OpenCredo Esper v 1.0

The OpenCredo Esper project builds on top of the excellent Complex Event Stream processing project Esper, see http://esper.codehaus.org . The OpenCredo Esper project is primarily motivated by a desire to make using Esper easier within messaging applications built on Spring Integration. Using Esper to provide views of the data/messages passing through a system is an excellent way to improve the comprehension of complex event driven/asynchronous messaging based applications in Spring Integration. In order to implement the Spring Integration integrations, the OpenCredo Esper project also created a Spring-style template which makes calling Esper extremely simple regardless of whether the application is using Spring Integration.

EsperTemplate

The EsperTemplate allows the simple creation of statements and optionally the association of listeners with those statements along with a mechanism for publishing events to the Esper engine.

The following code is taken from the samples. The example creates a statement and registers a listener that will receive all sleep events.

```
EsperStatement sleepStatment =
       new EsperStatement("select * from Sleep");
sleepStatment.addListener(new LoggingListener());
template.addStatement(sleepStatment);
template.initialize();
```

The listener needs to implement the Esper interface `UpdateListener` as in the following example that logs the total property for new events.

```
private static class LoggingListener implements UpdateListener {
  public void update(EventBean[] newEvents, EventBean[] oldEvents) {
     for (EventBean newEvent : newEvents) {
      ...
     }
   }
}
```

Publishing events is then as simple as.

```
 template.sendEvent(new Sleep());
```

The module also provides a Spring namespace allowing xml based configuration of a template along with a set of associated statements and listeners.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:esper="http://www.opencredo.com/schema/esper"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
       http://www.opencredo.com/schema/esper
       http://www.opencredo.com/schema/esper/opencredo-esper-1.0.xsd">
  <esper:template>
    <esper:statements>
        <esper:statement
              epl="select * from org.opencredo.esper.sample.SampleEvent">
              <esper:listeners>
                 <bean class="CallRecordingListener" />
              </esper:listeners>
        </esper:statement>
    </esper:statements>
  </esper:template>
</beans>
```

Spring Integration Esper support

The Esper support includes three integration points between Esper and Spring Integration. Firstly there is an inbound channel adapter which receives Esper events and makes them available as Spring Integration messages. Secondly there is a wire tap implementation which intercepts messages using the standard channel interception interface and publishes those messages as events to Esper. Thirdly there is a throughput monitor built on top of the wire tap that monitors the number of messages being sent to a channel.

Inbound channel adapter

The inbound channel adapter implements the Esper interfaces UpdateListener and UnmatchedListener allowing it to be registered with a template to publish either. The example below shows how to publish periodic counts of sheep the Spring Integration channel inboundEsperMessages. Note the use of the esper namespace prefix for the template and the si-esper namespace prefix for the inbound channel adapter.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:esper="http://www.opencredo.com/schema/esper"
       xmlns:si-esper="http://www.opencredo.com/schema/esper/integration"
       xmlns:si="http://www.springframework.org/schema/integration"
       xsi:schemaLocation="http://www.springframework.org/schema/beans

       http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
                                        http://www.opencredo.com/schema/esper
       http://www.opencredo.com/schema/esper/opencredo-esper-1.0.xsd
       http://www.opencredo.com/schema/esper/integration
       http://www.opencredo.com/schema/esper/integration/opencredo-integration-esper-
1.0.xsd
       http://www.springframework.org/schema/integration
       http://www.springframework.org/schema/integration/spring-integration-1.0.xsd">

<esper:template id="template" configuration="esper-configuration.xml">
              <esper:statements>
                     <esper:statement epl="select count(*) from
Sheep.win:time_batch(1 second)">
                           <esper:listeners>
                                  <esper:ref bean="inboundAdapter" />
                           </esper:listeners>
                     </esper:statement>
              </esper:statements>
       </esper:template>

       <si-esper:inbound-channel-adapter id="inboundAdapter"
              channel="inboundEsperMessages" />
```

Esper Wire Tap

The Esper Wire Tap allows the transparent publication of channel interception events. This covers the four interception points defined in the Spring Integration ChannelInterceptor interface, pre send, post send, pre receive and post receive. The wire tap can be configured to publish events to Esper for one or more of theses interception points. The example that follows configures a wire tap which will publish events immediately after a message is sent to any channel with a name matching the

specified pattern, 'orders' in this case. This example also sets the send context flag to true meaning that the event broadcast to Esper will be an instance of org.opencredo.esper.integration.MessageContext that will contain a reference to the channel as well as the Message instance that was sent. In the case that send context is false the event will simply be the Message instance itself.

```
	<si-esper:wire-tap id="wiretap" sourceId="defaultWiretap" template-ref="template"
post-send="true" send-context="true" />

	<si-esper:wire-tap-channels default-wire-tap="wiretap">
		<si-esper:channel pattern="orders" />
	</si-esper:wire-tap-channels>

	<esper:template id="template" configuration="esper-configuration.xml" />
```

Esper Throughput Monitor

The throughput monitor is designed to provide easy access to throughput information for messages passing over Spring Integration channels. The example below shows a throughput monitor configured to collect the number of messages sent to the Spring Integration order channel in each five second interval. This information can then be exported using JMX or picked up by Spring Integration by configuring a default inbound channel adapter to poll the getThroughput method of the configured throughput monitor instance.

```
<si-esper:channel-throughput-monitor id="throughput" sourceId="exampleThroughput"
    channel-ref="orderChannel" time-sample="5 second" />
```