SocEDA

Cloud based platform for large scale social aware EDA

ANR-10-SEGI-013

Document name:	Esper Engine versus CEP requirements
Document version:	**V-1.0**
Task code:	
Deliverable code:	D2.2.1
WP Leader (organisation):	OrangeLabs (FT)
Deliverable Leader (organisation):	OrangeLabs
Authors (organisations):	OrangeLabs
Date of first version:	June 2012

Table of Contents

1. Executive Summary

The goal of SocEDA is to develop and validate an elastic and reliable federated SOA architecture for dynamic and complex event-driven interaction in large highly distributed and heterogeneous service systems. Such architecture will enable exchange of contextual information between heterogeneous services, providing the possibilities to optimize/personalize the execution of them, according to social network information. The main outcome will be a platform for event-driven interaction between services, that scales at the Internet level based on the proposed architecture and that addresses Quality of Service (QoS) requirements. The platform consists of:

- Federated Middleware layer: a peer-to-peer overlay network combined with a publish/subscribe mechanism, that has the task to collect events coming from the heterogeneous and distributed services,
- Distributed complex event processor: an elastic, distributed computing cloud based engine for complex processing of events coming from different services in order to detect interesting situations that a service should react on,
- Social aware event modeling and matching plus an event based work-flow engine for filtering events and proposing adaptation and changes in running business processes and services,
- A monitoring and governance framework as well as a Mashup front end to describe and manage services as well as business events.

This document focuses on the Event Processing aspect. The ESPER engine [1] has been selected as an important open source component since the beginning of the SocEDA project. This deliverable describes the ESPER engine and its ability to perform the treatment required by SocEDA platform. The study takes input from the deliverable D4.1.1 *SOTA in CEP* [2] and is based on the requirements elicited in the SocEDA deliverable D2.2.3 *Identification of requirements* [3]. We focus on potential limitations offered by ESPER and explain why a DiCEPE component based on ESPER or another engine is mandatory to be designed and developed.

2. Introduction

2.1. Purpose

SocEDA project aim is to provide an open distributed platform for event-driven interaction between services that scales at the Internet level. To achieve this goal, a federated architecture will be deployed to address the multiplicity and the heterogeneity of service networks. This platform is grounded on a DiCEPE component in order to deploy and run complex event rules.

In this perspective, this document describes the functionalities of how the ESPER engine fulfills or not the requirements of this DiCEPE component.

2.2. Document outline

After a short reminder of the ESPER Engine functionalities in Section 3, Section 4 gives an overview of how the ESPER engine can deal with the DiCEPE requirements.

2.3. List of Acronyms

Acronym	Definition
CEP	Complex Event Processing
DiCEPE	Distributed Complex Event Processing Engine
EDA	Event-Driven Architecture
ESB	Enterprise Service Bus
EPL	Event Processing Language
ESP	Event Stream Processing
CRUD	Create, read, update and delete
EPA	Event Processing Agent
EPN	Event Processing Network

3. Esper Engine

3.1. Features

The ESPER engine is a component for complex event processing (CEP), available for Java.

The ESPER (GPL) engine has been studied in [2] with the implementation of a short POC (see 4.1.3.3). The key functionalities delivered by ESPER (GPL) engine are reminded here:

- ESPER has a strong runtime architecture,
- ESPER engine can be deployed on any operating systems that support Java JSE,
- ESPER can be embedded in java applications,
- ESPER is secured and the platform is scalable,
- ESPER allow the deployment of CEP applications with scripts (A management console is available but required a licensed version),
- ESPER does not deliver a High Availability solution with the GPL version. Esper HA is a paying solution),
- ESPER provides a lot of input adapters (RSS, JMS, JDBC, HTTP, Socket, …) and allows the development of Custom adapters,
- ESPER offers small block of logical functions (through the EPL language) mainly Filtering, Pattern matching, Temporal, Event retention, streams, Event querying, Extensibility, and Enrichment/reference data (see [7] for a complete description of EPN and EPA),
- ESPER can receive new statement at runtime,
- ESPER benchmark announces 500 000 events/s.

4. Requirements from SocEDA

4.1. Requirements

The Requirements are described in the SocEDA Deliverable D2.2.3 [3].The goal now is to see how ESPER can deal with these elicitations. Requirements have been classified as **Functional, Interoperability, Performance** and **Scalability.**

4.1.1. Functional Requirements

Functional Requirements (D2.2.3 [3])		**ESPER**	**Status**
FR001	DiCEPE platform MUST take primitive event(s) and/or complex event(s), and rule(s) (described in Event Processing Language and called also “statements”) as inputs	ESPER understand the EPL syntax (SQL-like language with SELECT, FROM, WHERE, GROUP BY, HAVING and ORDER BY clauses).	OK
FR002	DiCEPE platform MUST return complex event(s) as output(s)	ESPER deals with Complex computations with applications that detect patterns among events and return new complex event in return	OK
FR003	DiCEPE platform MUST synchronize events that arrive from different sources in different time due to network delays	ESPER engine does not offer natively this function.	NOK
FR004	DiCEPE platform MUST support different kind of events	ESPER engine deals with different event representations (Java POJO, Java Maps, XML (DOM) , XML streaming)	OK
FR005	DiCEPE platform SHOULD know the address to which it will send (publish) the processing results (complex events)	ESPER engine has no native interface to publish/subscribe engine	NOK
FR006	Inter-DiCEPE interaction SHOULD be done over a subscription/publication process to exchange events and rules	The GPL ESPER engine version does not offer any mechanism for easily publish/subscribe events to another instance of ESPER	NOK

