IMPLEMENTATION OF RECORDING LOG EVENTS IN WSO2 ESB COMMUNICATION

Stefan Pitulić[1], Slaviša Ilić[2], Siniša Ilić[1], Vladimir Veljović[3]
[1]*Faculty of Technical Sciences, K. Mitrovica University of Priština, Serbia*
[2]*Faculty of Informatics and Computing University Singidunum, Serbia*
[3]*Faculty of Technical Sciences, Čačak University of Kragujevac, Serbia*

Abstract
In this paper the implementation of recording the log of communication events to the external database servers from WSO2 - open source Enterprise Service is presented. The WSO2 ESB is used to connect different Web services deployed on different endpoints. Also, the analysis of data from communication events stored in the external databases is demonstrated using the built-in ESB dashboards.

Keywords: WSO2 ESB communication tracking, ESB logging to an external database, ESB Dashboard

INTRODUCTION

Almost all companies today rely own business on the software applications they use. A typical scenario is that an enterprise runs hundreds or thousands of applications, which could be custom built, acquired from a third party or parts of legacy systems [1]. One of the main tasks then is to make these software applications and services work together to produce unified business functionality.

The task of plumbing different software applications, services, and systems, and forming new software solutions out of that is known as enterprise integration. The software application that is designed to enable that task is known as the Enterprise Service Bus (ESB). An ESB enables diverse applications, services, and systems to talk to each other, interact, and transact. It acts as the main messaging backbone in any Service Oriented Architecture (SOA); it's lightweight, built on top of open standards such as web services standards, and supports commonly used patterns in enterprise integration known as Enterprise Integration Patterns [2]

An enterprise service bus (ESB) is a software architecture construct that enables communication among various applications. Instead of having to make each of applications communicate directly with each other in all their various formats, each application simply communicates with the ESB, which handles transforming and routing the messages to their appropriate destinations [3].

Different services may be using different data formats and communication protocols. Physical locations of services can change arbitrarily. An ESB can be used to loosen these couplings between different services and service consumers [3].

In a such environment it is very important to keep a log of data exchange events between applications. A properly configured logging system is vital for identifying errors, security threats, and usage patterns [4]. The structured log data are usually stored in a reliable database, from where the functioning of the ESB can be analysed. In order to ensure business contingency of IT services in a production environment a redundancy of log database is usually implemented. Such a database environment must be reliable and with help of reporting and analytic tools capable to produce different usage reports and analyses.

In this paper we described the re-configuration of default settings of WSO2 – the open-source ESB in order to enable standard logging configuration architecture in a production environment. In more details we described implementation of logging events and data using the external database and implementation of analyses of data exchange events and usage of Web Services APIs.

DEFAULT LOGGING SETTINGS IN THE WSO2 ESB

In the WSO2 ESB server, which needs to run side by side with the WSO2 ESB Analytics server, tracking can be enabled for execution of specific services such as REST and SOAP APIs and Proxy services, connecting to different EndPoints where Web Services are deployed.

There are several ways to log events in WSO2 ESB. WSO2 ESB is a Java application underneath, and as it is a typical in this type of application, it uses *log4j* for logging purposes. It means that one can manually edit the configuration file that can be found in <ESB_HOME>/repository/conf/log4j.properties to suit the architectural needs.

On the other hand, this configuration can be changed also through the management console. The changes are persisted in the registry so that they are available after restarting the server and overwrite the configuration in the log4j.properties. The logging configuration settings can be found in the Configure/Logging window of the Management Console.

By default, WSO2 ESB store logs on the OS filesystem and H2 database.

WSO2 log database

H2 database is default database for all WSO2 ESB products. According to the recommendation of WSO2 developers, the embedded H2 database is NOT recommended in enterprise testing and production environments. It has lower performance, clustering limitations, and can cause file corruption failures [3]. One more disadvantage of using H2 is that since it stores the data in the memory, it is volatile. H2 uses the memory much more than others databases especially when the size of data increases [5].

WSO2 ESB uses three H2 databases to store: configuration (regdb), metrics (wso2 metrics), and log analytics (wso2_analytics, wso2_analytics_processed).

