SEAMLESS FAILOVER USING WSO2

A PROJECT REPORT

Submitted by

S.ANUMITHA (12C13)

S.MEENA PREETHI (12C53)

C.K.PIRAVEEN GANESH(12C68)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

THIAGARAJAR COLLEGE OF ENGINEERING , MADURAI-15

(A Government Aided ISO 9001:2008 Certified Autonomous Institute Affiliated to Anna University)

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2016

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI-15

(A Government Aided ISO 9001:2008 Certified Autonomous Institute Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report "**SEAMLESS FAILOVER USING WSO2** "is the bonafide work of "**ANUMITHA.S(12C13) , MEENA PREETHI.S (12C53),PIRAVEEN GANESH.C.K(12C68)"** who carried out the project under my supervision during the Academic Year 2015-2016.

SIGNATURE	**SIGNATURE**
Dr. S MERCY SHALINI	Dr. P. CHITHRA
HEAD OF THE DEPARTMENT	**ASSISTANT PROFESSOR**
COMPUTER SCIENCE AND ENGG	Mr.K.NARASIMAMALLIK ARJUNAN
THIAGARAJAR COLLEGE OF ENGG,	**ASSISTANT PROFESSOR**
MADURAI-625015.	Mr. N.SHIVAKUMAR
	ASSISTANT PROFESOR
	COMPUTER SCIENCE AND ENGG
	THIAGARAJAR COLLEGE OF ENGG
	MADURAI-625015

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our sincere gratitude to our Honorable Correspondent **Mr. Karumuthu T. Kannan** for his encouragement and guidance in our academic life at Thiagarajar College of Engineering. We wish to express our profound gratitude to our honorable Principal **Dr.V.Abhai Kumar** for his overwhelming support provided during my course span in Thiagarajar College of Engineering.

We are grateful to our department head **Dr.MERCY SHALINI**, for all the academic help extended in our project. We thank our supervisor **Mr. Shankar**, for all the guidance for the successful completion of this project and led us from the scratch of this project's commencement with undivided attention and skillful expertise in developing this project in a systematic and professional manner.

We express our sincere thanks to all the staff members and also the lab technicians of the Department of Information Technology for their constant and dedicated service to brighten our career.

We thank God Almighty and our parents for helping me procure such a challenging and interesting project, and completing the same in due course without much difficulty.

Abstract

In real time environment, when a backend system component fails the ability of the front end component to failover to other active backend components is limited and often done manually except few routine error scenarios. To add there is tight coupling between the frontend and backend components .The focus of the project is to make the failover seamlessly without manual intervention and increase the scope of failover conditions by adding Drools rules to route the request or failover from the front end components based on certain pre-configured rules. The project also stresses the need for establishing a middleware layer like ESB to facilitate failover and to have loose coupling between the frontend and backend components. The capabilities of the proposed failover framework and its advantages are demonstrated by providing a middleware integrating wso2 ESB and Drools as the rules engine. In our project, applications of different domains and using different message formats can communicate/failover among each other using the wso2/Drools. For example when there is an error code from the backend and within a particular time interval of the day, the request is redirected to the secondary server based on the rules defined in the rules engine (Drools).

Chapter 1

1. INTRODUCTION

1.1 Enterprise service bus:

An enterprise service bus (ESB) is a software architecture model used for designing and implementing communication between mutually interacting software applications in a service-oriented Architecture (SOA). As a software architectural model, for distributed computing, it is a specialty variant of the more general client server model and promotes agility and flexibility with regard to communication between applications. Its primary use is in enterprise application integration (EAI) of heterogeneous and complex landscapes.

1.2 ESB as Software:

The ESB is implemented in software that operates between the business applications, and enables communication among them. Ideally, the ESB should be able to replace all direct contact with the applications bus, so that all communication takes place via the ESB. To achieve this objective, the ESB must encapsulate the functionality offered by its component applications in a meaningful way. This typically occurs through the use of an enterprise message model. The message model defines a standard set of messages that the ESB transmits and receives. When the ESB receives a message, it routes the message to the appropriate application. Often, because that application evolved without the same message model, The ESB is implemented in software that operates between the business applications, and enables communication among them. Ideally, the ESB should be able to replace all direct contact with the applications on the e ESB has to transform the message into a format that the application can interpret. A software adapter fulfills the task of effecting these transformations, analogously to a physical adapter.

ESBs rely on accurately constructing the enterprise message model and properly designing the functionality offered by applications. If the message model does not completely encapsulate the application functionality, then other applications that desire that functionality may have to bypass the bus, and invoke the mismatched applications directly. Doing so violates the principles of the ESB model, and negates many of the advantages of using this architecture.

The beauty of the ESB lies in its platform-agnostic nature and the ability to integrate with anything at any condition. It is important that Application Lifecycle Management vendors truly apply all the ESB capabilities in their integration products while adopting SOA. Therefore, the challenges and opportunities for EAI vendors are to provide an ation solution that is low-cost, easily configurable, intuitive, user-friendly, and open to any tools customers choose.

1.3 Failover:

Failover is a backup operational mode in which the functions of a system component (such as a processor, server, network, or database, for example) are assumed by secondary system components when the primary component becomes unavailable through either failure or scheduled down time. Used to make systems more fault-tolerant, failover is typically an integral part of mission-critical systems that must be constantly available. The procedure involves automatically offloading tasks to a standby system component so that the procedure is as seamless as possible to the end user. Failover can apply to any aspect of a system: within an personal computer, for example, failover might be a mechanism to protect against a failed processor; within a network, failover can apply to any network component or system of components, such as a connection path, storage device, or Web server.

An automatic server failover solution is that it works by having two servers with identical content on them – a primary server and a secondary server. A third server monitors the primary server and detects if there is a problem. If there is, it will automatically update the DNS records for your website so that traffic will be diverted to your secondary server. Once your primary server is functioning again, traffic will be routed back to your primary server. Most of the time your users won't even notice a thing.

