Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

D.5.4.2: FI-WARE User and Programmers Guide

Project acronym: FI-WARE
Project full title: Future Internet Core Platform
Contract No.: 285248
Strategic Objective: FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform
Project Document Number: ICT-2011-FI-285248-WP5-D.5.4.2
Project Document Date: 2013-05-15
Deliverable Type and Security: Public
Author: FI-WARE Consortium
Contributors: FI-WARE Consortium

1.1 Executive Summary

This document contains a guide for users and programmers that ain at working with Generic Enabler developed within in the "Internet of the Things" chapter.

1.2 About This Document

This document comes along with the Software implementation of components, each release of the document being referred to the corresponding Software release (as per D.x.2), to provide documentation of the features offered by the components and interfaces to users/adopters. Moreover, it explains the way they can be exploited in their developments.

1.3 Intended Audience

The document targets users as well as programmers of FI-WARE Generic Enablers.

1.4 Chapter Context

FI-WARE will build the relevant Generic Enablers for Internet of Things Service Enablement, in order for things to become citizens of the Internet–available, searchable, accessible, and usable – and for FI services to create value from real-world interaction enabled by the ubiquity of heterogeneous and resource-constrained devices.

From a physical standpoint, IoT enablers have been spread in two different domains:

- **FI-WARE IoT Gateway.** A hardware device hosting a number of features of one or several Gateway Generic Enablers of the IoT Service Enablement. It is usually located at proximity of the devices (sensors/actuators) to be connected. In the FI-WARE IoT model, the IoT Gateway is an optional element aiming to optimize the network traffic sent to the Backend and IoT services efficiency and reliability. Zero, one or more IoT Gateways can be part of a FI-WARE IoT setting. Several m2m technologies introduce specific gateway devices too, where it is not feasible to install FI-WARE gateway features. Those gateways are considered plain devices grouping other devices and not FI-WARE IoT Gateways.
- **FI-WARE IoT Backend.** A setting in the cloud hosting a number of features of one or several Generic Enablers of the IoT Service Enablement. It is typically part of a FI-WARE platform instance in a Datacenter. In the FI-WARE IoT model, a single IoT Backend is mandatory and it is connected to all IoT end devices either via IoT Gateway(s) and/or straight interfaces. Normally, during FI-WARE Releases R1 and R3 timeframes, the Backend will refer to the IoT Backend enablers installed in the FI-WARE Testbed or Open Innovation Lab (OIL), as described in the project Catalogue.

A key design statement is that, whenever present, IoT Gateways are not expected to be permanently connected to the Backend as per communications design or failures. Another relevant remark is that IoT Gateways are expected to be constrained devices in some scenarios. Therefore, light-weight implementations of the same GEs plus additional GEs interfaces helping to save unnecessary features/GEs are specially considered in the Gateway domain.

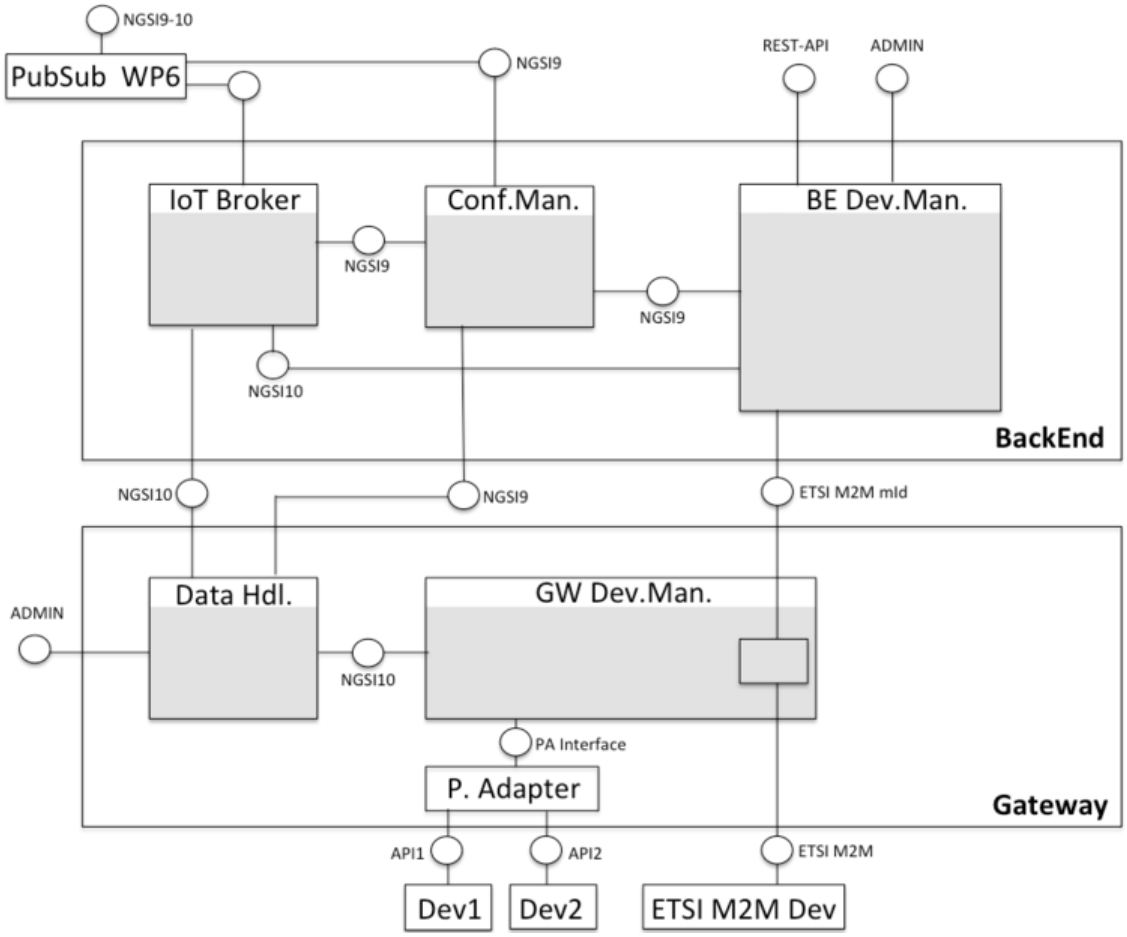
From the functionality point of view, FI-WARE IoT design aims to expose the "Things"

abstraction to services developers, cope with different vertical m2m applications and provide a uniform access to heterogeneous m2m hardware and protocols. There is a number IoT features which are somehow duplicated in the Backend and the gateway domains in order to fulfill the goals and statements described above. For instance, a CEP engine at the Gateway level reduces the network overload and improves condition-based-events triggering time. Application developers will be able to access Things and devices observation and control interfaces in two ways:

- Directly, by using Northbound IoT interfaces as described in this Wiki.
- Throughout Data/Context GEs, by configuring Backend IoT GEs (IoT Broker) as NGSI notifications Context Providers of Data/Context Publish-Subscribe-Context-Broker GE.

Nota Bene: For the reader, we are using in the following chapters the same vocabulary as in the FI-Ware Product Vision chapter:

- **Thing.** Physical object, living organism, person or concept interesting from the perspective of an application.
- **Device.** Hardware entity, component or system that either measures properties of a thing/group of things or influences the properties of a thing/group of things or both measures/influences. Sensors and actuators are devices.
- **IoT Resource.** Computational elements (software) that provide the technical means to perform sensing and/or actuation on the device. The resource is usually hosted on the device.



More information about the IoT Service Enablement Chapter and FI-WARE in general can be found within the following pages:

http://wiki.fi-ware.eu

Internet_of_Things_Services_Enablement_Architecture

Materializing_Internet-Of-Things-Services-Enablement_in_FI-Ware

1.5 Structure of this Document

The document is generated out of a set of documents provided in the public FI-WARE wiki. For the current version of the documents, please visit the public wiki at http://wiki.fi-ware.eu/

The following resources were used to generate this document:

D.5.4.2_User_and_Programmers_Guide_front_page

Backend IoT Broker GE - User and Programmers Guide

Configuration Manager GE - User and Programmers Guide

Gateway Data Handling GE - User and Programmers Guide

Gateway_Data_Handling_-_Esper4FastData_Servlet_-_User_and_Programmers Guide

Gateway_Data_Handling_-_Esper4FastData_Mobile_-_User_and_Programmers Guide

Gateway Data Handling - SOL CEP User and Programmers Guide

1.6 Typographical Conventions

Starting with October 2012 the FI-WARE project improved the quality and streamlined the submission process for deliverables, generated out of the public and private FI-WARE wiki. The project is currently working on the migration of as many deliverables as possible towards the new system.

This document is rendered with semi-automatic scripts out of a MediaWiki system operated by the FI-WARE consortium.

1.6.1 Links within this document

The links within this document point towards the wiki where the content was rendered from. You can browse these links in order to find the "current" status of the particular content.

Due to technical reasons part of the links contained in the deliverables generated from wiki pages cannot be rendered to fully working links. This happens for instance when a wiki page references a section within the same wiki page (but there are other cases). In such scenarios we preserve a link for readability purposes but this points to an explanatory page, not the original target page.

In such cases where you find links that do not actually point to the original location, we encourage you to visit the source pages to get all the source information in its original form. Most of the links are however correct and this impacts a small fraction of those in our deliverables.

1.6.2 Figures

Figures are mainly inserted within the wiki as the following one:

```
[[Image:....|size|alignment|Caption]]
```

Only if the wiki-page uses this format, the related caption is applied on the printed document. As currently this format is not used consistently within the wiki, please understand that the rendered pages have different caption layouts and different caption formats in general. Due to technical reasons the caption can't be numbered automatically.

1.6.3 Sample software code

Sample API-calls may be inserted like the following one.

```
http://[SERVER_URL]?filter=name:Simth*&index=20&limit=10
```

1.7 Acknowledgements

The current document has been elaborated using a number of collaborative tools, with the participation of Working Package Leaders and Architects as well as those partners in their teams they have decided to involve.

1.8 Keyword list

FI-WARE, PPP, Architecture Board, Steering Board, Roadmap, Reference Architecture, Generic Enabler, Open Specifications, I2ND, Cloud, IoT, Data/Context Management, Applications/Services Ecosystem, Delivery Framework , Security, Developers Community and Tools , ICT, es.Internet, Latin American Platforms, Cloud Edge, Cloud Proxy.

1.9 Changes History

Release	Major changes description	Date	Editor
v0	First draft of deliverable submission generated	2013-04-22	TID
V1	Final version	2013-05-15	TID

1.10 Table of Contents

1.1 Executive Summary 2
1.2 About This Document 2
1.3 Intended Audience 2
1.4 Chapter Context 2
1.5 Structure of this Document 4
1.6 Typographical Conventions 4

1.6.1 Links within this document 4
1.6.2 Figures 5
1.6.3 Sample software code 5
1.7 Acknowledgements 5
1.8 Keyword list 5
1.9 Changes History 5
1.10 Table of Contents 5
2 Backend IoT Broker GE - User and Programmers Guide 8
2.1 Introduction 8
2.2 User Guide 8
2.3 Programmer Guide 8
2.3.1 Accessing the IoT Broker NGSI-10 Interface from a Browser 10
3 Configuration Manager GE - User and Programmers Guide 12
3.1 Introduction 12
3.2 User Guide 12
3.3 Programmer Guide 12
3.3.1 Starting iotConfigMgr 12
3.3.2 Database 13
3.3.3 Registering entities 13
3.3.4 Discovering Entities 15
3.3.5 Connecting to iotConfigMgr 18
3.3.6 Sending requests to iotConfigMgr 19
3.3.7 Debugging iotConfigMgr 19
4 Gateway Data Handling GE - User and Programmers Guide 21
5 Gateway Data Handling - Esper4FastData Servlet - User and Programmers Guide 22
5.1.1 Introduction 22
5.2 User and Programmer Guide 27
5.2.1 Accessing the Esper4FastData Servlet Interface from a Browser 27
5.2.2 database methods 27
5.2.3 ngsi methods 27
5.2.4 Engine methods 41
5.2.5 EventType methods 43
5.2.6 EventType transform methods 47
5.2.7 Statement methods 51
5.2.8 Send Event methods 57
5.2.9 Use Case from end to end 58

6 Gateway Data Handling - Esper4FastData Mobile - User and Programmers Guide 65
6.1.1 Introduction 65
6.1.2 User and Programmer Guide 70
7 Gateway Data Handling - SOL CEP User and Programmers Guide 74
7.1 Introduction 74
7.1.1 Addressed Topics 74
7.2 Programmers Guide 74
7.2.1 Configuration 74
7.2.2 Restrictions 75

2 Backend IoT Broker GE - User and Programmers Guide

You can find the content of this chapter as well in the wiki of fi-ware.

