Framework for monitoring JMS based Integration Bus

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Computer Science and Engineering

Submitted By:

Palka

(801332017)

Under the supervision of:

Dr. Ajay Kumar

(Assistant Professor)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

July 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "***Framework for monitoring JMS based Integration Bus***", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Ajay Kumar* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Palka 14/7/15

Signature:

(Palka)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

A Kumar 14/7/15

(Dr. Ajay Kumar)

Assistant Professor, CSED

Countersigned by

(Dr. Deepak Garg)

Head

Computer Science and Engineering Department

Thapar University

Patiala

(Dr. S. S. Bhatia)

Dean (Academic Affairs)

Thapar University

Patiala

Acknowledgment

No volume of words is enough to express my gratitude towards my guide, Dr. Ajay Kumar, for giving me the guidance, encouragement, counsel throughout my research. Without his invaluable advice and guidance it would not have been possible for me to complete this dissertation. I am greatly indebted to him for his constant encouragement and invaluable advice in every aspect of my academic life. I consider it my good fortune to have got an opportunity to work with such a wonderful person.

I am indebted to Dr. Deepak Garg, Head-Department of CSED, and all faculty members and staff of the CSED for their sympathetic cooperation.

I am very thankful to Oracle Retail, for giving me platform to learn and explore in real time environment. I am also thankful to Integration team, for supporting me throughout the internship, for being always there for me in need of the hour and giving me the opportunity to learn new things.

Most importantly, I would like to thank my parents and the Almighty for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

Finally, I would like to thank all of them whose names are not mentioned here but have helped me in any way to accomplish the work.

Palka

(801332017)

Abstract

Integration of applications in the enterprise has become the basic need, as the number of applications increasing manual integration is becoming harder. There are some technologies available for integration of an application, like RMI, RPC, JMS, and Web Services. Using these technologies enable to create robust and cohesive integration software. All these technologies handle the communication part. All of them have their own benefits; some provide highly robust communication and guaranteed delivery. With all the features available we can't rely on software blindly, software behaves unexpectedly sometimes. Due to this reason the requirement of monitoring of software is becoming mandatory, especially for the software that deal with network, anything can go wrong, the users might not even have a hint of what is going inside the system.

For technologies like Web services it is easy to monitor, there is no additional requirement of any extensions; they already expose their data through services that can be used to create monitoring API's. However, with the systems which are JMS, JCA or RMI based we need to extend the system to expose the inner health and status of the system. Study of recent research shows that there have been many developments in monitoring of SOA-based integration, on the basis of the proposed integration frameworks in recent research; the proposed framework is based on web services.

To address the monitoring needs of the JMS based integration bus, a framework has been proposed. JMS implementations do not provide API's to an external system, so for monitoring of the system requires API's to be created over the JMS clients. The proposed implementation is web services based, which are basically of two types SOAP and ReSTFull, a comparative study of both the services and optimization of SOAP has been done. Optimization of SOAP is required because it is more suitable for the proposed architecture and according to the requirement of the system.

Table of Contents

List of Keywords

JMS	Java Messaging Service
MOM	Message Oriented Middleware
RPC	Remote Procedure Call
RMI	Remote Method Invocation
JCA	Java Connector Architecture
SOAP	Simple Object Access Protocol
ReST	Representational State Transfer
SOA	Service Oriented Architecture
GUI	Graphical User Interface

List of Figures and Tables

Chapter 1

Introduction

This chapter gives an introduction to integration. It explains some basic terms related to integration and integration topologies. A brief summary of the contents of this report is also given in this chapter.

1.1 System Integration

System Integration is a method to connect the cohesive system that has been created and evolved independent of each other. [7] System integration is a basic requirement these days, as no one wants to synchronize the independent but related applications manually. We are moving towards automation of almost everything and system integration is one the step towards automation of the complete process software can perform. We have enormous number of applications which are used for performing the day to day activities. Most of the applications are inter-related to each other but can work independently, for such situations system integration is a must requirement to provide a complete solution to the users.

Gone are the days when large and complex applications were made to fulfill all the requirements of the users. Those applications were difficult to maintain and were not at all cost effective. These days the applications are broken into smaller applications. They perform their part of the job independently but need a communication link between each other, which communication link is provided by the system integration applications.

System integration applications are usually network based applications. They don't have a front-end, but with the number of applications and their complexity being increased day by day we need to provide the integration application the ability to expose the inner working and status of the system, so that it can be monitored.

Integration is becoming more crucial with the latest technologies like cloud computing. The software are being moved to cloud, and some of them are on-premise,

or a hybrid cloud is being used then integration of application becomes complex task that needs monitoring, so as to keep a check on things is working right or not. [9]

There are many methods that have been adopted by different organizations for integration; different protocols have been used. The latest boom in integration has been brought in the industry by the Service Oriented Architecture (SOA). It has been accepted industry-wide, and it is the best way to make independent applications and integrate them using web services, which doesn't have much overhead.

1.2 Integration Topologies

Integration topology defines the physical architecture of the integration system, commonly the following Topologies are used for Integration: [5]

1.2.1 Bus Integration

In a bus integration topology, all the applications connect to the same bus. Bus integration is useful in case of small network, with a small number of applications. This topology is robust in the sense, if one of the application is down, other applications can still communicate with each other.

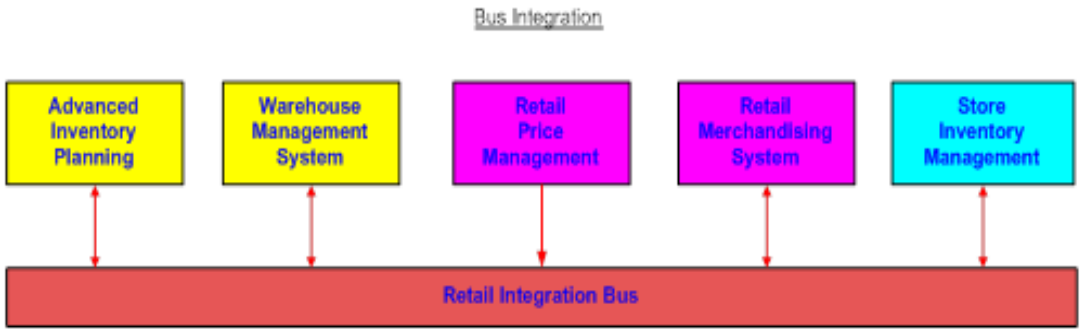

Figure 4.1: Bus Integration Topology

1.2.2 Hub-and-Spoke Integration

In a hub and spoke integration, one application is the center of communication, all the communication happens through the central application. This integration is also useful for small networks, with a small number of applications.

Figure 1.5: Hub and Spoke Integration Topology

Here the central application is the bottle-neck. If the central application goes down, no communication will happen.

1.2.3 Point-to-Point Integration

Point to point topology connects all the applications using a one-to-one link all applications are directly connected to each other, this is a highly robust topology, but it is not a cost efficient method.

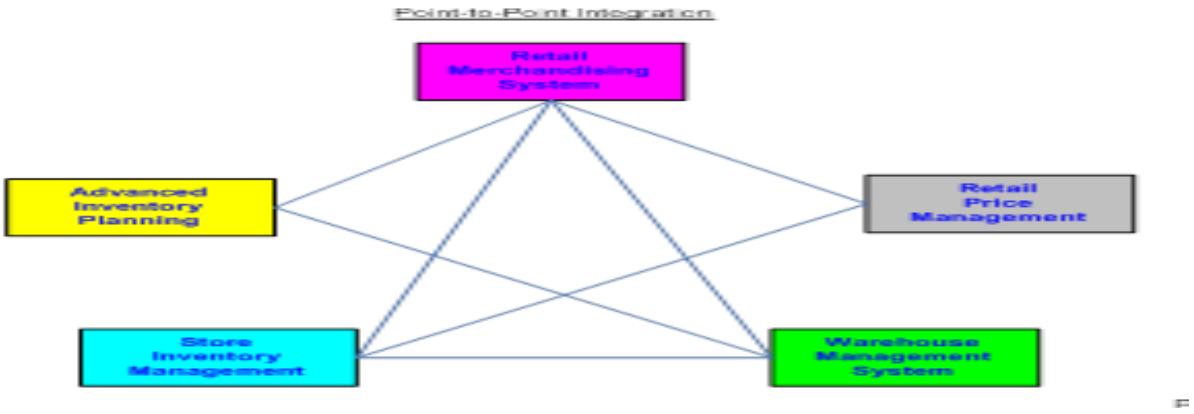

Figure 1.6: Point to Point Integration Topology

1.3 Middleware Technologies

Middleware is the system that allows communication between different applications to provide interoperability. Middleware is required to allow integration of distributed system. Middleware simplifies the distributed architecture and allows an easy way to communicate, exchange data and synchronize. [11]

There are different middleware technologies for communication between two remote Java programs, which are: [10] [1]

- RMI: Remote Method Invocation
- JMS: Java Messaging Service
- Web Services
- JCA: Java Connector Architecture

Remote Method Invocation is the middleware technology to be used for communication when there is a requirement of synchronous communication, and a coupled client-server system is acceptable. RMI does not provide reliable communication so it cannot be used if the application is critical one and can't afford to lose any data.

On the other hand, Java messaging service provides a highly cohesive, asynchronous and reliable communication. JMS can be used for the communication between the applications which are independent of each other and require reliable communication middleware. Java Messaging Service is a message oriented middleware (MOM).

Web services are independent of Java or any other programming language, and it is useful to use them in case the system has heterogeneous, highly cohesive application and require inter-operability. Web services are a reliable way of communication.

JCA is a tightly coupled and synchronous communication middleware architecture it works on request/reply mode. JCA allows resource adaptations. The evolving implementation of JCA is moving towards asynchronous communication.

1.4 Definitions

1.4.1 Remote Procedure Calls

Remote procedure calls, means calling methods from a remote machine, usually called client, when a client wants some services from a server, and a server exposes the methods through interfaces, which can be called by the client remotely. The client knows the method signatures by the interfaces. [12]

1.4.2 Synchronous and Asynchronous Communication

Synchronous communication means that the data sent while communicating is sent at a regular interval of time at a constant rate. Sometimes synchronous communication requires clock signal with the data.

Asynchronous communication is opposite of synchronous; the data can be sent at any time any interval and any size.

1.4.3 Message Oriented Middleware

A message oriented middleware is one in which the communication is entirely based on messages. MOM provides cohesive integration. Any middleware infrastructure which provides messaging capabilities is referred as MOM. [2]

1.5 Thesis Outline

This thesis has been organized in six chapters; the first chapters give a brief introduction to integration, topologies, available options for middleware and related terms. The second chapter focuses on Java Messaging Service, which is the middleware of the target system; this chapter gives a clear understanding of Java messaging service. The third chapter focuses on the related work. The fourth chapter defines the problem statement, the motivation behind it, and solution of the designed problem, which includes the solution of a sub-problem. The fifth chapter covers the testing and results of the proposed framework. The last chapter concludes the thesis and set the future scope.