The main disadvantage of using automatic server failover is application downtime.Once the failover occurs, at some point you usually need to fail back to the original principal again, which entails more application downtime. So,we move on to ESB for integration and rule engine for generating rules for failover.

1.4 WSO2 Enterprise Service Bus:

WSO2 Enterprise Service Bus is a lightweight, high performance, and comprehensive ESB. 100% open source, the WSO2 ESB effectively addresses integration standards and supports all integration patterns, enabling interoperability among various heterogeneous systems and business applications. The cloud-enabled, multi-tenant WSO2 ESB is also available on the cloud as a service.

WSO2 ESB can be deployed at the heart of an SOA architecture or on the edge to mediate, enrich, transform messages across a variety of systems, including legacy applications, SaaS applications, as well as services and APIs.

In its fifth generation, WSO2 ESB has been deployed in production at hundreds of customers and is used in a wide variety of use cases: as a service gateway, mediation engine across SAP, Salesforce, Microsoft .NET services, and as a healthcare hub or in IoT scenarios, thanks to its support for transports such Apache Kafka and MQTT.

WSO2 ESB comes with a large set of building blocks, called mediators, which are used to construct mediation flows. Mediators cover data manipulation, data enrichment, connections to external systems, invoking external services and

APIs, business events generation, and database integration among others. Additionally, WSO2 ESB supports the entire set of enterprise integration patterns. Mediation flows can be reused as-is or transformed into templates for further reuse across integration applications.

1.4.1 Proxy Services:

A proxy service is a virtual service hosted in the ESB runtime. For a service client it appears to be a real Web Service which accepts service requests. The proxy service mediates any accepted requests and forwards them to a specified endpoint, most of the time to an actual Web Service. The responses coming back from the target endpoint are mediated back to the client which sent the original service request. Proxy services often make references to sequences, endpoints and local entries.

An 'inSequence' in a proxy service would decide how the message would be handled after the proxy service receives the message. The 'outSequence' defines how the response is handled before it is sent back to the client. Unlike sequences and endpoints which can be stored and loaded from the registry, proxy services cannot be loaded from the registry. However a proxy service can make references to sequences and endpoints stored in the registry. The following proxy service is used for this scenario.

1.4.2 Performance:

Designed for high throughput and low latency, the WSO2 Enterprise Service Bus handles thousands of transactions per second (TPS) and clocks in a few ms latency. We recently conducted a comparison across multiple open sources ESB. Improving the performance of WSO2 ESB is to allow users to run more transactions with less hardware. ESB 4.9 brings up to forty percent more performance over version 4.8.At the same time, ESB reduced transaction latency.

1.4.3 Features:

Route, Mediate and Transform:

- Routing: header based, content based, rule-based and priority-based routing
- Mediation: support for all Enterprise Integration Patterns or EIPs, including guaranteed delivery and message enrichment database integration, event publishing, logging & auditing, validation

Acts as Message, Service, API, and Security Gateway:

- Expose existing applications & services over different protocols and message formats

- Enforce and manage security centrally, including authentication, authorization, and entitlement
- Ensure load balancing for scalability and failover for high availability of business endpoints

Get High Performance, High Availability, Scalability & Stability:

- Get declarative development with configuration instead of code
- Enable easy configuration of fault tolerant mediations with support for error handling
- Ensure server customization via feature provisioning of any WSO2 middleware
- Integrate with WSO2 Developer Studio, Eclipse-based IDE for all WSO2 products

1.4.4 ADVANTAGES:

- 100% OPEN SOURCES

With no tricks or hidden agendas the WSO2 Platform delivers, rapid innovation by integrating Apache and other open source projects, affordability by not having to pay heavy licensing costs, visibility into how the products operate under the hood, and flexibility in configuring and extending the open source code to meet your requirements.

- Comprehensive Platform

This ensures making innovative connections that advance your organization is seamless, fluid, and rapid taking less engineering resources and integration time. Security, identity, logging, monitoring, and management services combined with interoperable protocols enable you to leverage what you know and what you may already have. You have the flexibility to easily evolve and grow your technology in a consistent and stable manner.

1.5 ACTIVEMQ:

Apache ActiveMQ is an open source message broker written in Java together with a full Java Message Service (JMS) client. It provides "Enterprise Features" which in this case means fostering the communication from more than one client or server. Supported clients include Java via JMS 1.1 as well as several other "cross language" clients. The communication is managed with features such as computer clustering and ability to use any database as a JMS persistence provider besides virtual memory, cache, and journal persistency.

ActiveMQ is used in enterprise service bus implementations such as Apache ServiceMix and Mule. Other projects using ActiveMQ include Apache Camel and Apache CXF in SOA infrastructure projects.

1.5.1 FEATURES:

- Supports a variety of Cross Language Clients and Protocols from Java, C, C++, C#, Ruby, Perl, Python, PHP
- OpenWire for high performance clients in Java, C, C++, C#
- Stomp support so that clients can be written easily in C, Ruby, Perl, Python, PHP, ActionScript/Flash, Smalltalk to talk to ActiveMQ as well as any other popular Message Broker
- support allowing for connections in an IoT environment.
- full support for the Enterprise Integration Patterns both in the JMS client and the Message Broker
- Supports many advanced features such as Message Groups, Virtual Destinations, Wildcards and Composite Destinations
- Fully supports JMS 1.1 and J2EE 1.4 with support for transient, persistent, transactional and XA messaging

- Spring Support so that ActiveMQ can be easily embedded into Spring applications and configured using Spring's XML configuration mechanism
- Includes JCA 1.5 resource adaptors for inbound & outbound messaging so that ActiveMQ should auto-deploy in any J2EE 1.4 compliant server
- Supports very fast persistence using JDBC along with a high performance journal
- Designed for high performance clustering, client-server, peer based communication
- REST API to provide technology agnostic and language neutral web based API to messaging
- Axis Support so that ActiveMQ can be easily dropped into either of these web service stacks to provide reliable messaging
- Can be used as an in memory JMS provider, ideal for unit testing JMS.

