocation (RMI), Component Object Model (COM), Distributed Component Object Model (DCOM), Common Object Request Broker Architecture (CORBA) come to play.

2.3.5 Integration Platforms

Integration platform is a set of technologies used to implement bi-directional enterprise applications integration. There are many integration platforms in the market for EAI implementation. Among the common key integration platforms that facilitate EAI, Enterprise Service Bus (ESB), Message Oriented Middleware (MOM) and Integration Broker are the main enterprise applications integration platforms.

Enterprise Service Bus

An enterprise service bus (ESB) is a middleware solution that enables interoperability among integrated enterprise applications using a service-oriented model [14]. It is a software architecture that enables intercommunication among various integrated applications. Through its high scalability feature, ESB enables businesses to connect any application and synchronize data across numerous integrated systems and services to ensure consistency and keep data synchronized across the enterprise. Integration components in the ESB are connected through the bus as logical endpoints that are exposed as event-driven services but abstractly decoupled from each other.

One advantage of connecting services through the enterprise service bus is that clients only need to look for services in a single location. If a client service is moved, it is required only to reconfigure the endpoints of the ESB. The clients still can access the service through the ESB. Since ESB is highly scalable in-terms-of adding new components into the integration, adding a new application is also simple.

There are two types of ESB products Commercial and Open source; Commercial ESB Products are proprietary software products with usage-based licensee fee for the vendor and source code is not available. These products are expensive; some common ESB commercial products are IBM's WebSphere ESB, Tibco's ActiveMatrix ESB, Oracle's Oracle ESB, and Microsoft's BizTalk ESB can be mentioned. Open source ESBs do not have a usage-based license fee and the source code is freely available. In addition, services and support is available free for open source ESBs. Some common open source ESBs are WSO2 ESB, Mule ESB Community, Apache ServiceMix, JBoss ESB.

ESB provides various functionalities to facilitate the integration between applications through different hardware and software platforms. The following are the major functionalities and services provided by most ESBs:

Transformation: Applications may use different message structures and data formats to represent real-world concepts. Thus, ESB as an intermediary supporting different message structures, formats, protocol, and data translates from one format to another to allow greater flexibility and scalability for integrated applications and systems.

Transport mediation: integrated systems uses various transport protocols; hence, applications integration requires efficient mediation among the various transport protocols. **Intelligent routing**: ESB should have the capability of routing messages based on their address, priorities, contents, rules, and logic while load balancing between available service providers. To perform message routing the ESB requires to have some level of semantic understanding of the services.

Service Mapping: Service mapping is the ability of an ESB to translate a business service into another service implementation and provide location information and binding. Usually, service mapping contains core information like implemented service name, binding information, service-specific routing information, service protocol and, protocol specific information. It could be implemented through a database, an XML, or can be embedded within the mediator ESB components.

Service Orchestration: Service orchestration is the ability to managing the coordination of multiple service implementation

Security: Only authenticated and authorized service users should be capable of accessing and consuming service and messages containing sensitive information may need to be encrypted at runtime.

Message Oriented Middleware

Message Oriented Middleware (MOM) is a middleware that provides connectivity to integrated applications for program-to-program communications through message passing. A MOM is acting as a message mediator between message sender and message receiver [43]. Messages are addressed and router to receivers that subscribe to the messages. There is no requirement for sender and receiver applications to be connected to each other in a MOM to send and receive messages. A sender doesn't know the existence of the receiver; This allows loose coupling between sender and receiver applications.

Integration Broker

Integration broker or sometimes called message broker built for complex EAI architecture is an intermediary technology that facilitates interactions between integrated applications. Integration brokers provide at least message translation, transformation, and routing services. In the broker architecture, each integrated application has only one integration point connected to the broker. The broker then handles the message processing and routing before forwarding the message to receiver applications interested in the message. The main purpose of Integration Broker is to enable business processes to be fully automated between supplier, customer and the organization [42]. The leading integration brokers packages in the market include IBM MQSeries Integrator, webMethods, SeeBeyond, and Java Message Service (JMS).

2.4 Integration Middleware

Linthicum defines Integration middleware as 'any type of software that facilitates communication between two or more software systems [3].' Middleware can be considered as the central component in complex application integration that facilitates data and business logic sharing between integrated applications. It controls the connection between applications through remote procedure calls, handle data transactions, data interactions, and maintain message queues. Middleware focuses on integrating applications inside the company however modern middleware considers also the aspects of external integration business to business (B2B).

Middleware employs two types of communication mechanisms, synchronous and asynchronous. In synchronous communication a calling application wait for remote application respond; this creates a tight coupling between integrated applications. Whereas in asynchronous communication, the sending application doesn't wait for the receiving application to respond; the middleware is decoupled from the applications it works like 'fire and forgets' principle; therefore, will continue processing other requests until the receiving end responds. Generally, it is recommended that tightly coupled point-to-point connections should be avoided between integrated applications due to the complexity and expensiveness maintaining it.

2.5 Service Oriented Architecture

Service-oriented architecture (SOA) is a software systems architecture that addresses the requirements of loosely coupled, modular, standardized platform and protocol independent services [39] [41] [44]. In SOA Services are independent basic building block entities that expose data and functionalities as loosely coupled service that are designed for business function implementation. Loosely coupled Applications services can be integrated and executed independently. These applications are integrated from different independently developed modules.

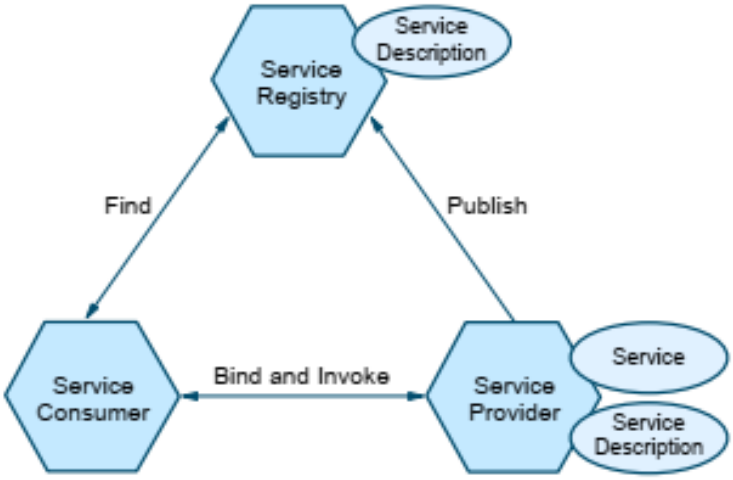


Figure 5. rolls in SOA (source oracle documentation) [based on 36]

As shown in Figure 5 service-oriented architecture is consisted of three roles; the service provider role, the service consumer role and the service registry/service broker role [36].

Service provider: The service provider is an entity that accepts and executes service requests from consumers. It publishes its interface contract and services to service registry so that the consumer can discover and access.

Service consumer: The service consumer is a software module, an application, or another service that requires a service. The service consumer initiates enquiring service in the registry, binds to a service over a transport, and executes service functions. It executes services based on the interface contract.

Service registry: A service registry is responsible for service discovery. It is a repository that contains a list of available services and allows for service lookup of the service provider interfaces to interested service consumers.

There are also three operations in a service-oriented architecture. The find, bind, and excite/invoke operations.

Publish: To be discovered and invoked by a service consumer, a service description must be published.

Find: a query by a service requester to locate services in the service registry.

Bind and invoke: once the service description is retrieved, the service consumer can continue invoking the service based on the service information in the description.

2.5.1 SOA Design Principles

The core of the Service-oriented architecture (SOA) is exposing and representing various unit of business logic and solution components as a service [40]. SOA design principles are generalized and accepted industry practices to provide rules and guidelines that determine how solution logic should be shaped and decomposed into services.

The service-orientation paradigm advocates the following nine distinct design principles, each of which supports fundamental design characteristics that form the target solution logic as service-oriented [16] [17]. A brief description of the fundamental SOA design principles is discussed below. The solution model in this research is designed with the consideration of these design principles.

Standardized service contract: All service description, communication protocol, purpose, and message format should be documented in a standard based service description format service contract.

Loose coupling: Loose coupling emphasizes that services should be designed to have minimal dependencies on each other by achieving logical separation of concerns [17]. Services shall not be tightly coupled. While designing services from the service contract to service implementation different layers of service loose coupling should be achieved.

Service abstraction: Services encapsulate the logic they provide from the outside world avoiding the proliferation of unnecessary service information internal implementation, technology, logic, and function away from users of the services. This helps greatly in developing applications without the need to review and analyze unimportant details about the system.

Service reusability: Service reusability emphasizes the potential of designing components of a solution, intentionally and independently designed services to use again and again.

Service autonomy: Service autonomy ensures two primary benefits, system reliability, and behavior predictability. The service benefit is only acquired by the service implementation; thus, no service is controlled by another service.

Service statelessness: Service statelessness refers to incorporating state management approach within service designs. It is important to keep track of the states of the history of the service.

Service discoverability: Service discoverability implies the discovery of services by applications in a systematic way. It is about the service registry that stores the service description records.

Service composability: Services are designed to be reconfigured easily based on new requirements so that can be used in different applications.

Service interoperability: Service interoperability promotes the usage of standardized service so that the services can be reusable in different platforms. Thus, special care should be given to the use of message formats and transport protocols while designing the services.

2.5.2 Web Services

The motivation behind Web Services is to facilitate businesses to interact and integrate with other businesses and clients. Without having to go through integration design or to expose its confidential internal application details; integration is possible through web service [24]. Web service is an implementation of SOA, in SOA the two software that communicates each other is the service provider software and the service consumer software. The service provider sends its service request to the service provider and the service consumer sends the service response back to the consumer.

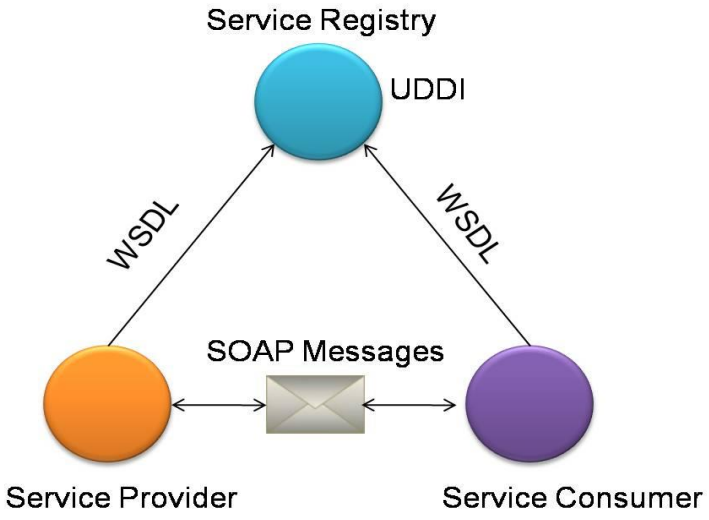


Figure 6. Web service overview (based on [18])

Figure 6 shows, the main characteristics of web-service architecture; all service provider software publishes its service description in a certain directory. The service consumer makes queries against this directory to find out what services are available, and how to communicate to the provider. WSDL is an industry standard language to create service descriptions and placed it in a directory. Simple Object Access Protocol (SOAP) is an industry standard protocol to talk to WSDL directories. The service provider will communicate with the directory using SOAP protocol in order to send its service description to the directory and service consumer would query against this directory using the same protocol as well. Then service consumer directly formulates its message to the service provider based on the specifications that are stored in the registry using extensible markup language (XML) which is a tag-based language. And the provider also formulates its response message to the consumer in XML format based on the specifications in the registry.

Webservice Standards

in a large variety of heterogeneous enterprise data environment, integration of data requires a structured standard data representation. Extensible Markup Language (XML) is designed and recommend as an industry standard markup language, it can shield heterogeneity of data from different sources. Its structured definition of data is based on self-describing open standard approach. It offers data access and exchange model [34] 35]. Data from different sources can be integrated using XML.

The core standards used in web-service implementation are SOAP, WSDL, UDDI, and application transfer protocols like HTTP, FTP, SMTP etc. But other standards used for web-service management and web-service security also exists.

Simple Object Access Protocol (SOAP): SOAP is a standard messaging format used to communicate between web-service architectural components. It is based on Extensible Markup Language (XML), it is independent of platform and other programming languages.

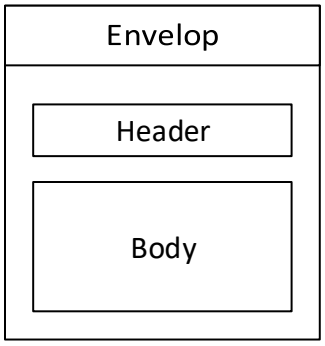


Figure 7. SOAP message format

As shown in Figure 7 the SOAP message format structure with its message header and message body.

Webservice Definition Language (WSDL): WSDL is an XML based language that describes the implementation of services. It is used both by the service provider and service requester. WSDL document includes information about the location of the services and their message format.

UDDI: UDDI is a web service used to register the description of services makes it available for other service consumers to discover [27].

2.5.3 Service Oriented Application Integration

Service-Oriented Application Integration (SOAI) is an approach that allows enterprises to share common application services as well as information/data among integrated applications [3]. This can be accomplished by providing integration infrastructure and by defining shared application services. Application services can be shared either by hosting them on a central server or by accessing them remotely through Applications Programming Interfaces (APIs) application such as through web service interfaces. Service interfaces define how to interact with a service [44]. In Service Oriented Architecture (SOA) software is packaged as services in a loosely coupled way that are platform and language independent, and can be located, invoked and combined with each other dynamically [45]. SOA formalizes the principles of loose coupling to achieve flexibility, independence, and modularity in application integration. Service Oriented integration approach tries to offer guidelines for promoting interoperability, reusability, scalability, and loos-coupling.

Service-oriented integration must be designed to Integrate once and connect many applications over time; each system hall be integrated once into the enterprise service bus, rather than many times for every point-to-point connection.

2.6 Data Synchronization

Data synchronization is the process of maintaining consistency of data across all consuming integrated applications and data stores and the continuous harmonization of the data over time in all participating entities in the data synchronization process [9]. It ensures that the same copy or version of data is used in all integrated applications that participate in the data synchronization from source to destination, and regardless of the data modification, all data changes are merged with the original data source. Data synchronization is based on the common data that would be shared between integrated applications. The common data may originate in one application, but the same data would be used by different integrated applications. The same way when the common data is updated or modified the change is required to be synchronized to create consistency among these integrated applications which use the common data.

In a heterogeneous enterprise environment synchronizing the enterprise data and maintain data consistency among integrated systems is a challenging task. it requires a good understanding of the applications and data nature which is exchanged between integrated systems and a careful data integration and synchronization design.

2.6.1 Data Inconsistency

In a complex enterprise IT environment, maintaining data consistency across heterogeneous applications is challenging; in such complex enterprises sometimes, the possibility of data discrepancy is an unfortunate reality. If not discovered and managed, bad data may lead to incorrect decision-making, failed service-level agreements, and ultimately, operational, legal, and financial risk. The common potential causes of data discrepancies in an enterprise are [46]:

Configuration Errors: Improper and unintended configuration of replication products may cause data discrepancies.

Gaps in Replication: Although data replication is configured between the source and target databases(s) and it is working well, there are instances in which data inserted on the source would not be replicated to targets.

Replication Latency: in asynchronous replication, there would be a short lag between changes and delivery of those changes to the target databases. Failure to meet the maximum latency threshold could potentially violate a data compliance requirement.

Application Errors: Applications that use target database can potentially change data due to faulty logic or, during application upgrades.

Migration error: when data is moved/migrated to the new application databases migration tools may use before replication begin facilitating initial load of the target database. Differences in configuration between migration tools and the replication products could result in data discrepancies. For example, source and target database configuration difference; incompatible character sets, date/time format can cause errors.

2.6.2 Data Synchronization Characteristics

Enterprise data synchronization can be characterized by two functional attributes, the enterprise data coherence, and the enterprise functional coherence [14]. Enterprise Data Coherence deals with maintaining multiple data copies synchronized so that all integrated data sources become symmetrical systems of records for the core/common enterprise data. Whereas in functional Coherence Decoupled and asynchronous notifications exchanged via the messaging broker that keep systems unaware and independent of each other while maintaining scalability as needed. If new applications are added to communicate their own data copy or sync with other applications, they will start listening to notifications and start sending notification to the broker to be dispatched as needed throughout the enterprise.

The main key quality Attributes for any data synchronization systems are the ability of the system to add additional systems in the data synchronization mechanism, maintainability, traceability, availability and performance of the data synchronization system are the main data synchronization system quality attributes.

Scalability: Without any architectural changes to the integration framework or the domain systems, new systems can be added to this topology and can be enabled to participate in the integration. It is a requirement that these systems expose a data notification service endpoint to handle enterprise notifications and to be able to raise and react to such data notifications appropriately, while being aware of the canonical model as the lingua franca of the enterprise integration.

Testability: Although additional testing frameworks for the integration components must be designed and built, individual systems will continue to be tested independently of each other or independent of the integration middleware.

Maintainability: ease of maintaining the data synchronization system for any defect identified during system test.

High Availability: is the ability of the system to operate seamlessly without failure.

Performance: the performance of the data synchronization system to execute transactions at a time or the ability of the system to handle the data synchronization workload.

2.6.3 Data Synchronization Technologies

Different technologies and mechanisms are involved to synchronization data in enterprise systems [55]. These technologies can be broadly classified into two categories, Data Movement Technologies, and Heterogeneous Data Access Technologies. These technologies optimally map to the three fundamental integration problems [56]: synchronizing data between systems, isolating applications from each other, and automating multistep business processes within companies and between companies.

Data Movement Technologies: One way to synchronize data between different systems is to move data between the systems themselves. Batch data transfer (which is the default approach to synchronize data), manual file and data transfer, Database replication technology and manual individual or batch data transfer effectively synchronize data in a batch transfer or in pre-scheduled transactions. The need for Realtime data consistency in enterprises requires data movement using automated technologies such as database replication.

Heterogeneous Data Access Technologies: An alternative to moving data between systems is to keep data in one place and access it from multiple heterogeneous applications. This eliminates the necessity to constantly synchronize data by moving it between systems. To simplify, all data that must be accessed by different systems are stored in one central place to which all applications requiring subsets of the data have access.

2.7 Basic Integration Components

In any enterprise application integration, a variety of components could be involved. The basic and main integration components selected to be discussed in this sub-topic are the Message filter, Message processor, Message endpoint, Knowledgebase, Data Service, Data mapping and, data notification components are introduced as follows.

Message Filter: A Message Filter component of the EAI determines whether to further process the Message or forward to an endpoint or to drop the message. This simply requires a test method that checks against a certain criterion. If the Message is accepted, it would be processed within the EAI internal components, else either sent to an endpoint or it would be dropped. A message filter is sometimes used by a message processor in which it determines could handle an incoming message.

Message Processor: Message processor is used to implement different messaging patterns and integration patterns. Message processors receive the incoming messages and do further processing. A message processor processes and make change to the message as per the designed mediation and send to the back-end service at a given rate.

Endpoint: An endpoint is an external destination for an outgoing message. It is represented by the endpoint Uniform Resource Identifier (URI), it encapsulates component-Prefix which is component specific URI. The remaining part of the URI has a syntax defined by the component. A message endpoint is an interface between the sender application and a message receiving system.

Knowledgebase: The knowledgebase is a central database structure for integrated enterprise applications which is derived from the integrated enterprise applications common information/data. A Knowledgebase is a central data repository that systematically captures, organizes, and categorizes information as a knowledge for future use.

Data Service: Data service is the process of exposing data as a web-services so that the data would be accessible to a wide variety of clients. Data services refer to SOA applied to data sources to access as a standards-based service from the World Wide Web and provide controlled and

governed access to back-end systems and data services enable reuse and sharing of structured data (such as relational Tables), unstructured information (such as RSS feeds, Web applications content, online commercial data) semi-structured information (such as Extensible Markup Language (XML) documents) [38].

Data Mapping: Data mapping is the process of creating link between different distinct data models. Data mapping can be done by comparing the attribute information from both entities of the conceptual and logical model with the corresponding elements in the database. To associating the functional entities of one application to entities of the other application without introducing awareness of the existence of the other domains direct dependencies that tie these enterprise application entity and data mapping service is introduced.

Data Notification: To maintain consistency among integrated applications any update in the common data should be communicated among integrated applications that participate in the integration so that consistent data would be available in their data store. All applications participating in the integration may notify the Knowledgebase about relevant data updates without being aware of the existence of the other applications that might need the information or even the data would be consumed by other applications. All applications participating in the integration would be notified of relevant data updates that may occur across the enterprise applications through update notifications. This in turn, allows applications to keep their data copy to be synchronized with the data in the knowledge-base and other integrated applications.

2.8 Chapter Summary

In this Chapter, the core concepts that construct the data synchronization solution model are discussed in detail. This Chapter is organized in seven major Sections. Section 2.2 defines and describes the nature and characteristics of enterprise applications. Section 2.3 discusses the main concepts behind the integration of enterprise applications and systems. This Section contains seven subsections that discuss deeper concepts about EAI such as its layers, levels, architecture, technology, standard interfaces etc. Section 2.4 discusses the general concept of integration middleware. Section 2.5 discusses SOA, and in the subsections, SOA design principles, its benefits, web-service and ESB in relation to the service-oriented architecture is discussed in detail. Section 2.6 discusses data synchronization and its characteristics. Section 2.7 discusses EAI basic components and its artifacts that are important for the design of the DSSM.