Chapter 2

Java Messaging Service

This chapter explains Java Messaging Service in detail. It also includes models, architecture, features and other gory details.

2.1 JMS Models

Java messaging service has been widely accepted throughout the IT industry for integration of applications. Going with the popularity of Java over the internet it can be assumed that there are large numbers of JMS solution. Java messaging service has two types of model defined for messaging, which are: [2]

- Point-to-Point messaging domain
- Publish/Subscribe messaging domain

2.1.1 Point-to-Point messaging domain

This model is useful for one to one communication, i.e., for each message published there is only one subscriber. When a message is published it is put on a queue, the subscriber reads the message from the queue until then the message stays on the queue. [2]

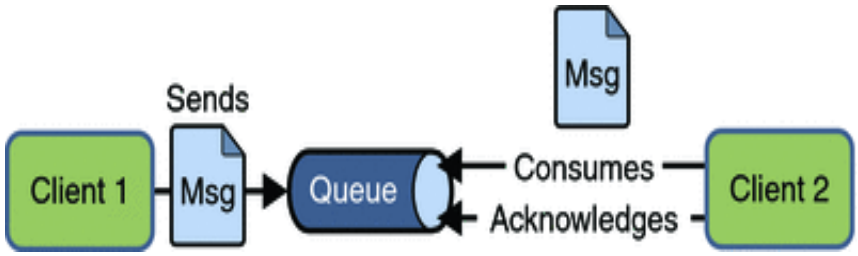

Figure 2.1: Point-to-Point messaging domain

After the message is received on the queue the message is consumed by the client whenever the client is available. The client after consuming the message sends the acknowledgment to the sender.

2.1.2 Publish/Subscribe messaging domain

This model is useful for scenarios where there can be more than one subscriber for one message. When an application wants to send a message, it publishes the message on to the topic. The topic keeps the message until all the registered subscriber to the topic didn't consume the message. This approach is loosely coupled approach and useful in case a single message has to be sent to multiple applications. [2]

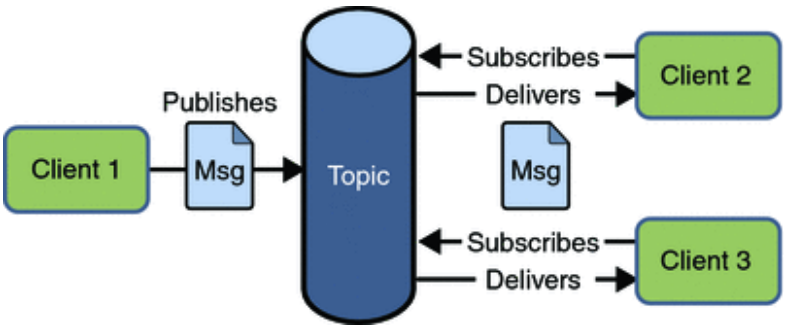

Figure 2.2: Publish/Subscribe messaging domain

2.2 JMS Programming Model

The programming model in JMS works as follows:

The message publisher takes an instance from the connection factory. The connection factory can be of two types, TopicConnectionFactory and the other is QueueConnectionFactory. According to the requirement the publisher takes the connection instance from the appropriate connection factory, and a session is created.

Using the session object, the message is published. On the other hand the subscriber also takes and instance from the session factory. The message is detected by the message listener at the client side. The listener enables the client to consume the message, which is then consumed by the subscriber. An acknowledgement is sent to the message publisher on successful consumption of message by the entire registered subscriber. Then the message is removed from the topic or the queue.

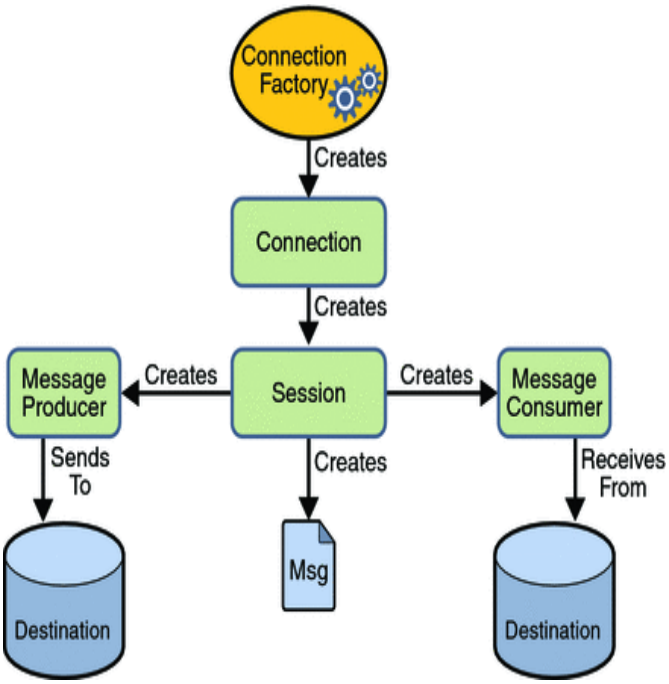

Figure 2.3: JMS Programming Model

2.3 JMS Features

Java messaging services provide many features, which make the communication through JMS reliable; the features which make JMS one of the best middleware solution are following: [3] [5]

2.3.1 Guaranteed Messaging

Guaranteed messaging is the most important feature of the JMS; this feature handles reliable communication. Guaranteed messaging implementation is not just a single component work; it takes much work and complexities to provide guaranteed messaging. The components that ensure guaranteed messaging are:

- Message Autonomy
- Store and forward messaging
- Message Acknowledgments and Failure Conditions

Message Autonomy: A message is an autonomous unit, it can be sent, processed, re-sent, etc. The contract between the sender and the server is that the server will keep the message until all the expected message subscriber clients have not processed the

message. There may be many contracts between the sender and the receiver, the receiver might need to process the message then edit it and then send the message to some other clients. The sending client is done with its work when it sends the message to the JMS rest is the responsibility of the JMS or the receiver.

Store and forward messaging: Storing and forwarding the message is the responsibility of JMS, as it is part of JMS contract with the sender, in case of persistent message, Storage mechanism is used, so that the message can be recovered in case of failure of sender or the receiver. JMS implementation can differ, with storage implementation; it could be central, distributed or an intelligent hybrid approach. Forwarding the message is also JMS responsibility, the message has to be routed and forwarded to the subscribers of the message.

Message Acknowledgements and Failure Conditions: It is a crucial part of the JMS guaranteed messaging service, there are number of acknowledgements being sent during course of a message, whenever the state of the message is changed an acknowledgement is sent to the sender, these acknowledgements allow tracking of the messages.

2.3.2 Performance and Scalability

JMS performance and scalability both go together, performance refers to the processing time taken by JMS to process a message, from sender to the receiver, and scalability refers to the number of clients a JMS can support at one time, so all in all the working of JMS can be judged by considering both the factors together, i.e., data being processed at a time, with respect to the number of clients active.

The performance and scalability requirements of the system are evaluated before choosing the JMS implementation, because some of the implementation might be good and most suitable for one type of application and another implementation might be good for some other application.

2.3.3 Multicasting

The JMS Publish/Subscribe model is beneficial only in cases a multicasting method is available, for multicasting over-internet many decisions need to be made for selecting

the underlying network protocols for communication. Among UDP and TCP, both have their advantages and disadvantages. UDP supports multicasting, but it is not a reliable transport layer protocol. TCP is a reliable protocol, but multicasting is not supported, and sending a single multicast message as different messages will take much overhead. Due to this reason and to comply with other JMS features, most of the JMS implementation use UDP and do the Error handling, duplication, ordering and reliability of messages at their end so that the efficiency of UDP can be used. UDP works fine on the intranet for multicasting but the internet does not support multicasting, because of the old routers that don't support multicasting there can addressing issues.

2.3.4 Security

Whenever we think about communicating over a network, we need to work on security, JMS is no exception. JMS implementations provide security in using, authentication, authorization, and secure communication.

Authentication: JMS uses authentication for various purposes, authentication can be used for a server to authenticate the clients or vice versa. The basic login authentication is explicitly supported by the JMS API. Some implementations of JMS use better authentication methods for more security.

Authorization: JMS supports authorization also to limit the access to the messages by random clients who shouldn't be accessing the messages, or it is used to control the ACL's (Access control Lists), or defining the groups. JMS usually uses a tree structure to implement authorization. Trees distribute the subscribers in different levels; these levels can be used to decide the authorization of the subscribers for a message. There are many intelligent algorithmic implementations for authorization.

Secure Communications: JMS communication channels can be either secured using hardware or software. Hardware security is about creating point to point links between the clients; this is not at all cost effective method and can't be used over the internet. Software security can be achieved by encryption, encryption could be at two levels, SSL or Payloads encryption, JMS supports SSL, so in case complete communication is required to be secured then SSL can be used, but in case only a few

messages are required to be encrypted than Payload encryption is preferred, with Payload encryption, the messages or the queues that are required to be secured are encrypted, this method saves a lot of unnecessary encryption and decryption.

2.3.5 Connecting with the outside world

There are some applications outside of your system you want to communicate with, this application might not implement JMS client to communicate with your system, in this scenario, connectors are required at which act as JMS client, these connectors can communicate with other protocols and perform all the processing of converting a JMS message to HTTP message and vice versa.

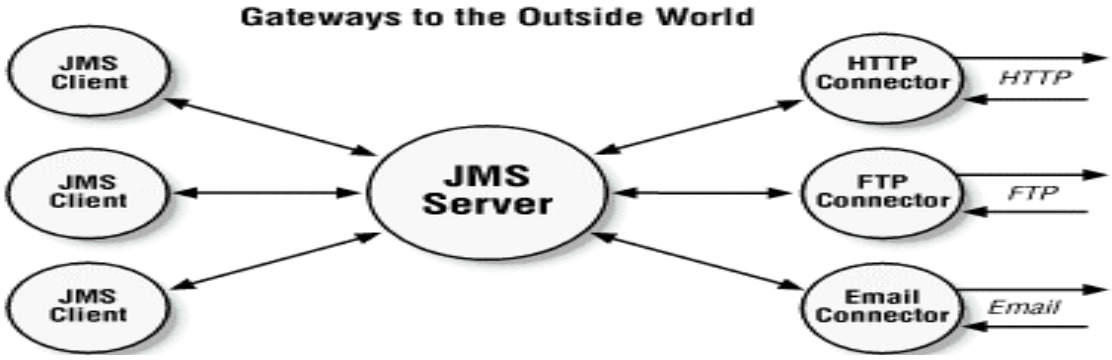

Figure 2.4: Gateways to the outside world

For each protocol a different connector is to be used, this connector take the message from JMS convert it into destination protocol message and send them over, when they receive message from the assigned protocols they convert them into JMS message and then publish it to JMS topic.

2.3.6 Communicating with other JMS systems

There are scenarios when client of one JMS needs to communicate with client of other JMS, in this scenario, they can't directly talk, because the underlying implementation and architecture might be entirely different from each other, for solving this problem a client is required which can talk to both the JMS, it will act as a pass through bridge. However, we need to consider the robustness of the common client because all the communication relies on that. This client will convert the message from one JMS to the format of other.