1.5.2 ActiveMQ Queue:

A JMS Queue implements load balancer semantics. A single message will be received by exactly one consumer. If there are no consumers available at the time the message is sent it will be kept until a consumer is available that can process the message. If a consumer receives a message and does not acknowledge it before closing then the message will be redelivered to another consumer. A queue can have many consumers with messages load balanced across the available consumers. So Queues implement a reliable load balancer in JMS.

1.5.3 ActiveMQ Topics:

In JMS a Topic implements publish and subscribe semantics. When you publish a message it goes to all the subscribers who are interested - so zero to many subscribers will receive a copy of the message. Only subscribers who had an active subscription at the time the broker receives the message will get a copy of the message.

1.5.4 ActiveMQ Connections:

An Apache ActiveMQ connection can be configured by explicitly setting properties on the ActiveMQConnection or ActiveMQConnectionFactory objects themselves via the bean properties or using the following URI syntax.

1.6 Drools:

Drools is a Business Logic integration Platform (BLiP). It is written in Java. It is an open source project that is backed by JBoss and Red Hat, Inc. It extends and implements the Rete Pattern matching algorithm.

In layman's terms, Drools is a collection of tools that allow us to separate and reason over logic and data found within business processes. The two important keywords we need to notice are Logic and Data.

Drools is split into two main parts:

- Authoring
- Runtime

Authoring: Authoring process involves the creation of Rules files (.DRL files).

Runtime: It involves the creation of working memory and handling the activation.

1.7 Rule Engine:

Drools is Rule Engine or a Production Rule System that uses the rule-based approach to implement and Expert System. Expert Systems are knowledge-based systems that use knowledge representation to process acquired knowledge into a knowledge base that can be used for reasoning.

A Production Rule System is Turing complete with a focus on knowledge representation to express propositional and first-order logic in a concise, non-ambiguous and declarative manner.

The brain of a Production Rules System is an Inference Engine that can scale to a large number of rules and facts. The Inference Engine matches facts and data against Production Rules – also called Productions or just Rules – to infer conclusions which result in actions.

A Production Rule is a two-part structure that uses first-order logic for reasoning over knowledge representation. A business rule engine is a software system that executes one or more business rules in a runtime production environment.

Rule:

Rules are pieces of knowledge often expressed as, "When some conditions occur, then do some tasks.

The most important part of a Rule is its when part. If the when part is satisfied, the then part is triggered.

```
rule  <rule_name>
    <attribute> <value>
    when
      <conditions>
then
 <actions>
end
```

1.7.1 Advantages of a Rule Engine:

Declarative Programming

Rules make it easy to express solutions to difficult problems and get the solutions verified as well. Unlike codes, Rules are written in less complex language; Business Analysts can easily read and verify a set of rules.

Logic and Data Separation

The data resides in the Domain Objects and the business logic resides in the Rules. Depending upon the kind of project, this kind of separation can be very advantageous.

Speed and Scalability

The Rete OO algorithm on which Drools is written is already a proven algorithm. With the help of Drools, application becomes very scalable. If there are frequent change requests, one can add new rules without having to modify the existing rules.

Centralization of Knowledge

By using Rules, you create a repository of knowledge (a knowledge base) which is executable. It is a single point of truth for business policy. Ideally, Rules are so readable that they can also serve as documentation.

Tool Integration

Tools such as Eclipse provide ways to edit and manage rules and get immediate feedback, validation, and content assistance. Auditing and debugging tools are also available.

Chapter 2

2. LITERATURE SURVEY

2.1 A lightweight, fast, and free ESB

Of the WSO2 enhancements, the Web management console is one of the most useful. Although the underlying XML-based configuration files are not terribly difficult to understand, and clear examples are provided for most common EAI patterns, the console makes mistakes less likely.This is especially true in environments where the operators and administrators of the ESB are not developers, commonly the case in larger enterprises. Proxies, end points, and sequences can all be created and managed via the DHTML-based management console. Although configuring the ESB does require some knowledge of the underlying ESB principals, the task is significantly easier in this environment.

2.2 PROGRESS IN ESB:

An ESB is often the entry point along the road toward SOA adoption. A service bus makes it easy to begin sharing services among applications with little development effort and minimal impact on existing infrastructure. Organizations with broader requirements for an ESB should consider WSO2 for non-critical applications until a HA strategy is rolled out by the vendor. Development of the product is active and proceeding quickly; if progress continues at the current pace, a few more releases would make WSO2 worthy of consideration in most enterprise-level ESB deployments.

2.3 WSO2 Releases Synapse-based Open-Source

Open-source start-up WSO2 on Monday released an open-source enterprise service bus based on any Apache Synapse project Called WSO2 ESB, the server software is designed to integrate different applications by translating between different protocols and converting different XML formats. The product is based on Synapse, an open-source ESB done at the Apache Foundation with the participation of WSO2 employees. The company adds additional features on top of Synapse including a Web-based administration console and a registry and repository, said Paul Fremantle, WSO2 co-founder and its vice president of technical sales. There are several open source ESB product in the marketplace, such as MuleSource, as well as those from entrenched integration vendors, such as IBM. WSO2, which was started by IBM employees worked on Web services protocols, distinguishes itself from the competition by being very lightweight and designed for XML.

