SocEDA

Cloud based platform for large scale social aware EDA

ANR-10-SEGI-013

Document name:	D4.1.4 Extension of the CEP Editor for mashup and events
Document version:	V-1.0
Task code:	
Deliverable code:	D4.1.4
WP Leader (organization):	OrangeLabs (FT)
Deliverable Leader (organization):	OrangeLabs
Authors (organizations):	OrangeLabs
Date of first version:	16/09 2013

Table of Contents

1. General description of the CEP Editor

1.1. Context

1.1.1. Objectives

The SocEDA project will provide an open distributed platform for event-driven interaction between services that scales at the Internet level. The platform environment should cover the whole event-driven application life cycle:

- Design, deploy and run complex and distributed EPL Statements,
- Define complex event patterns,
- Enrich existing events.

As described in the Complex Event Processing state of the art document [2] a key issue is to simplify the CEP design aspect with graphical tools. This document is about the CEP Editor which aims at facilitating the CEP rules creations in the context of the whole SocEDA platform [3].

In the SocEDA project one of the targeted CEP is the ESPER engine [1], so we will be talking about EPL for event processing language. The main goal is to translate a designed model and a high level definition of conditions into EPL Rules (ESPER-EPL as a first step).

The main differences with the previous V1 version are the following ones: Link with the SeaCloud environment. The consequence is the link with the event type repository of the SocEDA platform and the link to push rules to the platform.

2. Terms and Concepts definition

2.1.1. Acronyms

EPL	Event Processing Language
CEP	Complex Event Processing
Event type	Event type (event class, event definition, or event schema) represents a class of event objects
DCEP	Distributed Complex Event Processing
IDE	Integrated Development Environment
DiCEPE API	Distributed Complex Event Processing API
REST	Representational State Transfer
Servlet	A servlet is a small program that runs on a web server See Java Servlet Application Programming Interface (Sun API)

3. General Architecture

3.1.1. Main principles

In order to make this EPL rule design aspect easier, we provide a drag & drop tool where you can use event types coming from an event type repository.
Basically, it allows the designing of a small diagram representation of rules in order to push EPL queries within an ESPER instance without having to interact with any IDE.