2.1 Introduction

Welcome the IoT Broker GE User and Programmer Guide. The IoT Broker is the component implemented by NEC in the IoT WP (WP5). It is programmed in JAVA based on OSGi using standard interface NGSI 9/10 to comunicate with the other components/GEs. The online documents are being continuously updated and improved, and the FI-WARE wiki will be the appropriate place to get the most up-to-date information on this GE.

2.2 User Guide

The NGSI-10 reference (OMA_NGSI-10) describes how to use the NGSI-10 API in detail. For using the IoT Broker you need to contact via HTTP the server on port 80 with one of the REST HTTP METHOD (GET, POST, PUT, DELETE) according to the NGSI-10 reference document.

In the following section it is detailed how to use the Iot Broker GE from a user or developer perspective.

2.3 Programmer Guide

The IoT Broker exposes an NGSI-10 interface, which is a RESTful interface over HTTP. This means that it is possible query the IoT Broker regardless of the programming language.

First, for checking if the IoT Broker GE is running and which operations are supported, is possible to send a GET on http://thingsman.lab.fi-ware.eu/ngsi10 .The response will be like this:

IoT Broker GE ver. 2.2.0 Running..

NGSI 10 REST INTERFACE - Operations Supported

GET - /contextEntities/{EntityId}

PUT - /contextEntities/{EntityId}

POST - /contextEntities/{EntityId}

DELETE - /contextEntities/{EntityId}

GET - /contextEntities/{EntityId}/attributes

PUT - /contextEntities/{EntityId}/attributes

POST - /contextEntities/{EntityId}/attributes

DELETE - /contextEntities/{EntityId}/attributes

GET - /contextEntities/{EntityId}/attributes/{attribute}

POST - /contextEntities/{EntityId}/attributes/{attribute}

DELETE - /contextEntities/{EntityId}/attributes/{attribute}

GET - /contextEntities/{EntityId}/attributes/{attribute}/{value}

PUT - /contextEntities/{EntityId}/attributes/{attribute}/{value}

GET - /contextEntities/{EntityId}/attributeDomains/{attributeDomain}

GET - /contextEntityTypes/{type}

GET - /contextEntityTypes/{type}/attributes

GET - /contextEntityTypes/{type}/attributeDomains/{attributeDomain}

POST - /contextQuery

POST - /updateContext

Documentation

FI-WARE IoT Broker GE OpenSpecification

FI-WARE IoT Broker GE Vision

FI-WARE IoT Broker GE User & Programmer Guideline

© Copyright NEC Europe Ltd 2013. All Rights Reserved.

Now for getting a feeling of how NGSI-10 works, let us send a GET request on http://thingsman.lab.fi-ware.eu/ngsi10/contextEntities/Kitchen (QueryContextRequest). The HTTP request and response headers are showed below:

```
<GET /ngsi10/contextEntities/Kitchen HTTP/1.1
<Host: localhost:80
<User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:13.0)
Gecko/20100101 Firefox/13.0.1
<Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
<Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
<Accept-Encoding: gzip, deflate
<Connection: keep-alive
>
<HTTP/1.1 200 OK
<Server: Apache-Coyote/1.1
<Content-Type: application/xml
<Transfer-Encoding: chunked
<Date: Tue, 10 Jul 2012 15:46:49 GMT
```

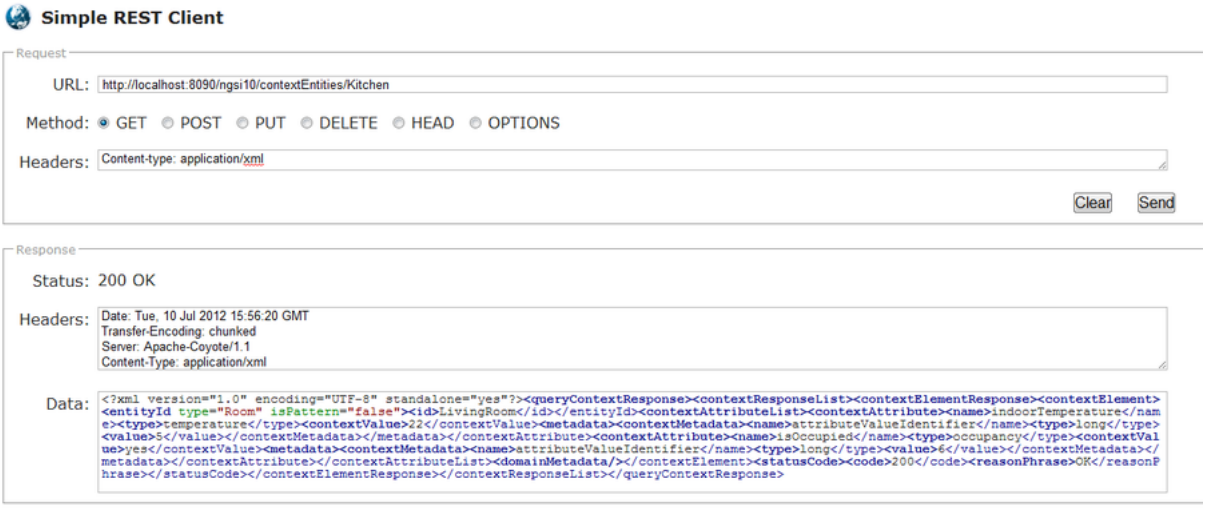
It is important to note that the NGSI-10 API, exposed by IoT Broker GE, required as Content-Type "application/xml", which means that an application can send only xml content and receive only xml as content.

2.3.1 Accessing the IoT Broker NGSI-10 Interface from a Browser

The following example interactions can be executed using the Chrome browser [1] with the Simple REST Client plugin [2] in order to send http commands to the IoT broker. You can use it also in Firefox through RESTClient add-ons [3].

We give two different example for the GET and POST request:

1. GET request:



2. POST request:

Simple REST Client

Request

URL: http://localhost:8090/ngsi10/queryContext

Method: GET POST PUT DELETE HEAD OPTIONS

Headers: Content-type: application/xml

Data:
```
<?xml version="1.0" encoding="UTF-8"?>
<queryContextRequest>
    <entityIdList>
        <entityId type='Room'>
            <id>Kitchen</id>
        </entityId>
    </entityIdList>
    <attributeList><attribute>indoorTemperature</attribute></attributeList>
</queryContextRequest>
```

Clear Send

Response

Status: 200 OK

Headers:
```
Date: Tue, 10 Jul 2012 16:32:27 GMT
Transfer-Encoding: chunked
Server: Apache-Coyote/1.1
Content-Type: application/xml
```

Data:
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><queryContextResponse><contextResponseList><contextElementResponse><contextElement>
<entityId type="Room" isPattern="false"><id>LivingRoom</id></entityId><contextAttributeList><contextAttribute><name>indoorTemperature</nam
e><type>temperature</type><contextValue>22</contextValue><metadata><contextMetadata><name>attributeValueIdentifier</name><type>long</type>
<value>5</value></contextMetadata></metadata></contextAttribute></contextAttributeList><domainMetadata/></contextElement><statusCode><code
>200</code><reasonPhrase>OK</reasonPhrase></statusCode></contextElementResponse></contextResponseList></queryContextResponse>
```

3 Configuration Manager GE - User and Programmers Guide

You can find the content of this chapter as well in the wiki of fi-ware.

3.1 Introduction

The Configuration Management component of the Things Management GE is the component responsible for the registration and discovery of entities. The executable is distributed as an RPM package and installed as /usr/bin/iotConfigMgr. iotConfigMgr uses a MySQL database as persistant storage and it is highly recommended for the MySQL server to run in the same node as the iotConfigMgr.

iotConfigMgr is a REST server implementing the FIWARE NGSI-9 interface, using the name FIWARE and not OMA to indicate that the interface used may have slight variations from what OMA publishes. In the first FIWARE release, the iotConfigMgr will support only two of the NGSI-9 requests, namely:

- registerContextRequest
- discoverContextRequest

3.2 User Guide

This component belong to the backend side, so it's meant to use by people with programmer profile. Therefore no content is provided for the user guide.

3.3 Programmer Guide

3.3.1 Starting iotConfigMgr

iotConfigMgr supports a number of command line options, the most important being:

- -u (print the complete usage on screen)
- -fg (run in foreground)
- -port (port to receive new connections)
- -reset (reset database at startup)
- -dbhost (host where the database server runs)
- -dbuser (username to login to database)
- -dbpwd (password to login to database)
- -db (name of the database)
- -psbHost (hostname for Pub/Sub Broker)
- -psbPort (port for Pub/Sub Broker)

The first time iotConfigMgr is started it will find an empty database and will then create all the necessary tables for it to function. A complete reset of the database is accomplished using the '-reset' option at starting iotConfigMgr. The '-fg' option is used when debugging iotConfigMgr, avoiding the executable to turn itself into a daemon process. To change the port where iotConfigMgr accepts incoming connections, the '-port' option is used. Starting iotConfigMgr with the '-u' options makes it print the full usage on the screen and after that it dies.

3.3.2 Database

Before starting iotConfigMgr, a MySQL server need to be running and the database to be used must be created and configured to that the database user of iotConfigMgr has access to it. All this is explained in the 'Installation and Administration Guide'.

The name of the database is 'cm' by default and this name is changed using the command line option '-db'.

The name of the database server is 'localhost' by default and this name is changed using the command line option '-dbhost'.

The name of the database user is 'cm' by default and this name is changed using the command line option '-dbuser'.

The name of the database password is 'cm' by default and this name is changed using the command line option '-dbpwd'.

The database tables used are the following:

- attribute
- attributeMetadata
- entity
- entityAttribute
- metadata
- registration
- registrationMetadata

To enter mysql and check what's in the database, the following mysql client command is used:

```
% mysql -u cm -p cm
password> cm
mysql> SELECT * FROM entity;    # or any of the other tables
...
```

3.3.3 Registering entities

To register an entity, the NSGI-9 REST request 'registerContextRequest' is used.

3.3.3.1 ***Request***

The REST path of a register request has the form (example using curl):

```
% curl host:port/ngsi9/registerContext --request POST --header
'Content-Type: text/xml' -d <data>
```

The data part of a register request is in XML form and the structure of the data is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<registerContextRequest>
  <contextRegistrationList>
```

```
      <contextRegistration>
        <entityIdList>
          <entityId type="" isPattern="false/true">
            <id></id>
          </entityId>
          <entityId type="" isPattern="false/true">
            <id></id>
          </entityId>
        </entityIdList>
        <contextRegistrationAttributeList>
          <contextRegistrationAttribute>
            <name></name>
            <type></type>
            <isDomain></isDomain>
            <metaData>
              <contextMetadata>
                <name></name>
                <type></type>
                <value></value>
              </contextMetadata>
            </metaData>
          </contextRegistrationAttribute>
        </contextRegistrationAttributeList>
        <registrationMetaData>
          <contextMetadata>
            <name></name>
            <type></type>
            <value></value>
          </contextMetadata>
        </registrationMetaData>
        <providingApplication>URI</providingApplication>
      </contextRegistration>
    </contextRegistrationList>
    <duration></duration>
    <registrationId></registrationId>
  </registerContextRequest>
```

This request is used both to register entities and to update entites already registered. When updating an entity, the 'registrationId' field must be filled in with the Registration

Id that was output in the entity's initial registration. If updating more than one entity, all entities must have been registered together initially, otherwise the attempt will fail (as some of the entities have an incorrect registration id). In this case, no update will be done, not even to the entites that match the registration id.