2.4 WSO2 ESB PERFORMANCE TESTING ROUND 1:

The first set of articles that measures and compares the performance of the WSO2 ESB against other leading implementations - both Open Source and proprietary. In order to test the scenarios, we used three separate systems. They were for load generation, ESB deployment and to host the service implementation backend. The hardware and software configurations and the OS level tuning parameters used are listed under the 'Notes' section below. Please note: in order to get high levels of concurrent connections it is essential to tune the TCP/IP stack... Conclusion The results show the initial performance benchmark results of the WSO2 ESB / Apache Synapse, and proves its scalability to handle thousands of concurrent connections. It also shows that the WSO2 ESB is able to perform more XPath based CBR routing decisions, as well as XSLT transformations than one of the leading proprietary ESBs, with the results being almost twice as better in some scenarios. In addition, the memory footprint of the WSO2 ESB is about half the size of the competitor, meaning that - for example - the WSO2 ESB could be co-located with an existing service client or server.

Chapter 3

Problem Formulation:

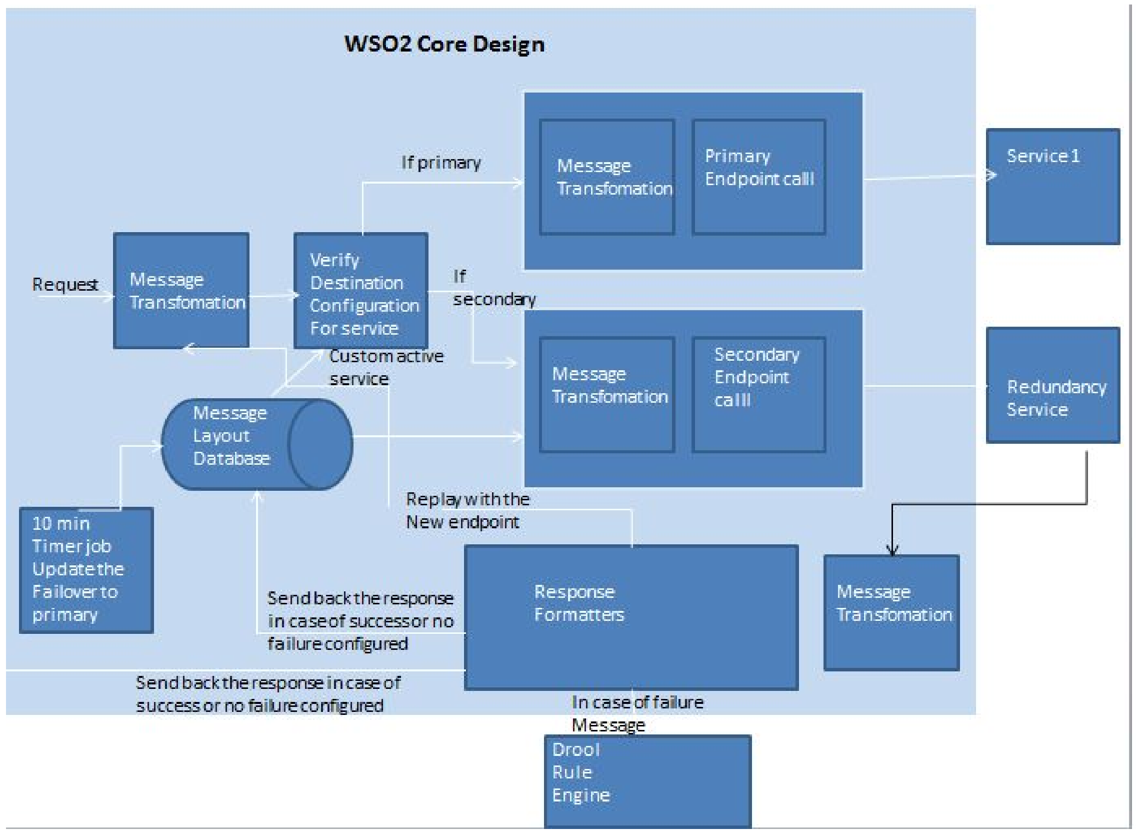
3.1 Problem statement

A middleware solution that enables interoperability among heterogeneous environments using service oriented architecture is needed.

Seamlessly redirect to other servers, when the request can’t be served by the requested server.

3.2 Existing approach

Generally database is used for communication between the distributed processes. When database is used to store messages to communicate between the processes, as soon as the message is received, a row insert and delete for each message is to be performed.



3.3 Disadvantages of Existing approach

When you try to scale it up, communicating thousands of messages per second, databases tend to fall over.

Chapter 4

Requirement Analysis and Specification

Hardware Specification

- Processor: Intel(R) Core(TM) i5 CPU
- CPU Clock:3.20 GHz
- RAM:8GB
- Hard Disk:4GB

Software Specification

- Operating System: Windows 7
- Language: Java
- Tools: Eclipse, Apache ActiveMQ, WSO2 ESB, DROOL

Chapter 5

PROPOSED METHOD

Proposed System

In this project, we integrate diverse systems using WSO2 and ActiveMQ. In ActiveMQ, the messages will be deleted very quickly and can do optimizations to avoid the overhead. And when the request can’t be served by the requested server, rules are generated by the rule engine drool to seamlessly failover to another server. Using WSO2 an effective solution for separating business logic-which needs to be changed and evolved more rapidly, from the infrastructure code-and bringing rules into the well-defined management and governance practices developed for SOA.It allows business rules to be encapsulated in more accessible forms, ensuring that rules are accurate and current reflections of the business needs. WSO2 provides proven performance and reliability. With WSO2, Drools develops a simple and straightforward way to integrate rules with their SOA implementations to provide a flexible, modular architecture. By adding the rule functionality of WSO2 BRS, rules become a first-class component of the complete WSO2 Carbon platform.

It is to make the failover seamlessly without manual intervention and increase the scope of failover conditions by adding Drools rules to route the request or failover from the front end components based on certain pre-configured rules. The project also stresses the need for establishing a middleware layer like ESB to facilitate failover and to have loose coupling between the frontend and backend components.

Chapter 6

IMPLEMENTATION:

The following products and the respective versions were used for the implementation:

- WSO2 ESB 4.8.1
- ActiveMQ 5.11.1
- Apache Ant 1.9.6
- Drools

6.1 Installation:

1.Open a command prompt (or a shell in Linux) and go to the <ESB_HOME>\bin directory. Then run the ant command as shown below to build the build.xml file.

```
C:\wso2\wso2esb\bin>ant
```

2.Run the ESB in the DEBUG mode.