3.1.1.1. CEP Editor Tool positioning

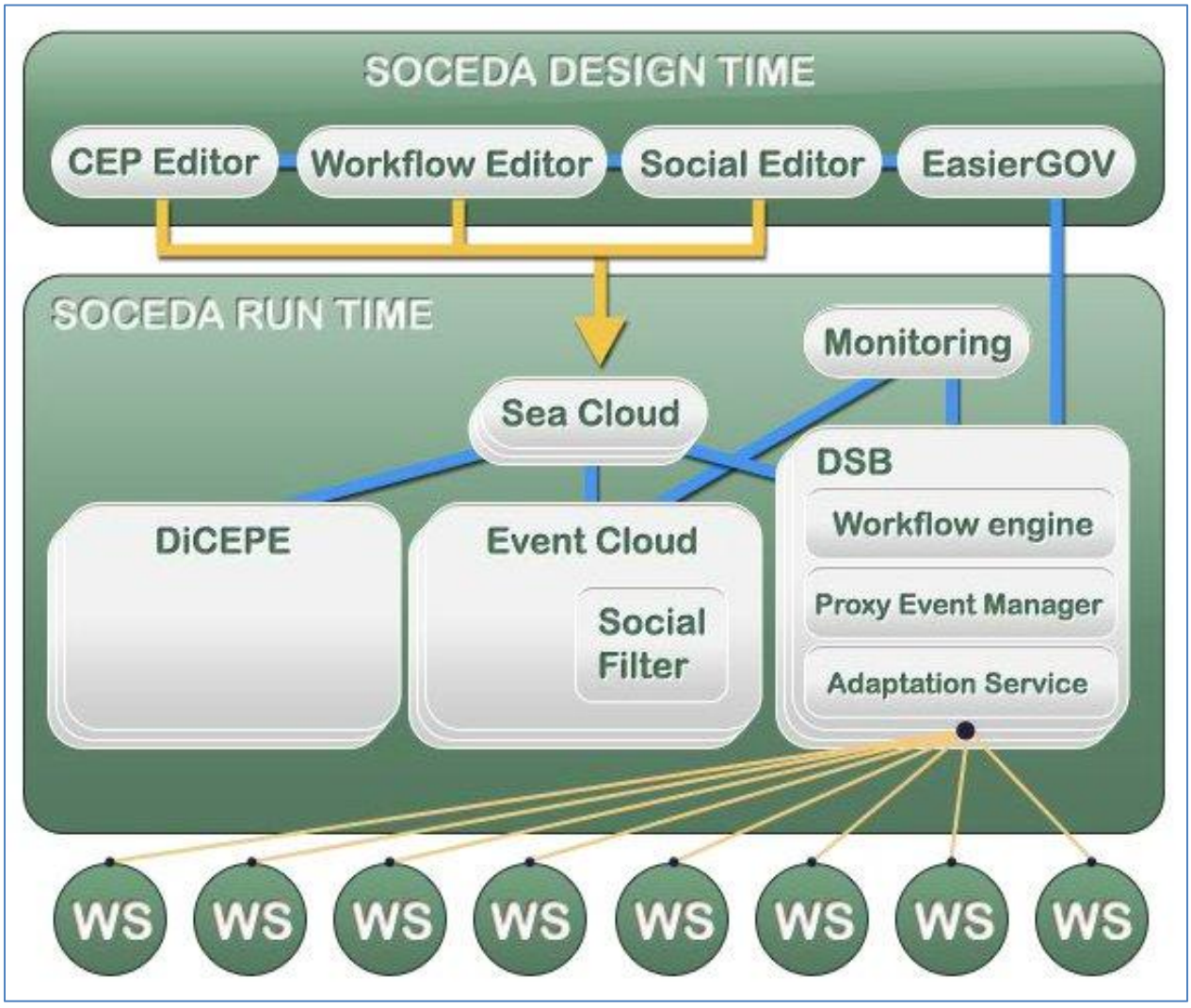

Figure 1: SOCEDA conceptual architecture

Since the v1 we added the link between the CEP Editor and the SeaCloud environment by setting the SeaCloud endpoint as a CEP Editor URL parameter.
e.g:
http://localhost:8080/cep-editor-v0.4/?seacloudUrl=http://localhost:9000/SeaCloudAdmin&easiergovUrl=http://localhost:9005/services/eventManager

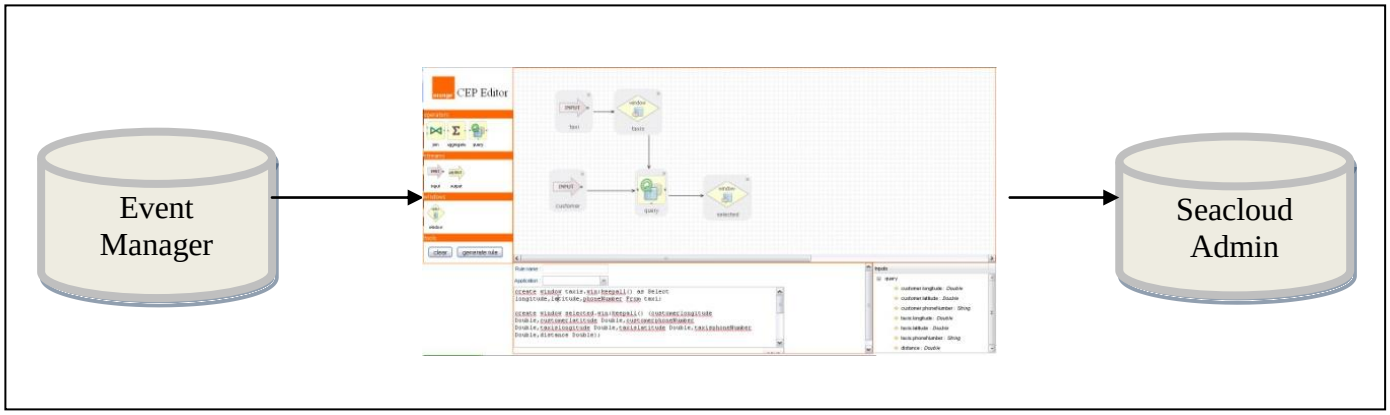

Figure 2: CEP Editor within the SocEDA platform

The eventManager allows retrieving all event type repository managed by the SocEDA platform. The Editor loads them when the web application is launched; if the link is broken and nothing is retrieved, the Editor loads a minimum event type (10 event types) description.

The Seacloud Admin is used to push the generated rules to the DiCEPE through the SeaCloud component. These two web services are proposed via the SocEDA platform and via the web application through the server side.

We have a DiCEPE Administration API and an Event Type Repository API to bind with the EPL Editor. The interface provided by the SeaCloud is the following:

```
listAllStatements(ListAllStatements parameters)
addStatement(String statementId, String statement)
deleteStatement(String statementId)
getStatementById(String statementId
updateStatement(String statementId, String statement
addStatementWithActions(AddStatementWithActions parameters)
```

These calls have been linked to the appropriate feature within the CEP Editor and the Android UI designer. These changes are the main new features compared to the previous CEP Editor version.

A jar library has been provided by Partners in order to make the Web service link easy to use.

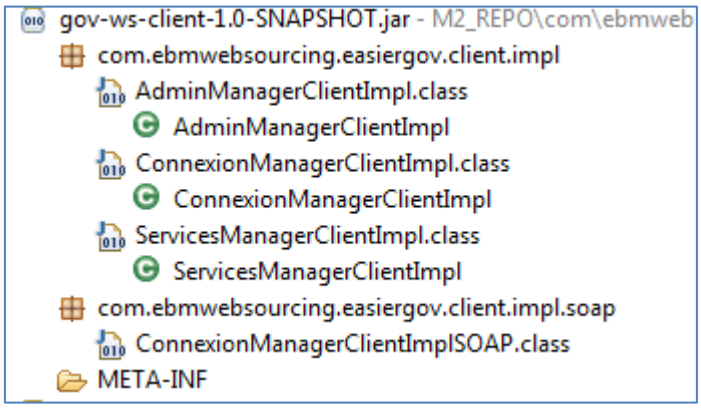

Figure 3: description of the jar library

Then we can setup the endpoint provided within the URL in order to instantiate the appropriate component and use the SocEDA platform according to our needs.