If the request is sent with empty registrationId, the iotConfigMgr will see this an an attempt to add new entities and if any of the entities in the request exist, the entire request will fail - nothing will be done.

When updating entities, attributes can be added to the entity, or metadata can be added to an already existing attribute of an entity. Also, registration metadata can be added (the registration already existed, of course - otherwise the update request would fail).

3.3.3.2 *Response*

The response of a registerContextRequest is an XML document of the following form:

```
<?xml version="1.0" encoding="UTF-8"?>
<registerContextResponse>
  <duration></duration>
  <registrationId>REGISTRATION_ID</registrationId>
</registerContextResponse>
```

Very important here to save the REGISTRATION_ID together with the registered entites as this registration id must be used in consequent updates of the entities registered. The duration is not mandatory in the response, but if it is present, it simply reflects the duration from the request corresponding to the response.

3.3.4 Discovering Entities

To discover entities, the NSGI-9 REST request 'discoverContextRequest' is used.

3.3.4.1 *Request*

The REST path of a discovery request has the form (example using curl):

```
% curl host:port/ngsi9/discoverContextAvailability --request
POST --header 'Content-Type: text/xml' -d <data>
```

The data part of a register request is in XML form and the structure of the data is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<discoverContextAvailabilityRequest>
  <entityIdList>
    <entityId type="" isPattern="true/false">
      <id></id>
    </entityId>
    <entityId type="" isPattern="true/false">
```

```
        <id></id>
      </entityId>
    </entityIdList>
    <attributeList>
      <attribute>temperature</attribute>
      <attribute>occupancy</attribute>
      <attribute>lightstatus</attribute>
    </attributeList>
    <restriction>
      <attributeExpression></attributeExpression>
      <scope>
        <operationScope>
          <scopeType></scopeType>
          <scopeValue></scopeValue>
        </operationScope>
        <operationScope>
          <scopeType></scopeType>
          <scopeValue></scopeValue>
        </operationScope>
      </scope>
    </restriction>
  </discoverContextAvailabilityRequest>
```

An entity is identified by the combination of its type and id.

3.3.4.2 *Response*

The response of a discoverContextAvailabilityRequest is an XML document of the following form:

```
<?xml version="1.0" encoding="UTF-8"?>
<discoverContextAvailabilityResponse>
  <contextRegistrationResponseList>
    <contextRegistrationResponse>
      <contextRegistration>
        <entityIdList>
          <entityId type="Room" isPattern="false">
            <id>OfficeRoom</id>
          </entityId>
          <entityId type="Room" isPattern="false">
            <id>OfficeRoom</id>
```

```
            </entityId>
          </entityIdList>
          <contextRegistrationAttributeList>
            <contextRegistrationAttribute>
              <name>temperature</name>
              <type>degree</type>
              <isDomain>false</isDomain>
              <metaData>
                <contextMetadata>
                  <name>ID</name>
                  <type>string</type>
                  <value>1110</value>
                </contextMetadata>
                <contextMetadata>
                  <name></name>
                  <type></type>
                  <value></value>
                </contextMetadata>
              </metaData>
            </contextRegistrationAttribute>
          </contextRegistrationAttributeList>
          <registrationMetaData>
            <contextMetadata>
              <name>ID</name>
              <type>string</type>
              <value>2212</value>
            </contextMetadata>
            <contextMetadata>
              <name></name>
              <type></type>
              <value></value>
            </contextMetadata>
          </registrationMetaData>

<providingApplication>http://192.168.100.1:70/application
          </providingApplication>
        </contextRegistration>
      </contextRegistrationResponse>
```

```
    </contextRegistrationResponseList>
    <errorCode>
      200
      <reasonPhrase>Ok</reasonPhrase>
      <details>a</details>
    </errorCode>
  </discoverContextAvailabilityResponse>
```

On failure, the entire contextRegistrationResponseList part is omitted.

On success, the errorCode part is omitted (here I have a bug in iotConfigMgr - on success I reply with both parts ...).

3.3.5 Connecting to iotConfigMgr

This executable is implemented in 'C', and this code example to connect to it is in 'C' as well. It is easy enough to find connect examples in any language.

```
 int serverConnect(const char* host, unsigned short port)
 {
   int                 fd;
   struct hostent*     hp;
   struct sockaddr_in  peer;
   if ((hp = gethostbyname(host)) == NULL)
   {
     printf("gethostbyname(%s): %s\n", host, strerror(errno));
     return -1;
   }
   if ((fd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
   {
     printf("socket: %s\n", strerror(errno));
     return -1;
   }
   memset((char*) &peer, 0, sizeof(peer));
   peer.sin_family      = AF_INET;
   peer.sin_addr.s_addr = ((struct in_addr*) (hp->h_addr))-
>s_addr;
   peer.sin_port        = htons(port);
   if (connect(fd, (struct sockaddr*) &peer, sizeof(peer)) == -
1)
   {
     close(fd);
```

```
     printf("connect(%s, %d): %s\n", host, port,
strerror(errno));
     return -1;
   }
   return fd;
 }
```

3.3.6 Sending requests to iotConfigMgr

In a 'C' program, a normal 'write' is used to send REST commands to the iotConfigMgr. This 'write' will need to follow the REST standard but it is out of the scope of this document to explain REST in detail. From command line, using the tool 'curl', this is how to interact with iotConfigMgr, Examples of this are found above, both for 'register' and 'discover'.

To update a registration, the registration identifier that was output in the initial regiostration must be provided, otherwise the operation will fail.

3.3.7 Debugging iotConfigMgr

The iotConfigMgr executable maintains a log file at /tmp/iotConfigMgrLog. The more verbose/trace asked at starting iotConfigMgr, the more content the logfile will have. There are a number of command line options to turn on verbosity and trace levels:

- -v
- -vv
- -vvv
- -vvvv
- -vvvvv
- -t <t1-t2,t3-t4>
- -r
- -w

All these verbose/trace levels can also be altered at runtime, using a REST interface:

```
curl --request PUT    host:port/log/verbose/set/3
curl --request PUT    host:port/log/trace/set/0-5,19.34
curl --request PUT    host:port/log/reads/on
curl --request DELETE host:port/log/reads
```

To debug the internal lists of iotConfigMgr, another REST request is defined:

```
curl host:port/debug/entity
curl host:port/debug/registration
curl host:port/debug/attribute
curl host:port/debug/metadata
```

```
curl host:port/debug/regMetadata
```

The output of these commands go to the logfile (/tmp/iotConfigMgrLog).

A useful command to view the logfile:

```
% tail -f /tmp/iotConfigMgrLog
```

4 Gateway Data Handling GE - User and Programmers Guide

You can find the content of this chapter as well in the wiki of fi-ware.

- Gateway Data Handling - Esper4FastData Servlet - User and Programmers Guide
- Gateway Data Handling - Esper4FastData Mobile - User and Programmers Guide
- Gateway Data Handling - SOL CEP User and Programmers Guide

OSGI is a specific environment for Esper4FastData Servlet. It makes sense to create a specific installation guide for the Esper4FastData Servlet environment and the Unit test plan. The use takes place via the servlet part, the User Manual makes sense just for this one.

5 Gateway Data Handling - Esper4FastData Servlet - User and Programmers Guide

You can find the content of this chapter as well in the wiki of fi-ware.

5.1.1 Introduction

5.1.1.1 *Events overview*

Basically, an event is just « something that happens ».

In philosophy, events are objects in time, or instantiations of properties in objects, whereas in computing, they are action that are usually initiated outside the scope of a program and that is handled by a piece of code inside the program.

Complex Event Processing(CEP) is a usual solution for the following questions :

- How to handle massively growing data volumes ?
- How to make meanings of all events flowing through your system at the speed of your business ?
- How to preserve flexibility ?

5.1.1.2 *Value-added of a CEP system*

It takes subsequent action in real time, delivers high-speed processing of many events, operates across all the layers of an organization. It also filters the most meaningful events, merges data from many events and has to deal with event privacy and subscribing.

5.1.1.3 *The Esper CEP library*

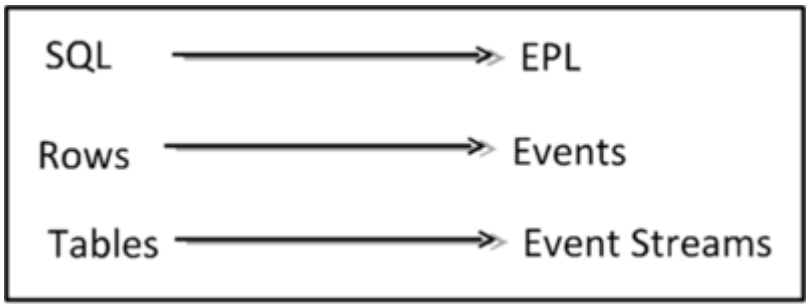
Esper4FastData engine is based upon the open-source Esper library. There is a very active community of developpers behind it.

Here are some technical facts about it :

- Is available under GPL v2
- Is available as a Java library (jar)
- Has been adapted to Android
- Has tiny footprint

5.1.1.4 *Esper Concepts*

Esper is like a database that has been turned upside-down, every database concept having its counterpart in the CEP world :



The Esper library is based on the Event Processing Language(EPL), that looks roughly like SQL. But the "querying" model is completely different, because CEP provides a continuous model of querying. This could be named "real-time data mining", in contrary to "store-now query-ater" model, which is equivalent to analyzing historical data.

Despite the syntax similarities between EPL and SQL, CEP is definitely not a database replacement.

Esper uses events windows to store event streams, which can be compared to tables in the database world. These windows have a predefined size, in terms of data volume, or retention time. They are in fact sliding windows that contain the events flowing THROUGH the EPL statement. This is a key point to understand the difference between a database and a CEP :

- In database world, one can instanciate an SQL statement to SYNCHRONOUSLY retrieve particular data
- In the CEP world it's the data that ASYNCHRONOUSLY flows THROUGH an EPL statement, which is trigerred only if properties criteria are fulfiled

5.1.1.5 ***Basic EPL statements and sliding events windows***

5.1.1.5.1 *Filter-free statement*

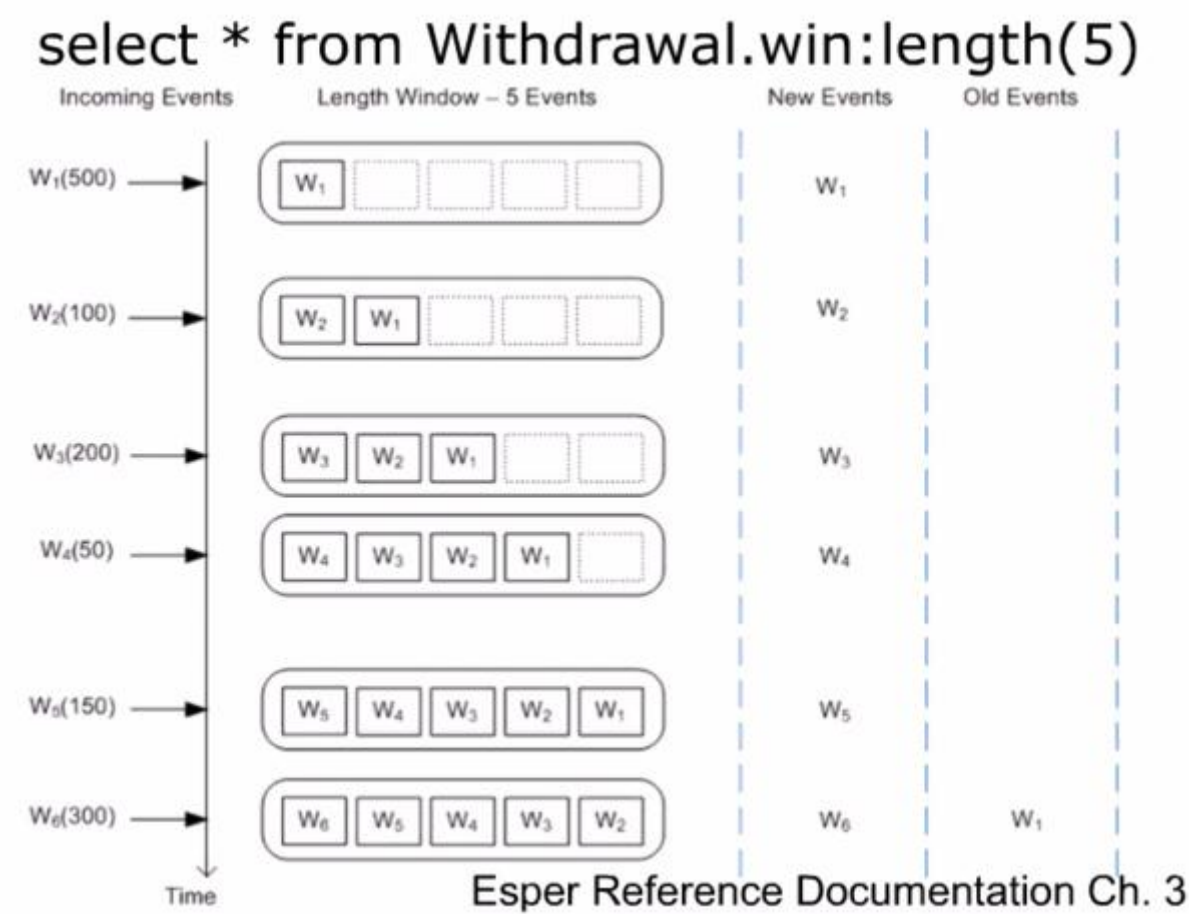
A length window instructs the engine to only keep the last N events for a stream. The next statement applies a length window onto the Withdrawal event stream. The statement serves to illustrate the concept of data window and events entering and leaving a data window:

select * from Withdrawal.win:length(5)

The size of this statement's length window is five events. The engine enters all arriving Withdrawal events into the length window. When the length window is full, the oldest Withdrawal event is pushed out the window. The engine indicates to listeners all events entering the window as new events, and all events leaving the window as old events.

While the term insert stream denotes new events arriving, the term remove stream denotes events leaving a data window, or changing aggregation values. In this example, the remove stream is the stream of Withdrawal events that leave the length window, and such events are posted to listeners as old events.

The next diagram illustrates how the length window contents change as events arrive and shows the events posted to an update listener :

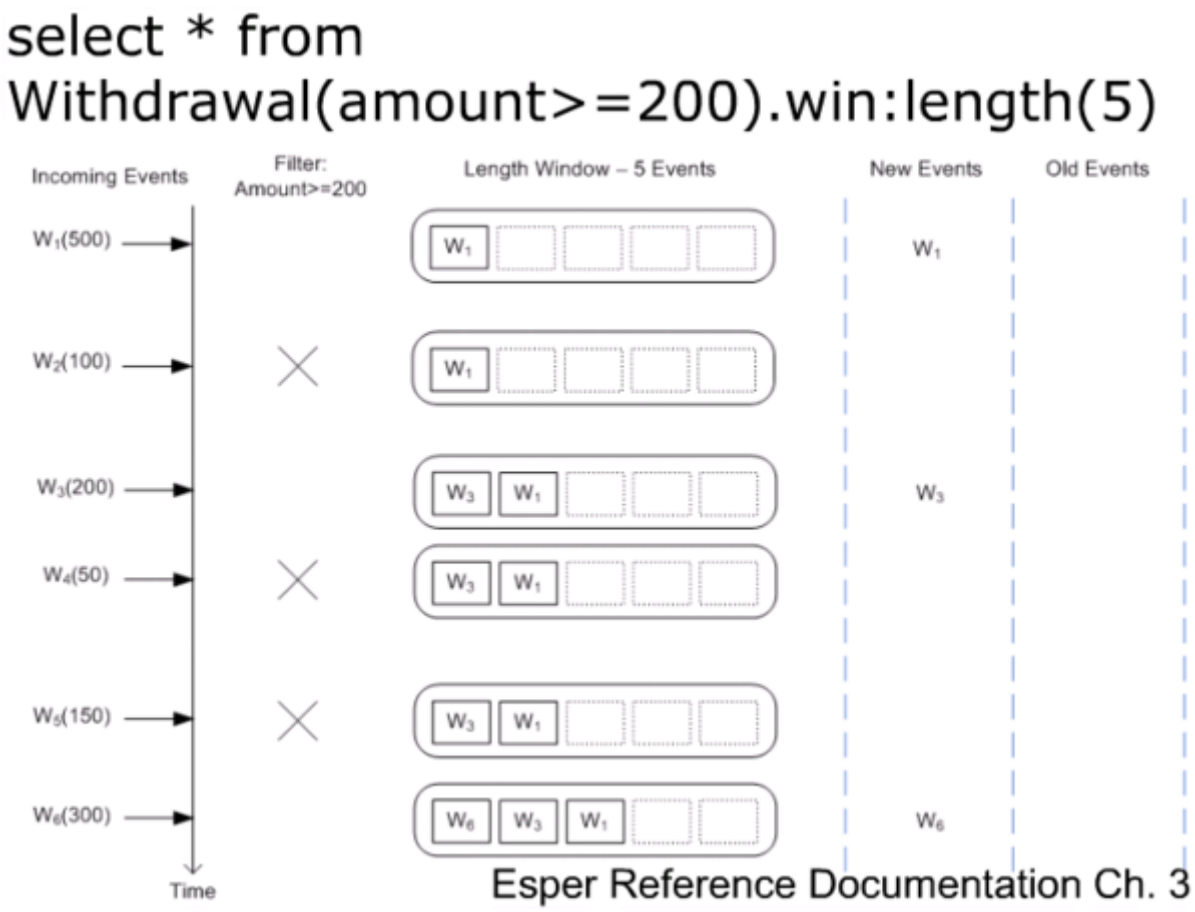


5.1.1.5.2 *Input filter statement*

Filters to event streams allow filtering events out of a given stream before events enter a data window. The statement below shows a filter that selects Withdrawal events with an amount value of 200 or more.

select * from Withdrawal(amount>=200).win:length(5)

With the filter, any Withdrawal events that have an amount of less then 200 do not enter the length window and are therefore not passed to update listeners :



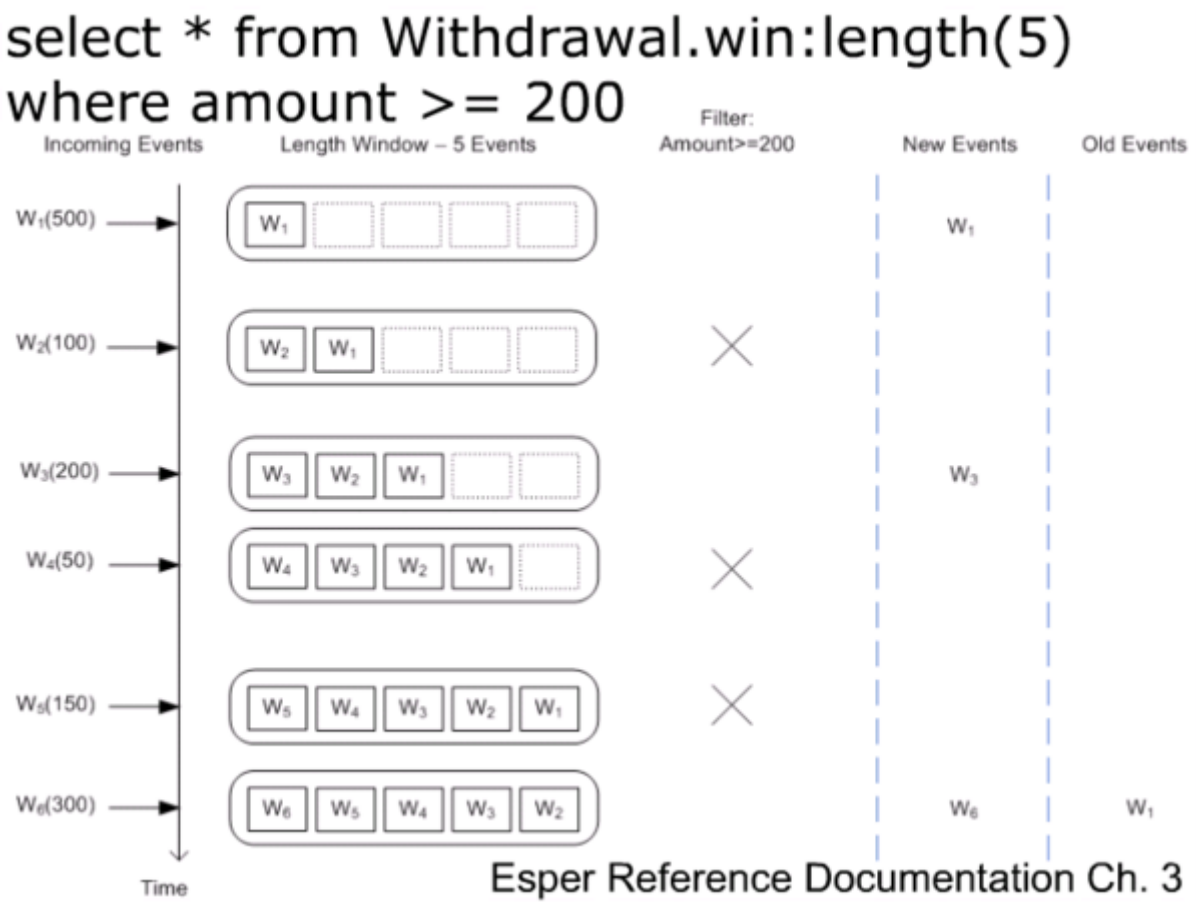
5.1.1.5.3 *Output filter statement*

The where-clause and having-clause in statements eliminate potential result rows at a later stage in processing, after events have been processed into a statement's data window or other views.

The next statement applies a where-clause to Withdrawal events.

select * from Withdrawal.win:length(5) where amount >= 200

The where-clause applies to both new events and old events. As the diagram below shows, arriving events enter the window however only events that pass the where-clause are handed to update listeners. Also, as events leave the data window, only those events that pass the conditions in the where-clause are posted to listeners as old events.



5.1.1.6 *Detailed EPL Reference*

See http://esper.codehaus.org/esper-4.6.0/doc/reference/en-US/html/epl_clauses.html

5.1.1.7 *Glossary of terms*

- Event: anything that happens, or is contemplated as happening
- Event Object, event message: an object that represents, encore or records an event, generally for the purpose of computer processing
- Event Type: a class of event objects. All events must be instances of an event type. An event has the structure defined by its type. Event types should be defined with a XML Schema Definition (XSD file)
- Event attribute or event property: a component of the structure of an event. An event attribute can have a simple or complex data type.
- Event processing: computing that performs operations on events, including reading, creating, transforming and deleting events.
- Timestamp: a time value attribute of an event. The time in which the event was created:creation time or observed:arrival time.
- Complex event processing (CEP): computing that performs operations on complex events, including reading, creating, transforming or abstracting them.
- Statement or event processing rules: a prescribed method for processing event. Event processing rules are described in Event Processing Language

- Event Processing Language: a high level computer language for defining the behavior of event processing agents

5.2 User and Programmer Guide

Welcome the Esper4FastData Servlet User and Programmer Guide. This component is a part of the Data Handling Generic Enabler and is a Rest Web service project using standard interface NGSI 9/10 to comunicate with the others components/GE. The following sections explains how to use the Esper4FastData Servlet as a user or developer. The Gateway Esper4FastData Servlet exposes methods to manage the CEP Engine which is a RESTful API via HTTP.

5.2.1 Accessing the Esper4FastData Servlet Interface from a Browser

The following example interactions can be executed using the Chrome browser [1] with the Simple REST Client plugin [2] in order to send http commands to the CEP Manager. You can use it also in Firefox through RESTClient add-ons [3].

All examples use the Chrome browser

5.2.2 database methods

5.2.2.1 *reset*

$rootService/database/reset
POST
URL : http://130.206.81.67/Esper4FastData/database/reset
NO payload
Result