Chapter Three: Related Works

In this Chapter, five researchers work related to enterprise data synchronization has been reviewed. The literature problem area, the proposed solution, the methodology used to solve the stated problems, the research validation result and limitation of their work is reviewed thoroughly and presented paper by paper as follows.

Song and Jing conducted a research [10], to grasp the production status of the branch cement factories in real-time using SQL database and its replication technologies. The research is conducted to provide a safe, reliable and low-cost solution for data synchronization in the cement industry using SQL replication technologies for data synchronization between servers. It copies, distributes and synchronizes data and database objects between databases to maintain consistency. Out of the four SQL server 2012 replications types (Snapshot Replication, transactional replication, peer-to-peer replication and merge replication), only the log reader agent of the snapshot to get the data in the publication database update is used due to different limitations on the others; then distribution agent synchronizes the data to the subscription database.

There were three database categories in the cement factory, the factory system management database (FSMDB), factory database (FDB) and data collection database (DCDB); and three types of synchronization models, multiple publications single subscription data synchronization model (to synchronized from FSMDB into the group system management database GSMD), single publications single subscription model (FDB and DCDB in each branch factory will be synchronized to the group in real time), single publication multiple subscription(Interactive data from clients is stored in the GSMDB and immediately replicated to branch factories).

The synchronization strategy is based on nodes which contain multiple tables that need to be synchronized; SQL replication technology leads to synchronization interrupt and that leads to subscription and publication revoke; it resulted in data differences between the publication and subscription databases and it requires maintaining the consistency of the two side databases to build nodes synchronization. Replication monitor is used to accurately monitor the condition of real-time data synchronization. It can also be added on the front desk management system platform to fully monitor the operational state of synchronization conveniently for in time maintenance. This research is based on the source database and destination database nodes; the writers don't consider synchronization of source and target data at application level logically and syntactically moreover, the solution is designed and tested for homogenous enterprise environment.

Jianjun and Xiangyun [9], conducted a research to overcome the limitations of existing data synchronization tools and strategies that lack generality and flexibility in solving data synchronization problems. The research was conducted in search of a common, flexible and safe data synchronization strategy, which will support a wider range of synchronization. As a solution, they design a data synchronization model based on Middleware and rule base (DSBMR). First, they define the taxonomy of synchronization rules and different rule tables are designed for different types of rules; then, the process of data synchronization is divided into five Links, which are data access, data capture, data transmission, data conversion, and security validation respectively. At last, they design each link based on the rule base and integrate them with the data source, target data, and working file. They validate the effectiveness of the designed model using XML as a data transmission carrier. The XML document syntax generated shows the accuracy of the synchronous data. The new model has been applied on municipal information management system and the real estate appraisal systems it passes the test of practice and gets a good reputation from users. Unlike the SQL server replication technology, this data synchronization solution is validated in a heterogeneous environment. But this synchronization system is user interactive and require users to choose the synchronization function and mode. Thus, it would be unrealistic to apply this solution in real-time automated enterprise application integration environment.

Mihaela Iridon's research in 2015 [14] to produce enterprise integration and data synchronization solution model through the implementation of "maintain data copies" data integration pattern based on custom broker messaging architecture using canonical data model. The data synchronization solution is implemented based on both data integration and functional integration. The research is realized based on three business domains common data HR, Finance and Benefit. Data notifications exchanged between these domains via a broker-based messaging architecture. The data payload of the messages is wrapped inside a context-based notification model, allowing participating systems to take action based on their own domain rules using the data received from the message broker. The model's only purpose is to capture and transport data notifications across the systems. A canonical model is used to describe entities that are shared between various domains of the enterprise. The data exchanged between the various domains is encapsulated inside a canonical model, which is the common data abstraction across the enterprise. This in turn, is wrapped inside a context-based, generic, and reusable notification model, allowing systems to react to these notifications based on their own business rules.

Functional characteristics (such as *Enterprise data Coherence and Enterprise Functional Coherence*) and non-functional characteristics (such as Scalability, Testability, Maintainability,

Availability, and Performance) have been considered in the solution architecture. The resulting architecture keeps Employee and Group demographics common data in sync among all enterprise systems. Features scalability, extensibility, and high-availability quality attributes, and near-real-time data synchronization between systems achieved by allowing integrated applications to operate without awareness of each other, while using their individual data formats, features, and domain rules. The solution is designed specifically for a light load environment; it may not fit for a high data transaction load environment like the ethiotelecom case.

To produce a data synchronization solution for LDAP, HR and Educations systems integration, Zhitong and Xiaoli conducted a research based on web service [15], XML and SOA technologies through ESB. The Enterprise applications are heterogenous in terms of programming language and the database development platform; some systems share data, some even share the function as well. To solve the difficulty of communication and data synchronization between these heterogynous systems, ESB is introduced. The data synchronization solution is designed based on SOA, Web Service (for its platform independence), ESB (for SOA implementation to integrate scattered services) and XML technologies in a four-step process. The four steps are the data synchronization requirement analysis, the common data service design, the business logic design, application of ESB based on Java Business Integration (JBI) specification. As a result, the changed data in HR system could successfully synchronize to the LDAP system; in addition, by applying flag the new system could capable of handling data migration and also data could be monitored in any given times. But unlike the ethiotelecom case, this solution is designed for light-weight applications and may not be applicable in a high transaction load environment.

Chapter Summary

Many researchers have conducted research on data synchronization worldwide, but the above-related works are selected much-related research works regarding data synchronization in an enterprise applications integration environment. From the reviewed literature, a great deal of lessons has been learned to apply in the proposed data synchronization solution model. For example, [10] provides data synchronization solution for the cement enterprise the researcher focuses on studying the enterprise environment and uses the available existing SQL server 2012 resource and its replication technology in the company for enterprise data synchronization. Whereas in [9], introduces the use of middleware to process and synchronize the heterogeneous data among different data sources by refereeing different rules form registry. Another literature [14], uses data notification technique to synchronize any data modification/update to other applications in the integration environment.

However, each of the research works has its own limitation to be implemented in ethiotelecom existing environment. One related research, [57] uses peer-to-peer architecture mode to produce a database synchronization system, which has a known scalability limitation and couldn't fit for large enterprise environment. Other research [10] focuses only on creating data consistency between databases and does not mention the effect of application integration in data synchronization. And [9] another related work is conducted based on SQL server 2012 replication technology which is not a heterogenous environment. As a result, none of the above researches could be used as a solution to address the company's specific data synchronization issues.

Moreover, unlike the other researcher's environment, in ethiotelecom, the integration environment is already established based on the traditional EAI layered architecture [Section 2.4.1] but due to vendors privacy protection for the data synchronization problem checking what is wrong in the source code in the integration environment and to modify it is not possible.

Chapter Four: Situational Analysis

In this Chapter existing ethiotelecom enterprise environment situation, its enterprise applications integration and SIM card sales business process in which the proposed solution is validated on is discussed in detail. At first, the details of existing system state is described; which covers the description and components involved in SIM card replacement business process. Then the existing implementation state evaluation is discussed against used tools and technologies in the enterprise environment.

4.1 Current State Analysis

Of all the enterprise systems implemented in the company, this research focuses on the NGBSS system implementation and its integration with other third-party applications. In the NGBSS system implementation, there are around 136 defined services [21]. This research particularly focuses on the SIM card replacement service. Figure 8 shows the SIM card sales environment front end and backend processing situations.

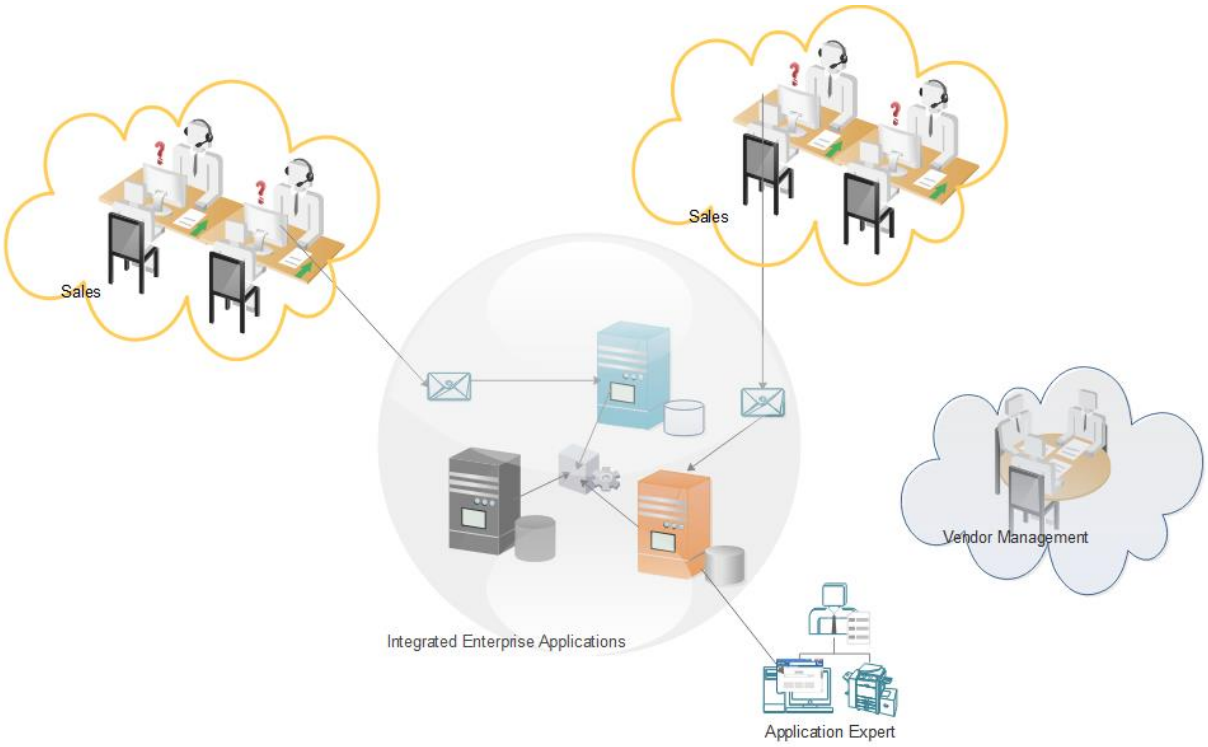


Figure 8. Existing System environment.

Figure 8 shows, existing enterprise applications integration environment. Salespersons in every ethio-shops are waiting for customers to manage his/her service request. Based on customers'

request the salesperson would execute from the list of services; for example, SIM card replacement service, new SIM sales service etc. The integrated enterprise applications are ready to process the service request messages from the sales terminals. The application experts are alert for the incoming error logs and events. Minor errors would be resolved immediately, or if it requires vendors involvement, they would forward the issue to vendors. But most of the time in a multi-vendor environment, most vendors try to push the problem to another party not to take responsibility; therefore, such issues would be resolved by organizing a multi-vendor team to solve the identified problem

The activity diagram in Figure 9 below shows the SIM card replacement business process implemented in ethiotelecom sales shops. The activity diagram is based on ethiotelecom's SIM card replacement business process and detail can be found in the sales manual document [22].

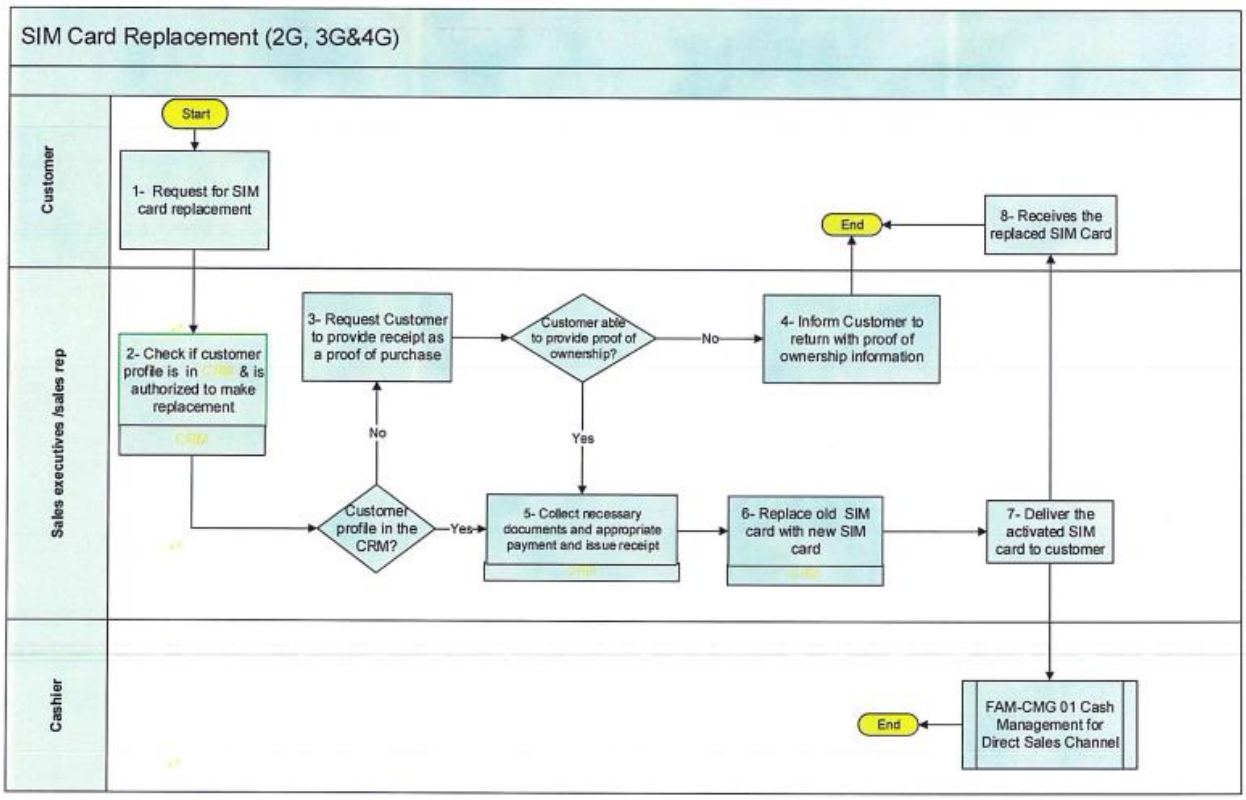


Figure 9. SIM card replacement process activity diagram [22]

Based on the NGBSS LLD document [21] the sequence diagram in Figure 9 below shows the sequence of activities performed by each actor and participated applications in the integration. The activities detail is also described per the sequence number next to the diagram below.

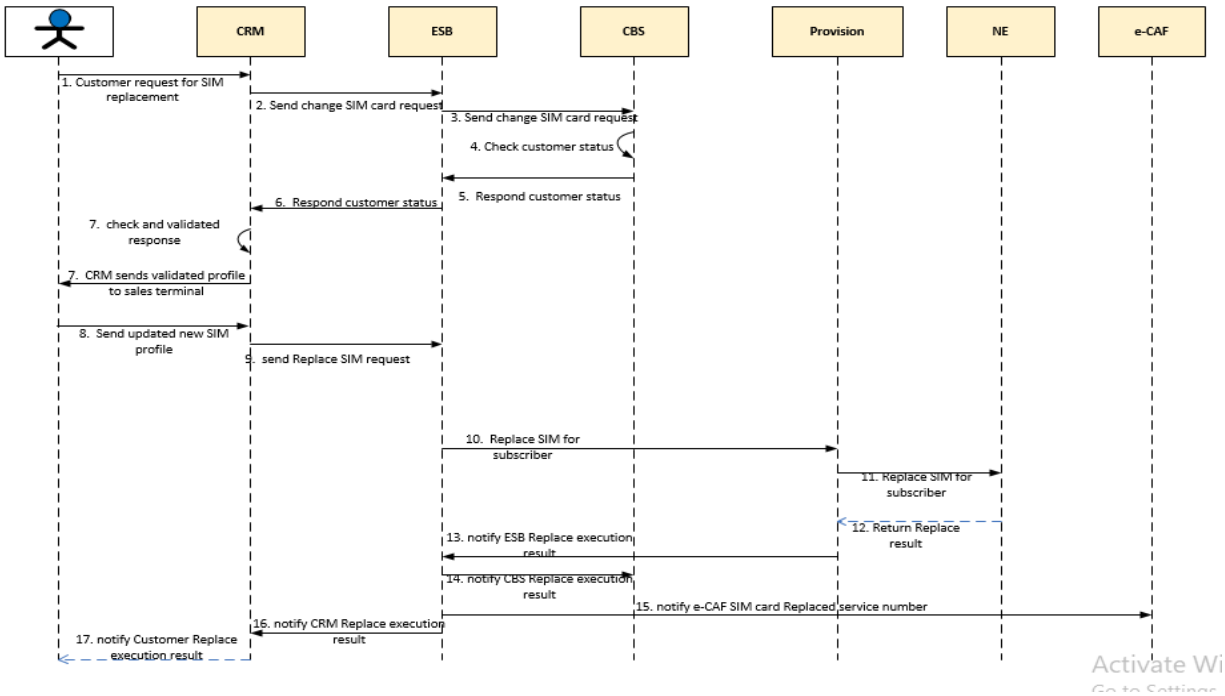


Figure 10. SIM Card replacement Sequence diagram [21]

The sequence diagram in Figure 10 above shows;

1. Customer in the sales office requests a SIM card replacement.
2. CRM sends the SIM card replacement request to the ESB
3. ESB sends the SIM card replacement request to CBS
4. CBS checks customer status
5. CBS send the response to ESB
6. ESB sends the response to CRM
7. CRM captures and validates the response
8. The customer profile is updated with new SIM information.
9. CRM sends the replaced SIM information to ESB
10. ESB sends replace SIM request to Provisioning
11. Provisioning sends SIM replacement request to NE
12. NE returns SIM replacement result to provisioning
13. Provisioning sends the SIM replacement result to ESB
14. ESB notify the SIM replacement to CRM
15. ESB notify SIM replacement information to CBS
16. CRM confirms the SIM replacement to the customer.

In the existing EAI implementation, the middleware called the Enterprise Service Bus (ESB) is not mediating all the enterprise applications; it is not actually Enterprise level service bus rather it mediates the NGBSS systems and some other third-party applications in the company. For example, it does not mediate any of the Operation Support System (OSS) applications. OSS elements such as the network element, integration with NGBSS is mediated through provisioning system.

4.2 Existing State Evaluation

To achieve the research objective collecting relevant data is one important activity which is conducted in the company through the following three techniques; through key informants' interview, company document review and system data collection. The interview questions are available at the end of this research document [Annex C]. The technical interview questions are intended for Key informants that are highly experienced application and the corresponding database administrators. This interview is intended to extract the technical details about the current system integrations information that could not be found on the company document; such as middleware design details, database design details, the data synchronization problem impact on the enterprise community, about events and error logs etc. Based on the methodologies specified above, required information has been gathered to discuss the current state of enterprise application integration implementation in ethiotelecom.

Due to the vendor privacy-policy of the enterprise applications, the source code is not visible, or the applications do not have interface to make change/modify applications in the company. Per this research's finding through key informant interview, even for a small maintenance related support if code modification is required the issue would be escalated/reported to the vendor for code modification.

For each integrated application, database design document could not be found in the company, instead, for this research, the database design is referred from the actual data queries and CRM, CBS and e-CAF model databases are redesigned for this research demonstration based on the actual data. Based on these model databases attribute the Knowledgebase is designed as described in Section 5.4.1.

Information and data communication among integrated applications are based on XML-based SOAP messaging [12] [13]. The message format and message content in the existing system uses web-service SOAP 1.1 protocol. And a sample message header and body format, request and

response message content is examined thoroughly so that the same message pattern can be created for the research demonstration. In the NGBSS system integration interface ID, Interface name, interfacing system, action, the destination system, etc. Key information's are included in the SOAP message [12] [13].

4.3 Chapter summary

Chapter four describes the current state of the company in which this research is based upon. It discusses the company's situation in two major Sections that describe the current situation of the company and the evaluation of the current situation. In Section 4.2 the current state of the company is analyzed by applying techniques and methodologies relevant to gather information about the current state of the company. Section 4.3, the information gathered through the methodologies applied in Section 4.2 is further evaluated for its relevance for the proposed solution model implementation. And finally, this Chapter summary concludes Chapter four.

Chapter Five: Proposed Solution Model Design

In this Chapter, the proposed data synchronization solution model design is discussed. At first, the proposed data synchronization solution system model (DSSM) design consideration is described in great detail; then, its Design considerations, solution model design, its architecture and components, and implementation design would have detailed clarification in this Chapter. At last the DSSM design process is summarized to conclude Chapter Five.

5.1 Design Considerations

When Considering the applications integration scenario and the business process, the common ground for all integrated applications is the middleware and the common data that defines the customer and the service. The main enterprise data common to these enterprise applications must be shared and need to be consistent in all applications datastore. To accomplish this fact, the main purpose of the proposed data synchronization solution model (DSSM) is designed to become an intermediary additional data synchronization layer in the traditional EAI layered architecture. As explained in Section 2.4.1, the traditional EAI layered architecture is organized as User interface layer, Middleware layer, and Data layer.

Also, since modification to existing enterprise system is not allowed [Section 4.3], the proposed solution should be designed independent of existing enterprise applications and the solution should not demand major modification in existing enterprise systems in the research domain.

Based on the findings from the situational analysis in Chapter four, the following basic considerations are made to produce the data synchronization solution model.

- A data synchronization model shall be designed to create data consistency among integrated systems.
- The design should be based on/ should consider existing system implementation.
- The solution model should not require major change to existing systems
- It should not violate the vendor's privacy policy
- It should be based on SOAP 1.1 messaging protocol which is the existing message protocol used in the company environment.