By default, in database regdb the following properties are set: configuration paths, user accounts to access WSO2 ESB Management Console, user roles in WSO2 ESB registry, history of configuration changes, etc. In the databases wso2_analytics and wso2_analytics_processed the encrypted data needed for analysis are stored, such as: number of connection requests to some EndPoint, number of usage of some API, statuses of requests, etc. In the wso2 metrics the following data are stored: server load, the memory used, the load speed of some classes, average processor load, etc.

The detailed configuration of database functioning can be set in the following configuration XML files: registry.xml (general configuration of datasources), master-datasources.xml (the main database configuration), metrics.xml (general configuration of metrics), metrics-datasources.xml (configuration of database for metric tables), analytics-config.xml (the analytics dataservice configuration) and analytics-datasources.xml (configuration of database for analytics).

WSO2 log files

The log files are typically stored in the <ESB_HOME>/repository/logs/ folder. The following types of log files can be found by default:

- **http_access_management_console.log:** the basic information of each HTTP access to WSO2 ESB.
- **patches.log:** the log information when any patches are installed.
- **tm.out:** Atomikos transaction logs.
- **wso2carbon.log:** This is the main log file of WSO2 ESB, and this is the file one can see when he/she accesses or logs to the management console. This is the log that will be checked most of the times.
- **wso2-esb-errors.log:** all warning and error messages from wso2carbon.log
- **wso2-esb-service.log:** specific log from the proxy services deployed in WSO2 ESB. This is a way to extract the service logging entries from the general log (wso2carbon.log).
- **wso2-esb-trace.log:** more detailed information about the steps performed in designed sequences, mediators, or services when tracing has been enabled on them.

By default, all these files are configured as daily rolling log files. These files can be found with the date attached to the core names of the files [6].

A user does not have to open log files in order to read the logs. They can be accessed by Management Console. In this case, the log shown in the console is the carbon_memory logger, that is by default set in log4j.properties file, and can be changed by a user.

CHANGING THE DEFAULT LOG WSO2 CONFIGURATION

In a production environment, the typical architecture for keeping the logs of data exchange is shown in Fig. 1.

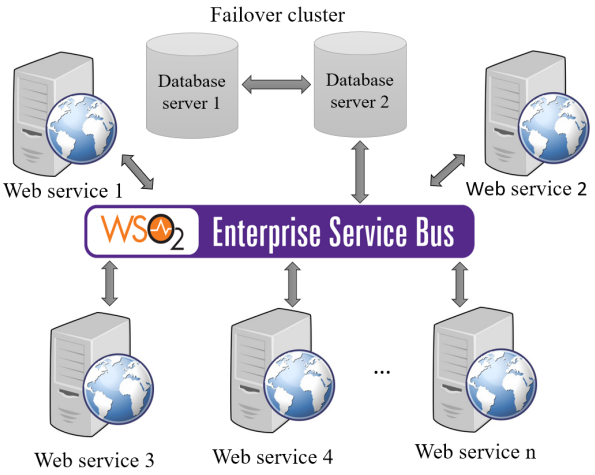

Fig. *1 Architecture of ESB in a production environment*

In order to keep track of all logs, a redundant database cluster (to support running of one database in the event of failure of another) should be created. In this experiment we decided to use and configure MySQL database in a master-slave replication mode.

Replacement of WSO2 default database

In order to redirect WSO2 ESB to send log data to MySQL database (instead to send it to the H2 DB), the configuration file regdb should be adjusted. The tags related to connection parameters (url, driverClassName, username, password) should be set to the MySQL jdbc values. The content of the adjusted master-datasources.xml file where the settings of regdb database can be set is shown in Fig. 2.