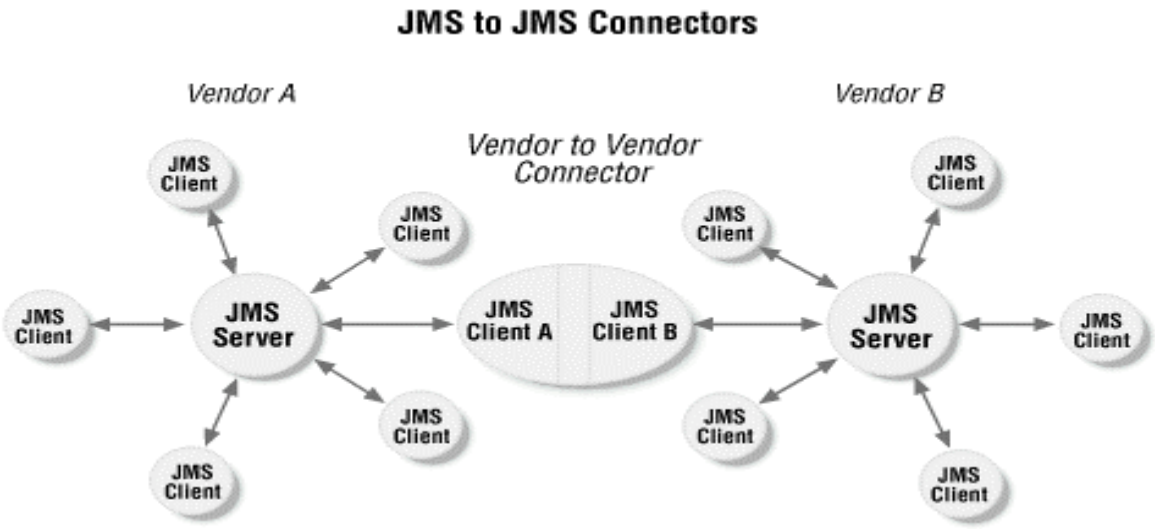

Figure 2.5: JMS to JMS Connectors

2.4 JMS Message Headers

Like any other communication protocol, JMS also has some message headers, which are used for communication and identification purposes, following are the message headers with their usage explained:

- **JMSDestination:** this header is used to specify the destination of the message, which can be topic or a queue, the topic and queue in JMS has super class Destination (javax.jms.Destination), so the instance of a Destination class is used to specify the value of this header. The purpose of this flag is for routing.
- **JMSDeliveryMode:** This header tells if the message is persistent or non-persistent. The value is set using the constants PERSISTENT and NON_PERSISTENT of the class, javax.jms.DeliveryMode. The purpose of this flag is for routing.
- **JMSMessageID:** This field in the message header uniquely identifies the message. The uniqueness could be global or local depending upon the JMS provider. The purpose of this flag is for routing.
- **JMSTimestamp:** The value of this field is set automatically when the send operation is invoked on a message. This field enables tracking, like tracking how long it has been after the send operation has been initiated. The purpose of this flag is for Identification.

- **JMSExpiration:** This field tells the expiration time of the message; if the expiration time has been crossed the message is no more processed. The purpose of this flag is for routing.
- **JMSPriority:** The field is used to set the priority of a JMS message, the value could be 0-4 and 5-9, 0-4 are normal priority messages and 5-9 and expedited priority messages, JMS is not enforced to deliver the higher priority message first, but it should try delivering the higher priority messages first. The purpose of this flag is for routing.
- **JMSRedelivered:** This field tells that if the message is delivered the first time, or it is a redelivery attempt, if the message was delivered earlier also, but the client didn't send the acknowledgement then the message is delivered again with a redelivered flag set to true. The purpose of this flag is for routing.
- **JMSReplyTo:** This flag is used to specify the address on which the subscriber can reply back. The purpose of this flag is for routing.
- **JMSCorrelationID:** This field is used to specify if the message is co-related with a previous message or not, if it is related to some previous message then that message ID is used as the value of this field. The purpose of this flag is for routing.
- **JMSType:** This field is used to define the message structure and the payload type. The purpose of this flag is for Identification.

2.5 Message Selectors

Message selectors in JMS helps the subscribers filter out the messages they receive on the topics or the queues they have subscribed, many times the messages received might not be of interest to the topic, in that case, the filters are required to be applied. Following can be used to create message selector:

- Identifiers
- Literals
- Comparison Operators
- Arithmetic Operators

2.5.1 Declaring a Message Selector

While declaring the message selector, it should be parsed to check if it is syntactically correct or not, if not then it will throw run time exception of javax.jms.InvalidSelectorException. Once a message selector is set for a subscriber then it cannot be changed, the subscriber should be deleted and recreated to have a new message selector.

2.5.2 Not Delivered Semantics

If a message is not selected by the subscriber, then in case or publisher/subscriber model the message can be consumed by other subscribers who are eligible to receive the message, whereas in case of point to point if the message selector of the prescribed subscriber filters out the message then the message is not visible to the subscriber, but another subscriber can see the message.

Chapter 3
Literature Review

Psiuk et al. [13] presented a monitoring system for SOA-based integration system which uses Enterprise Service Bus (ESB). A generic ESB Meta-model (EMM) defines the topology, and messaging mechanism for collection of monitoring data has been introduced. EMM provides the mechanism to cover all the monitoring parameters. The models applicability has been verified on Java Business Integration, which is a specification for Enterprise Service Bus. JBI itself lacks some trivial monitoring features, and the presented ESB Meta-model can be mapped to all the components of the JBI and add the lacking monitoring features to the JBI for a complete monitoring system. The EMM provides and independent and fully functional generic monitoring system that covers all the aspects of monitoring and enables diagnoses of the system. The components of the EMM, which are topology and the messaging services, clearly separate the components, artifacts lifecycle and monitoring interactions. EMM is a dynamic system that demonstrates the use of monitoring goals for updating the topology and the messaging mechanism. The results prove that the framework fulfills the requirements of the monitoring system, for business processes in integration SOA layer. Part of the information collected by the EMM framework can be interpreted as the business process monitoring. Some processing of the information like, filtering and augmenting with some artificial intelligence is required for complete business activity monitoring that hasn't been implemented in the current framework.

Baghermousavi et al. [14] presented a methodology for reducing QoS monitoring overhead in Enterprise Service Bus. The methodology focuses on the Enterprise service bus monitoring and provides a solution that is lightweight and doesn't have too much overhead in the business processing. The QoS is devised using time series forecasting in Neural Networks, every time a service is called the QoS is invoked. The methodology focuses on monitoring SOA-based integration, the SOA-based systems can be monitored at different places that are client side, service provider side or the Enterprise bus side, the approach focuses on the ESB side monitoring with QoS. SOA is the architecture style that is being adapted; the QoS monitoring enables a high-performance system that is the major requirement of the people moving towards SOA.

The performance of the given approach has been based on the response time, which resulted in an increase of performance for monitoring of an SOA based system.

Shi [15] presented an approach that combined the best of the SOAP and the ReSTFull services. The focuses have been on exploiting the best characteristics of both of them and then use the best performing character to devise the new approach. SOAP and RestFull have their benefits, and choosing one of them is entirely dependent on the requirements of the system. The approach is about accessing the functional aspects of any remote software using the distributed computing power. Currently, the services are not requester oriented; the services depend on the provider, if there is a change in the service, then the requester need to change its application that is consuming the service. This would also enable the same consumer's applications being used for the different version of one service. Using requester based service approach; same semantic service description framework can be used for SOAP and ReSTFull services, which reduces the gap between the semantic based and the HTTP, and makes the web services actually web based.

Shil and Ahmed [16] added some new functionality to the SOAP, WSDL, and UDDI for the better administration of the web services. These functionalities have been added to resolve the problems with the discovery of the appropriate services, which occurs due to lack of some information in WSDL, SOAP or UDDI. This approach will also enable in finding the association between different web services. The focus of this methodology is to provide traceability of the web services by devising association rules; adding some new information to the web service topology using the WSDL document; improving the description by UDDI using the metadata related to their usage, which can be deduced from the user's past request's; and a map is generated which contains information about, topology, partners, usage and importance that is required for enhancing the administration. Using this approach resulted in the optimized classification of web-services.

Lam and Rossiter [17] proposed an extension to SOAP for using it for delivering multimedia content. Current standards of SOAP doesn't support multimedia content delivery so there is a requirement for an extension which can be used with the existing SOAP standards, which would allow the transfer of multimedia content using SOAP over HTTP. A new SOAP streaming message exchange pattern has been added and the required HTTP binding for the Messages, these two are the new components of the extension. The framework proposed can deliver multimedia content using SOAP

over HTTP, furthermore using proxy server between the service requester and service provider improves the efficiency of the process as it uses caching.

Al-Zoubi et al. [18] proposed the use of ReSTFull web service architecture for distributed simulation over SOAP. The ReSTFul-CD++ simulation server, which communicates in ReST style over HTTP, due to which the server is the service part of the web which can be accessed from anywhere over the network and be used by any client, whereas SOAP is based on RPC calls which make the client and the server coupled together due to which they can't evolve independently to each other. The distributed simulation is covered by the server, as different servers communicate with each other using XML, to coordinate using the head/proxy algorithm that is used to bundle remote messages. SOAP services are heavy weight than the ReSTFull services which gives them another disadvantage as compared to the ReSTFull-CD++ server's functioning, the server supports SOAP based web services also.

Lee et al. [19] presented a framework for composing SOAP, non-SOAP and non-Web Services. With the increase in popularity of the SOA based development for mobile applications, there is a requirement for lightweight procedures then SOAP. There have been new approaches designed, which is lightweight, but at the same time those approaches cannot be used for complex processes and procedures. So, in this case, a requirement of a composite approach comes in, which can deal complex problems as well as be lightweight enough to be used with the mobile applications. To invent a composite approach there were two challenges, one of which is to integrate different web services like, SOAP, OSCGi and ReSTFull and the other one is to combine the web services with the non-web services. The approach has been to develop an extended BPEL bundled engine for invocation of SOAP, ReSTFull and OSCGi services on adapter patterns and a mechanism for transformation of web contents and other Android services to OSCGi services. The approach presented doesn't require a transformation of ReSTFull to soap externally the extended BPEL engine does that. The engine has been presented as a good option for the systematic heterogeneous integration of web services. The experimental results prove that the presented approach generates very less network traffic that reduces the network load and makes the processing of multiple requests faster and lighter, and another conclusion is that the turn-around time has been reduced dramatically, supporting the use of heterogeneous web-services.

Oliveira et al. [20] presented a comparative analysis of ReSTFull and SOAP-WSDL web services. Web services are becoming popular because of the low maintenance cost and highly cohesive approach; they are being used for integrating heterogeneous applications. On the other hand only considering maintenance and evolution there have been issues that have been ignored. The study covers the issues which are faced while maintainability of the web services, the experiments show that the ReSTFull services are difficult to maintain on the client side, as in case there are some changes on the server side, then it is easily scalable but the client implementation will have to be changed to incorporate the changes and the SOAP services are difficult to maintain at server side, as if there are some enhancements to be made the SOAP service implementations are quite complex, it will need a lot of work to achieve the required scalability, but at the client side same implementation can be used to communicate. So the study suggests that the service providers should be implemented in ReSTFull web services whenever possible, and the service requesters should be SOAP based.