Based on the above-mentioned considerations the message flow from the service requester through service register and to the service provider is narrated based on SOA integration rolls

[Section 2.6]. Based on the message flow in the company; at high level, the DSSM message flow design is described below and the high-level flow diagram is shown in Figure 11 below.

- Accept the SOAP message from application.
- Filter and process the required messages and pass the rest to the next layer.
- Process the filtered request message to appropriate backend service format.
- If the requested data is not available in the knowledge-base, check in other backend databases and update the KB.
- Notify other applications about the update/change to keep them consistent with others.

The proposed solution is designed to be implemented between application layer and data layer in EAI layered architecture. The whole process is, the DSSM will accept the request message and filters the message; based on the message content the solution would either forward to the next layer which is the Middleware or further processes the message. The DSSM filters only specific CRUD (create read update or delete) data requiring services. These services are already known and would be identified by their business Service identification number (BSID) in the incoming SOAP message. The following flowchart diagram further describes the message flow design.

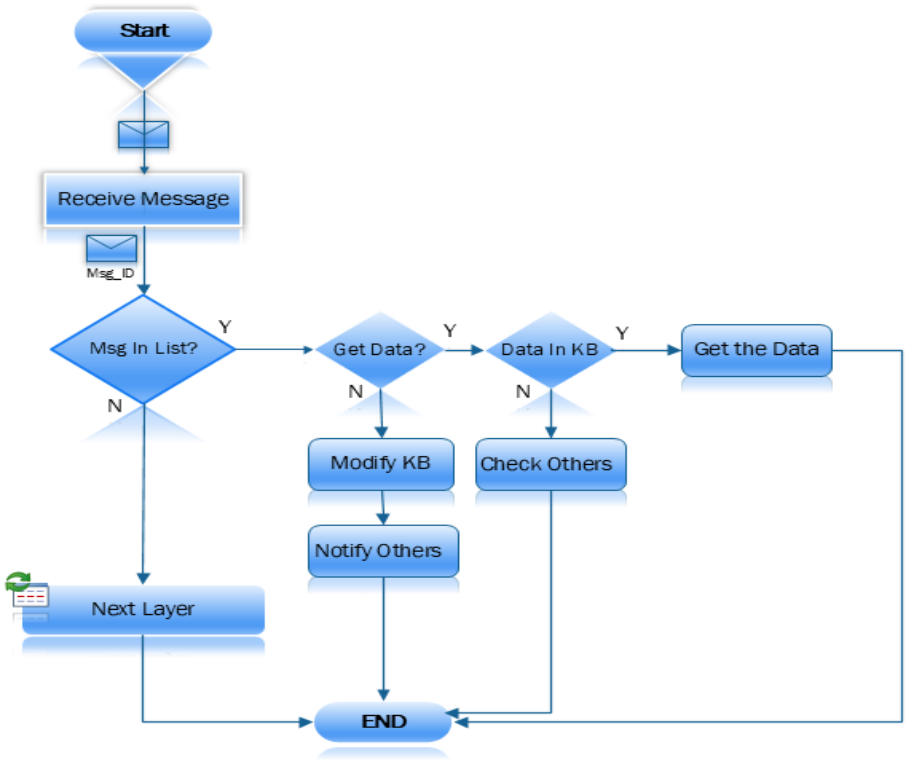


Figure 11. Message Flow in DSSM.

The data synchronization layer receives a message and checks if the message requires processing in the DSSM by its business service identification number (BSID) in the SOAP message, if the BSID is not in the list the message would be forwarded to the next layer which is to the existing middleware layer. If the BSID is in the list (for example, BSID=101 can be assigned for GetCustomerProfile service) the requested service number would be checked in the knowledge-base. If exists a response customer profile detail would be sent to the requester if not the data would be checked in other applications database and the result would be sent to the requester. On the other hand, if the operation is not to read data or BSID is not 101, then it requires to modify the data in KB and other applications participating in the integration through notification message about the data modification.

5.2 Proposed Solution

As indicated in the problem statement (Section 1.4) of this research document, the data synchronization problem identified in this research is a problem created by the data inconsistency among integrated enterprise applications. To solve the data inconsistency and discrepancy problem a data synchronized solution model is proposed. The proposed high-level Data Synchronization Solution Model (DSSM) is presented in Figure 12 below.

At a very high level, the proposed solution provides a knowledgebase/data repository for the common data exchanged between participating applications in the integration. All participating applications in the integration can access this Knowledgebase and if there is any change or modification in the Knowledgebase all participating application would be notified about the change. The Knowledgebase is careful designed so that there should not be any conflict or data mismatch with the other applications in the integration. If there is a data type, format or size difference in participating applications, it would be normalized in the DSSM. On the other hand, the data synchronization middleware filters incoming message and facilitates the message exchanged between integrated applications by providing data in receiver suitable format.

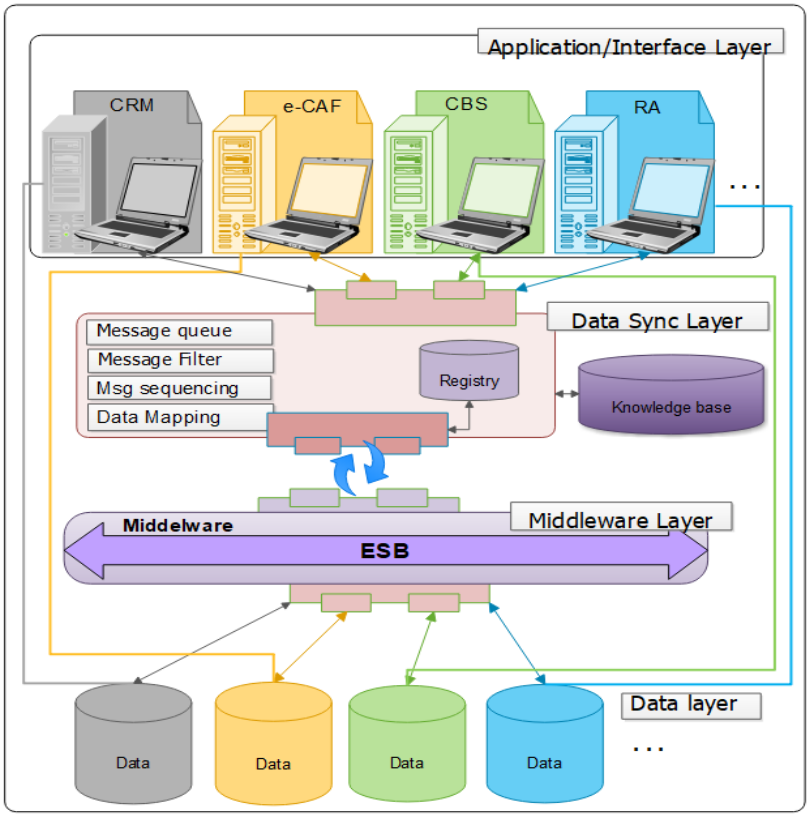


Figure 12. Data Synchronization Solution model.

Regardless of the heterogeneous enterprise environment in which these applications are deployed, the proposed solution is intended to provide a unified common data source across the integration environment and to implement additional data synchronization layer through integration middleware architecture. For this research implementation, the technology is carefully-chosen based on the identified requirements, reusable frameworks and services are created and a data synchronization models that fulfill the business need is designed.

5.3 Solution Model Architecture and Design

The data synchronization solution model is designed as an additional data synchronization layer in existing EAI architecture [Section 2.4.1] which is based on application layer, middleware layer, and data layer. It is designed based on SOA design principles [Section 2.6.1] to operate on top of existing middleware layer [Figure 12]. The design of the data synchronization solution model is inspired by the initial idea of Linthicum's white paper [7] "defining, designing and implementing data service layer" and similar idea is also available in Master Data Management (MDM), and Business intelligence literature.

"SOA-based data services offer the opportunity to get data under control and leverage an enterprise architecture that's more data-driven. It maps its mediated common data model to source and target data through the solution's internal components [7]*."*

The proposed data synchronization solution model (DSSM) [Figure 12] is designed based on its two major components. The Knowledgebase which is the common data repository for all enterprise applications that participate in the integration. And the message processing Middleware infrastructure which is composed of the message in-sequence, message receiving-sequence, message-out sequence, message fault-sequence and data services major components.

5.4.1 Knowledgebase Design

The Knowledgebase is designed based on the attribute and format of the common data exchanged between integrated applications. The Knowledgebase is designed from the normalized structure and data format of each application databases participated in the integration. For this research implementation the Knowledgebase format is referred from e-CAF to CRM Integration Control Document (ICD) document, but per the information found from the company, there is no ICD document for CRM to CBS integration, instead, the attributes of the two databases is referred from the actual data from the databases. The actual databases are very large to use for this research demonstration; for example, in CBS database there is more than 73 columns just in one table and in the e-CAF database there is around 145 columns just in one table. As a result, using the actual database structure is too difficult for a research demonstration; therefore, in this particular research a mini database model for the CRM, CBS, e-CAF [Annex B] and the Knowledgebase is created.

The Knowledgebase is intended to be designed based on existing integrated application databases. For this research demonstration, participating applications in the integration for SIM card replacements process are CRM, CBS, and e-CAF excluding the ESB middleware application. The Knowledgebase is a database designed based on the common data exchanged between these integrated enterprise applications by refereeing the integrated applications database structure.

The database design document for all integrated applications could not be found in ethiotelecom. Instead, the actual database tables and attributes are extracted from the actual data. Then actual database resembles sample CRM, CBS, and e-CAF databases are designed based on the extracted data. Finally based on these databases design the Knowledgebase structure is designed. In this research document, only the Knowledgebase database design is included; for the other applications database design, the reader can refer to Annex B.

For this research demonstration the following data items are identified and are prepared into five tables. The information required are, detailed information about the customer, its detail service subscription information, the kind of service customer wants when he/she come to sales shop, the information about the operator who serve the customer, and billing related information how the customer would be billed for the service he/she gets.

The conceptual model for the above data item is described in five tables with basic datasets and its primary key identified is described as follows table by table.

Operator_Information (Operator_ID, Region_ID, Department_ID, Section_ID, Shop_Code, Roll, First_Name, Middle_Name, Last_Name).

Customer_Information (Customer_ID, First_Name, Middle_Name, Last_Name, Email, POBox, Region, HouseNo, Category, OfficePhoneNo, MobilePhoneNo).

Subscription_Information(Service_No, Subscriber_ID, Customer_ID, Operator ID, Shop_Code, Status).

Service_Information (Order_ID, Service_No, Service_ID, Subscription_ID, Customer_ID, Service_Type, Network_Type).

Billing_Information (Bill_ID, Subscription_ID, BillSycle_ID, Status).

Detail description of each table and its attributes and other possible options that would be filled in the tables' attributes are described and tabulated as shown below.

Table 2. Database attributes detail Description.

Tables	Attribute	Description	Remark
OPER_INFO	OPER_ID	Operator identification number	
	REGION_ID	Region in which the operator is working	Per ethio ICD documents [13] there are 11 administrative regions (1-11)
	DEPT_ID	Department identification in which the operator is working	
	SECTION_ID	Section identification in which the operator is working	
	SHOPE_CODE	The shop in which the operator is working in.	
	ROLL	Roll of the operator	
	FIRST_NAME	First name of the operator.	
	MIDDLE_NAME	Middle name of the operator.	
	LAST_NAME	Last name of the operator.	

Tables	Attribute	Description	Remark
CUST_INFO	CUSTOMER_ID	Customer identification number	
	FIRSTNAME	First Name of the customer	
	MIDDLENAME	Middle Name of the customer	
	LASTNAME	Last Name of the customer	
	Email	e-mail address of the customer	
	POBox	Customer's Post office box number	
	WEREDA	Customer address woreda	According to the ICD document [13], there are 731 administrative regions (1-731)
	REGION	Customer address region	
	HOUSENo	Customer address House number	
	CATEGORY	Customer's category	1 for Normal/residential 2 for Ethio employee 3 for Public Service 4 for SOHO/SME 5 for Key Account 6 for ethiotelecom
	OfficePhoneNo	customer's office telephone number	
	MobilePhoneNo	customer's mobile telephone number	
SUB_INFO	Service_No	Service number in which the customer is subscribed	
	SUB_ID	Subscription identification number	
	CUST_ID	Customer identification number	
	OPER_ID	operator identification number	
	SHOPE_CODE	shop code in which the subscriber registered	
	STATUS	Status of the subscriber	
SERVICE_INFO	ORDER_ID	Order Identification number	
	SERVICE_NO	the service number for the requested service	

Tables	Attribute	Description	Remark
	SERVICE_ID	Service Identification number	which Is mapped with BSID
	SUB_ID	Subscription identification number	
	CUST_ID	Customer identification number	
	SERVICE_TYPE	Type of service the customer requests	
	NETWORK_TYPE	in which network the service is subscribed	Fixed, 2G, 3G, 4G, …
BILLING_INFO	Bill_ID		
	SUB_ID		
	BILL_CYCLE_ID		There are three bill cycles (1-3)
	STATUS		

The Knowledgebase conceptual model, normalized Entity Relationship (ER) diagram is shown in Figure 11 below.

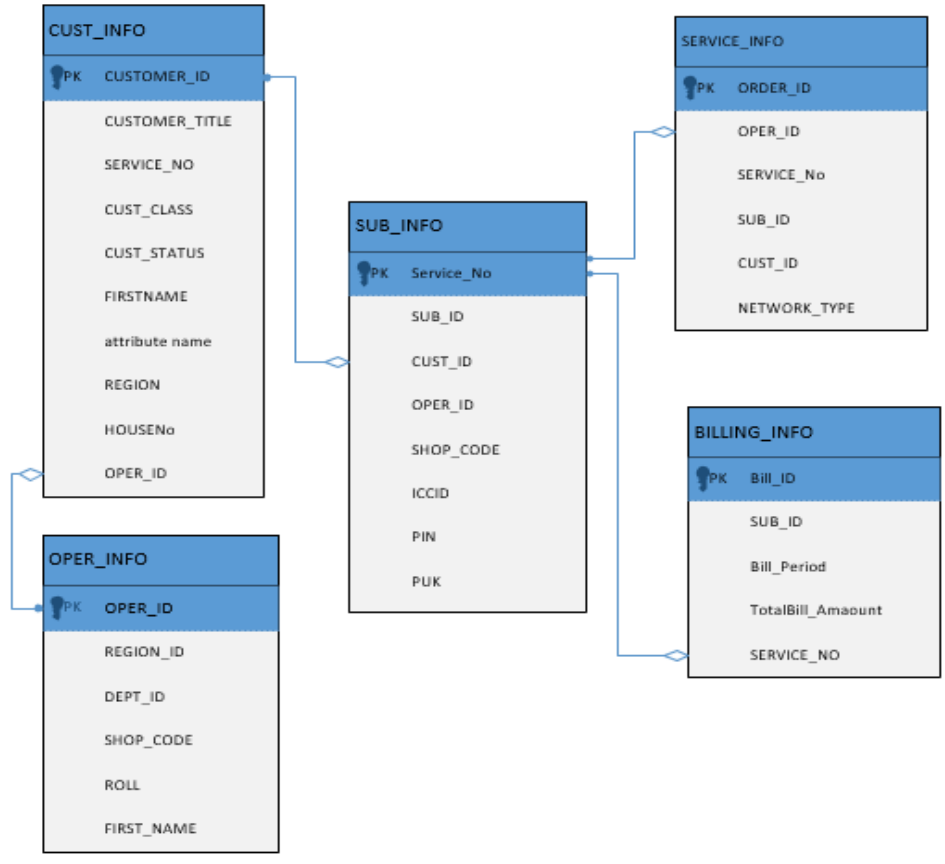


Figure 13. Knowledgebase ER diagram

For the Knowledgebase implementation design, low level logical design is described as follows for each table in the Knowledgebase to be implemented in Oracle 11g database.

create table OPER_INFO (OPER_ID Char(8) constraint oper_info_pk primary key, REGION_ID Number(4) NOT NULL, DEPT_ID Number(4) NOT NULL, SECTION_ID Number(4) NOT NULL, SHOPE_CODE Char(8), ROLL Chhar(16) NOT NULL, FIRST_NAME Char(64) NOT NULL, MIDDLE_NAME Char(64) NOT NULL, LAST_NAME Char(64) NOT NULL);

create table CUST_INFO (CUSTOMER_ID Char(16) constraint cust_info_pk primary key, FIRSTNAME Char(64) NOT NULL, MIDDLENAME Char(64), LASTNAME Char(64), Email Char(64), POBox Char(16), WEREDA Char(64) NOT NULL, REGION Char(16) NOT NULL, HOUSENo Char(16), CATEGORY Number(2) NOT NULL, OfficePhoneNo Number(16), MobilePhoneNo Number(16), created_date DATE defaullt SYSDATE, OPER_ID Char(8) NOT NULL constraint cust_info_fk foreign key (OPER_ID) references Oper_info (OPER_ID) on delete cascade);

create table SUB_INFO (Service_No Number(16) constraint subs_info_pk primary key, SUB_ID Char(16) NOT NULL, CUST_ID Char(16) NOT NULL constraint subs_info_fk foreign key (CUST_ID) references CUST_info (CUSTOMER_ID) on delete cascade, OPER_ID Char(8) NOT NULL, SHOPE_CODE Char(8) NOT NULL, STATUS Char(8) NOT NULL);

create table SERVICE_INFO (ORDER_ID Char(16) constraint service_info_pk primary key, SERVICE_NO Number(16) NOT NULL, SERVICE_ID Number(4) constraint service_info_fk foreign key (SERVICE_ID) references SUB_info (service_no) on delete cascade, SUB_ID Char(16) NOT NULL, CUST_ID Char(16) NOT NULL, SERVICE_TYPE Char(16), NETWORK_TYPE Char(16));

create table BILLING_INFO (Bill_ID Char(16) constraint billing_info_pk primary key, SUB_ID Char(16) NOT NULL, BILL_CYCLE_ID Number(2), Bill_Period Date defaullt SYSDATE, TotalBill_Amaount VarChar(16), Bill_AmtData Char(16), Bill_AmtLocal Char(16), Bill_AmtIntrntnl Char(16), STATUS Char(16), SERVICE_NO Number(16) constraint billing_info_fk foreign key (SERVICE_NO) references SUB_info (service_no) on delete cascade);

5.4.2 Message Processing Components Design

The DSSM message processing Middleware component is a major component with many functional units. The components are designed to performs the message processing end to end. This component design includes message endpoints, message processing mediators, sequences, proxy services, data services and webservice components. Based on the flow chart in Figure 11 above in Section 5.1, the components required to produce the DSSM message processing part is designed to have the following basic and major components.

Mediators: Mediators are the basic components of a mediation framework. WSO2ei has a comprehensive library of mediators. Mediator is a full-powered processing unit, at run-time, a mediator has access to all the parts of the middleware and can do virtually anything with the message. A mediator with some configuration of data can perform a certain operation to the message.

Sequences: A sequence is a chain of mediators each can execute its own specific tasks in the designed order. It is a sequential arrangement of mediators. When a message is delivered to a sequence, the sequence sends it through all its chained mediators. For example, a chain of property mediator, log mediator, send mediator.

Figure 14. Chain of mediators/Sequences

Proxy services: Proxy service in WSO2ei is a virtual service that receives a message, processes and forwards them to a specified endpoint [59]. This helps to perform necessary transformations and additional functionalities without changing existing services. A proxy service can be consist of three sequences; in-sequence, out-sequence, and fault-sequence. The incoming message will go to the in-sequence if an in-sequence is defined. A receiving sequence can also be defined in the in sequence to specify the sequence that receives the response message in the service chaining scenario. Or else the response message will be received by the out sequence by default if an out sequence is defined. A fault-sequence can also be defined for error handling purposes. For example, you could have a log mediator to log the fault error message and the drop mediator to drop the message depending on the requirement when error occurs.

Endpoints: An Endpoint defines an external service endpoint and any attributes or semantics that should be followed when communicating with that endpoint. Endpoint would be based on a service Address or a WSDL. Additionally, WSO2 ESB supports Failover and Load-balance endpoints - which are defined over a group of endpoints. An endpoint definition can be named for re-use or defined in-line or anonymously within a configuration. An Endpoint defines an external service endpoint and any attributes or semantics that should be followed when communicating with that endpoint. An endpoint definition can be named for re-use or defined in-line or anonymously within a configuration. Typically, an endpoint would be based on a service Address or a WSDL. In WSO2ei endpoints may be defined within the synapse-config configuration or within the Registry [19]. The following endpoints are used for this scenario.

An endpoint can be a URL, folders, registry or path. Based on the direction of the message, an endpoint can be inbound, outbound, or both.

Data service: Data service is a method of exposing application data as a web-service. Data services provide a service and resource interface to some data stored in a relational database. In a service interface, one must indicate how service requests map to queries against collections of tables in a relational database and how query results are mapped to service responses. In a resource interface, one must indicate how a set of resources map to queries and how query responses are returned as resource representations [19][20].

Message Filters: Message Filter checks an incoming message against a certain criterion that the message should adhere. If the criteria is met the message will be routed according to the defined output channel, otherwise, the message will be routed to the default channel.

Message stores: in a scenario where we have a back-end service that can only accept messages at a given rate and incoming traffic to ESB will have different message rates. To serve that traffic utilizing the back-end services, there should be a temporary storage for messages. Message Stores can be used as this temporary storage.