6.2 Starting the ESB with a sample configuration:

To start the WSO2 ESB with a selected sample configuration

Open a command prompt (or a shell in Linux) and go to the <ESB_HOME>\bin directory.

The following sections describe how to run the product.

- Starting the server
- Accessing the Management Console
- Stopping the server

Execute one of the following commands, where <n> denotes the number assigned to the sample.

```
wso2esb-samples.bat -sn <n>
```

6.3 Starting the Axis2 server

For the ESB samples, a standalone Apache Axis2 Web services engine is used as the back-end server, which is bundled with the WSO2 ESB distribution by default.

Once each back-end service is deployed to the Axis2 server, you need to start the Axis2 server before executing the sample client.

To start the Axis2 server

Open a command prompt (or a shell in Linux) and go to the <ESB_HOME>/samples/axis2Server directory.

Execute the following command

```
axis2server.bat
```

This starts the Axis2 server with the HTTP transport listener on port 9000 and HTTPS on port 9002 respectively.

The URL appears next to “Mgt Console URL” in the start script log that is displayed in the command window. For example:

The URL should be in the following format: https://<Server Host>:9443/carbon

You can use this URL to access the Management Console on this computer from any other computer connected to the Internet or LAN.

To run the client,Navigate to the <PRODUCT_HOME>/samples/axis2Client directory and type the following command:

```
ant stockquote -Daddurl=http://localhost:9000/services/SimpleStockQuoteService
-Dtrpurl=http://localhost:8280 -Dsymbol=IBM -Dmode=quote
```

To stop the server, press Ctrl+C in the command window, or click the Shutdown/Restart link in the navigation pane in the Management Console. If you started the server in background mode in Linux, enter the following command instead:

```
sh <PRODUCT_HOME>/bin/wso2server.sh stop
```

6.4 Deploying sample back-end services

The sample back-end services come with a pre-configured Axis2 server. These sample services demonstrate in-only and in-out SOAP/REST or POX messaging over HTTP/HTTPS and JMS transports using WS-Addressing, and WS-Security. The samples handle binary content using MTOM and SwA.

Each back-end sample service can be found in a separate folder in the <ESB_HOME>/samples/axis2Server/src directory. You need to compile, build and deploy each back-end service to the Axis2 server.

To build and deploy a back-end service

Open a command prompt (or a shell in Linux) and go to the required sample folder in the <ESB_HOME>/samples/axis2Server/src directory.

Run ant from the selected sample directory.

For example, to build and deploy SimpleStockQuoteService, run the ant command from the <ESB_HOME>/samples/axis2Server/src/SimpleStockQuoteService directory

6.5 ActiveMQ installation:

1.Download the latest release

2.Extract ActiveMQ from the ZIP file into a directory.

3.The recommended method of building ActiveMQ is the following:

REM add "-Dmaven.test.skip=true

mvn clean install

where [activemq_install_dir] is the directory in which ActiveMQ was installed.

6.6 Integrating WSO2 with ActiveMQ:

1)Check if wso2 is running independently.

2)Check if ActiveMQ is running independently.

3)Copy activemq-core-5.3.0.jar, geronimo-jms_1.1_spec-1.1.1.jar and geronimo-j2ee-management_1.0_spec-1.0.jar from activemq lib folder to wso2 repository/component/lib folder.

4)Replace the axis2.xml file attached in repository/conf/axis2.

5)Replace the sampleMQ.xml file in deployment/server/synapse-configs/default/proxy-services.

6)Restart the wso2 instance .

Then,create queues in ActiveMQ as what described in SampleMQ.xml and get corresponding request and response messages.

Creating the Proxy Service:

- Create the proxy service in console
- Create the in sequence for the proxy service but didn't create the out sequence.
- Create the Request sequences.
- Create the out sequence for the proxy service now.
- Before creating the out sequence, check the format of the response messages that service sends.Several response messages will be sent.

Since, several messages will be received by the out sequence, the out sequence must aggregate them into one message and sent to the user, because after all the user send his coordinates to receive information.

1)Go to Home>Manage>Web Services>List>Service Dashboard>Proxy Service>Design Sequence>Dashboard>Proxy Service

2)Add a log mediator

3)We will now add an aggregate mediator as shown below. Core -> Aggregate. The purpose of adding the aggregate mediator is to aggregate the messages and compose 1 message. An aggregate mediator usually always follows an iterate mediator.

4)Click on the aggregate mediator and add a log mediator.

Finally, add a send mediator under the aggregate mediator.

In the send mediator we will not specify an endpoint, thereby sending the message back to the original sender.

We had completed the creation of the out sequence,then click Finish.

Installation of Drools:

1)In Eclipse, go to Help → Software Updates.

2)Click on "Add" button in the install window. A dialog appears as shown below. In Eclipse 3.4.1, click on "Add Sites" button in the second tab.

3)Select DROOLS and jBPM Package and click next. It will redirect to install details screen. Click next and accept licensing term. Click "Finish". DROOLS plugin will start installing into eclipse. After the installation restart the eclipse.

To create DROOLS Project in Eclipse:

1) Go to File -> New -> Other->Rule Resource
2) Then create a Drool Project.
3) Go to File -> New -> Other->Rule Resource
4) Add a sample HelloWorld rule file to the project

5) Add a java class for creating and executing the HelloWorld rules.
6) Click on create a new DROOLS 5 Runtime Button.
7) Click on “OK” button of DROOLS Run Time window.