```
<informationMessage>
   <code>200</code>
   <message>OK</message>
</informationMessage>
```

5.2.3 ngsi methods

5.2.3.1 *Standard NGSI-9 operation resource: register Context*

$rootService/ngsi9/registerContext
POST
URL : http://130.206.81.67/Esper4FastData/ngsi9/registerContext
Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<registerContextRequest>
    <contextRegistrationList>
```

```
        <contextRegistration>
            <entityIdList>
                <entityId type="VehicleLocation"
isPattern="false">
                <id>Plane</id>
                </entityId>
                <entityId type="VehicleLocation"
isPattern="false">
                <id>Van1</id>
                </entityId>
                <entityId type="VehicleLocation"
isPattern="false">
                <id>Van2</id>
                </entityId>
            </entityIdList>
        <contextRegistrationAttributeList>
        <contextRegistrationAttribute>
            <name>latitude</name>
            <type>xs:double</type>
            <isDomain>false</isDomain>
        </contextRegistrationAttribute>
        <contextRegistrationAttribute>
            <name>longitude</name>
            <type>xs:double</type>
            <isDomain>false</isDomain>
        </contextRegistrationAttribute>
        </contextRegistrationAttributeList>
        <registrationMetaData>
            <contextMetadata>
                <name>timestamp</name>
                <type>xs:dateTime</type>
                <value/>
            </contextMetadata>
        </registrationMetaData>

<providingApplication>http://130.206.81.67/CEP2ManagerService/ng
si</providingApplication>
        </contextRegistration>
    </contextRegistrationList>
```

```
    <duration>P5D</duration>
    <registrationId/>
</registerContextRequest>
```

Result

```
<registerContextResponse>
   <duration>P5D</duration>
   <registrationId/>
   <errorCode>
      <code>200</code>
      <reasonPhrase>OK</reasonPhrase>
   </errorCode>
</registerContextResponse>
```

5.2.3.2 *NGSI-9 Convenience Operation ressources: registerContext*

Individual Context entity $rootService/ngsi9/contextEntities/{entityId}

POST

URL : http://130.206.81.67/Esper4FastData/ngsi9/contextEntities/Plane

Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<registerContextRequest>
    <contextRegistrationList>
        <contextRegistration>
            <entityIdList>
                <entityId type="VehicleLocation"
isPattern="false">
                    <id>Plane</id>
                </entityId>
            </entityIdList>
            <contextRegistrationAttributeList>
                <contextRegistrationAttribute>
                    <name>latitude</name>
                    <type>xs:double</type>
                    <isDomain>false</isDomain>
                </contextRegistrationAttribute>
                <contextRegistrationAttribute>
                    <name>longitude</name>
```

```
                    <type>xs:double</type>
                    <isDomain>false</isDomain>
                </contextRegistrationAttribute>
            </contextRegistrationAttributeList>
            <registrationMetaData>
                <contextMetadata>
                    <name>timestamp</name>
                    <type>xs:dateTime</type>
                    <value/>
                </contextMetadata>
            </registrationMetaData>

<providingApplication>http://130.206.81.67/CEPManagerService2/ng
si</providingApplication>
        </contextRegistration>
    </contextRegistrationList>
    <duration>P5D</duration>
    <registrationId/>
</registerContextRequest>
```

Result

```
<registerContextResponse>
   <duration>P5D</duration>
   <registrationId/>
   <errorCode>
      <code>200</code>
      <reasonPhrase>OK</reasonPhrase>
   </errorCode>
</registerContextResponse>
```

5.2.3.3 *Standard NGSI-10 operation resource: updateContext*

$rootService/ngsi10/updateContext

POST

URL : http://130.206.81.67/Esper4FastData/ngsi10/updateContext

Payload

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<updateContextRequest>
    <contextElementList>
```

```
    <contextElement>
        <entityId type="VehicleLocation" isPattern="false">
            <id>Plane</id>
        </entityId>

<attributeDomainName>VehicleLocation</attributeDomainName>
        <contextAttributeList>
            <contextAttribute>
                <name>latitude</name>
                <type>double</type>
                <contextValue>43.660266</contextValue>
            </contextAttribute>
            <contextAttribute>
                <name>longitude</name>
                <type>double</type>
                <contextValue>7.205032</contextValue>
            </contextAttribute>
        </contextAttributeList>
        <domainMetadata>
        <contextMetadata>
            <name>timestamp</name>
            <type>xsd:dataTime</type>
            <value>20130114T093810+0100</value>
        </contextMetadata>
        </domainMetadata>
    </contextElement>
    </contextElementList>
    <updateAction>UPDATE</updateAction>
</updateContextRequest>
```

Result

```
<updateContextResponse>
   <errorCode>
      <code>200</code>
      <reasonPhrase>OK</reasonPhrase>
   </errorCode>
   <contextResponseList>
      <contextElementResponse>
```

```
         <contextElement>
            <entityId isPattern="false" type="VehicleLocation">
               <id>Plane</id>
            </entityId>

<attributeDomainName>VehicleLocation</attributeDomainName>
            <contextAttributeList>
               <contextAttribute>
                  <name>latitude</name>
                  <type>double</type>
                  <contextValue>43.660266</contextValue>
               </contextAttribute>
               <contextAttribute>
                  <name>longitude</name>
                  <type>double</type>
                  <contextValue>7.205032</contextValue>
               </contextAttribute>
            </contextAttributeList>
            <domainMetadata>
               <contextMetadata>
                  <name>timestamp</name>
                  <type>xsd:dataTime</type>
                  <value>20130114T093810+0100</value>
               </contextMetadata>
            </domainMetadata>
         </contextElement>
         <statusCode>
            <code>200</code>
            <reasonPhrase>OK</reasonPhrase>
         </statusCode>
      </contextElementResponse>
   </contextResponseList>
</updateContextResponse>
```

5.2.3.4 *NGSI-10 Convenience Operation ressources: update Context*

Individual Context entity
$rootService/ngsi/contextEntities/{entityId}/{type}
PUT

URL : http://130.206.81.67/Esper4FastData/ngsi9/contextEntities/Van1/VehicleLocation

Payload

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<updateContextElementRequest>
    <attributeDomainName>VehicleLocation</attributeDomainName>
    <contextAttributeList>
                <contextAttribute>
                   <name>latitude</name>
                   <type>double</type>
                   <contextValue>53.660266</contextValue>
                </contextAttribute>
                <contextAttribute>
                   <name>longitude</name>
                   <type>double</type>
                   <contextValue>7.205032</contextValue>
                </contextAttribute>
    </contextAttributeList>
    <domainMetadata>
                <contextMetadata>
                   <name>timestamp</name>
                   <type>xsd:dataTime</type>
                   <value>20130301T093810+0100</value>
                </contextMetadata>
    </domainMetadata>
</updateContextElementRequest>
```

Result

```
<updateContextResponse>
   <errorCode>
      <code>200</code>
      <reasonPhrase>OK</reasonPhrase>
   </errorCode>
   <contextResponseList>
      <contextElementResponse>
         <contextElement>
            <entityId type="VehicleLocation">
               <id>Van1</id>
```

```
            </entityId>

<attributeDomainName>VehicleLocation</attributeDomainName>
            <contextAttributeList>
               <contextAttribute>
                  <name>latitude</name>
                  <type>double</type>
                  <contextValue>53.660266</contextValue>
               </contextAttribute>
               <contextAttribute>
                  <name>longitude</name>
                  <type>double</type>
                  <contextValue>7.205032</contextValue>
               </contextAttribute>
            </contextAttributeList>
            <domainMetadata>
               <contextMetadata>
                  <name>timestamp</name>
                  <type>xsd:dataTime</type>
                  <value>20130301T093810+0100</value>
               </contextMetadata>
            </domainMetadata>
         </contextElement>
         <statusCode>
            <code>200</code>
            <reasonPhrase>OK</reasonPhrase>
         </statusCode>
      </contextElementResponse>
   </contextResponseList>
</updateContextResponse>
```

$rootService/ngsi10/contextEntities/{entityId}/{type}
POST
URL : http://130.206.81.67/Esper4FastData/ngsi10/contextEntities/Parcel-0/ParcelStep
Payload

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<appendContextElementRequest>
    <attributeDomainName>ParcelStep</attributeDomainName>
    <contextAttributeList>
```

```
        <contextAttribute>
            <name>vehicleId</name>
            <type>string</type>
            <contextValue>Van2</contextValue>
        </contextAttribute>
        <contextAttribute>
            <name>deliveryStep</name>
            <type>string</type>
            <contextValue>Delivered</contextValue>
        </contextAttribute>
    </contextAttributeList>
    <domainMetadata>
        <contextMetadata>
            <name>timestamp</name>
            <type>xsd:dataTime</type>
            <value>20130301T193032+0100</value>
        </contextMetadata>
    </domainMetadata>
</appendContextElementRequest>
```

Result

```
<updateContextResponse>
   <errorCode>
      <code>200</code>
      <reasonPhrase>OK</reasonPhrase>
   </errorCode>
   <contextResponseList>
      <contextElementResponse>
         <contextElement>
            <entityId type="VehicleLocation">
               <id>Van2</id>
            </entityId>

<attributeDomainName>ParcelStep</attributeDomainName>
            <contextAttributeList>
               <contextAttribute>
                  <name>vehicleId</name>
                  <type>string</type>
```

```
                    <contextValue>Van2</contextValue>
                </contextAttribute>
                <contextAttribute>
                    <name>deliveryStep</name>
                    <type>string</type>
                    <contextValue>Delivered</contextValue>
                </contextAttribute>
            </contextAttributeList>
            <domainMetadata>
                <contextMetadata>
                    <name>timestamp</name>
                    <type>xsd:dataTime</type>
                    <value>20130301T193032+0100</value>
                </contextMetadata>
            </domainMetadata>
        </contextElement>
        <statusCode>
            <code>200</code>
            <reasonPhrase>OK</reasonPhrase>
        </statusCode>
    </contextElementResponse>
  </contextResponseList>
</updateContextResponse>
```