Message format conversion: describes the modification or conversion of the message contents (header and/or body) for various purposes based on the receivers' message format.

Registry: A registry is a content store and a data repository. Various SOA artifacts such as services, WSDLs and configuration files can be stored in a registry, keyed by unique paths. Local data repositories, configuration registries, and Governance registries are commonly used registries types in WSO2 application integration software.

Data Mapping: data mapping is the process of creating data element mappings between two distinct data models. it is used as a first step for data integration request in which data from a source is converted into a format that is compatible with the destination.

JDBC Driver: Java Database Connectivity (JDBC) is the industry standard Java API for database-independent connectivity between the Java programming language and variety of databases [60]. JDBC can access any kind of tabular data source, especially data stored in a Relational Database. JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. The JDBC library includes APIs for making connection to a database, Creating SQL or MySQL statements, Executing SQL or MySQL queries in the database, Viewing & Modifying the resulting records.

To construct DSSM in existing enterprise application integration layered architecture, it would be important to consider the practical implementation between the applications and the middleware layers in the traditional EAI layered architecture. The primary purpose of the solution design is to achieve data consistency in enterprise applications integration the solution layer needs to perform certain tasks. At high level the activities performed in the solution layer is extracted from Figure 12 message flowchart diagram in Section 5.2. Therefore, the data synchronization middleware is redesigned based on the message flow with WSO2ei components and artifacts as shown in figure 15 below.

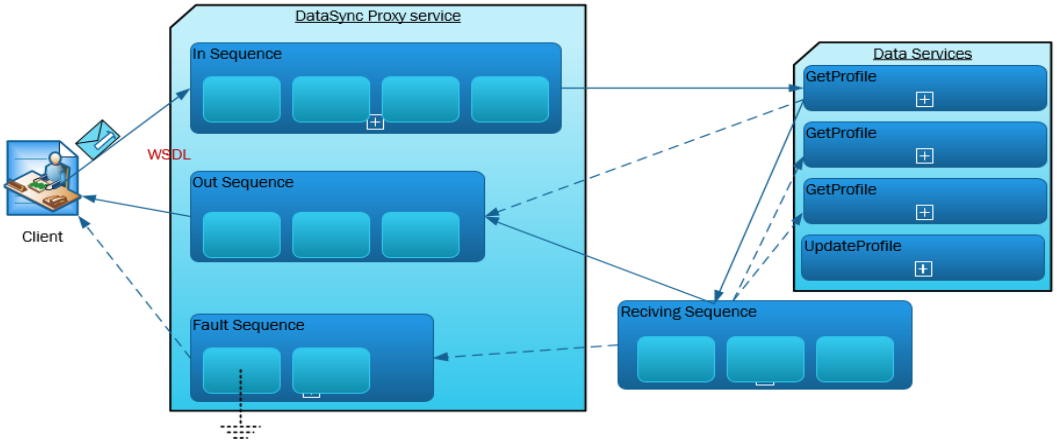


Figure 15. DSSM high-level process model

At high level the message flow in the DSSM is described in Figure 15. The Figure shows a message coming from client application would first be processed by the in-sequence of the DSSM proxy service; then the request would be sent to the data service (to read, update, create or delete data depending on the design specified). The response from the data services depending on the request message type (service) would either be sent directly to out-sequence or to the receiving sequence for further processed. Messages from the receiving sequence either would be forwarded to out-sequence or to other data service endpoints Before being forwarded to the out sequence. In case of error, the fault-sequence is the default error handler.

5.4.3 Backend Service Design

The backend service in this research is the data services defined for the GetProfile and UpdateProfile services. This service is designed based on the SIM card replacement business process requirements; which is to get/fetch customers profile data based on the customers' service number (Service_No). the UpdateProfile service updates the customer profile data based on the new SIM card information just replaced for the customer.

5.4 Solution System Implementation design

The proposed solution is intended to overcome identified problems in the problem statement Section [Section 1.4] by providing a data synchronization mechanism with loosely coupled modular service components based on SOA design principles. Moreover, the new DSSM is designed to be highly flexible to integrate additional services/solution-components into the new data synchronization solution layer and easy to maintain.

5.5.1 Implementation Design Overview

Requirement regarding the functionalities required from the DSSM could be evolving in real implementation scenarios, but the purpose of this research is to show the practicality of the hypothetical idea of adding additional data synchronization layer in the traditional EAI layered architecture [Section 2.4.1].

The implementation is possible both with opensource and commercial EAI implementation software. A variety of Integration implementation software both commercial and Free and Open Source (FOS) in which the proposed DSSM could be implemented are available worldwide. Some commercial software packages are even full-fledged and easy to use, but they are expensive, and the source code is not available to use flexibly. For this research due to the cost and the availability of the source code, the authors prefer to use open source integration software. The selected opensource integration software is WSO2 enterprise integrator (WSO2ei) by the Oxygen company. The tool is selected based on the requirements of the research and the ability of the tool to perform the DSSM implementation. As explained in Table 3, the selection comparison is made between some popular opensource integration software currently available [28-33]; however, the list does not exhaustively compare all available opensource software worldwide, rather the comparison is made between few selected software applications that are believed to do the task. WSO2 is found to be a better tool to validate the proposed solution practicality for this research.

Table 3. Open source integration software comparison Table.

Product	Fully open source?	ESB Capability	Data Service	Web services	Ese of use
Apache Camel	yes	Yes	No	XML REST-API	Require professional
JBoss	No	Yes	No (ODBC API not supported)	XML, SOAP, REST-API	easy to use
Openadaptor	yes	Yes	No	XML, SOAP, REST-API	easy to use
WSO2	yes	yes	yes	XML, SOAP, REST-API	easy to use and flexible to modify.
Apache ServiceMix	No			XML, SOAP, REST-API	easy to use
Talend	yes	yes	yes	XML, SOAP REST-API	easy to use but has limitation in some OS

WSO2 is user-friendly software with a comprehensive library of mediators; one can use available components or build custom component of its own. Moreover, WSO2 supports almost all operating systems and known databases available in the market including the H2 database for its data service implementation.

The functionality of mediators used in the implementation design of the DSSM based on WSO2 enterprise integrator to construct the proxy service and the receiving sequence are discussed as follows.

Property Mediator: The Property Mediator when configured can set or remove some property of the message based on the action specified.

Log Mediator: is used to log a mediated message. Depending on the configuration log mediator may take a full log or specified components of a message.

Payload-factory Mediator: Payload-Factory Mediator transforms or replaces the contents of a message. Each argument in the mediator configuration can be a static value, or an XPath or JSON expression can be specified to get the value at runtime by evaluating the provided expression against the existing SOAP message. The format of the request or response can be configured and mapped to the arguments provided.

Filter Mediators: The Filter Mediator can be used for filtering messages based on an XPath, JSONPath or a regular expression. If the test succeeds, the Filter mediator executes the other mediators enclosed in the sequence. It closely resembles the "If-else" control structure.

Switch Mediator: The Switch Mediator is an XPath or JSONPath filter. The XPath or JSONPath is evaluated and returns a string. This string is matched against the regular expression in each switch case mediator, in the specified order. If a matching case is found, it will be executed, and the remaining switch case mediators are not processed. If none of the case statements are matching, and a default case is specified, the default will be executed.

Send Mediator: The Send Mediator is used to send messages to an endpoint. The Send Mediator also copies any message context properties from the current message context to the reply message received on the execution of the send operation; so that the response could be correlated back to the request.

Drop Mediator: The Drop Mediator stops the processing of the current message. Drop mediator is the end of the message flow.

The three main components of the data synchronization solution model are the Proxy service (with In-sequence, Out-sequence and fault/error sequence subcomponents), The Receiving sequence and the back-end services in which the data services defined as shown in Figure 15 above. For DSSM implementation the DSSM high-level design in Figure 15 [Section 5.4.2] needs further redesign with appropriate WSO2ei components. The detail design of the DSSM is shown in Figure 16 below and the description of its major components is also discussed in below Sections.

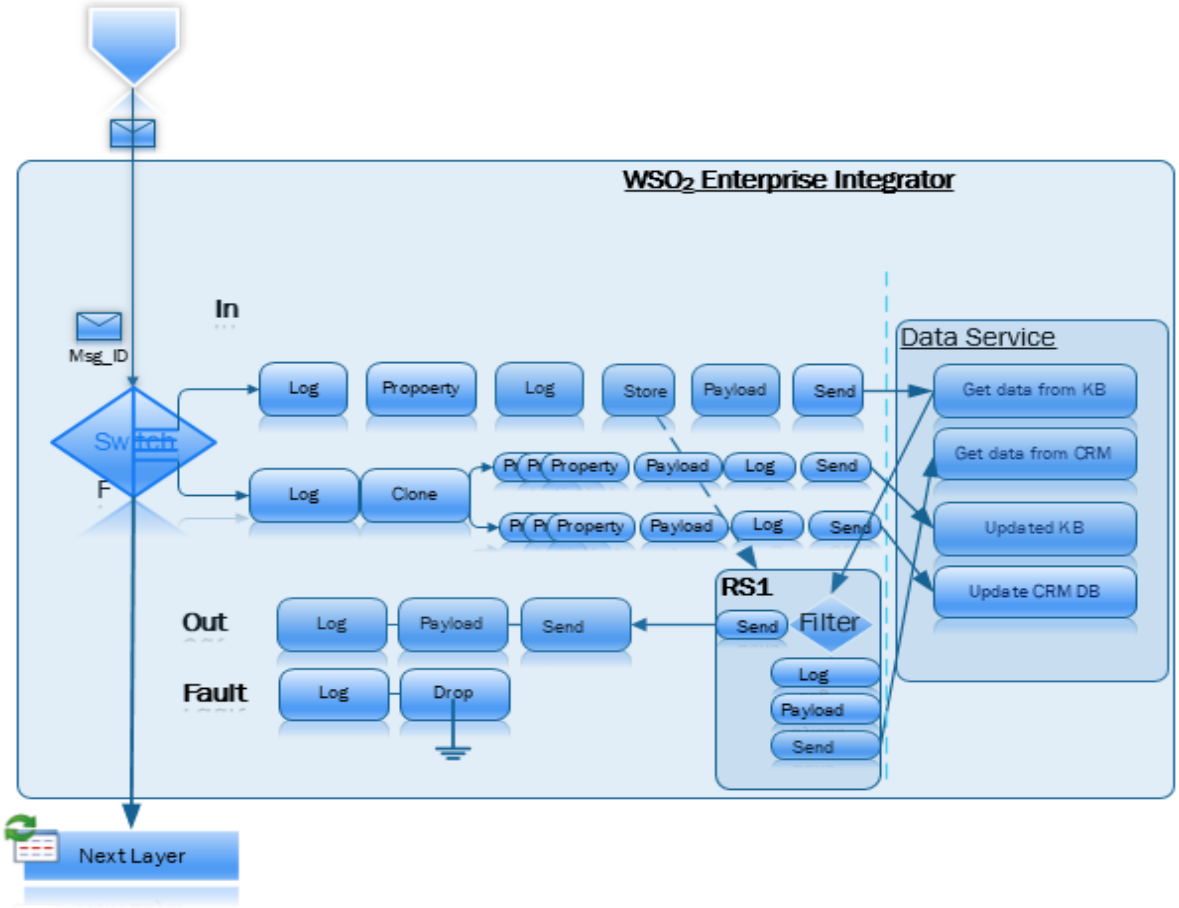


Figure 16. Data Synchronization Solution Model detailed design.

For example, Let's see how we can configure wso2ei to chain the two Data-Services in concern (the GetDataFromKB and GetDataFromCRM). When the proxy service is invoked, message 1st will go to the in-Sequence of the proxy service. In the in-sequence, we'll log the message, extract the 'service No' value; then in-sequence will construct a message with the extracted values according to the message format of the receiving data-service. Then using the send mediator specify the endpoint URL of the GetDataFromKB data service, we can also specify the receiving sequence (RS1). The message will be sent to GetDataFromKB data service; the response from the GetDataFromKB data service will be received by RS1. RS1 also (like the in-sequence) logs the message, extracts profile/service no value and construct a message and finally send the message to the other application DB (GetDataFromCRM) data service; then the response from the data service will be received by the out-sequence of the proxy service; then the out-sequence will send the response back to the service consumer. The implementation of this scenario using wso2ei functional components in wso2ei management console interface is described in Section 6.2.1 below.

5.5.2 Proxy Service Design

The proxy service design is based on the WSO2 community message processing design best practice that includes the message In-flow/In-sequence, Out-flow/Out-sequence, and Error-Flow/Fault-sequence. Each sequence in all proxy service components is based on a sequence of mediators that perform a certain operation to the message, and the sequence is designed per the requirements of DSSM design. For this research, the design of the proxy service with its In-sequence, Out-sequence and error sequence is designed as shown in Figure 17 below.

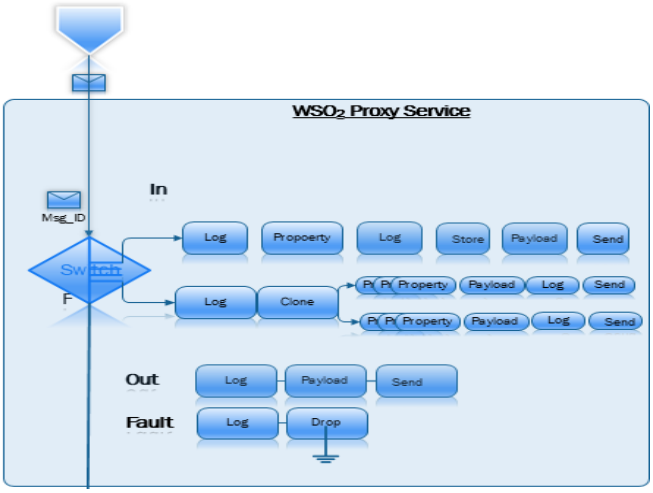


Figure 17. Proxy service design

As shown in the proxy service design Figure 17 above the incoming message would be received by the In-sequence of the proxy service; the switch mediator checks the SOAP message content Service ID (BSID) against the required service list which is intended to be processed in the DSSM; or the message would be forwarded to the next layer which is the ESB layer in ethiotelecom existing system architecture. In this research, the DSSM is being demonstrated for two services, GetProfile service, and UpdateProfile services. As shown in Figure 17, for UpdateProfile service the message would flow to the Log, Payload, send mediators sequence; for the getGrofile service, the message would flow to the property mediator flow based on switch mediator decision. Out-Sequence is the default response message processor and in case of error, Fault-sequence will handle the message flow.

5.5.3 Receiving Sequence Design

Receiving sequence is a chain of mediators that can be used to receive a message response sent by the in-sequence of a proxy service. Unlike the usual or default message flow in which a response

from a service provider is directed to the Out-sequence of a proxy service, the response from the backend service would be forwarded to receiving sequence and traverse all sequence of mediators in the receiving sequence chain. Therefore, for this research, the receiving sequence is designed based on the following mediators as shown in Figure 18 below.

Figure 18. Message Receiving Sequence

Figure 18 shows, the incoming message received by the receiving sequence one (RS1) is checked by the filter mediator if result is returned the response would be forwarded to the Out-sequence of the proxy service; if no result is returned the message would traverse the sequence of mediators to shape the message in destination suitable format and finally forwarded to other backend data services as shown in Figure 16 above.

5.5.4 Webservice Design and Configuration

To integrate the DSSM solution to external systems like enterprise applications and to the data source through web service, a SOAP or REST protocol can be used. But in this research based on existing system implementation SOAP 1.1 protocol is used for the message communication between DSSM and other integrated applications. Therefore, the web service is defined based on Web Service Definition Language (WSDL) as discussed below.

WSDL: For this research implementation web service is defined for SIM card replacement business process two operations; GetProfile and UpdateProfiles. The GetProfile operation is for extracting customers profile information for a particular service subscription of a customer based on the service number provided. The update profile is defined based on inputs from the GetProfile operation to update the modified profile data based on the New SIM just replaced. A portion of the WSDL definition code is described as follows.

Table 4. partial WSDL file for SIM card replacement service.

```
<xsd:complexType name="GetProfileRequestType">
        <xsd:sequence>
<xsd:element name="Service_No" type="xsd:double"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="GetProfileResponseType">
        <xsd:sequence>
<xsd:element name="FIRSTNAME" type="xsd:string"></xsd:element>
<xsd:element name="MIDDLENAME" type="xsd:string"></xsd:element>
<xsd:element name="LASTNAME" type="xsd:string"></xsd:element>
                        <xsd:element name="REGION" type="xsd:string"></xsd:element>
                        <xsd:element name="WEREDA" type="xsd:string"></xsd:element>
                        <xsd:element name="HOUSENo" type="xsd:string"></xsd:element>
<xsd:element name="CUST_CLASS" type="xsd:string"></xsd:element>
<xsd:element name="CUST_STATUS" type="xsd:string"></xsd:element>
                        <xsd:element name="CUST_ID" type="xsd:double"></xsd:element>
                        <xsd:element name="SUB_ID" type="xsd:double"></xsd:element>
                        <xsd:element name="SERVICE_No" type="xsd:double"></xsd:element>
                        <xsd:element name="SERVICE_ID" type="xsd:double"></xsd:element>
                        <xsd:element name="NETWORK_TYPE" type="xsd:string"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>
```

SOAP Send and Receive Messages

The message format of the SOAP request and response messages are designed based on the above WSDL file, the source code for the new GetProfile request SOAP messages is created as shown in Table 3 below. The same way a request message for UpdateProfile service is created.

Table 5. GetProfile request message content.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:por="https://www.ethiotelecom.et/intranet/portal">
  <soapenv:Header>
    <por:Header>
      <!--Optional:-->
      <Username>?</Username>
      <!--Optional:-->
      <Password>?</Password>
      <BSID>?</BSID>
    </por:Header>
  </soapenv:Header>
  <soapenv:Body>
    <por:GetProfileInput>
      <SERVICE_No>?</SERVICE_No>
    </por:GetProfileInput>
  </soapenv:Body>
</soapenv:Envelope>
```

The SOAP message response for the GetProfile service is also created as shown in Table 4 below. And the same way the UpdateProfile service SOAP message response is created.

Table 6. SOAP message response format for GetProfile service.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:por="https://www.ethiotelecom.et/intranet/portal">
  <soapenv:Header/>
  <soapenv:Body>
    <por:GetProfileOutput>
      <por:Profile>
        <FIRSTNAME>?</FIRSTNAME>
        <MIDDLENAME>?</MIDDLENAME>
        <LASTNAME>?</LASTNAME>
        <WEREDA>?</WEREDA>
        <HOUSENo>?</HOUSENo>
        <CUST_CLASS>?</CUST_CLASS>
        <CUST_STATUS>?</CUST_STATUS>
        <SUB_ID>?</SUB_ID>
        <ICCID>?</ICCID>
        <PIN>?</PIN>
        <PUK>?</PUK>
      </por:Profile>
    </por:GetProfileOutput>
  </soapenv:Body>
</soapenv:Envelope>
```

5.5 Chapter Summary

The proposed Data Synchronization Solution Model design detail explanation is discussed in this Chapter. This Chapter is organized in Four sections; Section 5.1 discusses the design considerations which are made based on the situational analysis information in Chapter Four. Section 5.2 explains the proposed DSSM and its components at high-level. Section 5.3 discusses the proposed solution model architecture and its components design in detail. Section 5.4 discusses the proposed solution implementation design; which is based on the selected data service and middleware development tool called WSO2 enterprise integrator. It is free and open source. The implementation is designed based on designed webservices, different databases, and the WSO2ei functional components.

Chapter Six: Implementation and Validation

This Chapter aims to describe the full motivation and process of designing a data synchronization Solution model implementation based on the assumption of the SIM card replacement business process in ethiotelecom. The Previous Chapter (Chapter five) describes the proposed solution system design in great detail. In this Chapter, the DSSM hypothesis implementation and validation would be discussed. Subtopics of this Chapter discuss proposed solution implementation, validation plan, validation environment, validation procedure, and the actual validation implementation performed. The validation checks the practicality and functionality of the data synchronization solution model in creating data consistency among integrated application databases.

6.1 Proposed System Implementation

To implement the data synchronization model designed in Chapter 5, per the design described in Figure 15 the following design assumptions are implemented. Off all the message transactions among integrated application only the data (Create, Read, Update, Delete) CRUD data operations need to be processed in DSSM. This is due to the fact that these operations are the key operations that could create consistency/inconsistency to the enterprise data depending on its design and implementation. The request messages from enterprise applications are known services that can be identified by its business Service identification number (BSID) which is part of the SOAP message format. For this research implementation, only message with specified BSID would be further processed by DSSM and other messages would be forwarded/passed to the next layer which is existing middleware layer [Figure 12].

The implementation design in Figure 16 is based on wso2 enterprise integrator version 6.2.0 components. As shown in the diagram excluding the knowledgebase, the DSSM solution has three major components; the proxy service (including the in-sequence, Out-sequence, and Fault-sequence), the Receiving Sequence, and the Backend services. All the three components and other important components configuration is discussed as follows.

7.1.1 Proxy Service Configuration

A proxy service is a virtual service with three major sequences; the In-Sequence, Out-Sequence, and Fault-Sequence [Section 5.3.2] [Section 5.4.2]. The DSSM proxy service is configured based on the components described in the DSSM implementation design which is shown in Figure 16

above. Per the implementation design, the proxy service has three main message flows; the In-flow/In-sequence, the Out-flow/Out-Sequence, and the error-flow/fault-sequence. Per the DSSM design diagram the proxy service configuration, the partial graphical view is shown in Figure 19 below.