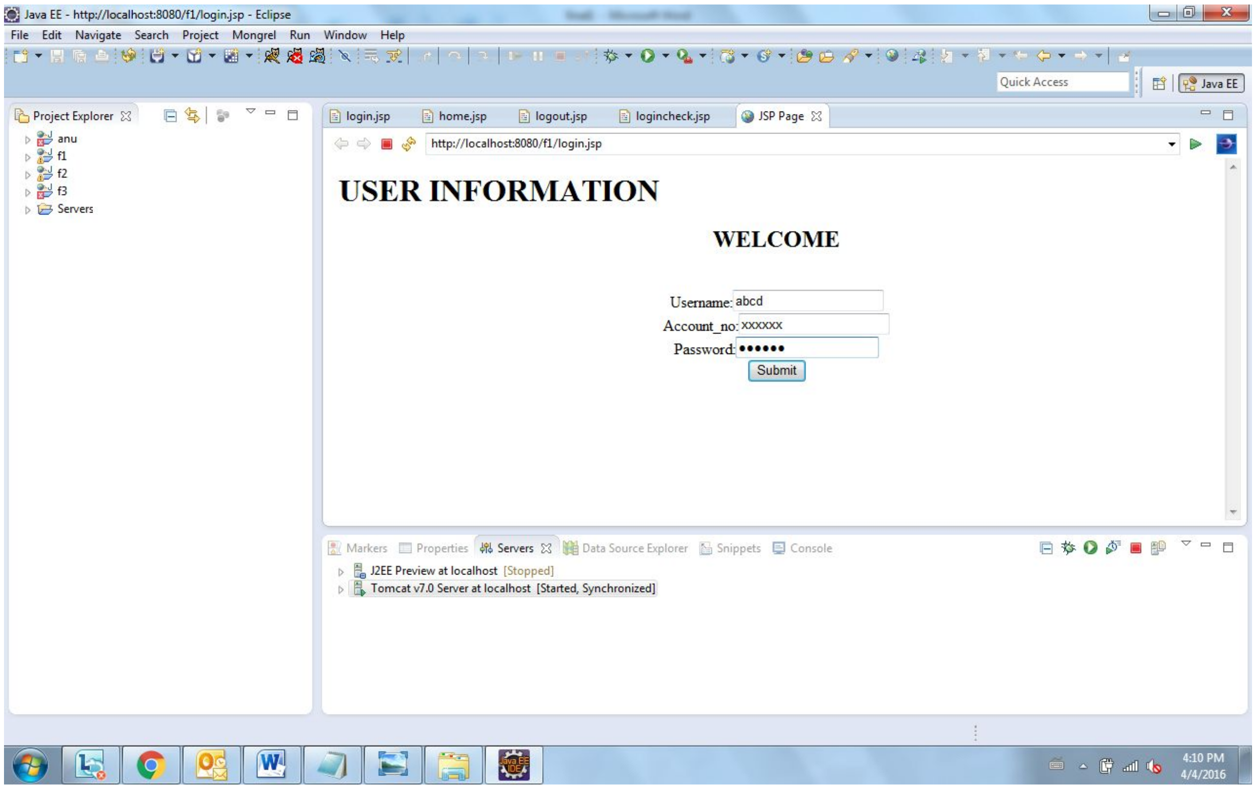
In the drool rules,

1) Create a drool files for the project.
2) To demonstrate that all rules are getting triggered from project rule files, use corresponding methods for the services.
3) More rules can be added in the drl files for the other services.
4) Call a external function from a drl file.

Chapter 7

Results and Disscussions:

We had created the frontend in which it gets the request from the user and then sent to ActiveMQ as XML message which is integarated with WSO2.