5.2.3.5 *Standard NGSI-10 operation resource: query Context*

$rootService/ngsi10/queryContext

POST

URL : http://130.206.81.67/Esper4FastData/ngsi10/queryContext

Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<queryContextRequest>
    <entityIdList>
        <entityId type="VehicleLocation" isPattern="false">
        <id>Plane</id>
        </entityId>
    </entityIdList>
    <attributeList>
        <attribute>latitude</attribute>
```

```
        <attribute>longitude</attribute>
    </attributeList>
    <restriction>
        <attributeExpression></attributeExpression>
        <scope>
        <operationScope>
            <scopeType></scopeType>
            <scopeValue></scopeValue>
        </operationScope>
        <operationScope>
            <scopeType></scopeType>
            <scopeValue></scopeValue>
        </operationScope>
        </scope>
    </restriction>
</queryContextRequest>
```

Result

```
<queryContextResponse>
   <contextResponseList>
      <contextElementResponse>
         <contextElement>
            <entityId isPattern="false" type="VehicleLocation">
               <id>Plane</id>
            </entityId>

<attributeDomainName>VehicleLocation</attributeDomainName>
            <contextAttributeList>
               <contextAttribute>
                  <name>latitude</name>
                  <type>double</type>
                  <contextValue>43.660266</contextValue>
                  <metadata/>
               </contextAttribute>
               <contextAttribute>
                  <name>longitude</name>
                  <type>double</type>
                  <contextValue>7.205032</contextValue>
```

```
                    <metadata/>
                 </contextAttribute>
              </contextAttributeList>
              <domainMetadata>
                 <contextMetadata>
                    <name>timestamp</name>
                    <type>xsd:dataTime</type>
                    <value>20130114T093810+0100</value>
                 </contextMetadata>
              </domainMetadata>
           </contextElement>
           <statusCode>
              <code>0</code>
           </statusCode>
        </contextElementResponse>
     </contextResponseList>
     <errorCode>
        <code>0</code>
     </errorCode>
</queryContextResponse>
```

5.2.3.6 *NGSI-10 Convenience Operation ressources: queryContext*

Individual Context entity
$rootService/ngsi/contextEntities/{entityId}
GET
URL : http://130.206.81.67/Esper4FastData/ngsi10/contextEntities/Van2
Result

```
<queryContextResponse>
   <contextResponseList>
      <contextElementResponse>
         <contextElement>
            <entityId isPattern="false" type="VehicleLocation">
               <id>Van2</id>
            </entityId>

<attributeDomainName>VehicleLocation</attributeDomainName>
            <contextAttributeList>
               <contextAttribute>
```

```
                    <name>latitude</name>
                    <type>double</type>
                    <contextValue>43.734391</contextValue>
                    <metadata/>
                 </contextAttribute>
                 <contextAttribute>
                    <name>longitude</name>
                    <type>double</type>
                    <contextValue>7.184658</contextValue>
                    <metadata/>
                 </contextAttribute>
              </contextAttributeList>
              <domainMetadata>
                 <contextMetadata>
                    <name>timestamp</name>
                    <type>xsd:dataTime</type>
                    <value>20130301T193032+0100</value>
                 </contextMetadata>
              </domainMetadata>
           </contextElement>
           <statusCode>
              <code>0</code>
           </statusCode>
        </contextElementResponse>
     </contextResponseList>
     <errorCode>
        <code>0</code>
     </errorCode>
</queryContextResponse>
```

Attribute container of individual Context entity $rootService/ngsi10/contextEntities/{entityId}/attributes

GET

URL : http://130.206.81.67/Esper4FastData/ngsi10/contextEntities/parcel-0/attributes

Result

```
<contextAttributeResponse>
   <contextAttributeList>
      <contextAttribute>
         <name>vehicleId</name>
```

```
            <type>string</type>
            <contextValue>Plane</contextValue>
            <metadata/>
        </contextAttribute>
        <contextAttribute>
            <name>deliveryStep</name>
            <type>string</type>
            <contextValue>DeliveryInProgress</contextValue>
            <metadata/>
        </contextAttribute>
    </contextAttributeList>
    <statusCode>
        <code>0</code>
    </statusCode>
</contextAttributeResponse>
```

Attribute of individual Context entity $rootService/ngsi10/contextEntities/{entityId}/attributes/{attName}

GET

URL : http://130.206.81.67/Esper4FastData/ngsi10/contextEntities/parcel-0/attributes/deliveryStep

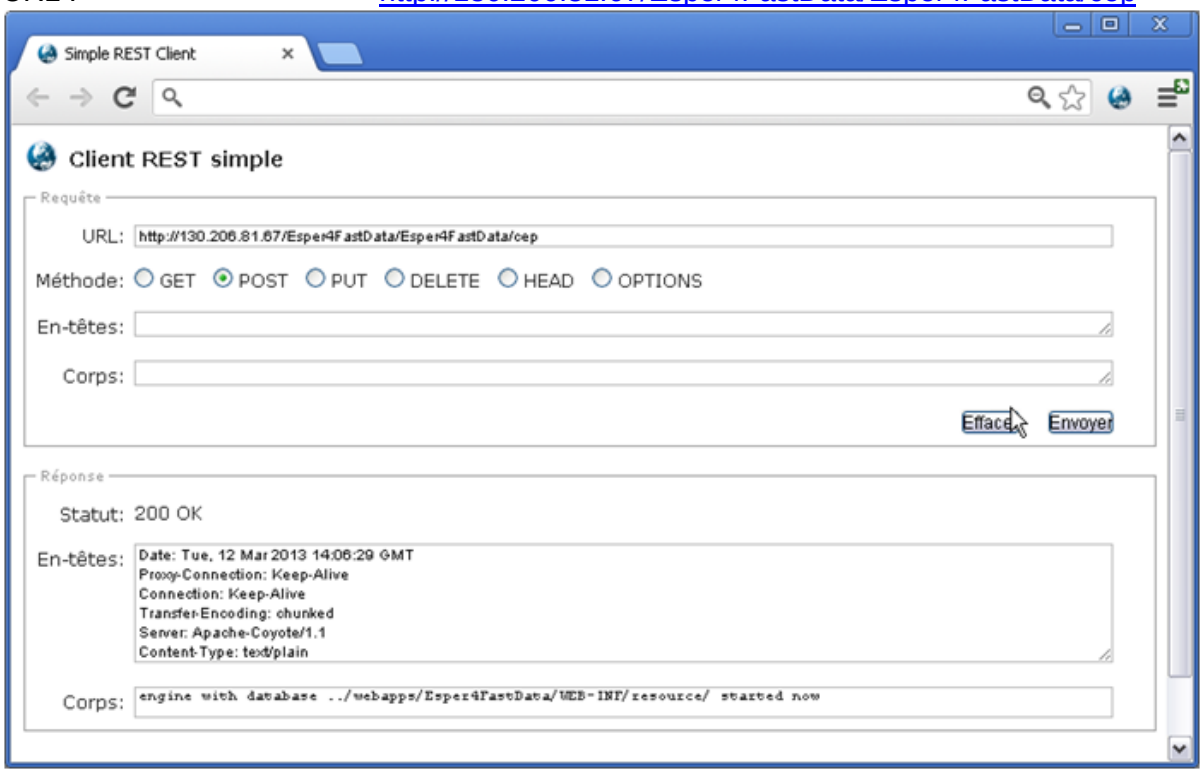
Result

