Getting Started Native Java

Summary

This example we show a simple use of the IoT Agent Statement API. As an example, we will count a number of events revived by the agent. For this we need:

1. to deploy and run an agent in one console or screen
2. to deploy a statement that does the counting
3. subscribing to the results of this statement
4. generate events so the agent can process them

Requirements

1. Java >= 1.8
2. Local MQTT broker
3. mosquito clients
4. curl
5. wget

Notes:

For running the IoT Data-Processing Agent is only needed the (1) and an MQTT broker no matter where. Anyhow, for realizing this tutorial the all above is needed. Please, notice that for this demo we need three console sessions.

1) Deploy and run the agent

In the console 1 download the agent. (NOTE: Update the version (1.8.2) to the latest in the URL)

Deployment (console 1)

```
curl -O "https://nexus.linksmart.eu/repository/maven-releases/eu/linksmart/services/events/gpl/distributions
/iot.learning.universal.gpl.agent/1.8.2/iot.learning.universal.gpl.agent-1.8.2.jar"
```

Result Deployment (console 1)

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 49.1M  100 49.1M    0     0  6935k      0  0:00:07  0:00:07 --:--:-- 6938k
```

Again in console 1, we started the agent.

Run (console 1)

```
# using bash
env_var_enabled=true cep_init_engines=eu.linksmart.services.event.cep.engines.EsperEngine
agent_init_extensions=eu.linksmart.services.event.ceml.core.CEML java -cp ./* "org.springframework.boot.loader.
PropertiesLauncher"

# using powershell
$env:env_var_enabled="true"
$env:cep_init_engines="eu.linksmart.services.event.cep.engines.EsperEngine"
$env:agent_init_extensions="eu.linksmart.services.event.ceml.core.CEML"
java -cp .\*  "org.springframework.boot.loader.PropertiesLauncher"
```

Result Run (console 1)

```
....
Lot of the output had being remove.
If no exception is given then this output is not relevant now.
....
2016-09-08 16:00:16.202  INFO 16010 --- [           main] eu.linksmart.services.Application        : Started
Application in 6.607 seconds (JVM running for 9.535)
```

2) Deploy a statement in the agent

In console three we deploy a statement.

Deploy Statement (console 2)

```
curl -v -H "Content-Type: application/json" -X POST -d '{"name":"countEvents","statement":"select count(*) as
obs from Observation.win:time(1 sec)"}' http://localhost:8319/statement/
```

The content of the result is not important what is important is that returns 201 created as the response.

Result part 1

```
> POST /statement/ HTTP/1.1
> User-Agent: curl/7.35.0
> Host: localhost:8319
> Accept: */*
> Content-Type: application/json
> Content-Length: 92
>
* upload completely sent off: 92 out of 92 bytes
< HTTP/1.1 201 Created
< Date: Thu, 26 Oct 2017 11:30:56 GMT
< Location: /statement/d88809505168bb760859e4651c15008d9d0a4435c9b9419555716dab3a78ecf1
< Content-Type: application/json; charset=UTF-8
< Content-Length: 3049
* Server Jetty(9.2.15.v20160210) is not blacklisted
< Server: Jetty(9.2.15.v20160210)
```

Prettify Result (console 2)


```
{
   "resources":{
      "6de4cb5b60053d24967379da038c946bdc4947437cfd56571b24163e68a64d33":{
         "cehandler":"eu.linksmart.services.event.handler.ComplexEventHandler",
         "restoutput":false,
         "persistent":false,
         "essential":false,
         "id":"6de4cb5b60053d24967379da038c946bdc4947437cfd56571b24163e68a64d33",
         "name":"countEvents",
         "statement":"select count(*) as obs from Observation.win:time(1 sec)",
         "output":[
            "LS/LA/bdb51cc6-acb6-4fc5-a1b3-593d4c36e41a/OGC/1.0/Datastreams
/6de4cb5b60053d24967379da038c946bdc4947437cfd56571b24163e68a64d33"
         ],
         "stateLifecycle":"RUN",
         "scope":[
            "outgoing"
         ],
         "agentID":"bdb51cc6-acb6-4fc5-a1b3-593d4c36e41a",
         "resultType":"eu.linksmart.services.payloads.ogc.sensorthing.linked.ObservationImpl",
         "publisher":"MQTT_PUB",
         "registrable":true,
         "logEventEvery":1,
         "discardDataOnFailPolicyOn":false
      }
   },
   "responses":[
      {
         "headline":"Created",
         "agentID":"bdb51cc6-acb6-4fc5-a1b3-593d4c36e41a",
         "producerID":"EsperEngine",
         "producerName":"CEPEngine",
         "message":"Statement 6de4cb5b60053d24967379da038c946bdc4947437cfd56571b24163e68a64d33 was successful*
Connection #0 to host localhost left
intact
",
         "status":201,
         "messageType":"SUCCESS",
         "topics":[
            "LS/LA/bdb51cc6-acb6-4fc5-a1b3-593d4c36e41a/OGC/1.0/Datastreams
/6de4cb5b60053d24967379da038c946bdc4947437cfd56571b24163e68a64d33"
         ]
      }
   ]
}
```


3) Subscribe to the statement

We subscribe to all statements.

Subscribe to an event (Console 2)

```
mosquitto_sub -t LS/LA/+/OGC/1.0/Datastreams/# &
```

Note:

The output topic can be configured to see IoT agents configuration.

In the default case, the '+' can be replaced by the id of the agent which can be set, or it is autogenerated and display at the start of the agent with a message such as:

> *[main] INFO e.l.s.event.core.DataProcessingCore - The Agent streaming core version 1.6.0 is starting with ID: e0e0c70e-2df4-4bbc-baee-df463bdb892d*

or it is shown every minute in the console as:

> *INFO 261 --- [Thread-34] e.l.s.event.core.DataProcessingCore   : The Agent with ID e0e0c70e-2df4-4bbc-baee-df463bdb892d is alive*

The '#' can be replaced with the id of the statement shown in location attributed located in the header response after the statement had been posted. In out case:

> *d88809505168bb760859e4651c15008d9d0a4435c9b9419555716dab3a78ecf1*

4) Generate data

Generate event (console 2)

```
mosquitto_pub -t LS/test/1/OGC/1.0/Datastreams/1/ -f event.json
```

Note:

The input topic can be configured to see IoT agents configuration.

event.json

```
{
   "Datastream":{
      "id":1
   },
   "Result":1,
   "Time":"2015-09-08T17:16:12.00Z",
   "Sensor":{
      "id":"1"
   },
   "id":"1"
}
```

Then in console 2, we can see the result. Quicker the data is being generated the "Result" will increase.

Result (Console 2)

```
{
   "datastream":{
      "@iot.id":"6de4cb5b60053d24967379da038c946bdc4947437cfd56571b24163e68a64d33"
   },
   "phenomenonTime":"2019-09-06T10:32:35.845Z",
   "result":1
}
```