



## JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY



WIPRO LIMITED

PROJECT REPORT

TITLE: AUTO LOG SEARCH

Training Period: 13<sup>th</sup> June, 2016 to 29th July, 2016

Project Manager: Gaurav Gupta

Designation: Program Manager

Company Organization's Address: Plot No 480-481, Udyog Vihar, Phase 3, Gurgaon, Haryana 122016

Name: Surbhi Jain

Department: Computer Science & Engineering

Enrollment Number: 13103633

## CONTENT

1. Acknowledgment
2. Declaration
3. Introduction
4. About Wipro Limited
5. About Project: Log Search
6. Project Workflow
7. Business Benefits
8. Result and Conclusion
9. References and Appendix

## ACKNOWLEDGEMENT

It has indeed been a great privilege for me to have **Mr. Gaurav Gupta**, Program Manager, Wipro Limited, as my mentor for this project. His superb guidance and constant encouragement are the motive force behind this project work. I take this opportunity to express my utmost gratitude to him. I am also indebted to him for his timely and valuable advice.

I am highly grateful to Ms. Mitali Bhattacharya and Ms. Medha for providing necessarily facilities and information and encouraging me during the course of work.

Surbhi Jain  
(13013633)  
Bachelor of Technology  
Department of Computer Science & Engineering  
Jaypee Institute of Information Technology

## DECLARATION

I (Surbhi Jain, 13103633) hereby declare that this project work entitled **AUTO LOG SEARCH** submitted to the **Department of Computer Science & Engineering, Jaypee Institute of Information Technology** during the academic year 2016-2017, is a record of original work done by me under the guidance of **Mr. Gaurav Gupta**, Program Manager, Wipro Limited, Gurgaon.

This project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering.

The information and data given in the report is authentic to the best of my knowledge and have not been submitted to any other university or Institute for the award of any degree.

Date:

Surbhi Jain

Place: Gurgaon

(13103633)

## INTRODUCTION

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).

The ESB approach to integration can provide the underlying integrated solution as a service based, loosely coupled, highly integrated and widely available network that extends beyond the boundaries of a traditional hub and spoke EAI broker. An ESB has the following characteristics:

- Highly Adaptable
- Distributed
- Ability to selectively deploy Integration components
- Secure and Reliable
- Ability to orchestrate processes
- Monitoring

The Telecommunication Industry forms a business case for using Enterprise Service Bus as an EAI broker particularly for an OSS/BSS solution.

In today's world of emerging IT technologies, business integration has become the top priority and concern for any company worth its salt. In the course of business integration, attention is focused on achieving the integration quickly but at the same time expecting an early return on investment. In the last two decades, the IT Industry has seen the life cycle of integration starting from traditional MOMs and then graduating to vendor driven EAIs along with various other Enterprise integration technologies. The latest and one of the most promising architecture to have emerged during this period is Enterprise Service Bus. An ESB oriented solution addresses lots of key challenges in the integration space and is out to establish itself as a key trend.

## ABOUT WIPRO LIMITED

The third-largest company in India, Bangalore-based Wipro Limited is an ever-growing and ever-diversifying global company that manufactures and sells products and services ranging from cooking oil and soaps to healthcare instruments and information technology (IT) consulting.

Wipro Technologies, the global IT business of Wipro Limited is a leading Information Technology, Consulting and Outsourcing company that delivers solutions to enable its clients do business better. Wipro champions optimized utilization of natural resources, capital and talent. Wipro Technologies delivers winning business outcomes through its deep industry experience and a 360° view of “Business through Technology” helping clients create successful and adaptive business. A company recognized globally for its comprehensive portfolio of services, a practitioner’s approach to delivering innovation and an organization wide commitment to sustainability, Wipro has 135,000 employees and clients across 54 countries.