```
<contextAttributeResponse>
    <contextAttributeList>
        <contextAttribute>
            <name>deliveryStep</name>
            <type>string</type>
            <contextValue>DeliveryInProgress</contextValue>
            <metadata/>
        </contextAttribute>
    </contextAttributeList>
    <statusCode>
        <code>0</code>
    </statusCode>
</contextAttributeResponse>
```

5.2.4 Engine methods

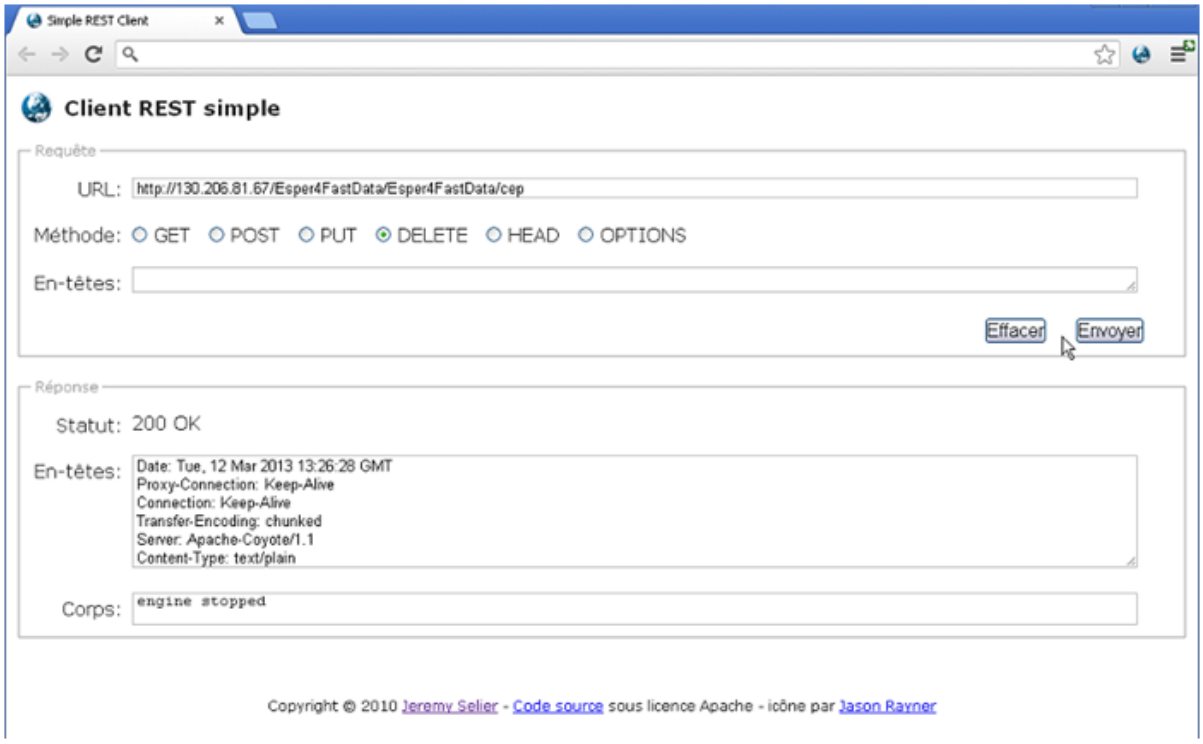
5.2.4.1 *Start CEP*

Start the CEP engine if it is not already started the engine must be started to call all Esper4FastData methods: adding eventType, statements…
$rootService/Esper4FastData/cep
POST
URL : http://130.206.81.67/Esper4FastData/Esper4FastData/cep



5.2.4.2 *Stop CEP*

Stop the running CEP engine a stopped engine has no more event Types and statements defined
$rootService/Esper4FastData/cep
DELETE
URL : http://130.206.81.67/Esper4FastData/Esper4FastData/cep



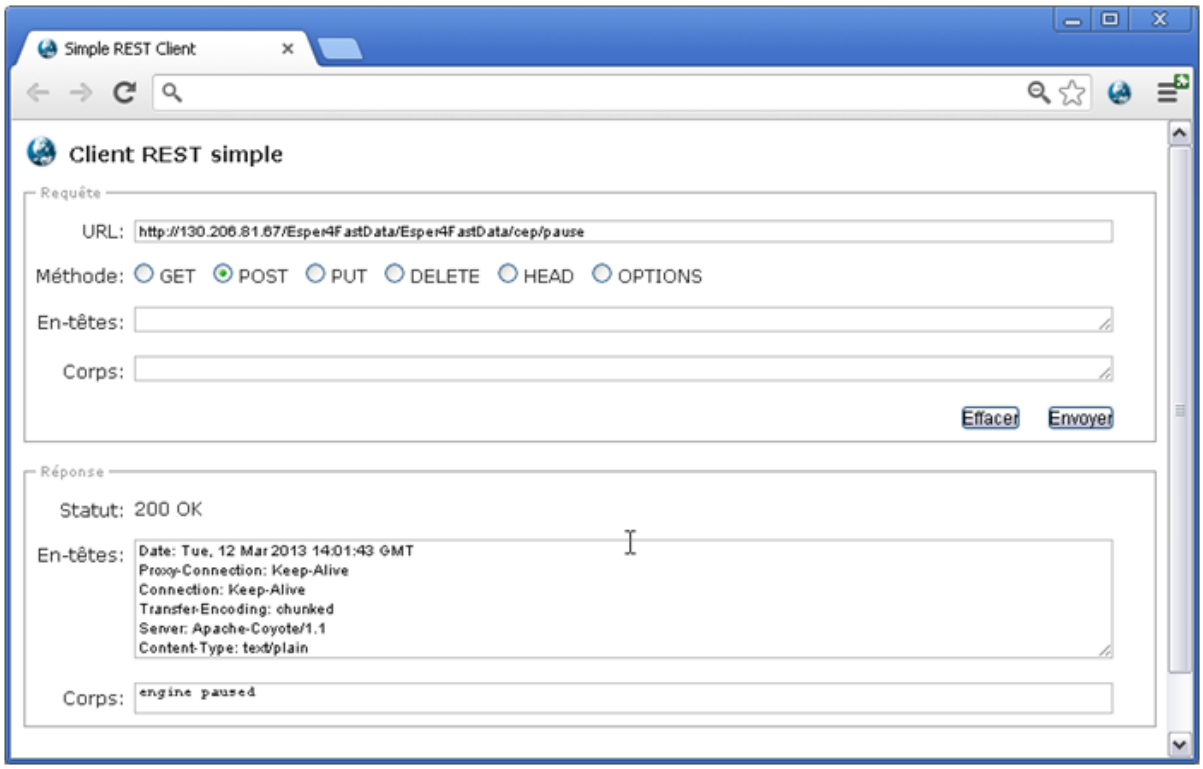
5.2.4.3 *Pause CEP*

Pause the running CEP engine Paused engine keeps eventTypes and statements defined

$rootService/Esper4FastData/cep/{cmd}

POST

URL : http://130.206.81.67/Esper4FastData/Esper4FastData/cep/pause



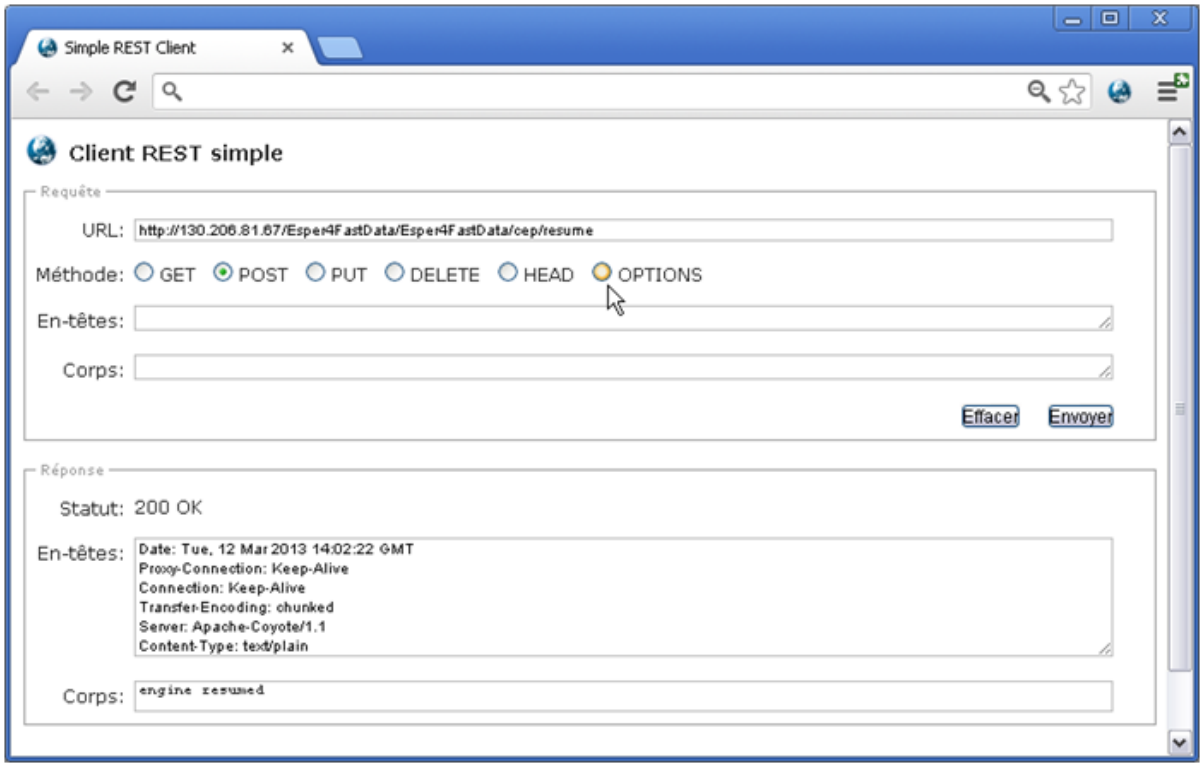
5.2.4.4 *Resume*

Resume the paused CEP engine Paused engine keeps eventTypes and statements defined

$rootService/Esper4FastData/cep/{cmd}

POST

URL : http://130.206.81.67/Esper4FastData/Esper4FastData/cep/resume



5.2.5 EventType methods

5.2.5.1 *Add Event Type*

add an event type to the CEP engine

$rootService/Esper4FastData/eventTypes/{name}

POST

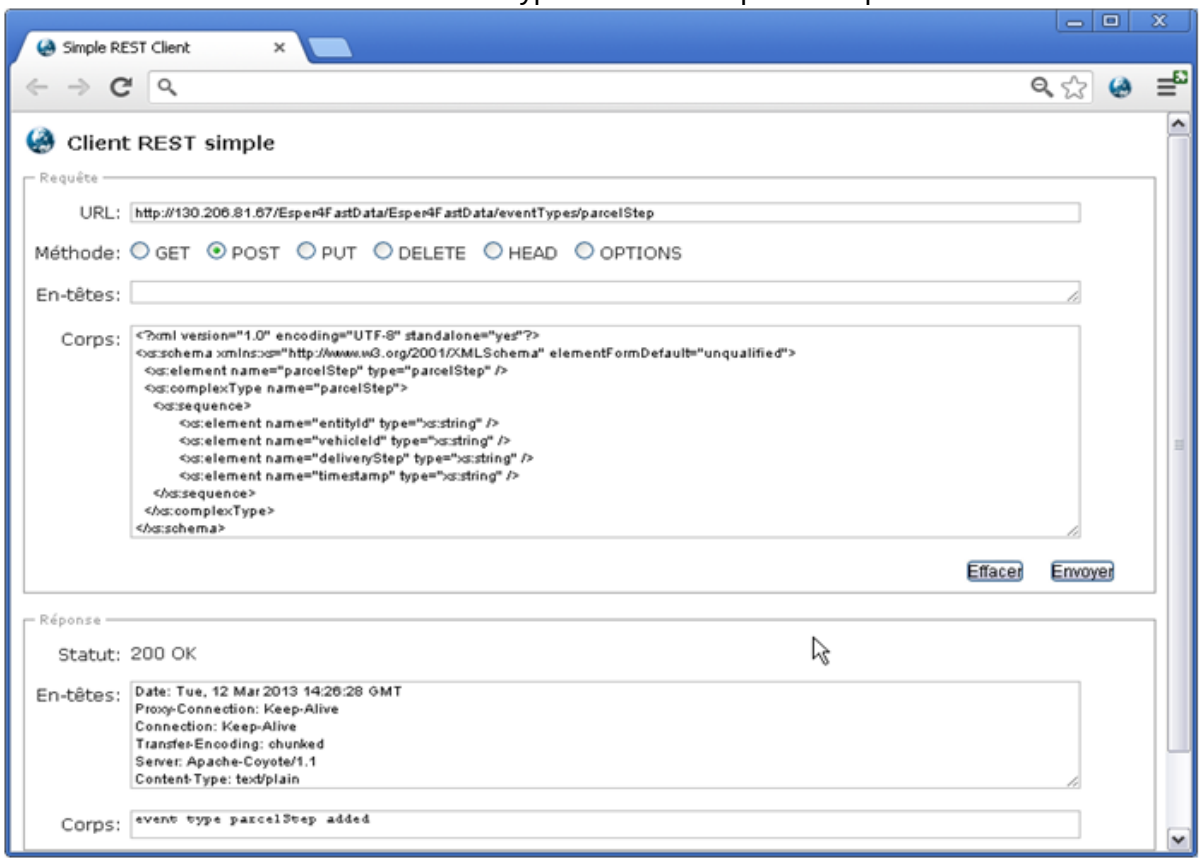
URL : http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/parcelStep

Payload

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified">
  <xs:element name="parcelStep" type="parcelStep" />
  <xs:complexType name="parcelStep">
    <xs:sequence>
          <xs:element name="entityId" type="xs:string" />
          <xs:element name="vehicleId" type="xs:string" />
          <xs:element name="deliveryStep" type="xs:string" />
```

```
            <xs:element name="timestamp" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Result event type parcelStep added

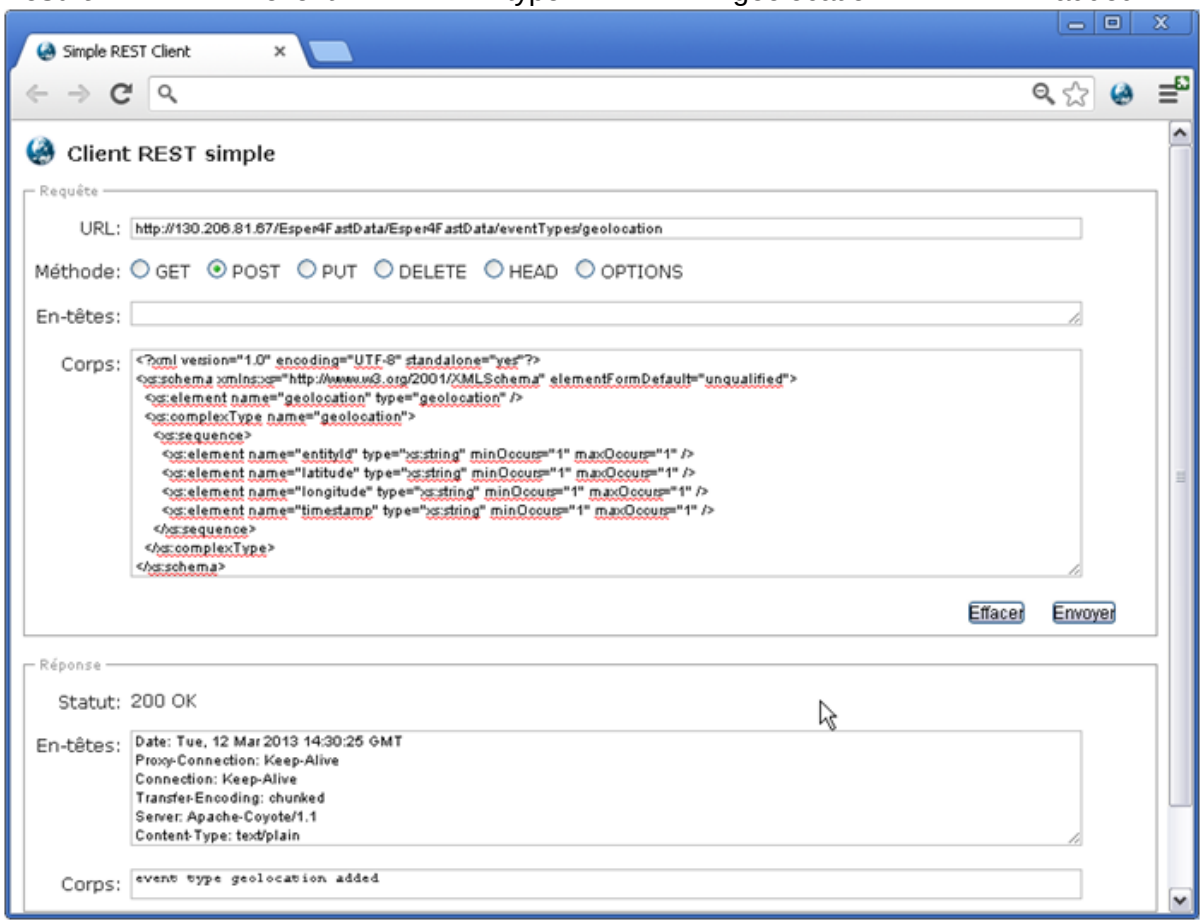


Example for geolocation eventType
POST
URL: http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/geolocation
Payload

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified">
  <xs:element name="geolocation" type="geolocation" />
  <xs:complexType name="geolocation">
    <xs:sequence>
      <xs:element name="entityId" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="latitude" type="xs:string" minOccurs="1"
maxOccurs="1" />
```

```
      <xs:element name="longitude" type="xs:string"
minOccurs="1" maxOccurs="1" />

      <xs:element name="timestamp" type="xs:string"
minOccurs="1" maxOccurs="1" />

    </xs:sequence>

  </xs:complexType>

</xs:schema>
```