Jiang et al. [21] presented the large-scale longitudinal comparison of SOAP vs ReSTFull services, with the popularity of the web services for development of large scale software, as it is easy to integrate much software and create one large-scale software, and the individual application got to communicate with each other using web services and evolve separately, independent of each other. Presently no analysis has ever been done on a large scale and real-time data for the performance evaluation of the web services. This paper presents the real-time large scale analysis of the publicly available SOAP and ReSTFull services. The collection of data has been done from five locations in the world the interesting statics which have collected state that the growth of ReSTFull based web services is the highest among the SOAP, ReSTFull and simple text-based web services. Another statics which have been observed say that the location for most of the services is the UK, US and Germany which affects the performance of the web services over the internet and gives a skewness effect to the QoS. If service requests are made from proximity location, then there is an improvement in throughput and performance.

Belqasmi et al. [22] represented a study on SOAP vs. ReSTFull web services for multimedia conferencing. The study uses existing ReSTFull and SOAP services, which provide same services. The results prove that ReSTFull services perform better for multimedia conferencing than SOAP. The reason behind the performance is that the ReSTFull services are lightweight, they don’t have any overhead of them self,

whereas SOAP-based messages have a lot of overhead associated with them which can't be entertained in case of high-performance requirement, ReSTFull service decrease the network load by two to three times if they are used instead of SOAP services.

AlShahwan and Klaus [23] presented and architecture for providing web services for mobile hosts, the study included the use of ReSTFull and SOAP-based web services for mobile hosts, the study explains that using ReSTFull services are beneficial over SOAP as they are lightweight, provide caching which decrease network usage, highly cohesive relation between the server and the client, and doesn't use too much of the mobile device resources for processing, whereas SOAP web services require heavy clients, which need high performing processors. However, there are some issues with the ReSTFull services regarding the security, addressing and accessibility only over HTTP, which requires further research on the topic to find the ideal solution for the mobile devices.

Feng et al. [24] presented ReST to be an alternative to RPC based web services. Most of the web services use RPC implementation. However, due to the complexity of RPC and the bottlenecks in web-scale applications ReST approaches should be explored which provide many benefits over RPC and exploits the web features fully. Both the architectural styles are compared and analyzed on the parameters scalability, coupling and security. ReST explores the usage of HTTP which makes it in web in-spite of being on the web; it uses the web for providing the services. The experimental results show that ReST provides better performance, scalability, and low coupling. The more and more the ReST architecture is becoming its development support for the architecture is increasing.

Abu-Ghazaleh et al. [25] presented a serialization approach for improving the performance of SOAP; the performance bottleneck of SOAP is the serialization of the outgoing messages to XML payloads, adding them to message buffers. To improve the performance of SOAP, a differential serialization approach has been proposed. According to the differential serialization approach, while serializing the content of the message should be compared to previous message, the same content shouldn't be serialized again, only the changed part of the message should be serialized, this saves a lot of time, as most of the part of the SOAP message remains same for a conversation. This approach has been tested with many messages dividing technique, linking chunking, overlaying, stuffing, etc. There is no need to change anything with

SOAP implementation. Only the serialization process has to be changed. The use of differential serialization improved the performance of SOAP to a great extent. This approach will not affect the performance of the applications that send messages that are completely different from each other; the types of application which can benefit from the approach have been also mentioned.

Govindaraju et al. [26] compared different implementations of SOAP, as popularity of SOAP is increasing and it is all set to support grid computing, more and more analysis is required for performance evaluation and optimization. The extensibility, robustness and interoperability provided by SOAP is useful in grid computing only if the performance can be optimized, and with a number of implementations of SOAP, we need to decide which one is the best-suited for our application. The paper provides insight into the features that are deciding factor in the performance of the SOAP web services. The analysis would help developers improve the performance of SOAP by considering the features presented, and the users can select the SOAP implementation according to the features their application requires.

Davis and Parasharz [27] compared the performance of different implementations of SOAP and compared it to JAVA RMI, CORBA, and HTTP. The objective of the comparison being performed is to find the performance bottlenecks in SOAP so that focus can be brought on improving them to improve the performance of SOAP. The sources of inefficiency in SOAP are mainly: serialization process, multiple system calls for one message, and XML parsing and formatting. To improve the efficiency of SOAP the performance of parsers, serialization and de-serialization algorithm need to be improved and decreasing the number of system calls, SOAP has all the capabilities required for creating cohesive, interoperable and scalable systems, but performance has always been a worrying factor, when it comes to implementing SOAP for high-performance system. The paper concludes that the performance of SOAP can be improved by using some of the capabilities of HTTP for example HTTP chunking can improve SOAP performance.

Li et al. [28] presented an approach for improving the performance of SOAP so that it can be used in high-performance application. The bottleneck for SOAP performance has been the XML parsing, so the approach replaces the parsing step with an optimized SOAP processing algorithm. The amateur prototype is unable to identify the namespace, but the performance improvement is huge, and the current prototype

can be used as a basis for improving performance, as namespace processing is not a very heavy task.

Rahaman and Schaad [29] presented a SOAP based secure conversation and collaboration approach. With the increasing popularity of SOAP for Remote method invocation, there has been a need for security for the communication to be reliable. There have been some security implementations of Web service security policies, but the fact that there are some loopholes, like re-writing of XML and the detection being delayed. The paper presented an approach for early detection of the re-written XML document. The name of the proposed approach is SOAP account. This approach allows early detection of XML re-writing, for the conversation held over a secure channel. The proposed approach works fine for homogenous applications, but no implementation has been done for heterogeneous applications, where the service provider and requester use different languages for implementation. The approach can identify the re-writing of XML at an early stage for homogenous applications.

Garcıa et al. [30] performed a comparative analysis of different implementations of SOAP, the implementations that are compared are JAVA (Axis2) and Erlang. The comparison was performed on two web services. The comparison factors that were used are throughput and load handling by mixing up stress and ideal phases. Erlang has been claimed as the most suitable language for distributed processing, and its lightweight inbuilt processes are very suitable for distributed environment. Using not so complex services resulted in Erlang outperforming Axis2, but as the complexity of the web services increases the performance using Erlang is downgraded as Erlang does not have good parsing capabilities for complex WSDL's. In the scenarios where the web services were complex Axis2 outperformed Erlang. Erlang does not even perform in good in case the computation load is high for the requested services. So keeping these factors in mind, to create a high-performance environment using SOAP, Erlang should be the preference for simpler web services and which have a lesser computational load. As using Eralang, improves the performance in these particular scenarios and creates a huge difference in response time as compared to Axis2 server.

Balfagih and Hassan [31] presented an agent based framework for monitoring SOA-based applications. SOA and web services are one of the best solutions available for integration of systems which consists of applications from different platforms and

enterprise. To maintain the quality of the service between the service requester and the service provider, an agreement is signed which is called the Service Level Agreement (SLA). The SLA defines the obligations, penalties and rights between the provider and the requester regarding the quality of service. There is a need to monitor the fulfillment of SLA at the requester and the provider side. The proposed framework is an automated agent-based framework that ensures the applicability of SLA at requester and provider side. A quality model has been introduced as a basis for the framework, which can identify the quality attributes. The framework deals with the non-functional requirements of the system. The research discovered a few drawbacks of the existing monitoring frameworks that monitor the QoS or the Functionality of the system. The current framework deals with single SLA; it will be in future extended to deal with multiple SLA's.

Tekli et al. [32] presented SOAP processing performance and enhancement approach. As with the advantages what web-services provide and SOA has we can't let go SOAP because of the issues with the performance, it has been proved that SOAP is not as good performer as other technologies like RMI. The performance of SOAP needs to be optimized so as to take the full advantages it provides; this survey paper is about evaluating the current research efforts made to improve the performance of SOAP. There has been much research going on this topic, which focuses on decreasing the serialization and de-serialization time. There has been much research which proved that XML processing is the most time taking process in the SOAP. The research evaluates all the steps in SOAP and the current research which can be used to optimize the step's performance and collectively how these will affect the performance of SOAP. The steps in SOAP where performance can be optimized are, message parsing, serialization, de-serialization, compression, multicasting, security evaluation and data/instruction-level processing, all together optimized can give us a high performing SOAP, we need efficient algorithms for each step. The major focus in improving the performance of all the steps is to reduce re-processing of data, as most of the SOAP messages contain the same data, so in-spite of re-processing the same data at every stage, it is better to identify the common data and process it once. Further the focus is on the processing architecture; the processing can be done at the micro and macro level so as to improve performance. The research presented an idea

of unifying the optimization of all the steps and making SOAP useful for high-performance environment.

Liu et al. [33] presented approach to improving performance of SOAP over the cloud. Cloud computing promises high-performance computing and SOAP has been the best option for providing extensibility, interoperability, and robustness, but not at all good performance. SOAP inherited the benefits as well as flaws of XML. That is why using SOAP on a cloud does not sound appropriate until and unless the performance of SOAP has been optimized. The study has proved that XML dealing is the bottleneck for SOAP. A framework has been proposed which is called HiSOAP, it means High-performance SOAP, and this framework can overcome the flaws of XML by using the techniques like, adaptable invocation of service, delaying the data binding and on-demand building. HiSOAP concurrently processes data binding and extended protocol processes. The research results show that using HiSOAP with delayed data binding and eliminated-reflection increases the performance at a great level. The presented framework needs further improvements as delayed data binding is unable to support message validation and type reference.

Rassan and Alyahya [34] presented SOAP compression framework for improving performance of web services that use XML. Web services use SOAP messages to communicate over network, SOAP messages are quite a bug, due to which communication part is slow, so to improve the performance of web services, SOAP messages compression has been chosen as an option, which wouldn't require any change in SOAP. The compression techniques that have been used are Tagged Sub-optimal Code (TSC), J-bit encoding (JBE) Algorithm, Byte Pair Encoding (BPE), and Huffman Encoding Algorithm. The algorithm designed to add compression to the SOAP, suggests that the compression would work at a lower layer than SOAP. This could reduce the size of SOAP message dramatically according to the algorithm being used and the application which is using compression, the decrease in size would improve the performance of web services over the internet.

Kennedy et al. [35] reviewed the optimization techniques, used for improving performance of SOAP in a loosely coupled environment that uses web services for communication. The use of XML degrades the performance of SOAP. Many techniques have been designed to improve the performance of SOAP. This paper

reviews the different techniques that have been devised to improve the performance of SOAP, client and server side caching, differential serialization and de-serialization, compression of SOAP message and SOAP binding. Because XML, documents are large and have redundant data, these techniques can be very helpful in improving SOAP performance. The comparison has been made in terms of throughput and bandwidth utilization. ReST is lightweight and a good alternative if you do not have complex web services and security requirements; otherwise SOAP is the preferred technology. So keeping in mind the alternative to optimize the performance of SOAP, it is possible for SOAP to out-perform any other technology in case of performance also other than being slow SOAP has provided many benefits no one wants to look over, this leaves us with the option of optimizing all the algorithms of SOAP to improve the performance to a level it is accepted by giving a second thought.