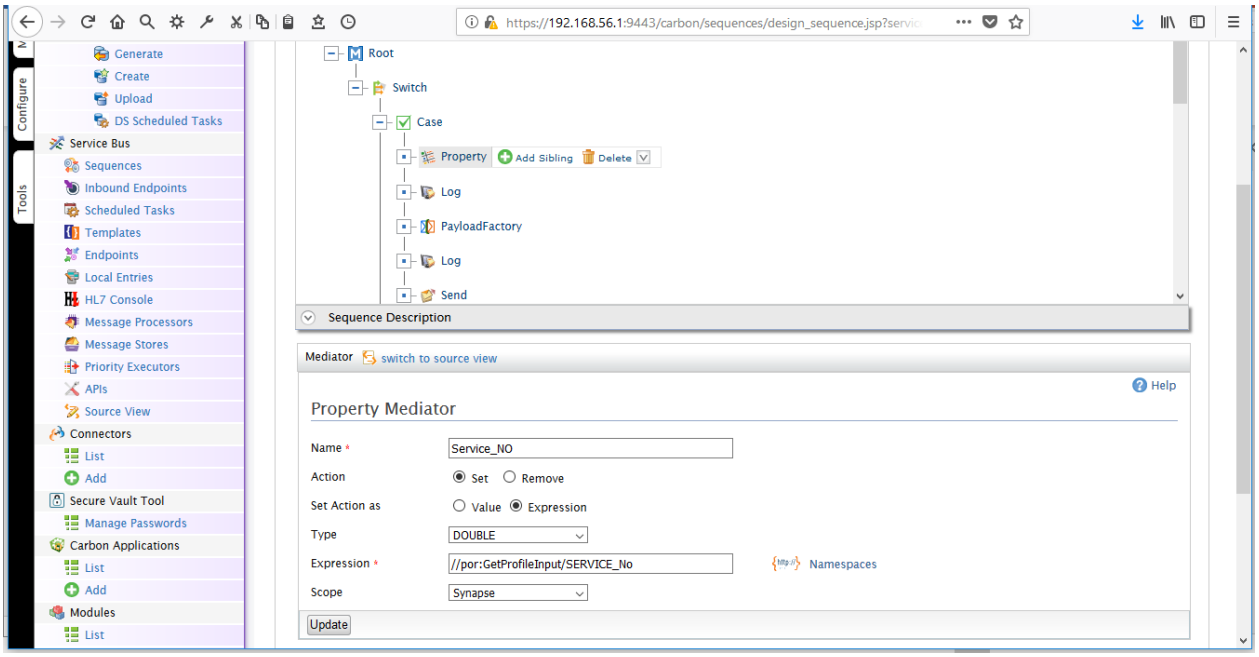


Figure 19. DSSM proxy service configuration partial view.

Figure 19 above shows the DSSM proxy service In-Sequence partial configuration with its selected sequence of mediators per the design diagram in Figure 16. The same way the Out-Sequence and the Fault-Sequences of the DSSM proxy service designed flow is configured. In this configuration the Switch mediator is the first key mediator that filters the incoming message by its Business Service ID (BSID); if the BSID is not specified in the Switch cases statement, the message would be forwarded to the default case which is log and/or drop the message per this research design. But in real implementation, the message would be forwarded to the next layer which is the middleware in existing architecture. Partial configuration code for the Switch mediator is shown in Table 6 below. After the switch mediators a sequence of other mediators such as Property mediator, log, payload factory, store, and send mediators are used. The detail about what these mediators are and what they do to the message is discussed in Section 5.4.1 above.

Table 7. Message filter sample configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<switch source="por:Header/por:BEID" xmlns="http://ws.apache.org/ns/synapse">
   <case regex="101">
      <property expression="//por:GetProfileInput/SERVICE_No"
        name="Service_NO"                    scope="default"                    type="DOUBLE"
xmlns:DS="http://ETISLGHWS0002:8280/services/DataServiceGetProfile"/>
      <log level="custom">
        <property expression="$ctx:Service_No" name="Service_No"/>
      </log>
          .
          .
          .
```

7.1.2 Receiving Sequence

Receiving sequence is a chain of mediators that can be used to receive a response message sent by the in-sequence of a proxy service. Unlike the usual or default message flow in which a response from a service is directed to the Out-sequence of a proxy service, the response from the backend service would be received by this receiving sequence. The incoming response message traverse the sequence of mediators in the receiving sequence chain. The DSSM Receiving-Sequence configuration graphical view is shown in Figure 20 below.

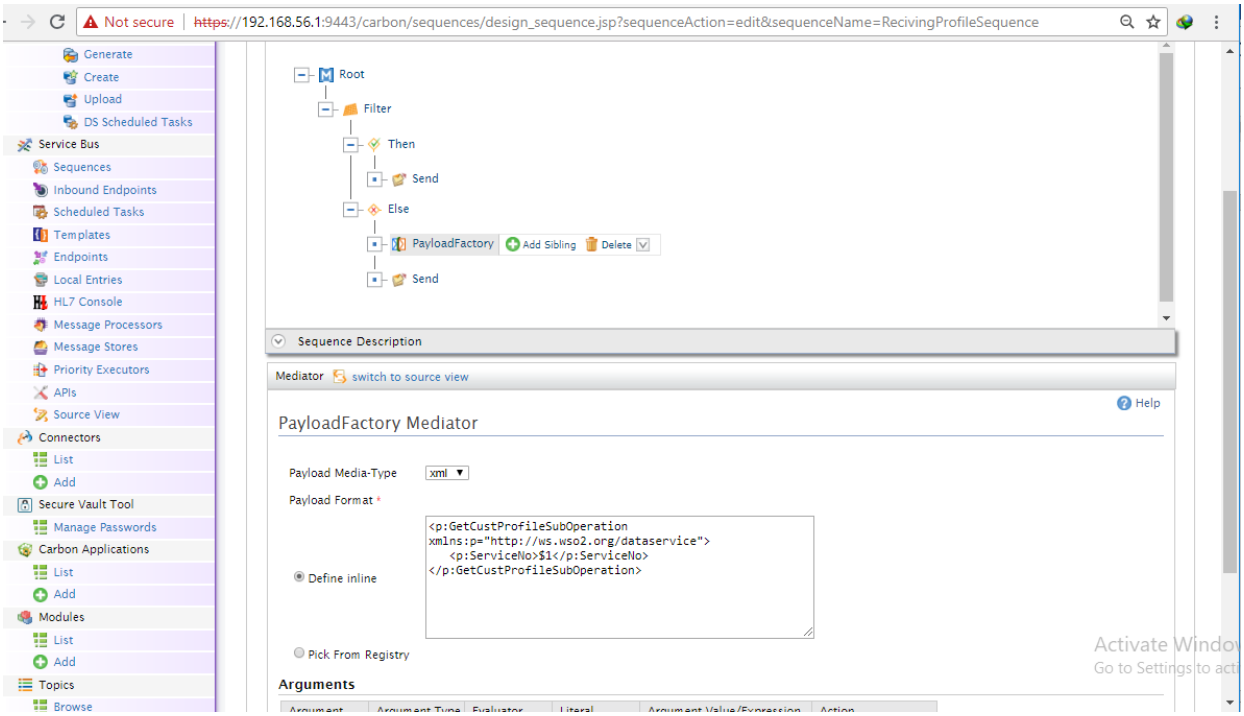


Figure 20. DSSM Receiving-Sequence Configuration Graphic View.

7.1.3 Data-Source Integration

To integrate the Data-Sources to the DSSM data service there are some prerequisites; install JDBC driver for Oracle 11g driver and enable the Relational Database Management System (RDBMS) as a data service in WSO2 enterprise integrator. For this research let's see the Knowledgebase integration to the DSSM data service. As explained in the detail design of the Knowledgebase in Section 5.3.1, the database design is implemented using Oracle 11g database. To integrate this database with the DSSM, JDBC driver for Oracle 11g is installed and the data service components are successfully configured; the configuration screenshot is shown in Figure 21 below.

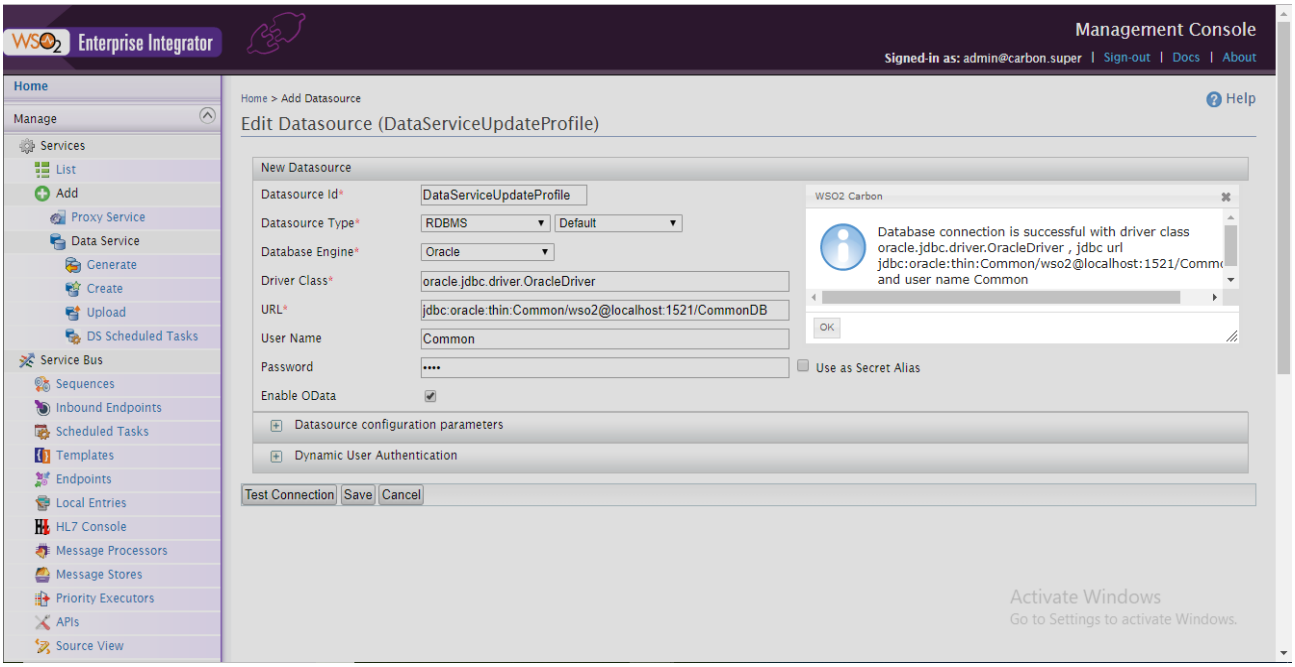


Figure 21. Oracle data source integration with DSSM.

After the database integration/binding to the WSO2 enterprise integrator data service complete, query configuration for the GetProfile and UpdateProfile data service is configured. Then Input and Output data mapping is configured to normalize the data format of the incoming SOAP message and the message format of the databases as shown in Figure 22 below.

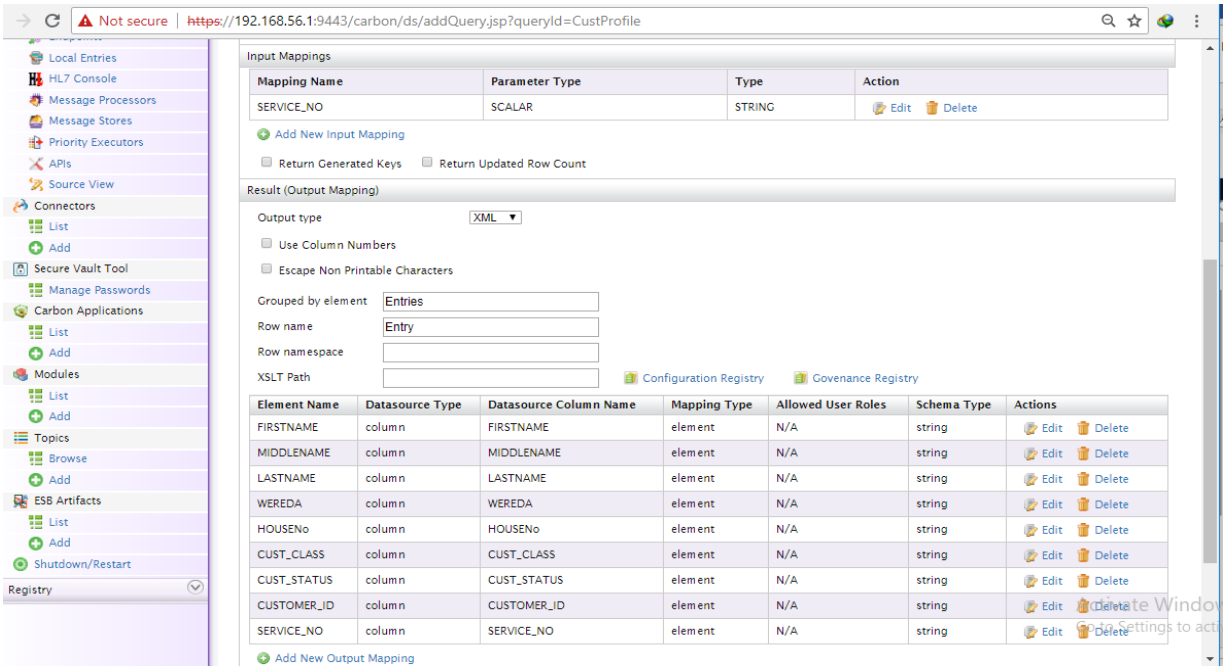


Figure 22. DSSM data service Input and Output mapping.

The two Operations created in DSSM implementation are the GetProfileOperation and the UpdateProfileOperation to execute the GetProfile and UpdateProfile queries defined above and the backend service configuration is finalized.

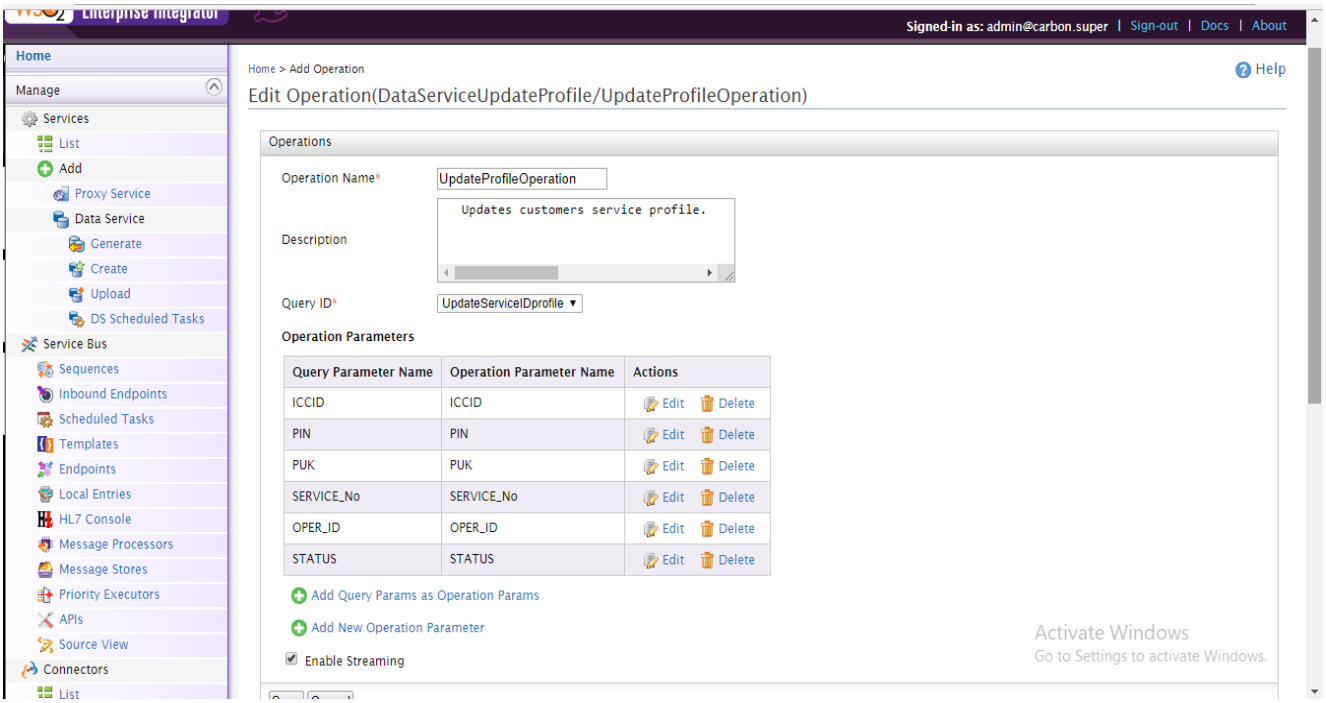


Figure 23. DSSM Data service UpdateProfileOperation Sample configuration.

7.1.4 Backend Services Configuration

wso2 data services component provides a service and resource interface to some data which is stored in a relational database. In a service interface, service requests can be mapped to queries against collections of tables in a relational database and query results also can be mapped to service responses [38]. The data service resource is defined using the Data Service Resource Language (DSRL); in which the <data> element defines a data element resource. In this data service data configurations, queries, operations, resources, and event-triggers can be defined. The data service is configured in three steps; the first step is binding/integrating the data sources to the DSSM data service. the second step is defining queries depending on the requirements what to do on the data sources. The third step is defining the data service operation.

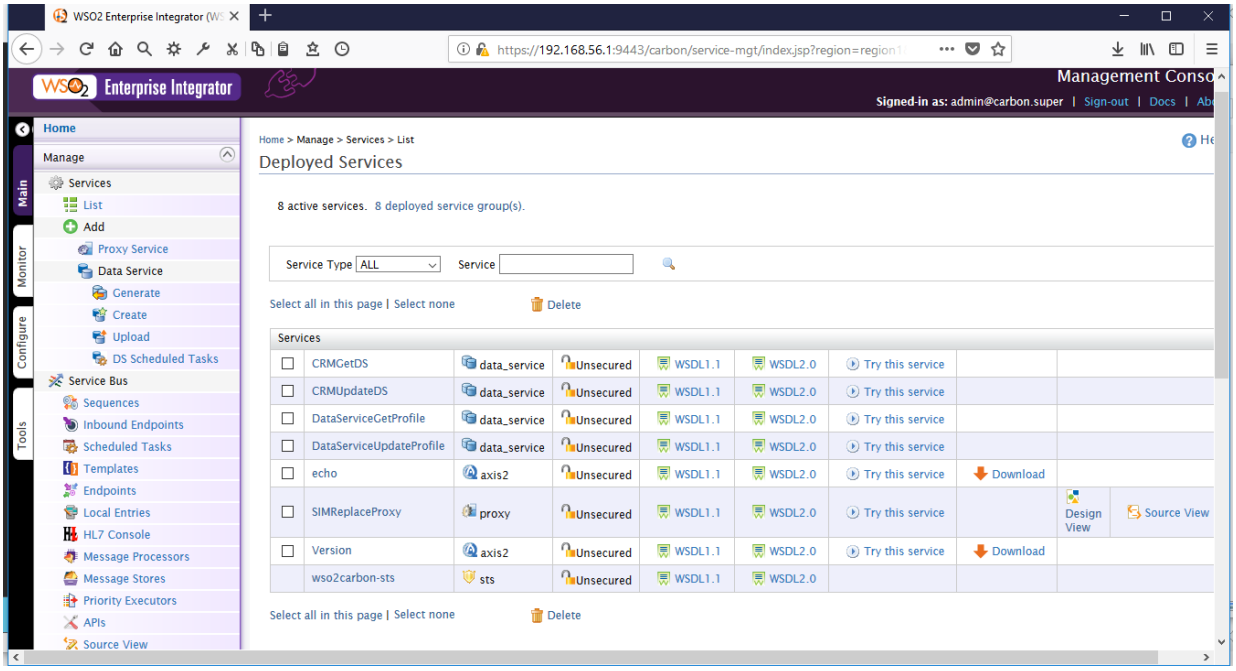


Figure 24. DSSM Deployed services Screenshot.

5.5.1 Endpoint Configuration

An endpoint is a specific destination for a message such as an address, WSDL, a failover group, destination services etc. [Section 2.8]. The endpoint sample configuration for the DSSM backend services is shown below.

Table 8. Service Endpoint sample configuration.