In a similar way the settings should be changed in other configuration files (analytics and metrics). Before starting the log collection to the new databases, a set of tables in databases should be created by running the predefined SQL query script. The WSO2 ESB has those scripts with SQL dialects for most of modern databases that can be found in <ESB_HOME>\dbscripts. After creation of new tables, the WSO2 ESB server should be started with the switch -Dsetup from a terminal in order to ESB copy data from default H2 databases to the new (MySQL) databases. From this moment ESB will send log data to the new MySQL databases.

```
<datasources-configuration
    xmlns:svns="http://org.wso2.securevault/configuration">
    <providers>
        <provider>
            org.wso2.carbon.ndatasource.rdbms.RDBMSDataSourceReader
        </provider>
    </providers>
    <datasources>
        <datasource>
            <name>WSO2_CARBON_DB</name>
            <description>
                The datasource used for registry and user manager
            </description>
            <jndiConfig>
                <name>jdbc/WSO2CarbonDB</name>
            </jndiConfig>
            <definition type="RDBMS">
                <configuration>
                    <url>jdbc:mysql://localhost:3306/regdb</url>
                    <username>regadmin</username>
                    <password>regadmin</password>
                    <driverClassName>
                        com.mysql.jdbc.Driver
                    </driverClassName>
                    <maxActive>80</maxActive>
                    <maxWait>60000</maxWait>
                    <minIdle>5</minIdle>
                    <testOnBorrow>true</testOnBorrow>
                    <validationQuery>SELECT 1</validationQuery>
                    <validationInterval>30000</validationInterval>
                    <defaultAutoCommit>false</defaultAutoCommit>
                </configuration>
            </definition>
        </datasource>
    </datasources>
</datasources-configuration>
```

Fig. *2 The content of adjusted master-datasources file*

The collection of log data to the analytics database in MySQL server is shown in Fig. 3.

In order to obtain the desired architecture of redundant database servers presented in Fig. 1. the MySQL master-slave replication configuration is created [7] by changing the default settings in mysqld.cnf files of both: master and slave database servers.

Configuring appender to send desired log data to the database

In general, WSO2 ESB stores complete log of all events in the files. Luckily, there is a tool called appender that can "catch" the content of a log and transfer it in a separate file or to the database.

Thus, by default, for data analysis purposes, the data from various events are sent to the H2 database in encrypted format. These data can be analysed in WSO2 Analytics server where are firstly decrypted and presented in the graphical format.

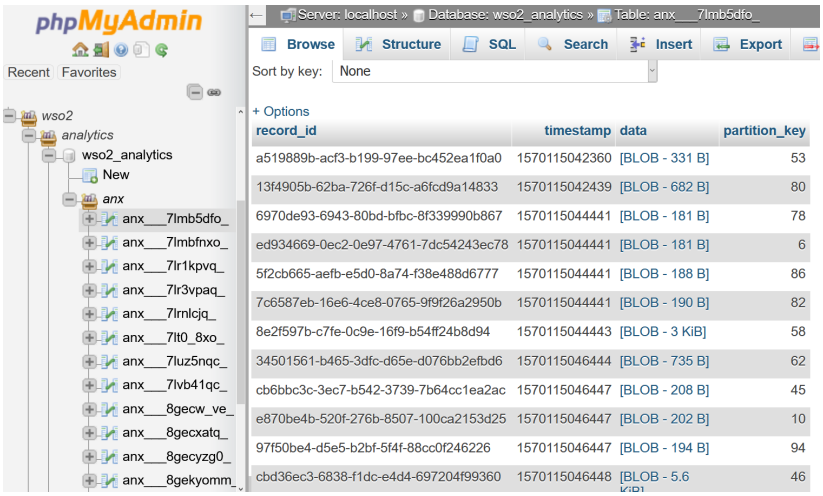

***Fig.** 3 Screenshot of MySQL analytics database*

WSO2 ESB APIs that connect different applications are created using the components called "mediators". Some of such mediators are: Log, Send, Property, Call, Respond etc. In the WSO2 ESB any service can be built by connecting afore mentioned mediators. The task of Log mediator is to allow tracking of all kinds of data that pass through that mediator. Using this component is common in most cases. The log mediator can be configured to log static and dynamic content and can also be configured to track properties of other mediators in an API by using method *get-properties ('propery_name')*. In order to configure WSO2 ESB to send messages that pass through a Log mediators of APIs to an external (MySQL) database server, a new database and the tables need to be created first.