http://localhost:8080/f1/login.jsp

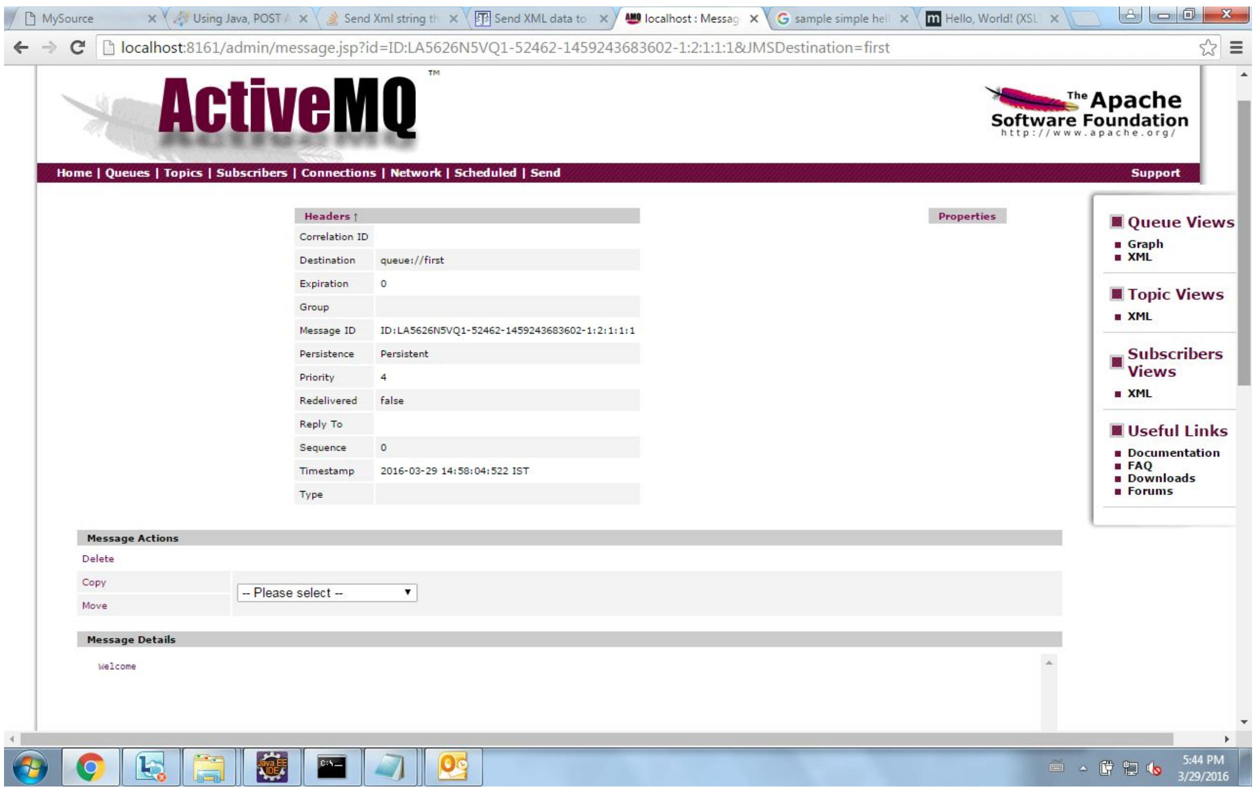
USER INFORMATION

WELCOME

Username: abcd
Account_no: xxxxxx
Password: ••••••
Submit

J2EE Preview at localhost [Stopped]
Tomcat v7.0 Server at localhost [Started, Synchronized]

/local_home/cole/Downloads/16.png



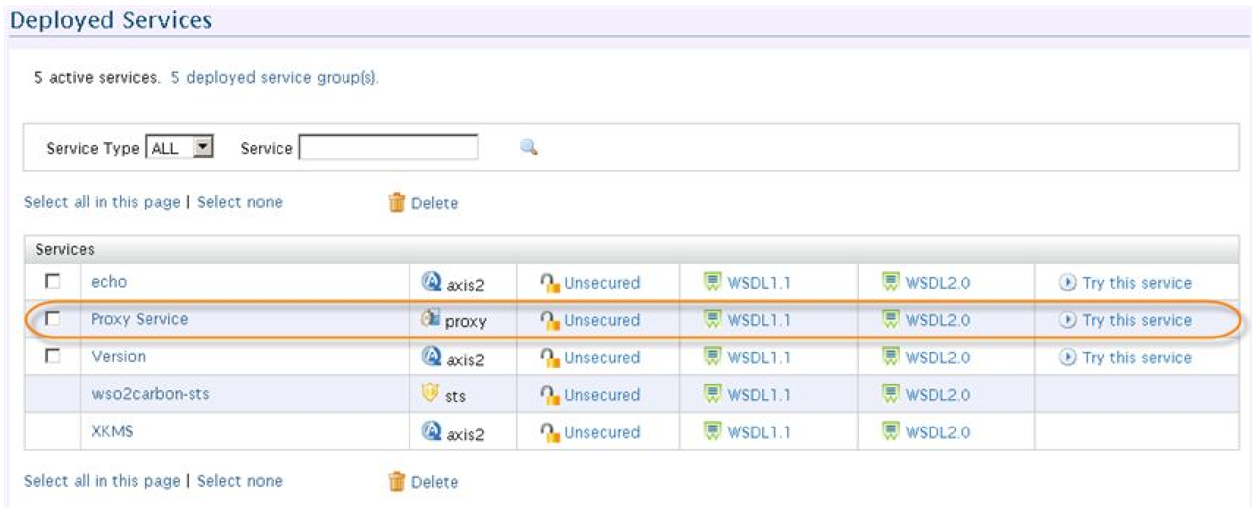
localhost:8161/admin/message.jsp?id=ID:LA5626N5VQ1-52462-1459243683602-1:2:1:1:1&JMSDestination=first

Headers ↑	
Correlation ID	
Destination	queue://first
Expiration	0
Group	
Message ID	ID:LA5626N5VQ1-52462-1459243683602-1:2:1:1:1
Persistence	Persistent
Priority	4
Redelivered	false
Reply To	
Sequence	0
Timestamp	2016-03-29 14:58:04:522 IST
Type	

Message Details

```
Welcome
```

We use ActiveMQ which acts as an interface.



Deployed Services

5 active services. 5 deployed service group(s).