The Aircel group is a joint venture between Maxis Communications Berhad of Malaysia and Sindya Securities & Investments Private Limited. Aircel was looking at setting up its National Technology Centre (NTC), a green field data center over 50,000 sq ft in Gurgaon. Aircel explored the technologies and best practices available that would help in this objective. It was Wipro’s Eco Energy practice that impressed the Aircel team the most. They chose Wipro because Wipro approaches the green initiatives in a conscious manner and not just as a fad. Their office is 100% green and they have green campuses in Bangalore. Wipro and Aircel drew up the blueprint for the data center. The data center uses many eco-friendly initiatives such as motion-sensor lighting, use of LED lights and CFLs, adaptive cooling, geothermal heat exchange and earth air tunnel for reducing load on chillers, solar water heating, and rain water harvesting.

## ABOUT LOG SEARCH

**Logs** are a collection of technical and statistical data about calls or requests made by the customer to the company regarding their various services. It does not encompass phone tapping or call recording. Such logs for a particular number called as MSISDN have to be downloaded using the shell script upon traversing to the path where the scripts are whenever needed. Our application aims at downloading these logs at the server and sending the needed logs to the client end of the application and showing as output only the required logs. Thus, easing the whole process of traversing, calling scripts, downloading and then finding the data in all the downloaded logs as they may contain logs for other MSISDN's as well.

Clustering is a technique for grouping similar instances or components to ease load to different components so that an individual request can be routed to a component that holds the specific data needed to process the request. This approach results in an increased performance as the load is redirected to specific instances in order to ameliorate the performance degradation. ESB is both horizontally and vertically clustered. There are 3 types of logs- BackLog, Rotation and Current.

- BackLog contains the old logs, i.e. logs till a day before the current date.
- Rotation has present day's logs till 10 minutes prior to current timestamp.
- Current logs contain logs of the current timestamp.

There are 6 clusters and each of these have a script for these three types of logs. The names are of the form, Cluster3BackLog.sh, where 3 is the cluster number. These scripts take the MSISDN as command line argument and once executed, take the date, month and year as input in case of BackLog and day and month in case of Rotation and download the required logs in .log.gz extension for BackLog and .log for Rotation.

Our application, upon user verification through a valid username and password, selection of ESB service, displays a page where MSISDN, Cluster Number, Date and Log type can be entered. These details upon validation are sent to

server, which downloads the required logs and transfers to the client, which in turn searches for the MSISDN in the received logs and copied them to an output file. Once the program is complete the output file is opened. Appropriate error message are displayed in case the values entered as not valid or the logs were not found, for example, if the user selected Backlog but selected the present date, error message would be displayed. Thus, the ESB team members would have to no longer go through the old time consuming process. The client application would do the work for them.

### **Software Used:**

#### **PYTHON 2.7**

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. As we require interaction with a Unix box and run a python script, that's why we chose python as our language to build the application as it provides a better interaction with Unix based systems.

#### **PY2EXE**

For the users not having python environment downloaded, we needed to make an executable file which would encapsulate the python requirement and wrap it in a folder. Hence we used this application to convert our Python program into an Executable file which would support the pre-requirements.

### **Concepts Used:**

- **Socket Programming:** There was a need to send data from server side to multiple clients with different IP addresses. Hence we used socket programming supported in python to help our case. Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents. The *socket* library provides specific classes for handling the common transports as well as a generic interface for handling the rest.
- **Threading:** Various clients need to download data from server using the same program; Hence to support this we used threads. Running several threads is similar to running several different programs

concurrently as multiple threads within a process share the same data space with the main thread and can therefore share information or communicate with each other more easily than if they were separate processes. Thus multiple clients have their own thread to communicate with server.