Chapter 4

Problem Statement and Proposed Solution

This chapter includes the motivation behind the thesis, and the problem statement with proposed solution for the problem. It also includes the case study for choosing the technology for the proposed solution.

4.1 Motivation

Integration systems mainly bridge the communication gap between the independent software, which are heterogeneous. [5] Heterogeneous here refers to the architecture, development language, servers being used for deployment (in case of web applications), the domains of the applications, etc. [4] In this sort of environment setup, the integration backbone plays an important role. But there is no existing solution for monitoring the working of the JMS based integration backbone. The user may not be aware at times even if there is a problem in communicating with an application, this could lead to serious problems like loss of integrity and consistency of data for the time the applications don't communicate. So, as to address this issue we need to give a clear picture to the user about the working of the integration backbone.

With the popularity of SOA, the focus has shifted to SOA-based integration, and SOA-based integrations are easier to monitor, they expose their data through the web-services, same way the monitoring can be performed without any additional hard work. Still there are enormous number of applications which aren't SOA based, the existing applications that use, JMS, JCA, or RMI, all these technologies focus on the inner working of the system, the user don't know what's going on in there, if everything is working as expected or not. Even if we use the best of the technologies, still software or hardware could response unexpectedly and things might go weird. There hasn't been much development to address these issues. It is very hard to add API's to already existing communications system, more likely the JMS based systems do not give up much information out and their complex implementations which make monitoring a tough task.

So, to address the monitoring challenges in a JMS environment we need architecture independent, secure, light (shouldn't have much overhead) and highly de-coupled system. The system should give the user a clear picture of all the applications and shouldn't disturb the actual working of the system.

Currently there is no user friendly way that enables the user to monitor the application, the user have to check the databases and logs to keep track of the system, which a layman can't do.

The requirement to address this problem is very serious, because most of the organizations to fulfill the needs of all their domains use a number of software, which are heterogeneous and they communicate through the integration bus. In case something goes wrong the layman can't identify the problem. For every issue the maintenance team is to be consulted, which is not at all cost effective.

JMS does its work reliably, but if there is something wrong at the application side then JMS can't communicate with that application. Due to which the data would be inconsistent for some time, and user might not know the problem. However, if we provide the user a clear picture of the applications, and their working then the user will be able to manage the applications if there is something wrong with them.

4.2 Problem Statement

The problem is to enable monitoring of the existing JMS based integration system. The monitoring framework should be lightweight, secure, and reliable. It should not affect the basic functionality of the integration system. The framework should be able to handle complex data and provide complete monitoring.

4.3 Objectives

1. Analyze and compare the available technologies which can be used for monitoring framework.
2. Design and implementation of the framework, with the best-suited technology
3. Validation of framework

An analysis of the available technologies is required to find the best-suited solution for the problem. On the basis of the most suitable technology, the framework should

be designed. The framework needs to be validated against the requirements set in the problem statement.

4.4 SOAP v/s ReSTFull: A case study

Web Services are an interface for interacting with the software; the W3C definition of Web Services says that they are software to provide interoperability over the internet. They are mainly categorized into two types: Soap Services and ReSTFull Services. Both the web services have different architecture and provide different types of interface. Selecting one of the web services is a major concern for designers and developers. The Soap web services have been accepted worldwide and W3C has defined standards for it, whereas ReSTFull web services are comparatively new, and the standards have not been defined for it, the ReSTFull web services are gaining popularity in software industry due to the less overhead required while developing them. ReSTFull services are based on the principle of using HTTP and URI for designing the web services which makes it easier to implement and access. SOAP web services use its standards for communication, WSDL for definition, UDDI for registering the services, SOAP for communication. SOAP is basically XML based; ReST provides the freedom of using XML, JSON, HTML, etc.

Integration of application is one of the biggest problems for every organization. In an integration environment where different applications need to communicate with each other, using web services can result in loosely coupled architecture. Now the question comes is which technology to be used SOAP or ReSTFull; both have their advantages and disadvantages, a thorough study of both the technologies is requires for making the decision.

4.4.1 SOAP

SOAP is a term used for Simple Object Access Protocol; it is a protocol for XML-based message exchange. SOAP is XML based; the request and response are XML messages. It is language and platform independent; this means the requester and providers can be written in any programming language and can be deployed on different platforms, SOAP is an accepted standard and recommended by W3C. The SOAP needs a service provider, which exposes the SOAP services and registers it to the UDDI, and a requester that requests the services exposed by the provider. The

service requester selects the service from UDDI; the UDDI returns it the service providers, the requester then selects the service provider, and then UDDI returns the WSDL of the service. Another way to exchange the WSDL of a service is, if a provider knows the service consumer then it can share its WSDL with the service consumer, in this case, the provider need not register its service to UDDI. SOAP uses WSDL (Web Services Definition Language) for defining the web services, the WSDL consists of request and response definition and the schemas for request and response, the WSDL are shared by the SOAP provider to the registry, using WSDL the requester can know the details of the web-service, the requester refers the schema's in WSDL to create the response and hit the services.

A SOAP message contains header, body and fault elements, which are encapsulated in envelope element that makes the XML identified as SOAP message, the SOAP header contains the information about the application, the body contains the request/response, and the fault element contains the fault code in case any fault occurs while invoking the SOAP services. SOAP is invoked over HTTP.

SOAP based web services expose their API's to the service requesters through their WSDL, the WSDL are of no meaning to the layman or sometimes even the programmer. The WSDL's are generated by machines for the web-services and understood by the machines to understand the definition of the web-services. SOAP based services use serialization to provide interoperability between machines having different architecture, operating system or the clients using different programming languages. The SOAP message is first serialized then converted into XML and then it is sent to the provider/requester then the message is de-serialized and converted to the destination machines understandable format.

The SOAP does not define how the messages will be transported over the internet, so for transporting the SOAP messages over internet transport protocol is required which we call as "SOAP Bindings". Transport protocols like HTTP, SMTP can be used for transportation of SOAP messages over the internet. Java messaging service (JMS) protocol is mostly used as a protocol binding for Java implementation of SOAP. According to the application requirements, the transport protocols can be used. For example HTTP can be used for applications that require synchronous communication.

4.4.2 ReSTFull

ReST (Representational State Transfer) architecture defines a stateless client-server communication setup, which enables web-services to be accessed as resources. ReSTFull web services use HTTP to communicate, and the operations used to access the resources are the four HTTP operations: Get, Post, Delete, Put. ReSTFull allows the use of XML, JSON, etc. for representation of the resources. The ReSTFull web services are easy to implement; they just need a server and a client setup. ReSTFull request is made over HTTP using URI, and the response could be XML or JSON representation of the resource. Unlike SOAP, the ReSTFull request and response is not encapsulated in an envelope. ReSTFull web services provide a high performance, scalable and loosely coupled system. Usage of the four HTTP methods is as follows:

- Get: It can be used for reading a resource using the URI. This method cannot be used for modifying the resource.
- Put: It can be used to create a new resource.
- Delete: It can be used to delete an existing resource.
- Post: It can be used to update a resource or create a new resource.

ReSTFull services identify every resource using a URI, so only one resource can be accessed per request.

ReSTFull request and response are stateless in nature; all the details should be included in the request and response. The ReSTFull responses can be understood by the layman as they are self-descriptive messages; resources are independent of their representation. ReSTFull web-services make the most use of web, and they are platform independent as they are HTTP based, which can be accessed from any machine.

4.4.3 Experimental Setup

Two web services have been created; one is the SOAP service, and another is the ReSTFull service. Both web services are deployed on the same server, and they return Customer Information, the following functions have been created in the web-services:

- count: returns the total number of customers.

- create: to create a new customer.
- find: to find a customer by id.
- findAll: to read data for all customers.
- edit: to update an existing customer.
- remove: to remove the entry of an existing customer.

SoapUI is used as client for calling both the services; load test has been run parallel to each other for evaluating the performance of both services with the following settings:

- Threads: 10
- Strategy: Threads
- Start Threads: 10
- End Threads: 10
- No of Runs per thread: 1000

4.4.4 Results

SOAP and ReSTFull have their benefits. Besides the high performing ReSTFull services, we discovered that ReSTFull does not fit our architecture because of the complexity and the security requirements. The following table shows the performance metrics for the comparison:

Table 4.2: Comparison of ReSTFull, SOAP and SOAP serialization

Service	Average Turn Around Time(milliseconds)	Bytes	Bps
ReSTFull	10.6626	68750000	644777
SOAP	13.2388	69450000	524593
SOAP with Selective Serialization	11.3603	69450000	611342

ReSTFull outperforms SOAP, but it does not provide the required feature. SOAP is useful when we need a standardized way to communicate and the service provider, and requester do not have much knowledge about each other. SOAP is also useful in case we need to design web services for complex API's.

ReSTFull is preferable in case a stateless service is required, and there are not too frequent changes in the data. ReSTFull services are only useful for simpler API's, but these are easier to integrate with existing system, and lesser over-head is required for implementing ReSTFull services, as the responses are understandable by browsers. So no additional client is to be developed. For using ReSTFull services the service requester and the service provider, needs to be in synchronization to understand the request and response.

The performance of SOAP is entirely due to the use of XML for communication, XML has advantages, but there are many disadvantages as well. XML documents are too huge and have redundant data. So optimizing the performance of SOAP has been the only option, to optimize the performance of SOAP, selective serialization using, templates is an efficient method. The use of templates and selective serialization dramatically improved the performance of SOAP, as XML parsing, including serialization and de-serialization is the bottleneck for SOAP's performance. Serializing and de-serializing only the values of the fields that change with every request is the most optimal way of performance because the XML documents have redundant data.

For internet based application, compression of XML documents also yield in lesser turn-around time for SOAP, as the XML documents are quite large and create too much of network traffic, using some compression algorithms can decrease the size of document before sending it on the network.

4.5 Proposed Solution

Integration software basically provide communication backbone to applications, the users of the integration software don't have a view of what's going inside the system, So for monitoring of the communication between the applications a framework has been designed and developed. While designing and developing applications and modules, the most requirement is mainly to have a highly decoupled system. We need

to maintain that independence of application while providing the monitoring framework. SOA based applications are easy to monitor as the web services can be used for creating API above them. So considering this benefit of SOA and keeping in mind the decoupling of the system web services has been used in the suggested architecture to expose the data which is required for monitoring of the system.

Using web services doesn't require much overhead, as they communicate over the internet in form of lightweight XML, JSON, HTML, etc. documents, and building web services doesn't affect the basic functionality if the application on which the web services has been created.

Web services are the most appropriate solution for the integration environments, as there are several applications that are heterogeneous in nature. Web services can be implemented above all types of application and developer need not worry about the underlying architecture of the machine or the character set or any other communication issue.