How to retrieve topics?

```
// get the event type list from the Web service
public ArrayList<EventType> getEventTypes() {
ArrayList<EventType> events = new ArrayList<EventType>();

List<EJaxbTopicType> topics = new ArrayList<EJaxbTopicType>();
try {
	topics = eventManager.findTopics("");
} catch (FindTopicsFault e1) {
	e1.printStackTrace();
}
String id, name, namespace;
GetElement request = new GetElement();
GetElementResponse response;
for (EJaxbTopicType topic : topics) {
	// call the eventManager in order to get the event description
	request.setIdElement(topic.getMessageTypes().get(0));
	try {
		response = dataManager.getElement(request);
		events.add(new EventType(topic.getMessageTypes().get(0)
				.getLocalPart(), parser.getPropertiesFromXML(topic
				.getMessageTypes().get(0).getLocalPart(),
				XMLPrettyPrinter.prettyPrint(SOAJAXBContext
						.getInstance().unmarshallAnyElement(
								response.getAny()))), topic
	.getMessageTypes().get(0).getNamespaceURI()));
	} catch (FaultMessage e) {
				e.printStackTrace();
			} catch (SOAException e) {
				e.printStackTrace();
			}
		}
```

How to add a rule ?

```
public void addRuleWithAction(String name, String rule,
        ArrayList<ft.orange.portail.shared.Action> actions) {

AddStatementWithActions statement = new AddStatementWithActions();
List<Action> l = statement.getAction();
for (ft.orange.portail.shared.Action currentAction : actions) {
        Action a = new Action();
        Fields f = new Fields();
        Entry e = new Entry();
        for (KeyValueProperties s : currentAction.getArguments()) {
                e = new Entry();
                e.setKey(s.getKey());
                e.setValue(s.getValue().contains("'") ? s.getValue() : "$" + s.getValue());
                f.getEntry().add(e);
                a.setFields(f);
                a.setType("event");

        }
        l.add(a);
        a.setName(new QName(currentAction.getEventNamespace2Generate(),
                        currentAction.getEvent2Generate(), "ns12"));
        a.getTopic()
                        .add(new QName(currentAction.getEventNamespace2Generate(),
                                        currentAction.getEvent2Generate() + "Topic", "ns12"));
        ArrayList<EventType> eventTypes = getEventTypes();
        EPStatementObjectModel stat = EPLValidator.esperEngine
                        .getEPAdministrator().compileEPL(rule);
        List<Stream> streams = stat.getFromClause().getStreams();

        List<Namespace> ln = statement.getNamespaceOfEventTypes();

        for (Stream s : streams) {
                for (EventType ev : eventTypes) {
                        if (ev.getName().equalsIgnoreCase(
                                        ((FilterStream) s).getFilter().getEventTypeName())) {
                                Namespace n = new Namespace();
                                n.setEventTypeName(ev.getName());
                                n.setPrefix("data");
                                n.setNamespace(ev.getNamespace());
                                ln.add(n);
                                break;
                        }
                }
        }
        Namespace n = new Namespace();
        n.setEventTypeName(currentAction.event2Generate);
        n.setPrefix("data");
        n.setNamespace(currentAction.getEventNamespace2Generate());
        ln.add(n);
}
statement.setStatement(rule);
statement.setStatementId(name);
AddStatementXMLBuilder xmlbuilder = new AddStatementXMLBuilder(
                statement);
//xmlbuilder.print();
try {
        seacLoud.deploy(xmlbuilder.asFile(), null);
} catch (CLoudManagementException e1) {
        System.err.println("impossible to deploy the rule on "+eventManager.getAddress()+" because of
"+e1.getMessage());
}

}
```

4. Roadmap

This tab lists the functionalities that have been considered in this document, and clarifies what has been implemented in the second version (V2) updating the previous version (V1).

Functionality	Version	Reference / Comment
Base environment and operations: Palette & Flow, drag & drop, components connection, opening properties, etc.	V. 0	3.1.1
EPL generation (basics)	V. 0	**Erreur ! Source du renvoi introuvable.**
EPL generation (advanced)	V. 1	**Erreur ! Source du renvoi introuvable.**
Binding with an event type repository	V. 0	**Erreur ! Source du renvoi introuvable.**
EPL validation	V. 0	3.3.1.2
Export/import	V. 1	**Erreur ! Source du renvoi introuvable.**
Binding with the easierGov	**V. 2**	**3.1.1.1**
CEP Manager	V. 1	3.5
Binding with Seacloud	**v. 2**	

5. References

[1] ESPER http://esper.codehaus.org/

[2] D4.1.1 State of the Art in CEP (Deliverable SocEDA)

[3] D1.2.1 Overall Framework Model (Deliverable SocEDA)