```
< endpoint xmlns="http://ws.apache.org/ns/synapse" name="DSgetProfileEP">
  <address uri="http://ETISLGHWS0002:8280/services/DataServiceGetProfile" format="soap11">
    <suspendOnFailure>
      <progressionFactor>1.0</progressionFactor>
    </suspendOnFailure>
    <markForSuspension>
      <retriesBeforeSuspension>0</retriesBeforeSuspension>
      <retryDelay>0</retryDelay>
    </markForSuspension>
  </address>
</endpoint>
```

7.2 Proposed Solution Validation

In this Section, the validation plan, the research environment in which the research is implemented and validated including the resources used to implement and validate the proposed solution model, the validation of the proposed solution model functionality is discussed.

7.3.1 Validation Plan

The purpose of this validation is to check the practicality of the data synchronization solution model through some functional and nonfunctional parameters. The functional components of the data synchronization model is designed and implemented based on ethiotelecom's SIM card replacement business process. Since the real enterprise environment is highly sensitive, and it is not possible to validate the data synchronization solution model in the real enterprise environment; therefore, the validation is performed by simulating and creating a heterogeneous virtual enterprise environment.

To validate the proposed solution two important issues, need to be considered and addressed. The first is the primary purpose of the data synchronization solution, which is to create data consistency among integrated applications. The second is the proposed solution is a new data synchronization solution model, which is built as an additional data synchronization layer in existing EAI layered architecture [Section 5.2 Figure 12]; the practicality of this hypothesis would be validated in this Section.

To validate the DSSM the validation procedure [Annex A] is prepared based on SIM card replacement business process in ethiotelecom. The SIM card replacement service is selected due to the fact that out of the data CRUD operation, validating this service involves the three operations (Create, Read, and Update). The business process is composed of simple procedures

[Figure 9]. The salesperson queries customer profile information based on the requested service number, the system responds the customers detail profile requested; then the salesperson cross-checks the customers profile validity from the response with the information provided by the customer; and update the customer profile data based on the new SIM card information; then issue the new SIM to the customer [Section 4.2]. In this business process validation mainly read data, and update data operations are involved.

7.3.2 Validation Environment

To create a real resemblance of the heterogeneous enterprise environment for this validation case, the authors create a heterogeneous operating systems platform and heterogeneous database environment using physical and virtual machines. By a validation environment, it is meant the whole hardware and software infrastructure as well as the configuration of internal subsystems involved in the specified scenarios. To create the validation environment the following resources are used as described in Table 9 below.

To validate the proposed solution a mock service (which is a dummy service only for testing purpose) is created using SOAPUI version 5.4.0 tool and the "tryit" module of the WSO2 enterprise integrator is used. The mock service is created from already designed GetProfile and UpdateProfile SOAP request and response messages [Section 5.1]. To validate the above-mentioned plan in Section 6.2.1, a heterogeneous enterprise environment is simulated by creating heterogeneous database environment in a heterogeneous operating system platform.

Table 9. Resources for validating the Data Synchronization Solution Model.

Tools	type	use	Description
Physical machine	Laptop Dell Latitude E5430	To perform the research.	Intel core i5 2.80Ghz each, 8GB RAM, x64bit based processor, 500GB Hard disc capacity. windows 10 enterprise edition 64-bit operating system.
Virtual machine	Using Oracle Virtual box	To install and use MySQL database for CRM system.	The virtual machine is required to create another independent server that can Simulate the CRM database environment.
Operating systems	windows 10 enterprise edition 64-bit	For DSSM solution implementation and testing.	To install WSO2ei 6.2.0, Oracle 11g database, java JDK8, Apache Maven, apache ant, SOAPUI 5.4.0, and other tools for DSSM implementation and test.

Tools	type	use	Description
	operating system.		
	Windows Server 2012R2	For CRM database installation based on MySQL database.	MySQL database is installed in this server to implement CRM databased already designed.
Databases	Oracle11g MySQL		To simulate heterogenous database environment.
WSO2	application	Middleware	Opensource middleware application
Java developer studio	Oracle JDK 1.8 /Open JDK 1.8	Platform	JDK 1.8 or above is a platform to run WSO2 carbon-based applications and to run apache ant and apache maven.
Apache Ant	Version 1.7.0 or later	Tool	To build the product from the source distribution (both JDK and Apache Maven are required).
Apache Maven	Version 3.0.x		JDK 1.8 or above is a platform to run WSO2 carbon-based applications and to run apache ant and apache maven.
Drivers	JDBC Oracle Driver	Driver	To integrate the Knowledgebase (Oracle 11g) to the data service of the DSSM solution.
	JDBC MySQL Driver		To integrate the CRM database (Oracle 11g) to the data service of the DSSM solution.
Web Browser		Browser/ interface	To access each product's Management Console. The Web Browser must be JavaScript enabled to take full advantage of the Management console.

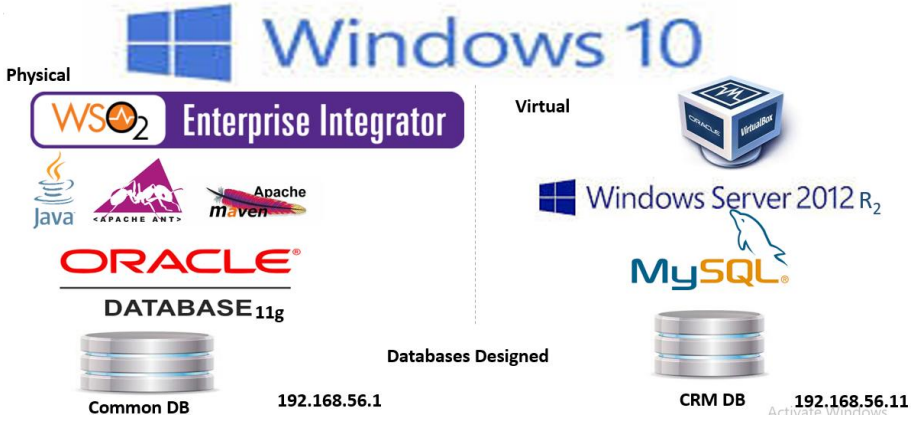


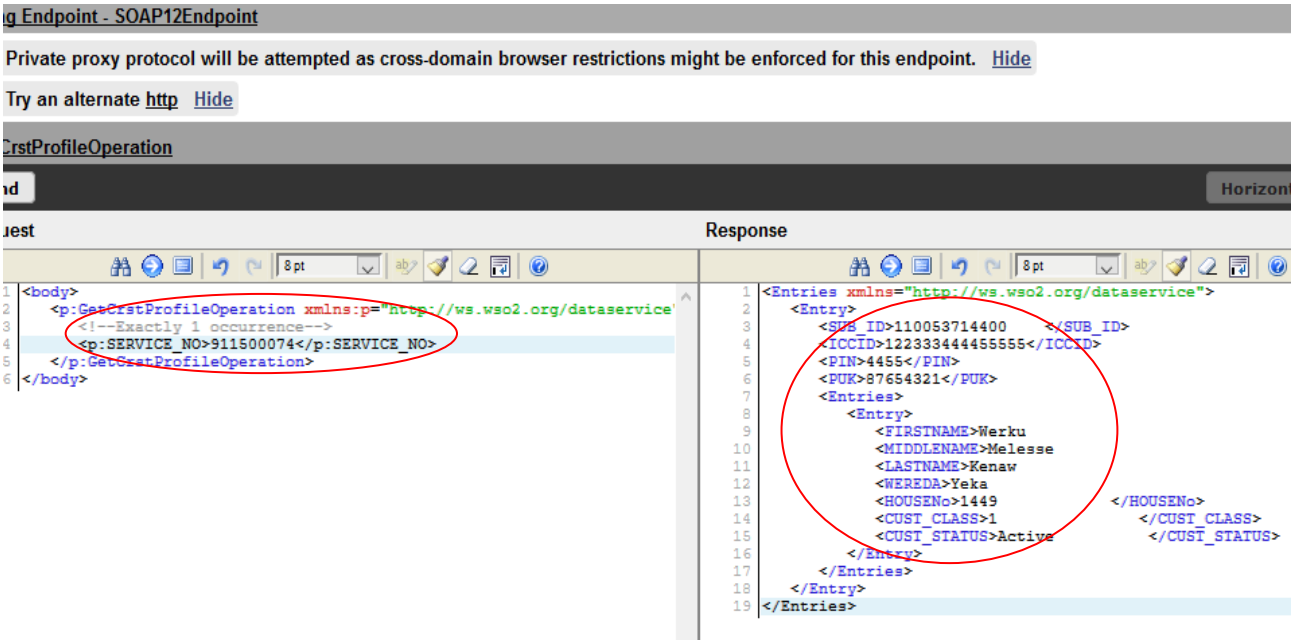
Figure 25. DSSM validation environment.

7.3.3 Proposed System Functionality Validation

To show the practicality of the proposed data synchronization solution model (DSSM) hypothesis, implementation of DSSM is designed based on WSO2 enterprise integrator version 6.2.0 components as described and discussed in Chapter Five. But in this subsection, the DSSM designed and implemented would be validated for its functionality.

Get Customer Profile Information

Based on the SIM card replacement business process [Section 4.1] the activities of the salesperson (through client-side sales application terminal) is simulated using SOAPUI mock service and the "tryIt" interface of the WSO2ei. For SIM replacement service, the salesperson sends the request based on the customer Service Number; then the customers profile response would be displayed. Therefore, the validation result screenshot in Figure 26 below shows, the successful request and response of the GetProfile Service performed by the DSSM. In the existing enterprise environment, the GetProfile Service operation response is the same; but the information retrieved from the CRM system may not be the same with the data in CBS system. For example, according to the customer profile retrieved from CRM system the customer status may be Active; that means the salesperson can replace the SIM card and update the new SIM information. But unlike in the CRM system the customer status in CBS may found blocked, therefore even if the customer gets new SIM card, he couldn't connect to the network for his status is blocked in CBS system.



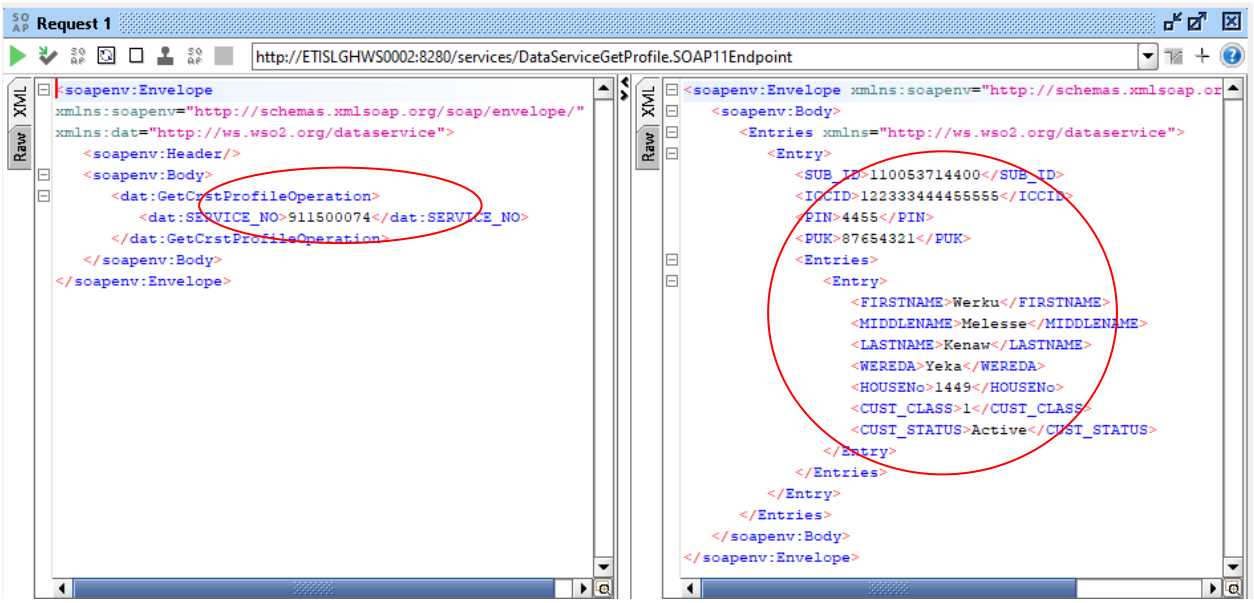


Figure 26. Webservice SOAP message request and response for GetProfile operation.

The left pan in Figur 26 shows the customer service number which is used to retrieve the customer's detail profile information. The right pan shows the successfully retrieved customer profile detail.

Update Customer Profile Information

The validation results in Figure 27 below shows, the successful request and response for the UpdateProfile Service performed by DSSM per the design in Section 5.4.1. It shows the customer profile is modified based on the new SIM information and submitted for update; in response, successful update notification returns a value one as a response. Per the design in Section 5.4.1 Figure 16 the update would be propagated to other applications participated in the integration through notification message to update their profile information. Moreover, the main data reference point for all integrated applications is designed to be the knowledge base. Therefore, unlike the existing implementation in the proposed solution the customer status in the Knowledgebase would be referred by both CRM and CBS systems and the customer who replaces his SIM card can connect to the network and get telecom services.

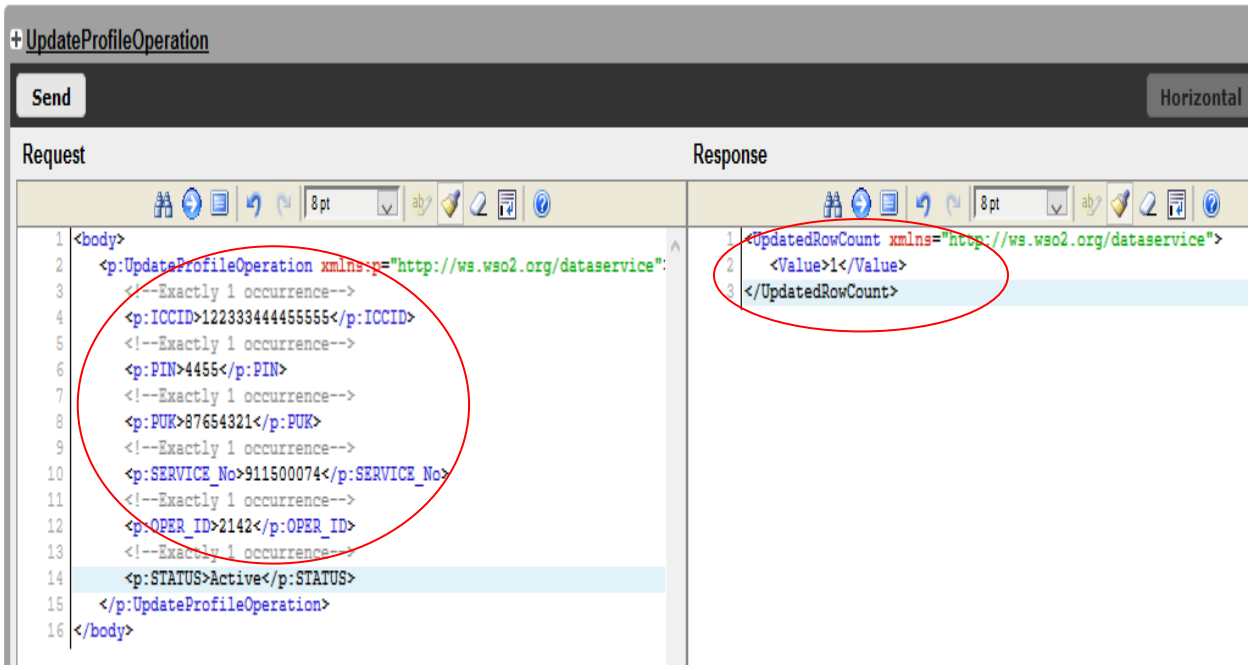


Figure 27. Successful customer profile update.

In Figure 27 above, the left pan shows the modified customer profile data based on the new SIM card information which updates the previous customer profile. Whereas, the information in the right pan shows the successful update conformation response message that indicates the UpdateProfile service can update customer profile successfully.

7.3.4 Data Consistency Validation

For data consistency test different customer profile data is entered into the databases. Then the data is modified several times through the GetProfile and UpdateProfiel service as shown in the functional tests Figure 27 above. To validate data consistency and synchronization, the query expressed in Annex A Section three is executed for multiple databases participated in the integration. And the result is compared for similarity per the data consistency validation procedure defined in Annex A. The result obtained is shown in Figure 28 and 29 below.

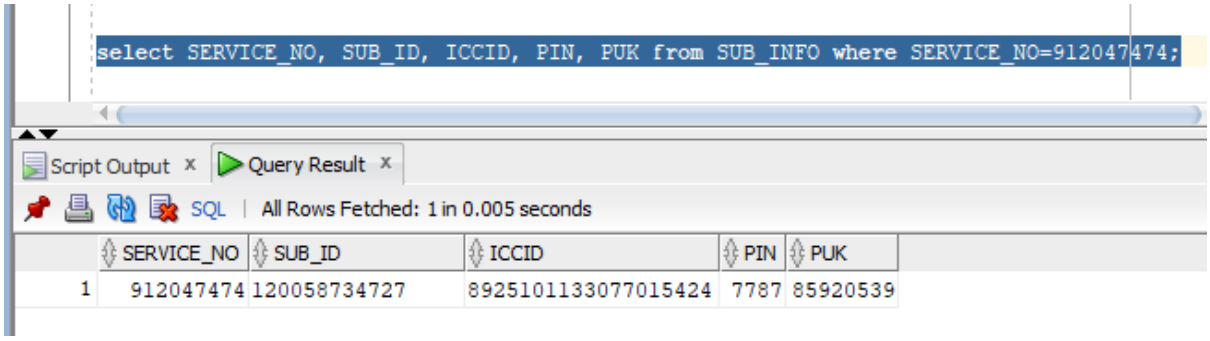


Figure 28. Query result from CommonDB (Oracle database).

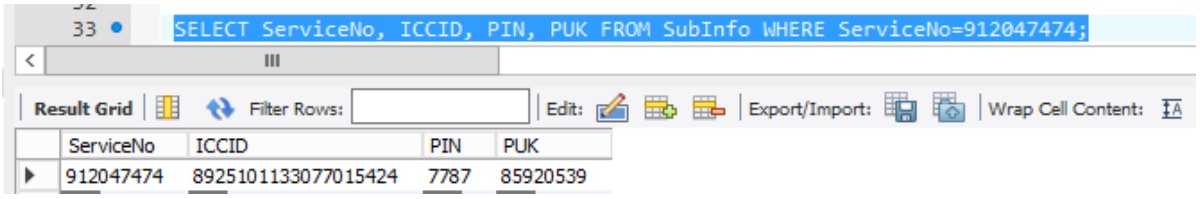


Figure 29. Query result from CRMdb (MySQL database).

7.4 Chapter Summary

In this Chapter, the proposed system implementation and validation is discussed in detail. Section 6.1 discusses assumptions and considerations of the proposed system implementation based on the selected WSO2 enterprise integrator components. In this Section, major components of the DSSM such as the proxy service and its subcomponents, the receiving sequence, the data sources and service endpoints are designed and configured. In Section 6.2 the DSSM is validated for its practicality to attain its intended data synchronization purpose in EAI environment. In this Section the validation plan which is a guiding plan for DSSM validation is explained, then the simulated virtual validation environment in which the DSSM implementation and validation is performed; then the proposed system is validated for its functionality and data consistency test is applied.

Chapter Seven: Result, Conclusion and Recommendation

8.1 Chapter Introduction

In previous Chapters the data synchronization solution model concepts are discussed by referring and referencing the state of the art and related works through literature about data synchronization in integrated enterprise applications; the situational analysis of the company in which this research is based on is discussed in Chapter four, the design, design considerations of the proposed DSSM is discussed and finally the implementation and validation of the DSSM is discussed in Chapter five and six. In this Chapter, the overall implementation and validation result of the proposed DSSM is discussed and interpreted. Conclusion, Recommendation, Future work, and Limitations of the research is also discussed in subsections of this Chapter and finally Chapter summery would conclude this Chapter and this research document.

8.2 Result

In this Section, the validation result is gathered together to analyze and comment on the proposed DSSM functionality evaluation. The detail validation performed is explained in Section 6.2; which is based on the validation procedure described in Annex A. Per the design described in Chapter Five, the proposed DSSM implementation is demonstrated and validated for its feature and operational behavior. The functionality validation document is prepared for each service implemented per the SIM card replacement business process as described in the functionality validation Section 6.2.3 for GetProfile and UpdateProfile data services.

As per the validation document Section one, the functionality of the GetProfile Service is evaluated, and all the procedures are tested and passed successfully. The same way in Section two, the functionality of the UpdateProfile service is validated and all the procedures are successfully passed the functionality validation. Also, the validation result performed in Section three which is data consistency validation shows that all the procedures in the validation document is passed.

Therefore, the result obtained from the DSSM functionality validation shows that all the functional requirements implemented in DSSM are fully functional per the design described in Chapter five and per the validation document in Annex A. The GetProfile data service could fetch the customer profile information and per the customer profile information obtained data modification can be made and the UpdateProfile data service could successfully update the customer profile as shown in the sample screenshot Figure 27.

8.3 Discussion and Interpretation

This research is conducted to provide a generic data synchronization solution in a heterogeneous enterprise environment. The research uses the ethiotelecom existing enterprise environment as a basis for designing and implementing the DSSM. In addition, the company real working environment is investigated, and the information gathered is used to build the proposed data synchronization solution model. For example, the existing EAI architecture, the messaging protocol and message format used, the nature of the data synchronization problem, the type of applications and middleware used in the integration, and other environmental variables like the Service Level Agreement (SLA) and vendor management issues that can be used as an input for the DSSM design and implementation are pre-investigated and taken as an input. Song and Ging [10] also could provide a data synchronization solution by investigating the cement enterprise real working environment.

However, in the existing system, as explained in the problem statement [Section 1.3] there is data inconsistency between integrated enterprise applications. For example, as shown in Figure 27 below, for the same customer service number, his subscription status is idle in CRM application and Barring in CBS application. Such inconsistency in integrated enterprise applications indicates the data synchronization problem in the integration environment.

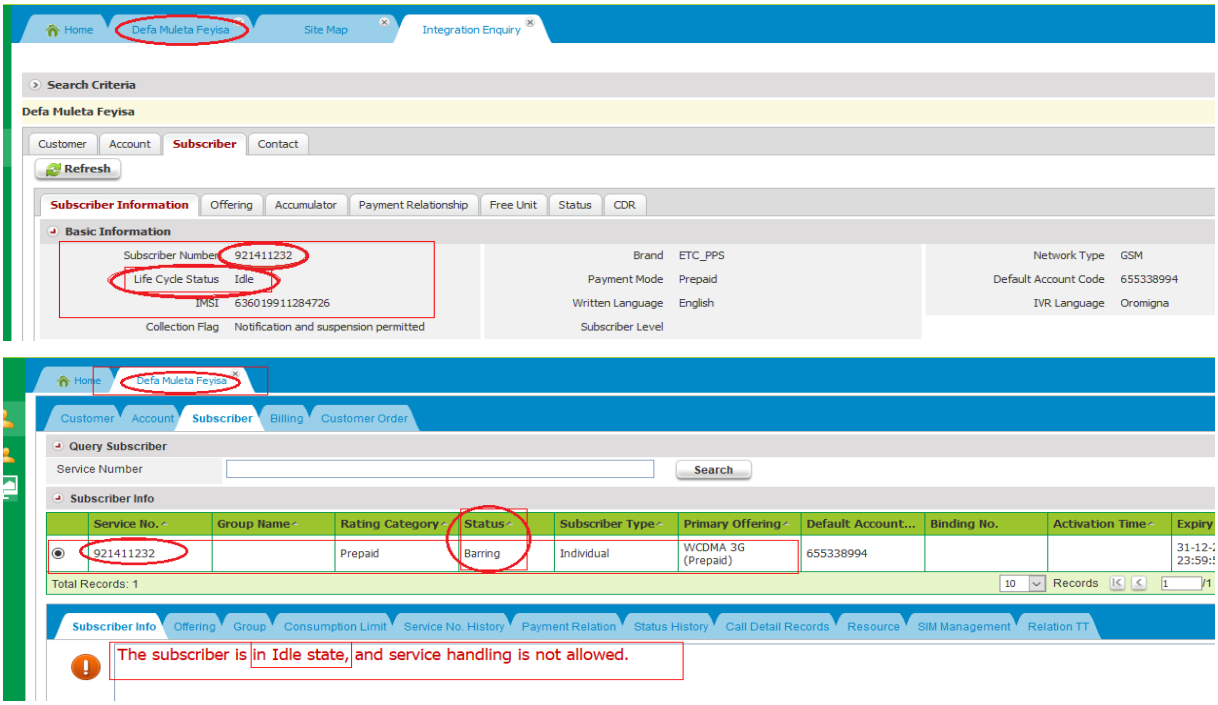


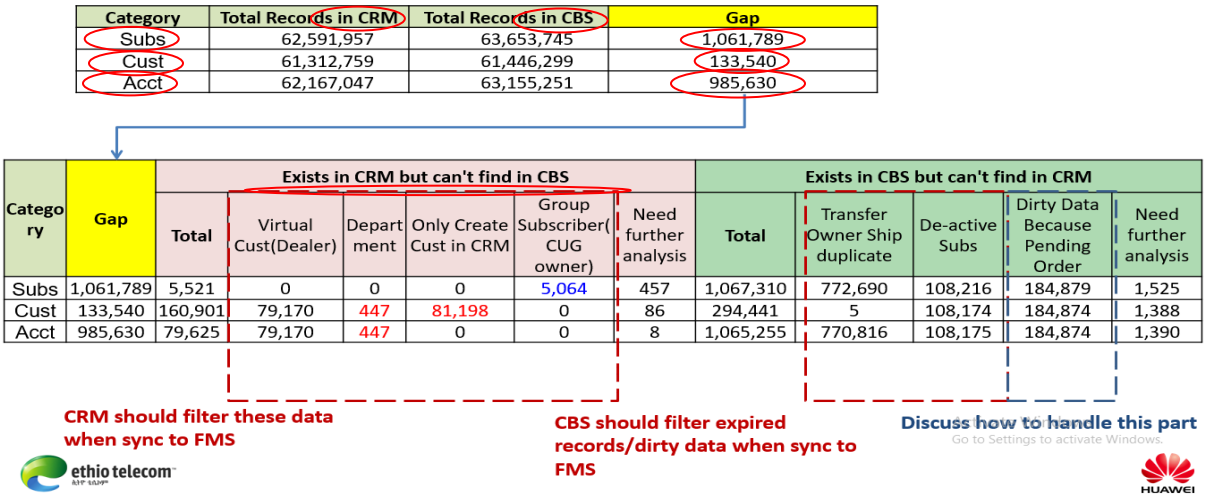
Figure 30. Data Inconsistency example in the existing system.

The objective of the DSSM is to create data consistency among integrated applications and to synchronize the data in applications' datastore. Data consistency can be achieved easily by creating a single/common data reference point for all integrated applications. And data synchronization can be achieved by exchanging notification message for any data change in the integration environment so that all integrated applications can consistently update their datastore. To achieve this, in the DSSM design a knowledgebase is used as a common data reference point for all integrated applications; and for the message exchanged a middleware as part of the DSSM is designed using WSO2 enterprise integrator tool.

The proposed solution is designed to be implemented in a heterogeneous enterprise environment, in which heterogeneous enterprise data exists. To maintain data consistency among the heterogeneous enterprise data sources and to maintain data synchronization among different data stores, different techniques that can maintain data consistency and avoid data conflict are used. Such as the JDBC drivers installed for each database types in the DSSM implementation is used to connect the DSSM data service endpoint to different application databases. The Input and Output mapping used in WSO2 data service is used to avoid entity naming and data type conflict, the use of factory payload mediator in the DSSM design is used for message transformation requirement to exchange data from one message format to another, and a data notification technique [14] [Section 2.8] is used to exchange update in the integration environment through notification message and it keeps the different application datastores synchronized and UpToDate.