Another plus point of using web services is that they can be accessed in multiple ways; they are understandable by humans as well as the programming languages. The data from different web services can be integrated into one and can be presented to the user. The web services use the standardized technologies and can be used over the internet that makes them portable, easy to use, and the developer, or the user need not buy any license to use these technologies.

Using web services gives the developers the freedom to work on any language, so the developers of different applications need not to be bound with each other for choosing the same development language.

Web services are a good solution for the applications which don't need to hold too much of data, which makes it very suitable for monitoring system, as the monitoring system needs only the current state of the system doesn't deal with the past state of the system. For an existing system, it is easy to implement web services because we do not need to change the existing working of the system.

4.5.1 Design and Architecture

The architecture of the monitoring framework defines two levels of monitoring one are application level and other is Global. Each Application need to have data collectors, which would be lightweight programs which collects the monitoring data at regular interval and event collectors which collect the data whenever some events happen within the system, so as to keep the system up to date, then we need a repository to store the system state for the interval until we get the latest data, so as to improve the availability.

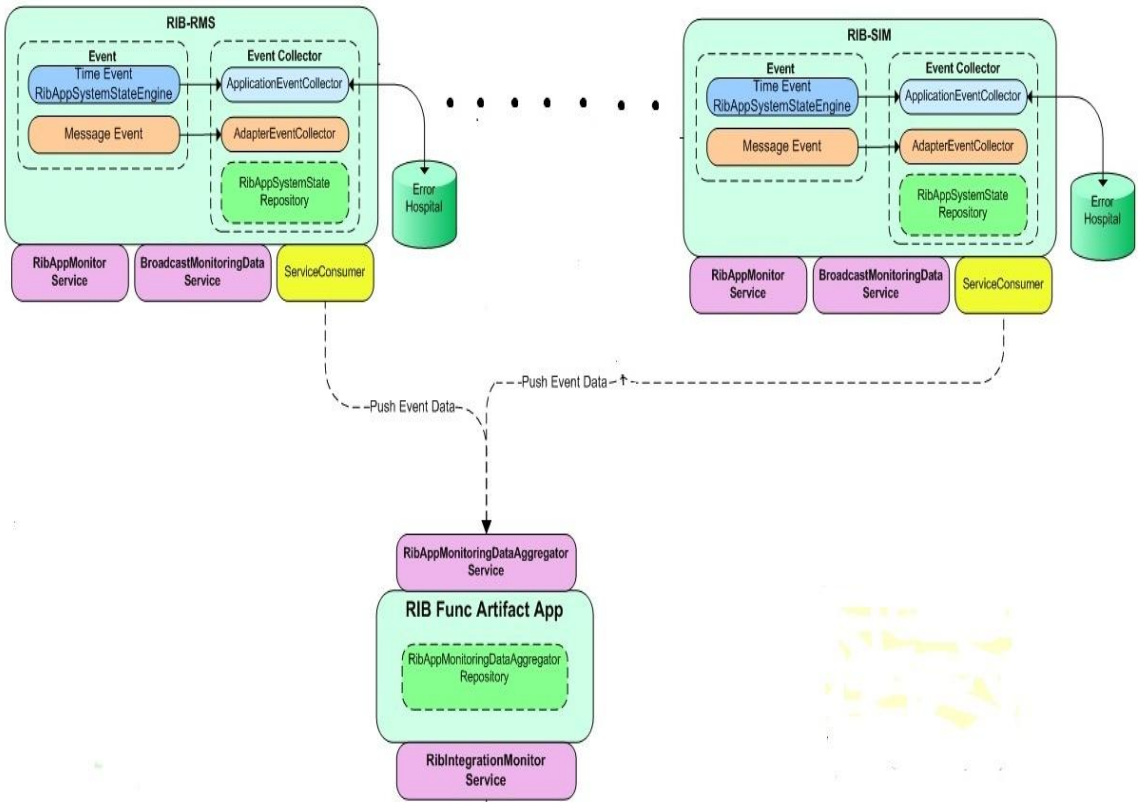

Figure 4.1 Architecture of the Monitoring Framework

For every application we need two web services, one for collecting the data and other for broadcasting, we are creating them as two separate web services because we are providing the application level monitoring of the system also, and the broadcasting service consumes the Global web services and sends them the application level monitoring data.

At the global level, we need two services one to collect the data from all the application that would be an aggregator service and other to expose the data collected by the aggregator service.

Event:

Events are of two types:

1. Time Event
2. Message Event

Time Event: Time events refer to the system status of the application, these events work at the application level.

Message Event: Message events basically refers to the event triggered when a message was published or subscribed, message events are adapter bound, work at adapter level, these events collect the data for an adapter are process it.

For different events we have different event collectors, these are:

1. Application Event Collector
2. Adapter Event Collector

Application Event collector that calls the Time events to collect the application level data, these event collectors collects the data from the error hospital database also. Application Event collector collects the following information:

- Status of the app
- Up since
- Total event counts
- Error Database
    - Total messages in error database
    - Total messages in error database due to dependency
    - Messages in error database for family
- Resource Usage
    - CPU usage
    - Current memory, max memory, and free memory

Adapter Event Collector, on the other hand, collects the adapter-specific data; it collects the data for all the adapters. Adapter Event collector collects the following information for all the subscribing and publishing adapters:

- Adapter Status
- Data collection time
- Adapter type
- Events processed
    - Total events count
    - Number of commits
    - Number of rollbacks
    - Most recent event adapter execution time
    - Most recent event integration API execution time

The repository is used for keeping the data between the data collection events, a simple EJB can be used for this purpose, this will make the repository in memory. The requirement of the repository is to make the system efficient so that for every request the services need not collect the data by calling the collectors; the collectors can be scheduled according to requirement. So the repository holds the data for the requests between the collectors scheduled calling. The Data collected by the collectors is then stored in this repository; the Rib App Monitoring service will collect the data from the repository and fulfills the request whenever a service request is received. This Rib App monitoring service could be hit by an external API and also by the broadcast monitoring service that pushes the data to the Integration monitoring service. The purpose of the broadcast monitoring service is to separate the application level and integration level monitoring. The broadcasting service will send the data received by hitting the app monitoring service to the integration level service.

There are two integration level services, one is Data aggregator service and another in Integration monitoring service, the data aggregator service is called by the application level broadcasting service that sends the application level data to the aggregator service. The aggregator service is to collect the data from all the applications and mange the expiration of data after a fixed interval so that any inconsistent information is not passed to the user. The data collected by the data aggregator service is used by the Integration monitoring service to expose the data to the API's. As we are using

Web services any programming language can be used to develop the API's. The API's can hit the services at regular interval of time and get the monitoring data.

A repository is required at the integration level also so as to maintain the data between the communication gap between the broadcasting service and the data aggregator service, an in-memory repository that could be developed using the EJB's is the most appropriate solution as we don't want to effect the systems performance. The aggregator service will save the data to the repository, and the integration monitoring service will read the data from the repository to service the requests received from the API's.

Using the repository in the system improves the performance of the system, but comes with a cost of a bit update data, but considering the performance, a bit delay in update of data can be accepted.

4.5.2 Security

Security of web services has always been a point under question, whenever we communicate over the internet we need to ensure security, if our system is exposing some sensitive data on the internet, then security requirements are even more crucial, keeping all these things in mind, Web service security policies has been used to secure the system.

Policy A: Policy A of web service security policies ensures the use of SSL and authentication for calling the web services; this ensures only authorized the use of the services that too over SSL.

Policy C: Policy C of web service security policies ensures the use of authentication for calling the web services; this ensures only authorized the use of the services.

Policy A is required for sensitive information being passed over internet and Policy C is the default policy defined for the services.

4.5.3 Optimization of SOAP

SOAP inherits the benefits as well as drawbacks of XML, using XML provide so many facilities, but at the same time it affects the performance and slows down

SOAP. ReSTFull services do not meet the standards required for our problem, so, we needed to optimize the performance of SOAP.

The techniques used for SOAP optimization is, in-spite of using the normal serialization procedure, the XML documents being used for sending out the data for JMS monitoring contains a lot of data which is static throughout the calls until moreover, there has been some new clients or some clients de-registering, which happens only if there is a change in business logic.

Due to the data like, adapters name, number of adapters, applications name, all these static data don't need to be serialized for every request, so to save time and optimize the performance, templates have been used to generate the XML documents that are transferred by SOAP, the template of the document will be recreated every time there are some new clients on JMS, that happens only when there is some business change, so it will rarely be required.

The improvement of results using selective serialization with the help of templates has been shown already, the performance of SOAP is close to ReSTFull, still ReSTFull is a bit better performance wise, but with the advantages, standards and security SOAP provides is unmatchable. For applications which can't compromise on security and have complex business logic, SOAP is preferred option. Many methods have been suggested by recent researches for optimization of SOAP, a unified approach towards optimizing all the steps of SOAP can help achieving the required performance of SOAP.

Chapter 5

Testing & Results

Testing has been performed for verification of working of the proposed framework, and a GUI has also been developed over the proposed framework to observe the expected working of the system. The testing of the web services has been performed using SOAPUI tool, the following screenshots show the testing results:

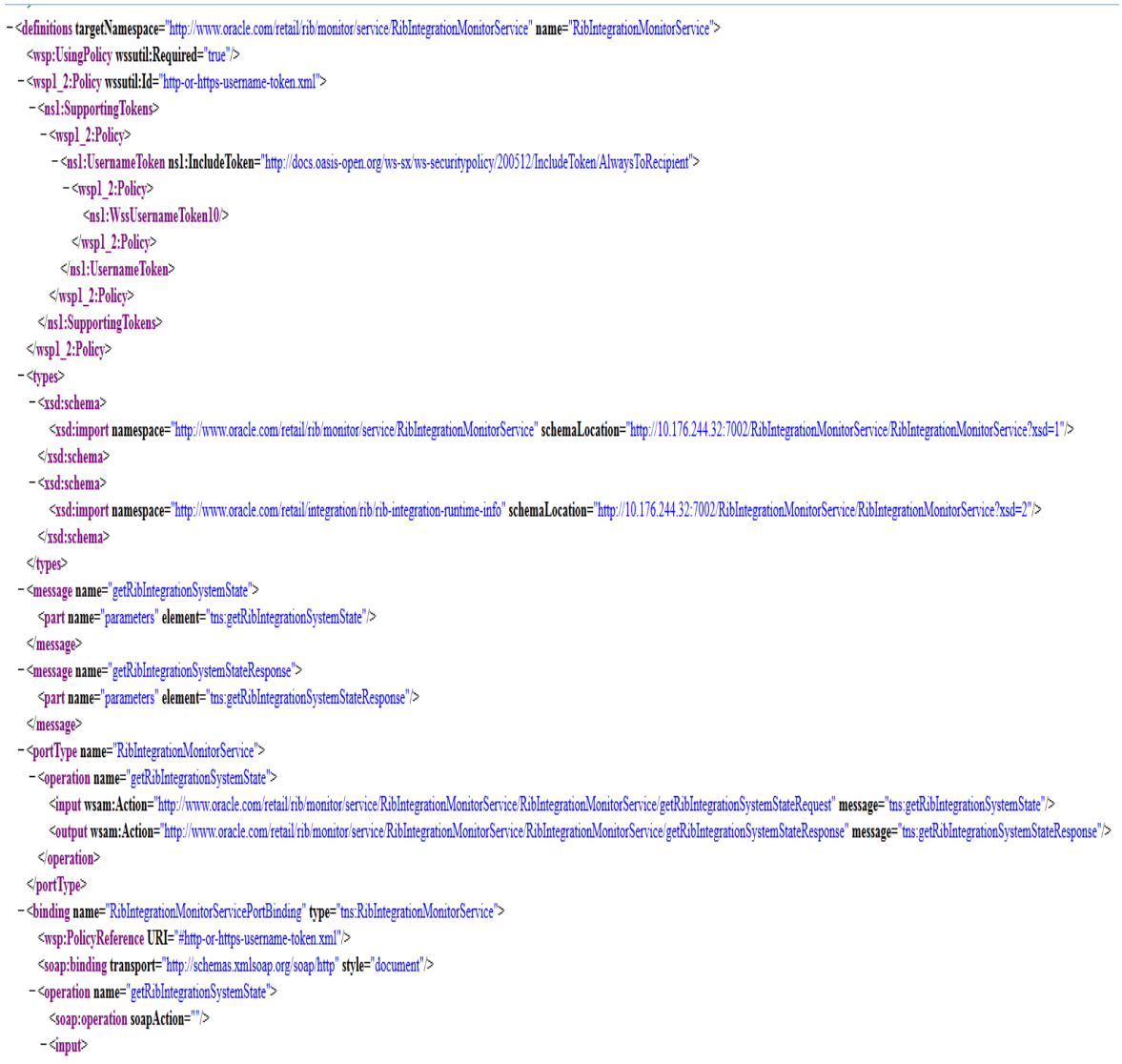

Figure 5.1: WSDL for Integration Monitoring Service

Figure 5.2: WSDL for Application Monitoring Service

Figure 5.3: WSDL for Data Aggregator Service

```
- <definitions targetNamespace="http://www.oracle.com/retail/rib/monitor/service/BroadcastMonitoringDataService" name="BroadcastMonitoringDataService">
    <wsp:UsingPolicy wssutil:Required="true"/>
  - <wspl_2:Policy wssutil:Id="usernametoken.xml">
    - <ns1:SupportingTokens>
      - <wspl_2:Policy>
        - <ns1:UsernameToken ns1:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/IncludeToken/AlwaysToRecipient">
          - <wspl_2:Policy>
              <ns1:WssUsernameToken10/>
            </wspl_2:Policy>
          </ns1:UsernameToken>
        </wspl_2:Policy>
      </ns1:SupportingTokens>
    </wspl_2:Policy>
  - <types>
    - <xsd:schema>
        <xsd:import namespace="http://www.oracle.com/retail/rib/monitor/service/BroadcastMonitoringDataService" schemaLocation="http://10.176.244.32:7003/BroadcastMonitoringDataService/BroadcastMonitoringDat
      </xsd:schema>
    </types>
  - <message name="transmitRibAppMonitoringData">
      <part name="parameters" element="tns:transmitRibAppMonitoringData"/>
    </message>
  - <message name="transmitRibAppMonitoringDataResponse">
      <part name="parameters" element="tns:transmitRibAppMonitoringDataResponse"/>
    </message>
  - <portType name="BroadcastMonitoringDataService">
    - <operation name="transmitRibAppMonitoringData">
        <input wsam:Action="http://www.oracle.com/retail/rib/monitor/service/BroadcastMonitoringDataService/BroadcastMonitoringDataService/transmitRibAppMonitoringDataRequest" message="tns:transmitRibAppMo
        <output wsam:Action="http://www.oracle.com/retail/rib/monitor/service/BroadcastMonitoringDataService/BroadcastMonitoringDataService/transmitRibAppMonitoringDataResponse"
        message="tns:transmitRibAppMonitoringDataResponse"/>
      </operation>
    </portType>
  - <binding name="BroadcastMonitoringDataServicePortBinding" type="tns:BroadcastMonitoringDataService">
      <wsp:PolicyReference URI="#usernametoken.xml"/>
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    - <operation name="transmitRibAppMonitoringData">
        <soap:operation soapAction=""/>
      - <input>
          <soap:body use="literal"/>
        </input>
```