- **Data File Handling:** The logs are downloaded at server end using python script. Our motive is to read the multiple downloaded logs from the server end and send it to the client side where it will write these logs into a single new file and concatenate all the data as to make it easier for the client to find the logs. Using file read and write functions of python we performed this task and to open zipped files and unzip it, we used **GZIP library** supported by Python. The Rotation log files are large in size, due to which it exceeds the maximum limit of data a compressed zip file can hold. Hence, we receive the data for log files in **text file** rather than gzip file.
- **Regular Expression (Regex):** “re” library supported by python is used to match the required logs within the single concatenated log file at the server end. We are required to fetch the part of logs only where the MSISDN is present. Hence we use regular expression to match the pattern and write these blocks into a new file.
- **GUI:** Tkinter Library of python was used to support GUI for the application. Buttons and fields are given to the client side to make them enter data in a user friendly and easier way.



## PROJECT WORKFLOW

A running server in UNIX box waits for client application to connect and send the required data.

The client application, an executable file, when launched asks the user to login. Upon verification, a new screen displays asking user to choose between integrated systems (2 systems, UDT and ESB are integrated as per scope).

Basis selection a new window would appear asking the user to enter the MSISDN, select the log type (BackLog, Rotation or Current), and select the date and the cluster number. Once the user submits the input, connection is made with the server and the data is validated for the following:

- MSISDN should be of ten digits or twelve, taking in consideration India code 91. But only the last 12 digits are sent to the server.
- None of the field entered or selected are empty.
- Incase user selected Rotation then date should be of the present day.
- In case of BackLog, date selected shouldn't be present or future day.
- The date should not be in future.

If no error is found, appropriate flag message is sent to the server and then the data is sent. In case the validation failed, flag is sent to server and it closes the connection with the user.

The server executes the shell script using the information received. For instance, if the user entered the following details:

MSISDN: 8801011429

Log Type: BackLog

Date: 04 June 2016

Cluster: 2

Then at the server end, the command executed will be: “ sh Cluster2BackLog.sh 8801011429 ”

When the script runs it takes the day, month and year as input and then downloads the logs if available.

Each cluster has a fixed prefix for the logs that are download. These are:

Cluster 1: Aircel\_MS

Cluster 4: AIRCEL\_ESB\_MS

Cluster 2: AircelESB\_MS

Cluster 5: ALSB\_MS

Cluster 3: ESB\_MS

Cluster 6: OSB\_MS

These downloaded logs are sent to the client in zipped form. In order to find the required logs from the ones downloaded, the application first searched for the MSISDN in the logs. Using “#####” as delimiter, it checks for the first occurrence of it in line above the line in which MSISDN was found and then in the lines below it. The two positions are saved, and lines in between are copied into a new text file. The file would be empty if no logs were found on server end. Hence, if the file is not empty, the file is opened at the end of completion and window closes terminating the connection with server and the window asking for user to select UDT or ESB is displayed again. At the server side, the names of files matching this prefix and date are transferred to a text file. And from that file, names are read one by one, and logs are sent to client. There can be multiple logs downloaded, but at the client end all of these are concatenated into a single file.

Several libraries were required for the execution of the python script, such as-

- os – for execution of UNIX commands and opening of text file.
- threading – for multithreading at server end
- gzip - in order to open zip files
- socket - for interprocess communication between client and server
- Tkinter – for GUI
- datetime- for date validation
- sys – for system-specific functionalities

The screenshots of the program's output can be seen:

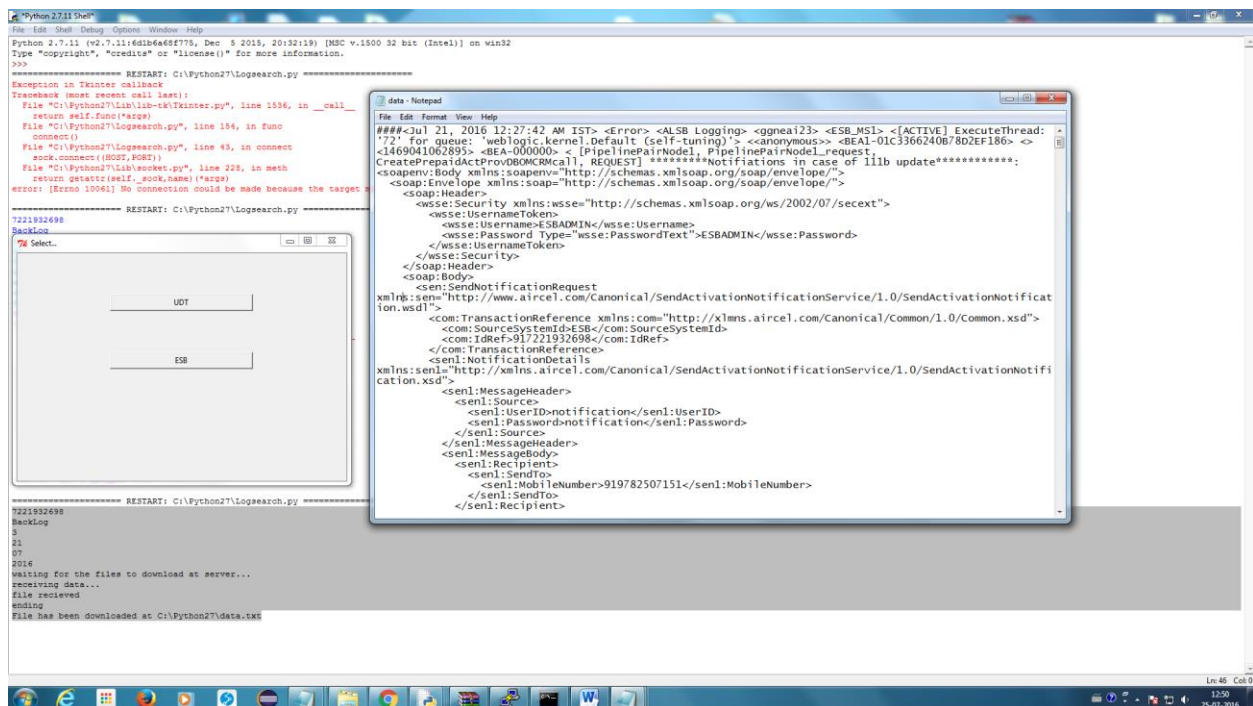
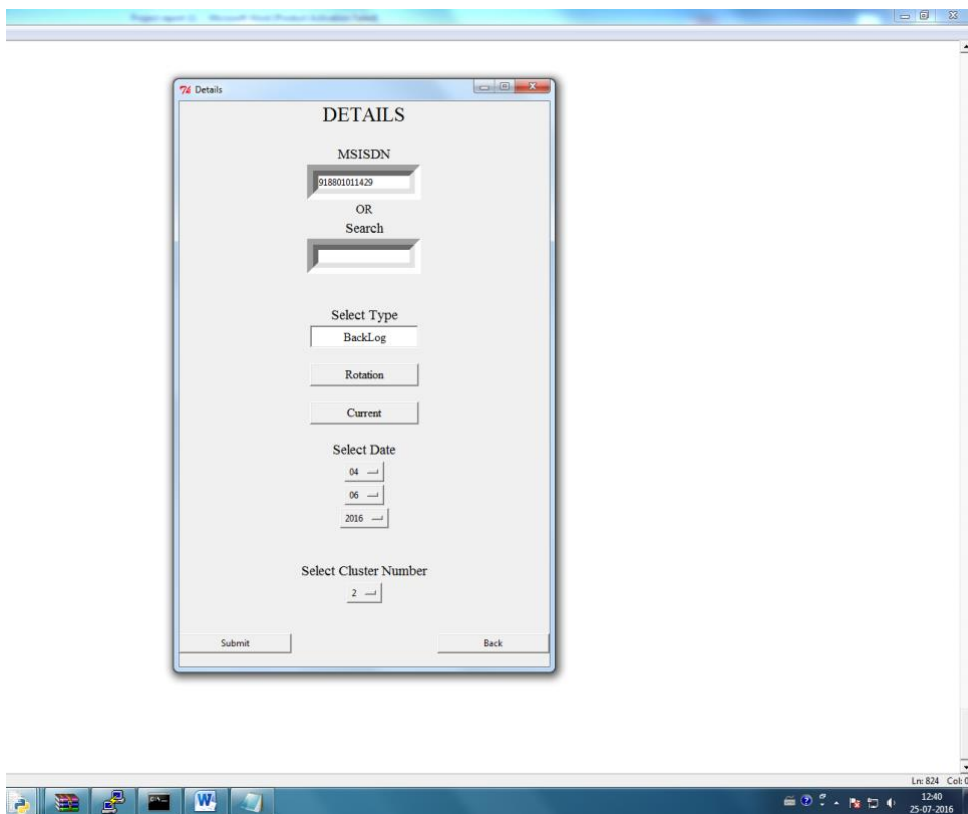
```
#Bind socket to local host and port
try:
    sock.bind((HOST, PORT))
except socket.error as msg:
    print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
    sys.exit()

#Start listening on socket
sock.listen(10)
print 'now listening , Waiting for connections'

#Function for handling connections. This will be used to create threads
"serverlog.py" 121 lines, 3506 characters
bash-3.2$ python serverlog.py
now listening , Waiting for connections
Connected with
msisdn : 7221932698
cluster 3
BackLog
date :21
month :07
year :2016
Enter Day
21
Enter Month
07
Enter Year
2016
wait for a moment...
-----
Downloading log files in current folder...
-----
ESB_MS1_2016_07_21_1 100% |*****| 319 KB 00:00
ESB_MS6_2016_07_21_0 100% |*****| 278 KB 00:00
ESB_MS6_2016_07_21_1 100% |*****| 296 KB 00:00
ESB_MS7_2016_07_21_0 100% |*****|
['ESB_MS1_2016_07_21_12_34.log.gz', 'ESB_MS6_2016_07_21_06_03.log.gz', 'ESB_MS6_
done sending file 1
done sending file 2
done sending file 3
done sending file 4
goodbye
```

```
waiting for connections !!

Connected with
msisdn : 8801011429
cluster 2
BackLog
date :04
month :06
year :2016
Enter Day
04
Enter Month
06
Enter Year
2016
wait for a moment...
█
```