The data consistency validation performed per the validation procedure in Annex A shows the data among integrated applications datastore is consistently synchronized. Mihaela Iridon [14] in her data synchronization model also obtain the same result using update notification message for data synchronization among integrated applications. But unlike the DSSM which uses SOAP messaging, the message notification she uses is broker-based messaging architecture.

Due to the data synchronization problem in existing system the company report generated from different applications was inconsistent and shows the big difference as shown in Figure 29 below; the data synchronization achieved by the new DSSM will illuminate such report inconsistency and report mismatch from different integrated applications.



Category	Total Records in CRM	Total Records in CBS	Gap
Subs	62,591,957	63,653,745	1,061,789
Cust	61,312,759	61,446,299	133,540
Acct	62,167,047	63,155,251	985,630

Category	Gap	Exists in CRM but can't find in CBS						Exists in CBS but can't find in CRM				
		Total	Virtual Cust(Dealer)	Depart ment	Only Create Cust in CRM	Group Subscriber(CUG owner)	Need further analysis	Total	Transfer Owner Ship duplicate	De-active Subs	Dirty Data Because Pending Order	Need further analysis
Subs	1,061,789	5,521	0	0	0	5,064	457	1,067,310	772,690	108,216	184,879	1,525
Cust	133,540	160,901	79,170	447	81,198	0	86	294,441	5	108,174	184,874	1,388
Acct	985,630	79,625	79,170	447	0	0	8	1,065,255	770,816	108,175	184,874	1,390

Figure 31. Data synchronization problem/Gap analysis report 2016 third quarter.

Figure 31 shows the Subscriber, Customer, and Account status report gap due to the data synchronization problem. The report is generated from CBS and CRM applications and the comparison for the Subscriber, Customer and Account status is made to identify the report gap between these applications.

8.4 Conclusion

This research has been conducted to answer the research question raised in section 1.3; what data synchronization solution can be applied to create data consistency in heterogenous EAI environment. To answer the research question, a Data synchronization solution model is designed based on middleware and knowledgebase. The Knowledgebase is designed to provide a single common data repository and data access point for all applications participated in the integration. Whereas, the Middleware is designed to process the message exchanged among integrated applications so that the message can be forwarded in a destination suitable format.

The general objective of the research [section 1.4], which intends to provide a service-oriented data synchronization solution model and all the specific objectives; are achieved through the activities performed in each chapter of this thesis document. In Chapter Two and Chapter Three different literatures have been referred to understand the state of the art and worlds' experience to deal with data synchronization problem in EAI environment. As described in Chapter Four the situation in which this case study is conducted has been analyzed through the use of the research methodologies for situational analysis described in section 1.5; in this Chapter the problem area, techniques and technologies used, the depth and detail of the problem, and the environment in

which the proposed solution is going to be implemented is studied. Based on the situational analysis findings in Chapter Four, in Chapter Five, some design considerations are made and a data synchronization solution model has been designed based on the considerations using SOA design principles. In Chapter Six, for the Solution model implementation, the service components are designed using an opensource WSO2 enterprise integrator tool. The solution model is validated by simulating a heterogenous virtual enterprise environment. Chapter Seven is the last chapter which contains the validation result, discussion, recommendation, limitation of the research and future work.

The design, implementation, and validation of this research shows that, the features and components of DSSM are functioning per the design specified in this research [Chapter Five]. The proposed DSSM practicality in synchronizing data in EAI environment is successfully validated by simulating a heterogenous enterprise environment using virtual machines. Moreover, by introducing additional data synchronization layer in EAI layered architecture, more scalability and flexibility can be achieved in terms of including other modules and data sources into the integration environment and in terms of modifying and maintaining the solution.

Therefore, Data integration and synchronization among integrated enterprise applications can be achieved through a custom integration architecture by carefully designing SOA based integration patterns based on integration middleware and knowledgebase. The research question raised in section 1.3 is answered by providing DSSM based on middleware and knowledgebase. The solution is successfully validated for its functionality and practicality in synchronizing data among integrated applications by creating a heterogenous enterprise virtual environment that highly resembles the environment in the case study. In the meantime, all the general and specific objectives of the research specified in Section 1.5 are achieved through this research activities as described chapter by chapter in this Thesis document.

8.5 Limitation of the Research

Unlike other researchers [9] [10] [14] who implement their research in real working environment, this research is implemented and validated in simulated heterogeneous virtual environment that highly resembles a real enterprise environment. This intern creates its own impact on the validation process. For example, in real environment, customer profile update is performed by first requesting the customer profile detail by his service number, then on the same screen the customer profile would be modified and sent for update. But in this DSSM validation, there is no client-side application, the test is performed based on the WSO2 ‘tryit’ feature and through

SOAPUI mock service; these tools could not automate GetProfile and UpdateProfile services in one screen. Therefore, the test is performed for GetProfile and UpdateProfile services separately.

In general, the authors describe the identified limitations of this research as follows.

1. The proposed solution is designed to scan and filter every message being transacted in the integration environment. This in-turn introduces delay, at list for the amount of time required to filter a single message which does not exist in the previous implementation.
2. It would be better if it was possible to test and validate the output solution of this research in real ethiotelecom working environment. But due to the sensitivity of the real ethiotelecom environment validating in real environment was not possible. Therefore, testing and validation is performed by simulating a heterogeneous enterprise-like environment using multiple databases and multiple operating systems in a virtual environment.
3. In the real enterprise integration environment, large number of applications usually involve in the integration. Unlike the real integration environment due to Hardware and software resource limitations, this research is tested only with two applications.
4. This research mainly focuses on testing the practicality of the hypothesis of implementing DSSM in existing EAI architecture; It would be preferable if it was possible to validate all the CRUD data operations. But based on the SIM card replacement business process, only the Read, Create and Update operations are Validated.
5. Due to the scope of this research, the client-side application is not developed. And the test is performed by using the SOAPUI mock service, and the “tryit” module of the WSO2ei, not with real client-side applications.
6. In the DSSM design, the knowledgebase is designed to be accessed by all applications participated in the integration for the common data exchanged between the applications. Such a design may create high pressure on the knowledge base to respond to all applications in real time. In real implementation, it may require some load balancing and implementation on multiple servers.

8.6 Recommendation

While designing and implementing enterprise applications, it is very important to study and consider the nature of the application's data required to integrate and exchanged between other integrated enterprise applications. Otherwise after implementation correcting and finding other workaround for compatibility is more expansive in terms of system performance, time, money and other resource.

The company needs to have a long-term infrastructure and application platform that considers the current and upcoming future applications. The trend so far is costing the company a lot of time, money and effort; because for every project introduced in the company the existing infrastructure requires redesign and major modification.

The existing Enterprise service bus is not serving all enterprise applications, rather it serves the NGBSS applications and some other third-party applications like e-CAF and ERP. It would be better for the company to carefully design and building enterprise-level service bus that could mediate all the current and upcoming enterprise applications including OSS applications.

8.7 Future Works

We believe that the DSSM can add modular service component like Froude suspect recommendation system, Security Policy enforcement, business rule implementation, etc. can be implemented in the data synchronization solution layer; without making major change to other enterprise applications and/or without violating the vendor's privacy policy. For the future, data recommendation system which could assist salespersons about previous data entry so that salespersons could avoid mistakes in data entry and to avoid/minimize data duplication is planned as a future work.

The proposed DSSM has a known performance limitation due to the message filter introduced in the normal message flow. However, the magnitude of the delay/performance tradeoff impact in the data exchange is not studied. For the future, it would be essential to study and improve the performance of DSSM.

Bibliography

[1] I. J. o. C. Applications, "Challenges and Future of Enterprise Application Integration," vol. 42, no. 7, p. 975, March 2012.

[2] J. WU, " assessment of EAI Problems in ethiotelecom," in *the Seminar Report Document AAU-AAIT, January 2018)*, Melbourne, Australia, July 14-17, 2012.

[3] D. S. Linthicum, Next Generation Application Integration: From Simple Information to Web Services, Addison Wesley, August 2003.

[4] E. C. Document, Ethiotelecom Department of Quality Assurance; Addis Ababa, 2016.

[5] W. L. a. v. Shankararaman, "Enterprise architecture and integration: Methods, Implementation, and Technologies.," USA, 2007.

[6] Z. Ming-bro, "Model-driven enterprise service Bus," in Proceedings of the 2009 International Symposium on Web Information Systems and Applications (WISA'09), Nanchang, P.R, China, May 22-24 2009.

[7] D. S. L. LLC, "Defining, Designing, and Implementing SOA-Based Data Services," 2009.

[8] F. Saleem, "Enterprise Application Integration as a Middleware," in *Science and Information Conference*, London, UK, August 2014.

[9] N. Malhotra, A. Chaudhary, " Implementation of Database Synchronization Technique between Client and Server," International Journal of Engineering Science and Innovative Technology (IJESIT), Volume 3, Issue 4, July 2014.

[10] S. D. a. J. Shaohong, "Data Synchronization Solution in Cement Enterprise;" in *Proceedings of the 35th Chinese Control Conference July 2*, Chengdu, China, July 27-29, 2016.

[11] X. Wei, "Heterogeneous Database Integration Middleware Based on Web Service," in *International Conference on Applied Physics and Industrial engineering*, Weifang, China, 2012.

[12] Ethiotelecom, Ethio-ICD-CBS-ETH Switch V2.0 Signed Version, 2018.

[13] E. I. Document, *ethio-ICD-CRM e-CAF,* Addis Ababa, 2014.

[14] M. Iridon, "A Practical Enterprise Data Integration and Synchronization solution," in *Enterprise Integration Modeling:* Dallas, USA, 2015.

[15] A. X. R. Zhitong Su, "Application of data Synchronization Based on ESB;" in *2nd International Conference on Geoscience and Remote Sensing;* 2010.

[16] D. L. P. W. tz, Service-Oriented Architecture: An Integration blueprint., Birmingham Mumbai: Packet Publishing enterprise., 2010.

[17] J. Bean, SOA and Web Services Interface Design: Principles, Techniques, and Standards. 1st Edition, Morgan Kaufman publisher, 2009.

[18] T. Erl, SOA Principles of Service Design, 1st Edition, Prentice Hall Publications, 2007.

[19] WSO2 enterprise integrator documentation, 24 December 2018. [Online]. Available: "https://docs.wso2.com/display/EI620"

[20] A. Dr. Adamko, Internet tools, and Services, 2014.

[21] Ethiotelecom, ethio application LLD_BSS v2.5-NGBSS Document, Addis Ababa, 2015.

[22] Ethiotelecom, SMO-SCA 02 SIM Card Replacement Process Manual, 2015.

[23] "http://www.africanews.com," 16 November 2017. [Online]. Available: http://www.africanews.com/2017/11/16/ethiopia-telecoms-monopoly-now-africa-s-largest-mobile-operator.

[24] M. R. a. G. A. Akyuz, "Enterprise Application Integration (EAI), Service Oriented Architectures (SOA) and their relevance to supply chain formation," *African Journal of Business Management Vol. 4,* pp. 2604-2614, October 2010.

[25] "SOA Reference Architecture – Integration Layer", Opengroup.org, 24 December 2018. [Online]. Available: www.Opengroup.org/soa/source-book/soa_refarch/p13.htm.

[26] "2018 5th International Conference on Computational Science/ Intelligence and Applied Informatics (CSII)," Mulesoft – Salesforce Integration Using Batch Processing, 24 December 2018. [Online]. Available: https://www.mulesoft.com/resources/esb/service-orchestration-and-soa.

[27] "SOAP Web Services Using JAX-WS - DZone Integration," dzone.com, 23 December 2018. [Online]. Available: https://dzone.com/articles/soap-web-services-using-cxfjibx-jax-ws.

[28] "Talend Products: Data Integration Platforms and Solutions," Talend Real-Time Open Source Data Integration Software 24 December 2018. [Online]. Available: https://www.talend.com/Products;

[29] "Integration - On-Premise and in the Cloud," Wso2.com, 28 December 2018. [Online]. Available: https://wso2.com/integration.

[30] "Dashboard - JBoss.org Documentation," Docs.jboss.org, 24 December 2018. [Online]. Available: https://docs.jboss.org/.

[31] "OpenAdaptor: Open source EAI," Open Source Unleashed, 24 December 2018. [Online]. Available: https://alexfletcher.typepad.com/all_bets_off/2006/05/openadaptor_ope.html.

[32] 24 December 2018. [Online]. Available: http://servicemix.apache.org.

[33] The Team, "Welcome to Apache ServiceMix!", Servicemix.apache.org, 24 December 2018. [Online]. Available: http://camel.apache.org/getting-started.html.

[34] "XML Soap," W3schools.com, 24 December 2018. [Online]. Available: https://www.w3schools.com/xml/xml_soap.asp.

[35] Don Box, " Simple Object Access Protocol (SOAP) 1.1," http://www.w3.org/TR/SOAP, 24 December 2018. [Online]. Available: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383490.

[36] IBM, Service-Oriented Architecture and Web Services, IBM, RedBook, 2004.

[37] Ethiotelecom, "Internal Communication," 3 September 2018.

[38] "Working with Data Services - Enterprise Integrator 6.2.0 - WSO2 Documentation," 23 December 2018. [Online]. Available: https://docs.wso2.com/display/EI620/Working+with+Data+Services.

[39] "SOA | Object Management Group," Omg.org, 23 December 2018. [Online]. Available: http://www.omg.org/technology/readingroom/SOA.htm.

[40] J. Bean, SOA and Web Services Interface Design: Principles, Techniques, and Standards; 1st Edition, Morgan Kaufmann Publications, 2009.

[41] T. Erl, SOA Principles of Service Design, 1st edition, Prentice Hall publications, 2007.

[42] A. Goel, "Enterprise integration EAI vs. SOA vs. ESB," *Infosys Technologies White Paper,* p. vol. 87, 2006.

[43] D. D. R. S. Bosanac, ActiveMQ in Action 1st edition, 2011.

[44] C. H. M. &. M. Apshankar, Web Services Business Strategies and Architectures, 2002.

[45] W.-J. v. d. H. Mike P. Papazoglou, "Service-oriented architectures: approaches, technologies, and research issues," *Service-oriented architectures,* March 2007.