Figure 5.4: WSDL for Broadcasting Service

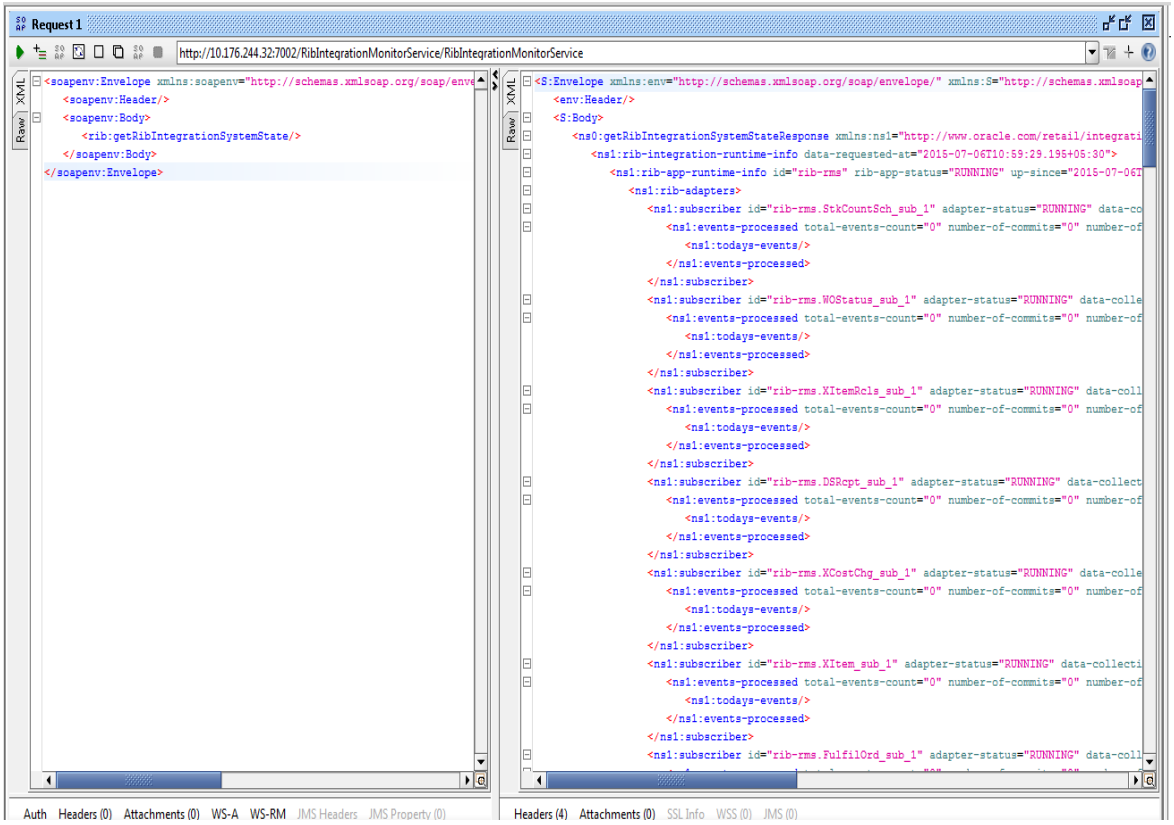

Figure 5.5: Request and Response for Integration Monitoring Service

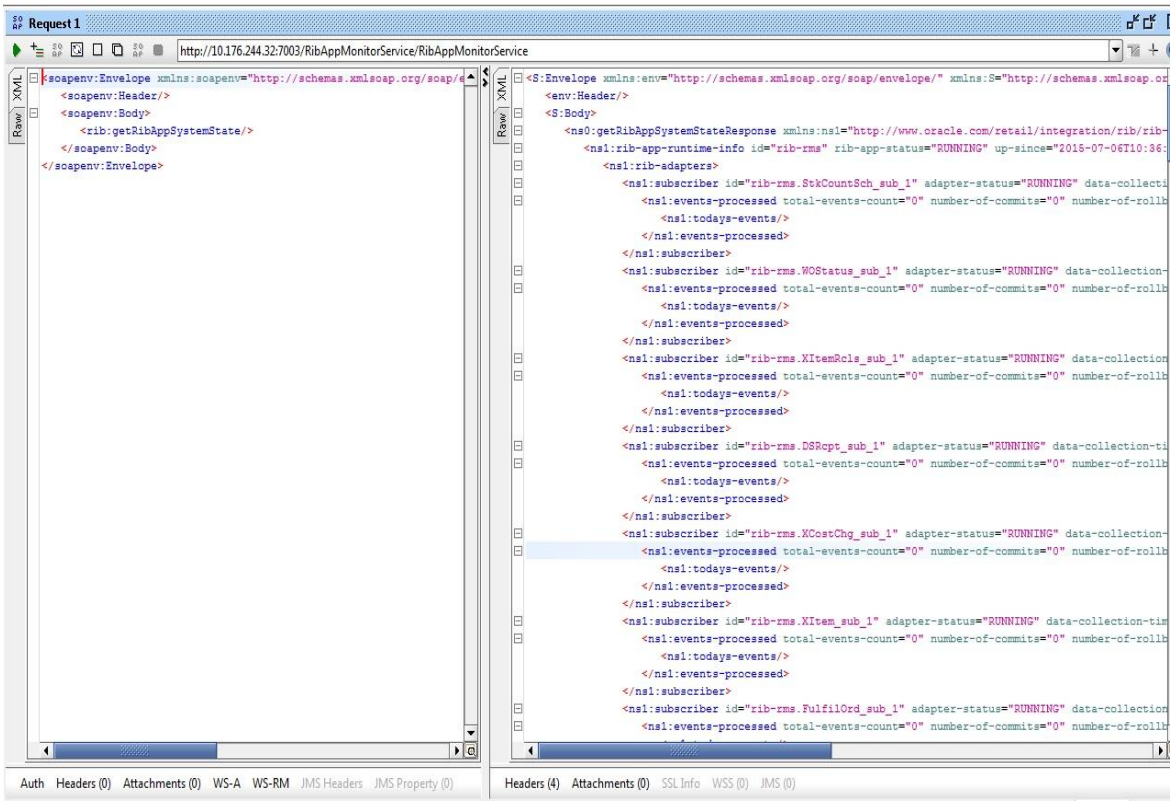

Figure 5.6: Request and Response for Application Monitoring Service

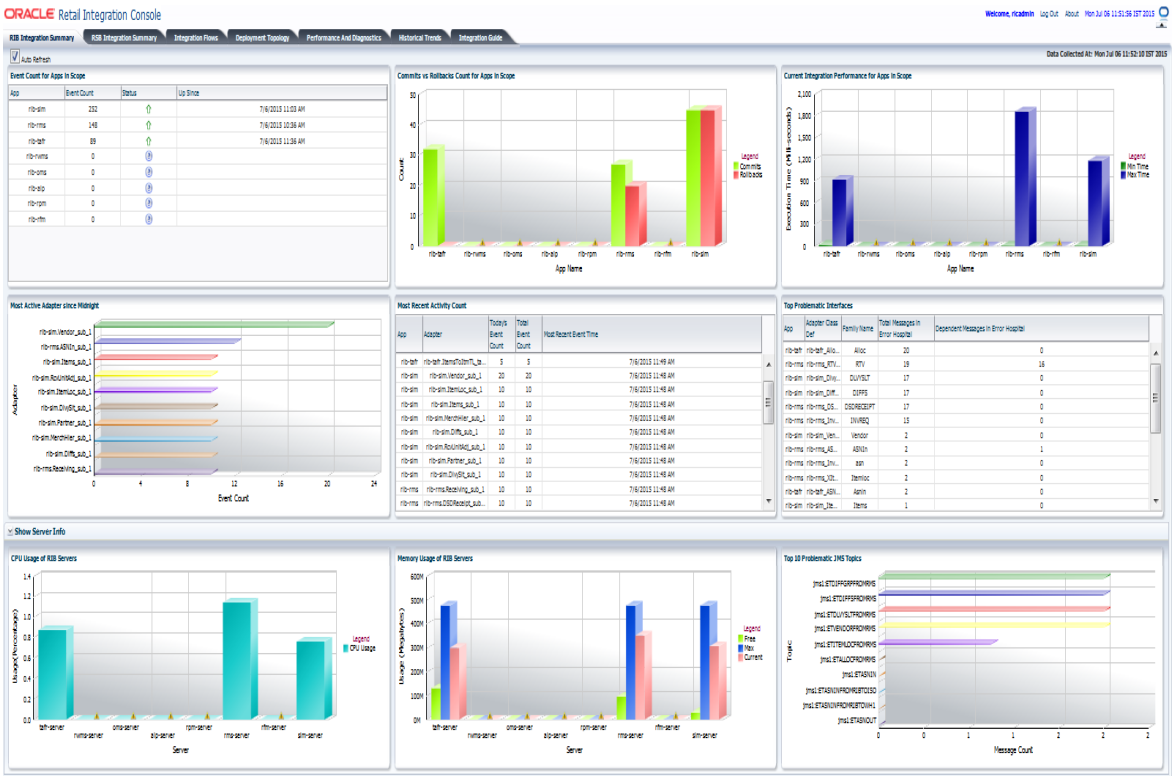

Figure 5.7 Screenshot of the GUI created the framework

Chapter 6

Conclusions & Future Scope

The proposed framework enables monitoring of JMS based integration system, and provides an approach to improve the performance of SOAP services. The proposed framework uses SOAP web services for monitoring. The framework fulfills the security and performance requirement of the monitoring framework. The performance have been kept under check by using selective serialization using templates for SOAP and using a repository to reduce the number of web services calls. The security has been ensured using authentication and SSL options of web service security policies.

6.1 Conclusion

Monitoring frameworks are required for back-end communication application, to let the users have the insight of the systems working. Web services are a very efficient method, for extending the existing system. Objectives of the thesis have been achieved and validated by testing the framework. All requirements of monitoring framework are fulfilled using web services. They are lightweight and don't interrupt the basic functionality of the current system and is easy to use and most efficient method. Web services are available over the internet so can be accessed from anywhere; this makes them the most appropriate solution for monitoring.

6.2 Future Scope

The current framework lacks in debugging capabilities, the future work will be concentrated on adding the debugging capability to the system. The system should be intelligent enough to find out the cause of the system and report the problems to the solution; this would require the use of some Artificial intelligent based on heuristics.

References

[1] W. Farrell, “Introducing the Java Message Service”, *IBM*, 08 June 2004.

[2] The Java EE 6 Tutorial, Part No: 821–1841–16, January 2013. Available at: http://docs.oracle.com/javaee/6/tutorial/doc/javaeetutorial6.pdf

[3] Oracle Retail Integration Bus Operations Guide Release 14.1, E57320-01, December 2014.

[4] Oracle Retail Integration Bus Installation Guide Release 14.1, E58590-03, March 2015.

[5] Oracle Retail Integration Bus Implementation Guide Release 14.1, E57321-01, December 2014.

[6] Oracle Retail Integration Bus Security Guide Release 14.1, E57322-01, December 2014.

[7] C. Mellon, Software System Integration, Software Engineering Institute, Available at: http://www.sei.cmu.edu/productlines/frame_report/softwaresi.htm.

[8] D. Chappell, SOAP v/s REST: Complements or Competitors? Available at: http://proceedings.esri.com/library/userconf/devsummit09/papers/keynote_chappell.pdf.

[9] Gold Stone Technologies, Enterprise Application Integration, Available at: http://www.goldstonetech.com/investor%20info/white%20papers/EAI%20Overview.pdf.

[10] R. Coqueret Engagement and M. Fiammante, “Choosing among JCA, JMS, and web services for EAI Positioning alternative interface technologies for enterprise integration” Available at : http://www.ibm.com/developerworks/library/ws-jcajms/ws-jcajms-pdf.pdf .

[11] Sun Java System Message Queue 4.3 Technical Overview, Available at: http://docs.oracle.com/cd/E19340-01/820-6424/aeraq/index.html.

[12] J. Waldo, “Remote procedure calls and Java Remote Method Invocation”, Concurrency, IEEE (Volume: 6, Issue: 3), 5-7, Jul-Sep 1998.

[13] M. Psiuk, T. Bujok and K. Zieliński, "Enterprise service bus monitoring framework for SOA systems", IEEE Transactions on Services Computing, (Volume: 5, Issue: 3), 450-466, 2012.

[14] S. Baghermousavi, H. Rashidi, and H. Haghighi, "An approach to reduce QoS monitoring overhead in ESB", in 8th International Conference on e-Commerce in Developing Countries: With Focus on e-Trust, ECDC'14, pp. 1-5, Mashhad, Iran, 2014, IEEE.

[15] X. Shi, "Sharing service semantics using SOAP-based and REST Web services." IT Professional (Volume: 8, Issue: 2), 18-24, 2006.

[16] A. Ben Shil and M. Ben Ahmed, "Additional Functionalities to SOAP, WSDL and UDDI for a Better Web Services Administration", in Information and Communication Technologies, ICTTA'06, 2nd, (Volume: 1), pp. 572-577. 2006, IEEE.

[17] G. Lam and D. Rossiter, "A SOAP-based streaming content delivery framework for multimedia Web services", in Asia-Pacific Services Computing Conference, APSCC'08, pp. 1097-1102, Yilan, 2008, IEEE.

[18] K. Al-Zoubi and G. Wainer, "Using REST web-services architecture for distributed simulation" in Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation, pp. 114-121, Lake Placid, NY, 2009, IEEE.

[19] J. Lee, S. Lee and P. Wang, "A Framework for Composing SOAP, Non-SOAP and Non-Web Services", IEEE Transactions on Services Computing, (Volume: 8, Issue: 2), 240-250, 2015.

[20] R. Ramos de Oliveira, R.V. Vieira Sanchez, J. C. Estrella, R. Pontin de Mattos Fortes and V. Brusamolin, "Comparative evaluation of the maintainability of RESTful and SOAP-WSDL web services", in 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems, MESOCA'13, pp. 40-49, Eindhoven, 2013, IEEE.

[21] W. Jiang, D. Lee and S. Hu, "Large-scale longitudinal analysis of soap-based and restful web services" in 2012 IEEE 19th International Conference on Web Services, ICWS'12, pp. 218-225, Honolulu, HI, 2012, IEEE.

[22] F. Belqasmi, J. Singh, S. Y. Bani Melhem and R. H. Glitho, "SOAP-based vs. RESTful Web services: A case study for multimedia conferencing", IEEE Internet Computing, (Volume: 16, Issue: 4), 54-63, 2012.

[23] F. AlShahwan and K. Moessner, "Providing soap web services and restful web services from mobile hosts", in Fifth International Conference on Internet and

Web Applications and Services, ICIW'10, pp. 174-179, Barcelona, 2010, IEEE.

[24] X. Feng, J. Shen and Y. Fan, "REST: An alternative to RPC for Web services architecture" in First International Conference on Future Information Networks, ICFIN'09, pp. 7-10, Beijing, 2009, IEEE.

[25] N. Abu-Ghazaleh, M. J. Lewis and M. Govindaraju, "Differential serialization for optimized SOAP performance", in 13th International Symposium on High Performance Distributed Computing, HPDC'04, June 2004, IEEE.

[26] M. Govindaraju, A. Slominski, K. Chiu, Pu Liu, R. Van Engelen and M. J. Lewis, "Toward characterizing the performance of SOAP toolkits", in proceedings of Fifth IEEE/ACM International Workshop on Grid Computing, pp. 365-372, 2004, IEEE.

[27] D. Davis and M. Parashar, "Latency performance of SOAP implementations", in 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002, pp. 407-407, 2002, IEEE.

[28] L. Lei, C. Niu, N. Chen and J. Wei, "High Performance Web Services Based on Service-Specific SOAP Processor", in International Conference on Web Services, ICWS'06, pp. 603-610, Chicago, IL, 2006, IEEE.

[29] M. A. Rahaman and A. Schaad, "Soap-based secure conversation and collaboration", in IEEE International Conference on Web Services, ICWS'07, pp. 471-480, Salt Lake City, UT, 2007, IEEE.

[30] A. Jose Garcıa, R. Blanco and A. Blanco, "A comparative performance evaluation of different implementations of the SOAP protocol", in Fifth European Conference on Web Services, ECOWS'07, pp. 109-118, 2007, IEEE.

[31] Z. Balfagih and M. F. B. Hassan, "Agent based monitoring framework for SOA applications quality", in 2010 International Symposium in Information Technology, ITSim'10, (Volume: 3), pp. 1124-1129, Kuala Lumpur, 2010, IEEE.

[32] J. M. Tekli, E. Damiani, R. Chbeir and G. Gianini, "SOAP processing performance and enhancement", IEEE Transactions on Services Computing, (Volume: 5, Issue: 3), 387-403, 2012.

[33] H. Liu, X. Liu, J. Li, Y. Zhao and Z. Li, "Building high-speed roads: Improving performance of SOAP processing for cloud services", in 2011

IEEE 6th International Symposium on Service Oriented System Engineering, SOSE'11, pp. 72-78. Irvine, CA, 2011, IEEE.

[34] I. Al Rassan and H. Alyahya, "Improve XML Web Services Performance Using SOAP Compression", in Proceedings of the International Conference on Semantic Web and Web Services, SWWS'14, Las Vegas Nevada, USA, 2014, CSREA.

[35] M. Kennedy Senagi, O. George, S. Arthur, C. Wilson, K. Jades and K. Nyeri, "A Review of SOAP Performance Optimization Techniques to Improve Communication in Web Services in Loosely Coupled Systems", International Journal of Computer Science Issues (IJCSI), (Volume: 11, Issue: 2), 2014.

List of Publications

- COMMUNICATED
    - Palka and Ajay Kumar Loura, "**Soap v/s ReSTFull: A Case Study"**