FR007	DiCEPE platform MUST support the add of event processing rules without stopping the system	ESPER engine offers CRUD functionalities	OK
FR008	DiCEPE platform MUST support the removal of event processing rules without stopping the system	ESPER engine offers CRUD functionalites	OK
FR009	DiCEPE platform MUST support the update of event processing rules without stopping the system	ESPER engine offers CRUD functionalities	OK
FR010	DiCEPE platform MUST support a listing feature of all event processing rules loaded in the DiCEPE without stopping the system	ESPER engine provides the facility for managing statements	OK
FR011	DiCEPE platform MUST be able to return an event processing rule according to its identifier without stopping the system	ESPER engine provides the facility for managing statements. This requirement is manageable with the ESPER API.	OK
FR012	DiCEPE platform SHOULD receive topic(s) to which it must subscribe AND must receive events as notifications when they occur	ESPER engine does not offer pub/sub interface	NOK
FR013	DiCEPE platform SHOULD be managed trough the CEPEditor AND/OR the SeaCloud which interact with exposed service interfaces as web services	Both the CEP editor component and the SeaCloud component are custom SocEDA components. Interfacing to the ESPER engine is to be developed This is a key requirement.	NOK

4.1.2. Interoperability Requirements

Interoperability Requirements		ESPER	Status
IR001	DiCEPE platform SHOULD be portable and interoperable	The ESPER engine by itself is a CEP component portable on many platforms (Java language) but not interoperable with other CEP engines.	NOK
IR002	Within the DiCEPE platform, CEPs MAY be heterogeneous (vendor-specific)	The ESPER engine by itself is just a CEP component. The described requirement here is to abstract the CEP component to use another CEP engine	NOK
IR003	DiCEPE platform must be an open solution that will allow the different CEPs to select their communication protocol, i.e. DiCEPE must support synchronous AND asynchronous communication protocols (e.g. REST, JMS, WS, etc)	The ESPER engine provides *Input and output adapters* accepting events from various sources (HTTP, JMS, Socket)	OK
IR004	DiCEPE platform MUST handle an interoperable and common format for exchanging events	The ESPER engine is adaptable to any XML format for exchanging events	OK

4.1.3. Interoperability Requirements

Performance Requirements		ESPER	Status
PR001	DiCEPE platform MAY be designed as SCA components with FraSCAti	The ESPER engine must be wrapped as a SCA component.	OK
PR002	DiCEPE platform SHOULD be capable to fragment statement(s) between involved remote DiCEPEs	The ESPER engine itself cannot deal with such complex statements and situations.	NOK

4.1.4. Scalability Requirements

Scalability Requirements		ESPER	Status
ScR01	DiCEPE platform MUST support a huge volume of events and rules in an optimal way	ESPER engine benchmarks are available (see 0).	OK
ScR02	DiCEPE platform SHOULD be able to distribute event-processing task over other DiCEPEs if it does not have all the capabilities to preform it OR to do a load balancing	ESPER offers partially a functionality of this type through is EsperHA product [6] (mainly for load balancing). But no fine grain functionality for the distribution of event-processing tasks	NOK

5. Conclusion

Taken into account the previous tables, the ESPER engine considered as a CEP component has serious limitations regarding the expressed requirements. A complete implementation of the described elicitations requires an adaptable wrapper around the ESPER engine (or another Engine, i.e. ETALIS Engine [5]).

So to fulfill the described requirements and to handle the main objectives of the DiCEPE component, an intelligent wrapper must be developed around the CEP engine to deliver a true DiCEPE component. The ESPER engine may be used as the basic CEP engine but other engines such as ETALIS may be also selected.

6. References

[1] Codehaus.org Esper online documentation set - http://esper.codehaus.org/tutorials/tutorials.html - 2007
[2] D4.1.1 Relevant SOTA in CEP system http://research.petalslink.org/download/attachments/2425486/D4.1.1+Relevant+state+of+the+art+in+Esper+CEP+system.pdf?version=1&modificationDate=1321440385000
[3] Soceda D2.2.3 - Identification of requirements to support distributed CEP http://research.petalslink.org/download/attachments/2425506/D2.2.3-Identification+of+requirements+to+support+distributed+CEPv1.2.doc?version=1&modificationDate=1339164305000

[4] ESPER Performances benchmark http://docs.codehaus.org/display/ESPER/Esper+performance
[5] ETALIS CEP engine http://code.google.com/p/etalis
[6] ESPER HA product http://www.espertech.com/products/esperha.php
[7] Event Processing in Action - Opher Etzion and Peter Niblett http://www.amazon.fr/Event-Processing-Action-Opher-Etzion/dp/1935182218