The appropriate settings must be then set in the log4jproperty file (/repository/conf) and the "sql" appender for the log4j.rootLogger must be added. The following code should be added to the log4jproperty file for database [8]:

```
log4j.rootLogger=ERROR,
CARBON_CONSOLE, CARBON_LOGFILE,
CARBON_MEMORY, CARBON_SYS_LOG,
ERROR_LOGFILE, SQL
log4j.appender.sql=org.apache.log4j.jdbc.J
DBCAppender
log4j.appender.sql.URL=jdbc:mysql://localh
ost/LOG_DB
# Set Database Driver, uname and pass
log4j.appender.sql.driver=com.mysql.jdbc.D
river
log4j.appender.sql.user=root
log4j.appender.sql.password=root
# Set the SQL statement.
log4j.appender.sql.sql=INSERT INTO wso2-
esb-api VALUES ('%x', now()
,'%C','%p','%m')
# Define xml layout for file appender
log4j.appender.sql.layout=org.apache.log4j
.PatternLayout
```

In order to test sending the log to a MySQL database from a running WSO2 Web Service (WS) API, the appropriate log data are stored in MySQL database (Fig. 4) by issuing the WS request from SoapUI application (Fig. 5).

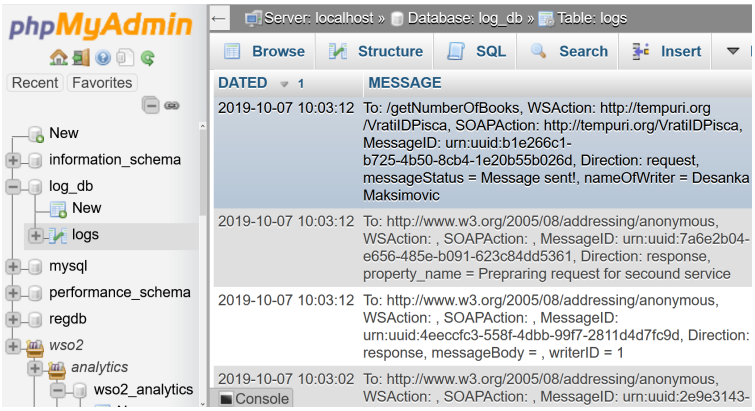

***Fig. 4** Log data in MySQL database received by getNumberOfBooks API*

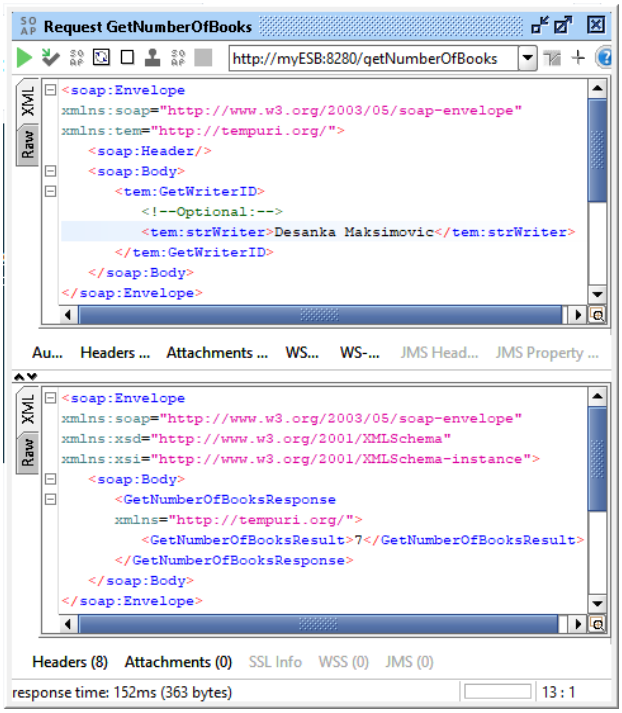

Fig. 5 *Request issued from SoapUI to API getNumberOfBooks of ESB*

This setup with running log appender can lead to performance reduction of ESB. One way to handle the performance issue is to define a large buffer to store the log data. In this way the data will be stored in a database when a buffer is almost full rather than to insert data on each log event. The size of a buffer can be set through the log4j method "bufferSize" where the number to be set is actually the number of log events to be stored in a buffer. Another way for achieving better performances is to filter the log events by keywords (log4j.appender.sql.filter.01.String ToMatch= "message") that will enable inserting only the filtered log events to a database. The similar approach can be used to capture Log mediator messages to a new file instead to a new database.