## BUSINESS BENEFITS

As the team members used to manually run the script to download the logs and had to find all the required logs manually from those, open them and find the required part from all the downloaded logs which didn't have the required information necessarily, the task was time consuming and a wastage of man power; our Auto Log Search software does all of the work by itself thus saving a lot of time of the employees. They are provided with all the necessary logs in a single file without having to search each log manually for the same. Our software makes the ESB team more efficient in handling the requests.

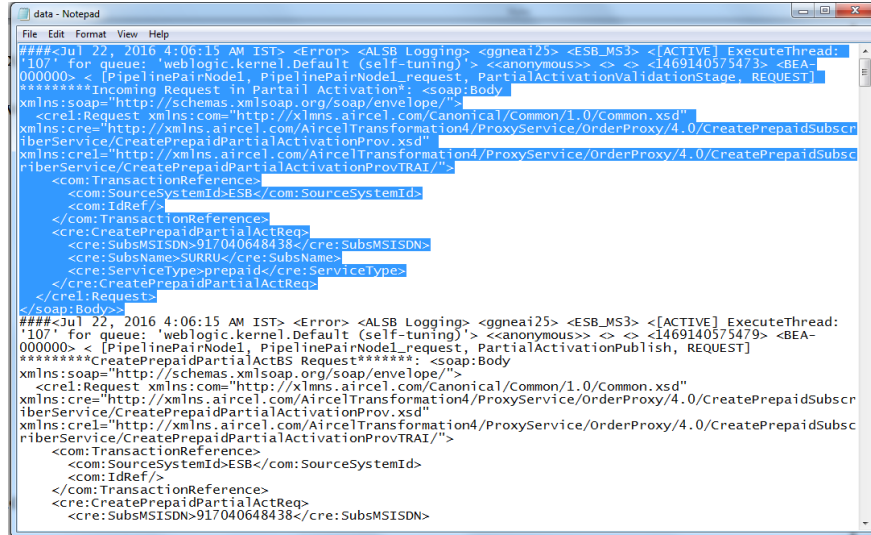
The software right now offers services for ESB and UDT teams but it can easily be expanded for all the other teams according to the type of logs that they require; hence it has the capability to save huge amount of time for these teams and make them more efficient and responsive in their work for respected departments.