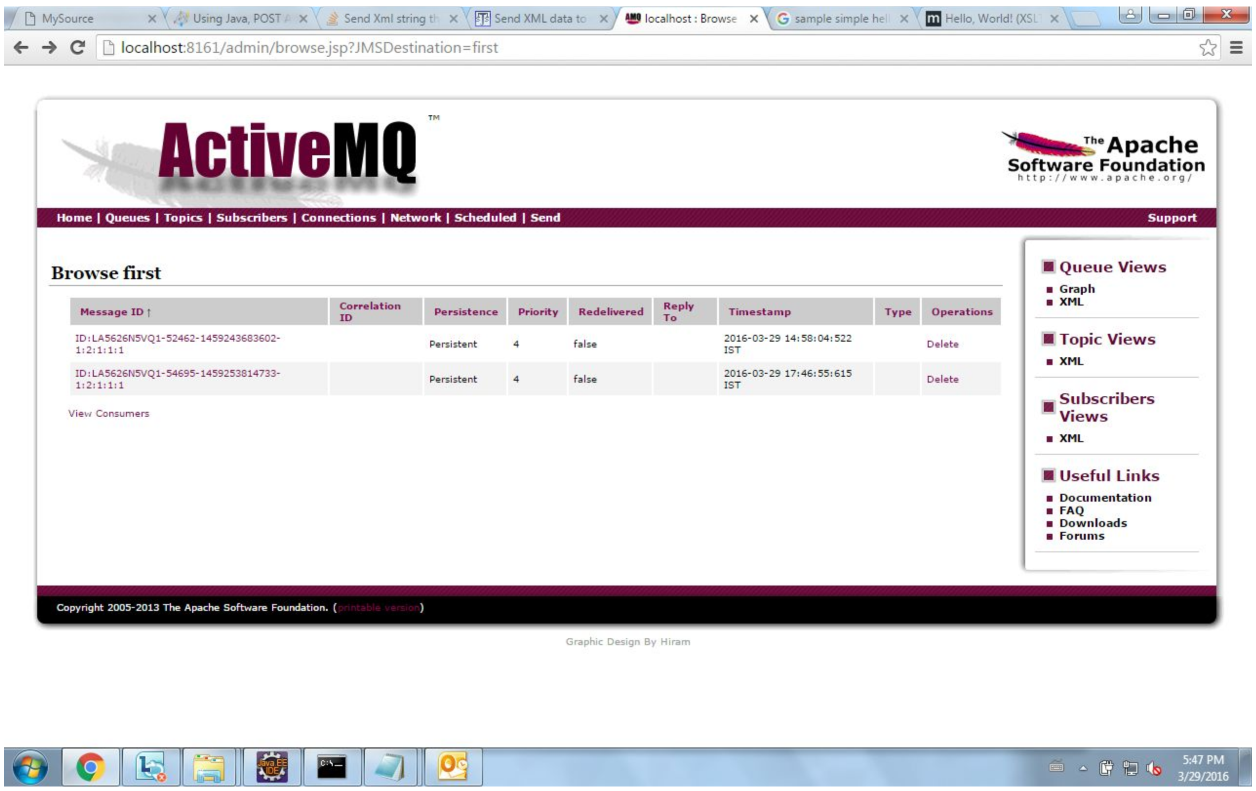
Service Type ALL Service

Select all in this page | Select none Delete

Services						
☐	echo	axis2	Unsecured	WSDL1.1	WSDL2.0	Try this service
☐	Proxy Service	proxy	Unsecured	WSDL1.1	WSDL2.0	Try this service
☐	Version	axis2	Unsecured	WSDL1.1	WSDL2.0	Try this service
	wso2carbon-sts	sts	Unsecured	WSDL1.1	WSDL2.0	
	XKMS	axis2	Unsecured	WSDL1.1	WSDL2.0	

Select all in this page | Select none Delete

We use wso2 to create proxy service and transfer messages



localhost:8161/admin/browse.jsp?JMSDestination=first

Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Browse first

Message ID ↑	Correlation ID	Persistence	Priority	Redelivered	Reply To	Timestamp	Type	Operations
ID:LA5626N5VQ1-52462-1459243683602-1:2:1:1:1		Persistent	4	false		2016-03-29 14:58:04:522 IST		Delete
ID:LA5626N5VQ1-54695-1459253814733-1:2:1:1:1		Persistent	4	false		2016-03-29 17:46:55:615 IST		Delete

View Consumers

Copyright 2005-2013 The Apache Software Foundation. (printable version)

Then we generated the rules for those servers in which errors were raised.

Chapter 8

Conclusion and FutureWork:

We integrated diverse systems using WSO2 and ActiveMQ. And when the request can’t be served by the requested server, rules are generated by the rule engine drool to seamlessly failover to another server.

And our future work would be applying the concept of service chaining. Service chaining is a popular usecase in WSO2 ESB. In lot of usecases, business functionalities defined in different services are exposed as a single service to the outside world.Since two or more services are aggregated using the ESB, a request to the ESB is served by the help of multiple services. Usually, the ESB has to call these services in a particular order to create the response. This kind of sequential service calling is being recognized as service chaining.

“Rules are becoming an increasingly popular part of the enterprise application toolbox, and a vital part of realizing the promises of agility and efficiency of service-oriented architectures,” said Dr. Sanjiva Weerawarana, founder and CEO of WSO2. “With the addition of the WSO2 Business Rules Server to our WSO2 Carbon platform, IT and business users can work even more effectively together to build and maintain competitive advantage in their industries.

Chapter 9

REFERENCES:

[1] Lean Weng YEOH, Ming Chun NG, Architecting C4I Systems, Second International Symposium On Engineering.Systems MIT, Cambridge, Massachusetts, June 15-17,2013.

[2] United States Department of Defense (USDoD) , Joint Doctrine for Command, Control, Communications&Computer (C4) Systems Support for Joint Operations,Joint Chiefs of Staff, 30 May 1995.

[3] [June 12, 2007] "Apache Synapse 1.0 and WSO2 ESB 1.0 Released." By Stefan Tilkov. From Infoq (June 12, 2007). "The Apache Software Foundation has released Apache Synapse 1.0; simultaneously, WSO2, the company behind Axis2, has released a commercial offering based on Synapse, called WSO2 ESB 1.0. As opposed to Axis2, Synapse is not programming environment.

[4] [June 11, 2007] "WSO2 Releases Synapse-based Open-Source ESB." By Martin LaMonica. From CNET News.com Blog (June 11, 2007). "Open-source start-up WSO2 on Monday released an open-source enterprise service bus based on any Apache Synapse project.

[5] Mike Somekh, Mark Foster, Rastislav Kannocz "GlassFisg ESB High Availability and Clustring, Sun Micro Systems", White paper, Chapter 1 Introduction, Dec 2013.https://www.sun.com/offers/docs/glassfish_esb_ha_wp.pdf

[6] Luis Garces-Erice, “Building an Enterprise Service Bus for Real-Time SOA: A Messaging Middleware Stack”, 33rd Annual IEEE International Computer Software and Applications Conference,pp. 79-84, 2013, doi: 10.1109/COMPSAC.2013.119.

[7] Robert Woolley, “Enterprise Service Bus (ESB) Product Evaluation Comparisons”, Utah Department of Technology Services, 34th Annual IEEE International Computer Software and Applications Oct 18, 2012.

[8] Tobias Kruessmann, Arne koschel, Martin Murphy, Adrian Trenaman, Irina Astrova, “High Availability: Evaluating Open Source Enterprise Services Buses” Proceedings of the ITI 2009 31th Int. Conf. on information Technology Interface, June 22-25, 2013, Cavtat, Croatia.

[9] Tijs Rademakers and Jos Dirksen, “Open Source ESBs in Action”, Manning Publications, ISBN 1933988215,Sep 20,2012.