IMPLEMENTATION OF WSO2 ESB DATA EXCHANGE EVENTS ANALYSES

As already mentioned above, in order to connect external database server for analytic purposes to WSO2 ESB, one has to change settings in analytics-datasources.xml and in metrics-datasources.xml files, and then to run SQL script from <HOME ANALYTICS>\dbscripts to copy data from existing H2 database. After these steps, the ESB Analytics Dashboard will use data from a new external database for data analysis. Which data can be used for analytical purposes must be configured in the Management console.

One of the typical dashboard screen is shown in Fig. 6. From the figure it can be seen the overview of all web services in last 30 days, total number of web services requests and responses, the average time of responses, number of requests to a different APIs, number of requests to different EndPoints, statuses of requests, etc.

The more detailed analysis can be obtained by selecting appropriate menu items on the left side of the dashboard.

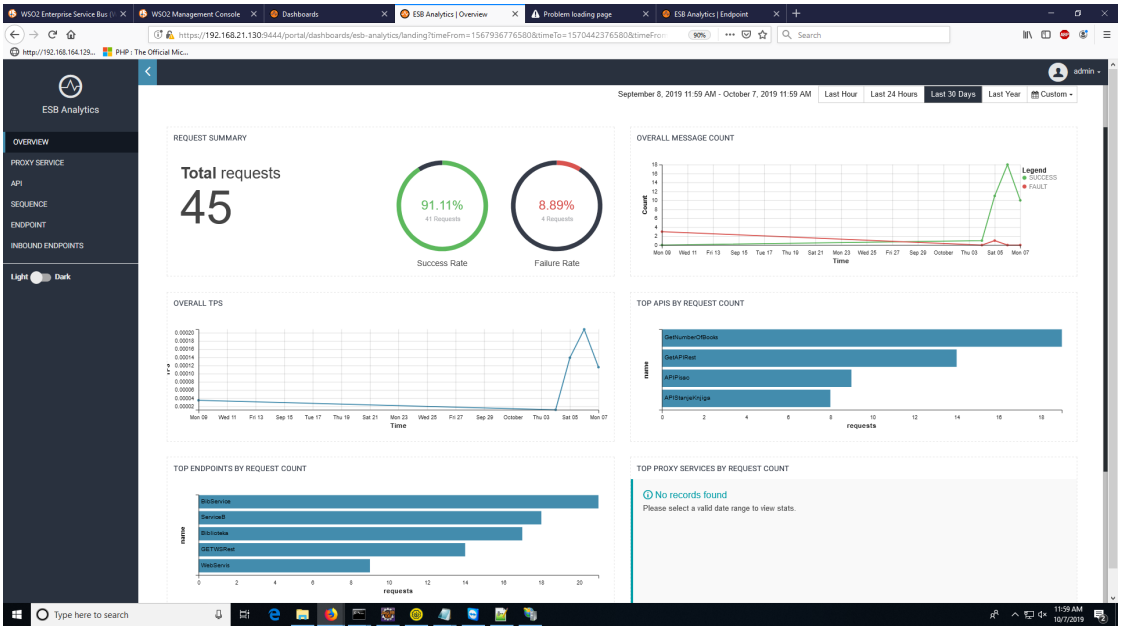

Fig. *6 Dashboard ESB Analytics Overview*

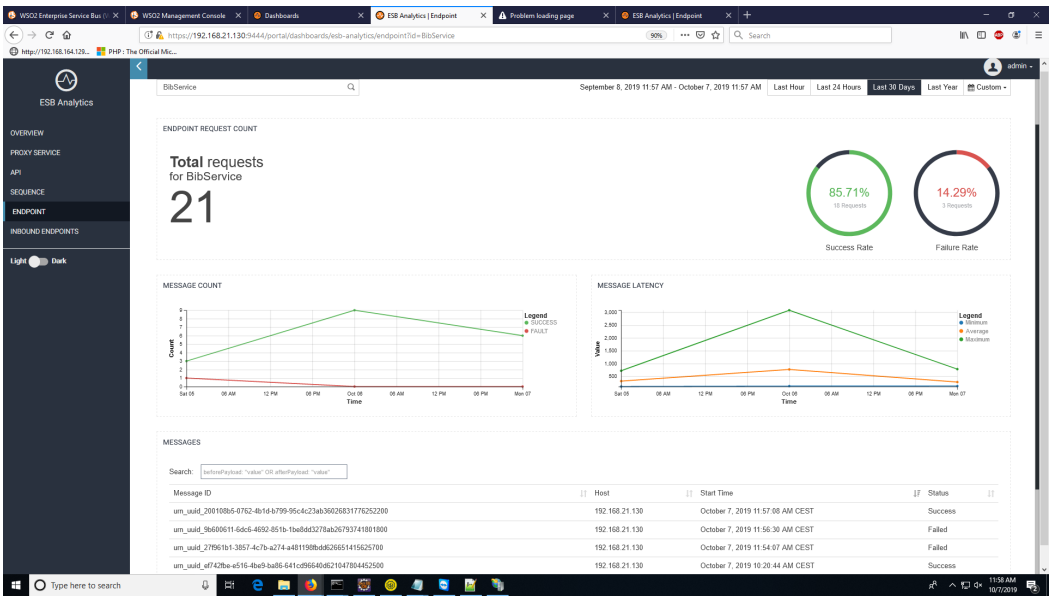

Fig. *7 Dashboard ESB Analytics for Web Service BibService*

The list of request messages to the appropriate endpoints with the graphical statistics of the statuses of requests, message count and latency for the Web Service can be seen in Fig. 7 by using data from external MySQL database server.

In the details pane of the dashboard it can be seen the following data related to each request: ID of the message, the URL of web service requested, timestamp of the issued request and the status of request.

CONCLUSION

Nowadays modern applications frequently exchange data in a controlled way using the Web Services. Usually it is easier to create a single point of access for all communication services rather than to establish multiple point to point connections. This backbone interface is the Enterprise Service Bus (ESB).