The same algorithms can be applied for any other types of files containing a block of data with some keywords which is required by other teams and thus our software can be a base line for other needs as well.

Thus our software is a promising one with ease of use and many benefits for future also.

## RESULTS AND CONCLUSION

We were able to make an executable multi-client server based python application, using Py2exe software which would take a number and its details and would fetch the generated logs and output the file containing the required part of the logs to the clients.



```
data - Notepad
File Edit Format View Help

###Jul 22, 2016 4:06:15 AM IST> <Error> <ALSB Logging> <ggneai25> <ESB_MS3> <[ACTIVE] ExecuteThread:
'107' for queue: 'weblogic.kernel.Default (self-tuning)'> <anonymous>> <?> <1469140575479> <BEA-
000000> < [PipelinePairNode1, PipelinePairNode1_request, PartialActivationValidationStage, REQUEST]
*****Incoming Request in Partail Activation*: <soap:Body
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <crel:Request xmlns:com="http://xmlns.aircel.com/Canonical/Common/1.0/Common.xsd"
  xmlns:cre="http://xmlns.aircel.com/AircelTransformation4/ProxyService/OrderProxy/4.0/CreatePrepaidSubsc
  iberService/CreatePrepaidPartialActivationProv.xsd"
  xmlns:crel="http://xmlns.aircel.com/AircelTransformation4/ProxyService/OrderProxy/4.0/CreatePrepaidSubc
  riberService/CreatePrepaidPartialActivationProvTRAI/">
    <com:TransactionReference>
      <com:SourceSystemId>ESB</com:SourceSystemId>
      <com:IdRef/>
    </com:TransactionReference>
    <cre:CreatePrepaidPartialActReq>
      <cre:SubsMSISDN>917040648438</cre:SubsMSISDN>
      <cre:SubsName>SURRUX</cre:SubsName>
      <cre:ServiceType>prepaid</cre:ServiceType>
    </cre:CreatePrepaidPartialActReq>
  </crel:Request>
</soap:Body>
*****CreatePrepaidPartialActBS Request*****: <soap:Body
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <crel:Request xmlns:com="http://xmlns.aircel.com/Canonical/Common/1.0/Common.xsd"
  xmlns:cre="http://xmlns.aircel.com/AircelTransformation4/ProxyService/OrderProxy/4.0/CreatePrepaidSubsc
  iberService/CreatePrepaidPartialActivationProv.xsd"
  xmlns:crel="http://xmlns.aircel.com/AircelTransformation4/ProxyService/OrderProxy/4.0/CreatePrepaidSubc
  riberService/CreatePrepaidPartialActivationProvTRAI/">
    <com:TransactionReference>
      <com:SourceSystemId>ESB</com:SourceSystemId>
      <com:IdRef/>
    </com:TransactionReference>
    <cre:CreatePrepaidPartialActReq>
      <cre:SubsMSISDN>917040648438</cre:SubsMSISDN>
```

## REFERENCES

- [www.stackoverflow.com](http://www.stackoverflow.com)
- [www.tutorialspoint.com](http://www.tutorialspoint.com)
- [www.python.org](http://www.python.org)

## APPENDIX

ESB	Enterprise Service Bus
UDT	User Data Transformation
MSISDN	Mobile Station International Subscriber Directory Number
EAI	Enterprise Application Integration
OSS	Operation Support System
BSS	Business Support System
MOM	Message Oriented Middleware
HLR/IN	Home Location Register / Intelligent Network