Prototype of Esper-based DAQ expert system

Tomasz Bawej

March 12, 2014

0.1 Overview

0.1.1 Nutshell

The program examines events, arriving in streams, looking for specified patterns in order to produce an appropriate notification or create new events using transfomed, derived or filtered information. The events, being arrays of Java Objects, are passed through matching instructions - *statements* - as they arrive and, once processed, they get discarded. Should the processed events match a given pattern, the information is available right away. This is often referred to as an inversed database paradigm with the queries being retained and the data passing through them.

0.1.2 Events

In the project setup events are played back from a database. Upon retrieval of a database row it gets converted into an array of Objects that gets passed to the engine. A special `CurrentTimeSpanEvent` is used to update the engine clock as the events with progressing timestamps are retrieved.

0.1.3 Statements

Statement is an instruction to the engine indicating the way that certain events should be processed. Within the Esper engine each statement is an `EPStatement` Object that several types of listeners can be attached to. Most importantly, a listener can be specified to receive updates from the statement - its output triggered by the most recent event. Such listeners are used to route the notifications as they are produced by the Esper engine.
While the statement objects can be created entirely by the means of invoking particular Esper API methods, Esper provides an SQL-like Event Processing Language (EPL) to facilitate the process. EPL statements can either be deployed one-by-one by passing a corresponding String literal or as whole files - modules.

0.2 Esper

0.3 Project

Project comprises the following modules:

- LOAD - the application controller object governing the lifecycle of federated objects
- EventsProcessor - Esper engine wrapper responsible
- FieldsTypeResolver

- EventsTap - an interface
  - DbFlashlistEventsTap
- EventsSink

0.3.1 Data

0.3.2 Architecture

0.4 Functionality

0.4.1 Implemented checks

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

0.4.2 Overview

0.4.3 Details