In a such environment it is very important to keep a log of data exchange events between applications. A properly configured logging system is vital for identifying errors, security threats, and usage patterns.

Some open-source ESB solutions by default do not offer professional setup of log events storage, but enable modification of initial setup.

In this paper we have shown that by modification of initial setup of WSO2 – open source ESB, we configured it to track communication events in redundant external database servers.

Using the built-in analysis tool (dashboard) we were able to create various reports from recorded data from the external database and make analysis of: statistics for defined period and detailed log of each request.

ACKNOWLEDGMENT

This work has been supported by the Ministry of Science and Technological Development of the R. of Serbia under Project No.TR-35026.

REFERENCE

[1] Menge F. Enterprise Service Bus, Free and open source software conference, 2007.
[2] Indrasiri K. Beginning WSO2 ESB, First Edition, 2016
[3] WSO2 Enterprise Service Bus, Version 5.5.0, 2016.
[4] ESB Documentation, 2017.
[5] Kabakus A. T, Kara R, A performance evaluation of in-memory databases, Journal of King Saud University – Computer and Information Sciences, (2017) 29, 520–525, available at: https://www.sciencedirect.com/science/article/pii/S1319157816300453, 2019.
[6] Prieto F, Ramón E, Lázaro G. WSO2 Developer's Guide, 2017.
[7] MySQL DB Replication, available at: https://medium.com/@madumalt/mysql-db-replication-63786ac8241e, 2019.
[8] Configuring Apache Log4j, available at: https://logging.apache.org/log4j/log4j-2.2/manual/configuration.html, 2019.