[46] S. K. Abhishek Verma (Author), "Data Synchronization in Heterogeneous Database Environment," in *2nd International Conference on Contemporary Computing and Informatics*, 2016.

[47] Oracle, "Ensuring Data Consistency," in *Oracle GoldenGate Veridata*, July 2013.

[48] H. a. A. P. Barki, "A Model of Organizational Integration, Implementation Effort, and Performance," *Organization Science,* p. 165–179, 2005.

[49] D. S. Linthicum, Enterprise Application Integration, Addison Wesley, November 1999.

[50] C. Jones Do More with SOA integration: Integrate, Automate and Regulate your business process with the best of Packet SOA book, packet Publishing enterprise.

[51] T. R. a. A. H. Awan, "Challenges and Future of Enterprise Application Integration," *International Journal of Computer Applications vol. 42, no. 7,* pp. 42-45, 2012.

[52] S. M. I. Li Da Xu, "Enterprise Systems: State-of-the-Art and Future Trends," *Transactions on industrial informatics,* vol. 7, no. 4, November 2011.

[53] "A roadmap to enterprise data integration. Information integration solutions," IBM, February 2006.

[54] P. Kumar, "An Overview of Architectures and Techniques for Integrated Data Systems (IDS) Implementation;" 2011.

[55] Oracle, " Overview of E-Business Integration," Docs.oracle.com, August 2017. [Online]. Available:https://docs.oracle.com/cd/A87860_01/doc/ois.817/a83729/adois01.htm#998301. [Accessed 07 September 2018].

[56] Oracle, " Methodology and Solutions," Docs.oracle.com, August 2017. [Online]. Available: https://docs.oracle.com/cd/A87860_01/doc/ois.817/a83729/adois02.htm#998203. [Accessed 07 September 2018].

[57] J. L. a. X. Zheng, "Research for a data synchronization model based on Middleware and rule base," in *the 1st International Conference on Information Science and Engineering*, 2009.

[58] A. Reeve, Book Managing Data in Motion: Data Integration Best Practice Techniques and Technologies, Elsevier Inc, 2013.

[59] WSO2, " Working with Proxy Services - Enterprise Integrator 6.2.0 - WSO2 Documentation," Docs.wso2.com, 11 4 2018. [Online]. Available: https://docs.wso2.com/display/EI620/Working+with+Proxy+Services. [Accessed 11, July 2018].

[60] Oracle, " Java Standard Edition Technologies - Database," Oracle.com, [Online]. Available: https://www.oracle.com/technetwork/java/javase/jdbc/index.html.[Accessed 11, October 2018].

Annex A: Validation Procedure

This is a guiding procedure to validate the proposed Data Synchronization Solution Model (DSSM) functionality and data synchronization capability. The validation procedure is organized and tabulated in three Sections as shown in (Table 9) below. The procedure in the first Section is to validate the DSSM components proper functionality through the GetProfile data service to fetch customer profile information. The second Section is to validate the DSSM functionality through UpdateProfile data service and the third Section is to validate the functionality of DSSM to perform data synchronization by creating data consistency among the integrated enterprise systems.

Table 10. Validation Procedure.

Validation Procedure			
Functionality test			
Preliminary			
1	Run WSO2 enterprise integrator as administrator from program files		
2	Open the WSO2 management console using a web browser.		
3	Enter credential, username =Admin and Password= Admin		
Section One: Functionality Validation for GetProfile service			
№	**Test Procedure**	**Expected result**	**Test result (Pass/Fail)**
1	From the main menu in the left of the page click Services list	List of available services would be displayed.	
2	from the list of wso2 services select the GetProfile Service.	The GetProfile data service dashboard would be displayed.	
3	Click 'try this service' link	The GetProfile data service would be displayed and prompted for service number entry.	
4	Enter service numbers 911500074 and click send	Detailed Customer profile information for service number 911500074 would be displayed	

№	**Test Procedure**	**Expected result**	**Test result (Pass/Fail)**
5	Close the result and return to WSo2 integrator main menu	The profile information would be closed and WSO2 integrator main menu would be displayed.	
6	Follow the same steps for service numbers 912047474, 0912140333, 911502570, 911501234, 0911500101.	WSO2 enterprise integrator main menu would be displayed.	
Section Two: Functionality Validation for UpdateProfile service			
No	Test Procedure	Expected result	Test result (Pass/Fail)
1	From the main menu in the left of the page click Services list	List of available services would be displayed.	
2	from the list of wso2 services select the UpdateProfile Service.	The UpdateProfile data service dashboard would be displayed.	
3	Click 'try this service' link	The UpdateProfile data service would be displayed and prompted for new SIM card information update entry.	
4	Update the SIM card replacement information for service number 0911500074 as; ICCID=122333444455555, PIN= 4455, and PUK=87654321	For successful update, number of Updated-Row-Count result value '1' would be returned/displayed.	
5	Close the result and return to WSo2 integrator main menu	The Update information would be closed and WSO2 integrator main menu would be displayed.	
Section Three: Data Consistency Test			
№	Test Procedure	Expected result	Test result (Pass/Fail)
1	Execute all the steps in Section Two above for UpdateProfile service for other services; as an example, for Service number 912047474, update as ICCID = PIN =	For a successful update, number of Updated-Row-Count result, value '1' would be returned/displayed.	

№	Test Procedure	Expected result	Test result (Pass/Fail)
	PUK =		
1	From the main menu in the left of the page click Services list	List of available services would be displayed.	
2	from the list of wso2 services select the UpdateProfile Service.	The UpdateProfile data service dashboard would be displayed.	
3	Click 'try this service' link	The UpdateProfile data service would be displayed and prompted for new SIM card information update entry.	
4	Update the SIM card replacement information for service number 912047474 as; ICCID=987654321234567 PIN= 4959, and PUK=43218765	For a successful update, number of Updated-Row-Count result, value '1' would be returned/displayed.	
5	Close the result and return to WSo2 integrator main menu	The Update information would be closed and WSO2 integrator main menu would be displayed.	
6	Run SQL developer for Oracle	Oracle SQL Developer interface will be opened.	
7	DoubleClick CommonDB	CommonDB will prompt for credential.	
8	Provide credential as; UserName='Common' Password='wso 2'	Oracle CommonDB will be opened with its own Worksheet, Query Builder, Tables, Views, etc.	
9	Provide the following query "select SERVICE_NO, SUB_ID, ICCID, PIN, PUK from SUB_INFO where SERVICE_NO=912047474;" and click 'Run statement'	The query result for service number 912047474 will be displayed.	
10	Keep the Query result for further comparison with the MySQL query result.		
11	Now Open the virtual box to access MySQL database.	Oracle VM VirtualBox Manager will be displayed	

№	Test Procedure	Expected result	Test result (Pass/Fail)
12	Start the data-service-server Machine	The DataServiceServer Machine will start windows server 2012 R2.	
13	Run MySQL workbench 8.0 console	MySQL workbench 8.0 interface will be displayed	
14	Open the SQL query editor	The MySQL query editor screen will be displayed.	
15	Execute the following query "update SubInfo set OperID = :OperID, Status = :Status, ICCID = :ICCID, PIN=:PIN, PUK=:PUK where ServiceNo = 912047474; "	the query result will be displayed.	
16	Compare the query result in CRMdb (MySQL database) with the above Query result in CommonDB (Oracle database).	The query result would be identical with the query from (CommonDB) Oracle database.	

Annex B: Database Design

CRM sample database Design

The information required to build the sample CRM database is organized into four Tables. These are Operator information Table, Customer Information Table, Subscriber information Table, and service information Tables.

The conceptual model for the above data item is described in four Tables with basic datasets is described as follows.

Operator_Information (OPER_ID, REGION_ID Number, DEPT_ID, SECTION_ID, SHOP_CODE, ROLL, FIRST_NAME, MIDDLE_NAME, LAST_NAME).

Customer_Information (CUST_ID, FIRSTNAME, MIDDLENAME, LASTNAME, Email, POBox, WEREDA, REGION, HOUSENo, CATEGORY, OfficePhoneNo, MobilePhoneNo, CUST_CLASS, CUST_STATUS, OPER_ID).

Subscription_Information (SUB_ID, CUST_ID, SERVICE_No, OPER_ID, SHOPE_CODE, STATUS, SUB_CLASS).

Service_Information (ORDER_ID, SERVICE_NO, SERVICE_ID, SUB_ID, CUST_ID, SERVICE_TYPE, NETWORK_TYPE, OPER_ID).

The CRM database normalized Entity Relationship (ER) diagram is shown in Figure 11 below.

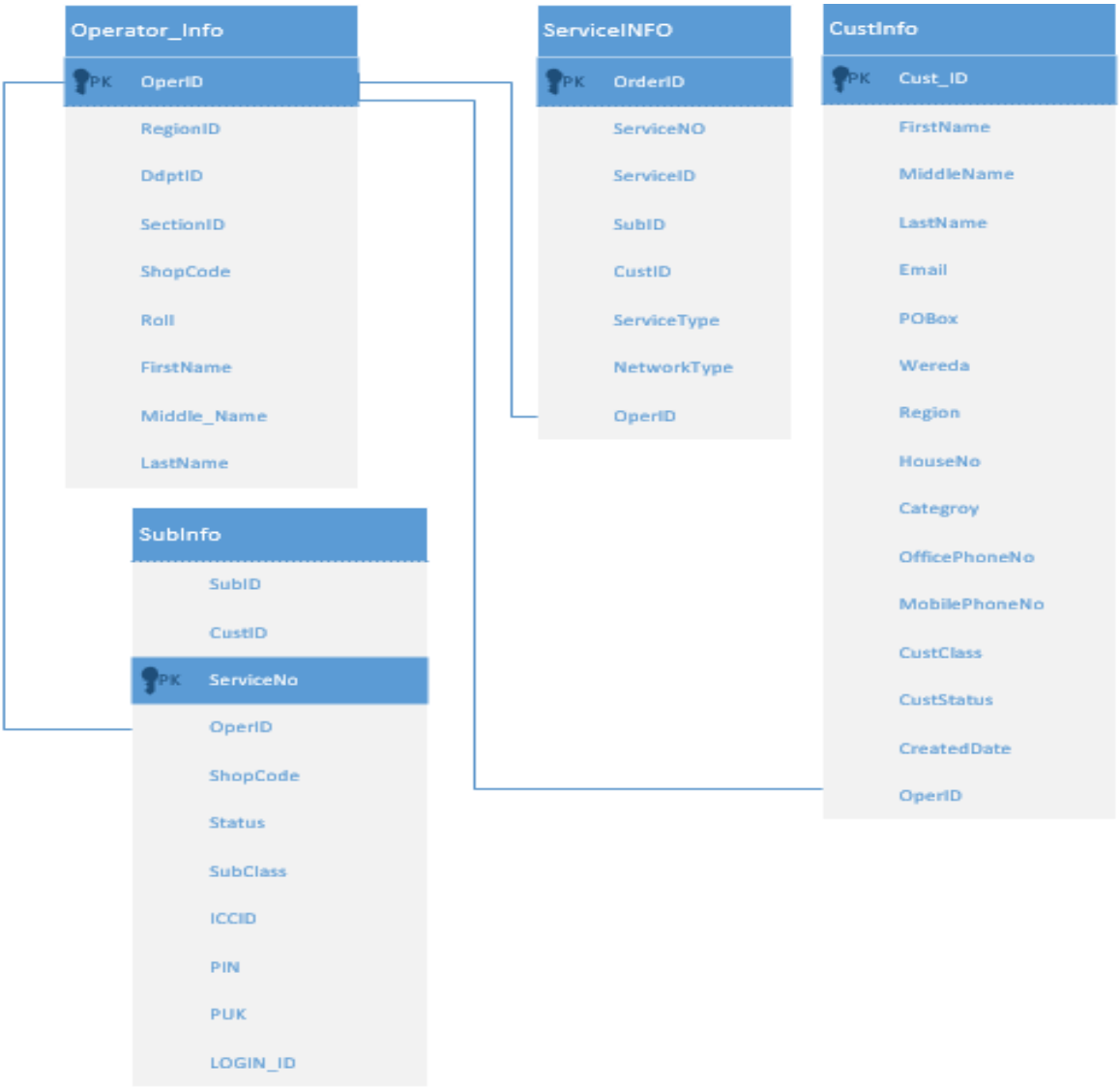


Figure 32. CRM database ER diagram

For the CRM database implementation, low level logical design is described as follows for each table in the CRM database to be implemented in MySQL database.

create table OPERATOR_INFO (OPER_ID Char(8) constraint oper_info_pk primary key, REGION_ID Number(4) NOT NULL, DEPT_ID Number(4) NOT NULL, SECTION_ID Number(4) NOT NULL, SHOP_CODE Char(8), ROLL Char(16) NOT NULL, FIRST_NAME Char(64) NOT NULL, MIDDLE_NAME Char(64) NOT NULL, LAST_NAME Char(64) NOT NULL);

create table CUST_INFO (CUST_ID Number(16) constraint cust_info_pk primary key, FIRSTNAME Char(64) NOT NULL, MIDDLENAME Char(64), LASTNAME Char(64), Email Char(64), POBox Char(16), WEREDA Char(64), REGION Char(16), HOUSENo Char(16), CATEGORY Char(16) CONSTRAINT custinfo_category_in check (category in ('Residential','Enterprise','Guest')), OfficePhoneNo Number(16),

MobilePhoneNo Number(16), CUST_CLASS Char(16) CONSTRAINT custinfo_custclass_in check (cust_class in ('Platinium','Gold','Silver','Normal')), CUST_STATUS Char(16) NOT NULL, OPER_ID Char(8) constraint cust_info_fk foreign key (OPER_ID) references Operator_info (OPER_ID) on delete cascade);

create table SUB_INFO (SUB_ID Number(16) NOT NULL, CUST_ID Number(16) NOT NULL, SERVICE_No Number(16) constraint subs_info_pk primary key, OPER_ID Char(8) constraint subs_info_fk foreign key (OPER_ID) references Operator_info (OPER_ID) on delete cascade, SHOPE_CODE Char(16) NOT NULL, STATUS Char(16) NOT NULL, SUB_CLASS Char(16) NOT NULL);

create table SERVICE_INFO (ORDER_ID Char(16) constraint service_info_pk primary key, SERVICE_NO Number(16) NOT NULL, SERVICE_ID Number(4), SUB_ID Number(16) NOT NULL, CUST_ID Number(16) NOT NULL, SERVICE_TYPE Char(16) CONSTRAINT serviceinfo_type_in check (service_type in ('NewSIM', 'SIMReplace', 'Change of Subscription')), NETWORK_TYPE Char(16), OPER_ID Char(8) NOT NULL constraint service_info_fk foreign key (OPER_ID) references Operator_info (OPER_ID) on delete cascade);

e-CAF sample database Design

To create e-CAF database the information required to build the sample database is organized in to two Tables. These are Agent information Table, and Subscriber Information Table.

The conceptual model for the above data item is described as follows.

AGENT_INFO (Agent_ID, Agent_ID_Status, ORG_ID, USERNAME, MIDDLENAME, SURNAME, SHOP_CODE, ROLE, PASSWORD, STATUS).

SUBSCRIBER_INFO (MSISDN_ID, CUSTOMER_ICCID, CUSTOMER_SERVICE_TYPE, SUBSCRIPTION_TYPE, CUSTOMER_ID, CUSTOMER_TYPE, CUSTOMER_TITLE, CUSTOMER_NAME, CUSTOMER_MIDDLENAME, CUSTOMER_SURNAME, CUSTOMER_CATEGORY, CUSTOMER_PIN, CUSTOMER_SIM, CUSTOMER_PUK, LOGIN_ID, CUSTOMER_CONTACT_PERSON, CUSTOMER_CONTACT_NUMBER, Agent_ID, CAPTURE_AGENT_ID, BACKLOG_SCAN_DATE, CAPTURE_DATETIME, APPROVED_AGENT_ID, APPROVED_REJECT_COLS_ID, APPROVED_ORG_ID, APPROVED_NOTE, APPROVED_DATETIME, APPROVAL_STATUS).

The e-CAF database normalized Entity-Relationship (ER) diagram is shown in Figure kk below.

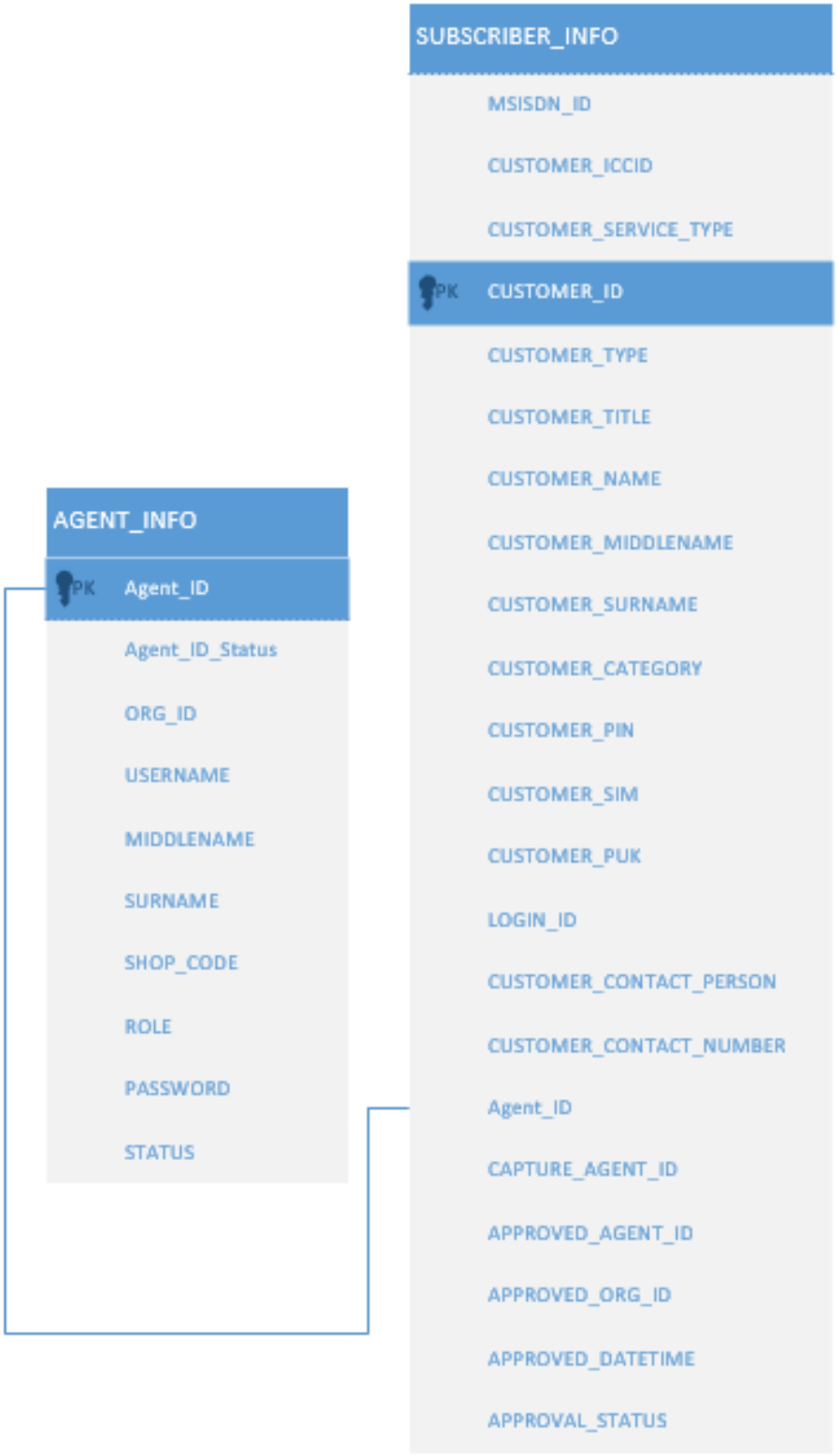


Figure 33. e-CAF database ER diagram

Annex C: Interview Questions

This interview question is a semi-structured interview in which for smooth communication between the interviewee and the interviewer, the interviewer may not strictly follow the questions sequentially. The purpose of this interview is to collect technical and operational related information about integrated enterprise applications implementation and other information in related to data synchronization problems in ethiotelecom. If not confidential the information gathered from this interview is intended to be used as an input for the research being conducted for the Master of Science in telecom engineering in Addis Ababa University Institute of Technology. The research title is 'Enterprise Application Integration Data Synchronization Solution Model the case of ethiotelecom'. Any confidential information that the interviewee specified as confidential will not be published or used in the research documentation. The interviewees are selected based on their better technical knowledge in the application domain in ethiotelecom.

Data synchronization is the process of ensuring data consistency among integrated applications. And, data synchronization problem is a problem that prohibits integrated applications from having consistent data in their datastore; or it is a problem that creates data mismatch or discrepancy among integrated enterprise applications.

Name of the Interviewee: _____

Current Position: _____

Roll: _____

Experience in the Position: _____

Gender _____

Technical Interview Questions

1. What are the Enterprise applications you are familiar with?
2. Your duties in these applications?
3. What kind of data causes inconsistency/data synchronization problems? Please be exhaustive and provide scenario.
4. What kind of data sync problem frequently reported?
5. How is the data sync process flow for existing SIM card sales? (Scope: New SIM sales, Reconnection, Transfer ownership, and termination)
6. Which application could reject data synchronization request? (the middleware or the receiver application)
7. What are the condition in which data synchronization request would be rejected? (what are the errors; Invalid account, already registered, Input string error, service No doesn't exist… (service profile errors))
8. What are the restrictions in data update/ change (which application can modify data e.g. customer profile, billing related,)?
9. When there is data inconsistency which data source is more reliable/acceptable? Is there a prioritization logic?
10. What are the business rules and security policies that should be considered in EAI data communication?
11. How is the message transaction and data mapping work?
12. What functionalities are performed by the ESB?
13. Is that possible to modify/make change to existing application integration (ESB, interface) for ethiotelecom experts?
14. If you have any supportive document or screenshots or …?
15. Is any additional information you want to tell me about data synchronization?

Interview Questions for System User

1. What are the integrated application you are familiar with?
2. Your duties in these applications?
3. What kind of data sync problem frequently occur?
4. How is the data sync process flow for existing SIM card sales?
5. how frequent is the data synchronization problem? Can you quantify?
6. How long does it take a data synchronization problem to be fixed?
7. what is the impact of data synchronization problem on your work (from customer and company perspective.)
8. What kind of quick fix or workaround is in practice to minimize customer impact
9. What kind of data create inconsistency/data synchronization problems? Please be exhaustive and please provide scenario.
10. If you have any supportive document or screenshots or …?
11. Any important point about data synchronization problems you want to tell?

Declaration

I declare and certify that the enclosed Master Thesis entitled 'A Data Synchronization Solution Model for Enterprise Applications Integration the case of ethiotelecom' is entirely the result of my own personal effort. I have accurately cited all sources, including books, journals, company documents, and unpublished manuscripts, as well as any other media, such as the Internet. This thesis contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree; except where acknowledged. This thesis work is done under the guidance of Dr. Mesfin Kifle (Ph.D.), In Addis Ababa Institute of Technology (AAIT), Addis Ababa**.**

Candidate: Werku Melesse Kenaw

Signature_____

Date_____

In my capacity as supervisor of the candidate's thesis entitled 'Enterprise Applications Integration Data Synchronization Solution Model the case of ethiotelecom', I certify that the above statements are true to the best of my knowledge.

Advisor: Mesfin Kifle (PHD)

Signature_____

Date_____