Result event type geolocation added



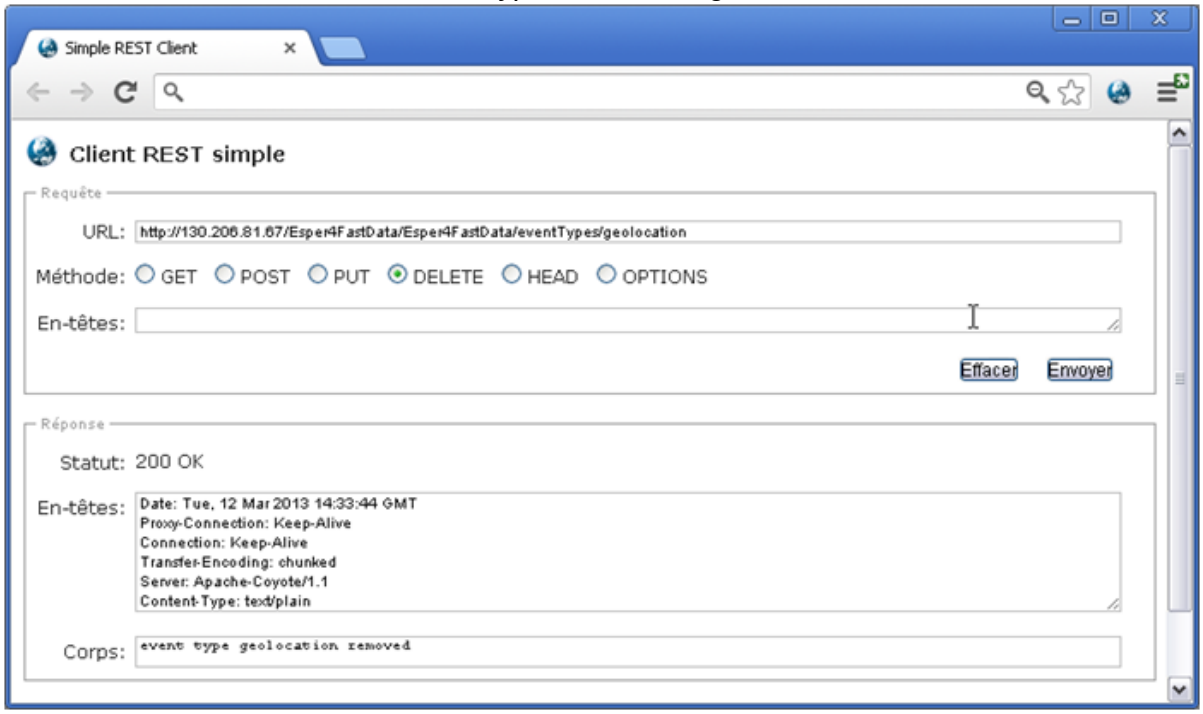
5.2.5.2 *Remove Event Type*

$rootService/Esper4FastData/eventTypes/{name}

DELETE

URL: http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/geolocation

Result event type geolocation removed



5.2.5.3 *Get Event Types*

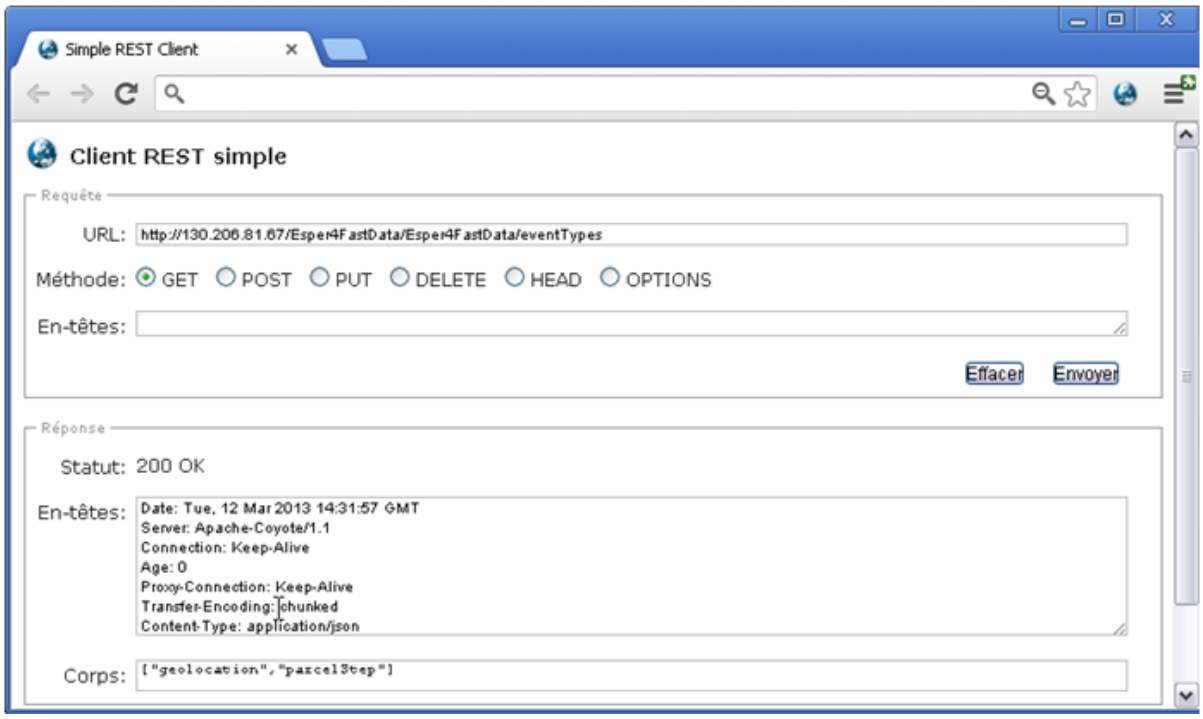
$rootService/Esper4FastData/eventTypes

GET

URL: http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes

Result

```
["stateorder","geolocation","customerlocation","pollution","cont
extElement","availability"]
```



5.2.6 EventType transform methods

5.2.6.1 *Add transform file to Event Type*

Add a transform file associated to an event type. These transform file is used to transform event from the specified event Type out from the CEP.

$rootService/Esper4FastData/eventTypes/{name}/transform

POST

URL :

http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/parcelStep/transform

Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       version="1.0">
       <xsl:output method="xml" indent="yes" />
       <xsl:template match="/*">
       <xsl:variable name="eventType">
        <xsl:value-of select="name(/*)" />
       </xsl:variable>
       <xsl:element name="updateContextRequest">
       <xsl:element name="contextElementList">
       <xsl:element name="contextElement">
```

```
       <xsl:element name="attributeDomainName"><xsl:value-of
select="$eventType" /></xsl:element>
                      <xsl:call-template
name="entityId"></xsl:call-template>
                      <xsl:call-template
name="contextAttributeList"></xsl:call-template>
                      <xsl:call-template
name="timeStamp"></xsl:call-template>
              </xsl:element>
              </xsl:element>
              <xsl:element
name="updateAction">UPDATE</xsl:element>
              </xsl:element>
    </xsl:template>


       <xsl:template name="entityId">
              <xsl:element name="entityId">
        <xsl:element name="id"><xsl:value-of select="entityId"
/></xsl:element>
        </xsl:element>
     </xsl:template>


       <xsl:template name="contextAttributeList">
      <xsl:element name="contextAttributeList">
                      <xsl:call-template name="attribute">
                      <xsl:with-param
name="name">vehicleId</xsl:with-param>
                      <xsl:with-param
name="type">string</xsl:with-param>
                      <xsl:with-param name="value"
select="vehicleId"></xsl:with-param>
                      </xsl:call-template>
                      <xsl:call-template name="attribute">
                      <xsl:with-param
name="name">deliveryStep</xsl:with-param>
                      <xsl:with-param
name="type">string</xsl:with-param>
                      <xsl:with-param name="value"
select="deliveryStep"></xsl:with-param>
                      </xsl:call-template>
              </xsl:element>
       </xsl:template>
```

```
    <xsl:template name="attribute">
    <xsl:param name="name" />
    <xsl:param name="type" />
    <xsl:param name="value" />
    <xsl:element name="contextAttribute">
    <xsl:element name="name"><xsl:value-of select="$name"
/></xsl:element>
    <xsl:element name="type"><xsl:value-of select="$type"
/></xsl:element>
    <xsl:element name="contextValue"><xsl:value-of
select="$value" /></xsl:element>
    </xsl:element>
    </xsl:template>

       <xsl:template name="timeStamp">
       <xsl:element name="domainMetadata">
       <xsl:element name="contextMetadata">
       <xsl:element name="name">timestamp</xsl:element>
       <xsl:element name="type">xsd:dataTime</xsl:element>
       <xsl:element name="value"><xsl:value-of
select="timestamp" /></xsl:element>
       </xsl:element>
       </xsl:element>
       </xsl:template>
</xsl:stylesheet>
```

Result

```
transformFile ../webapps/CEPManagerService/WEB-
INF/resource/transformGeolocation2ngsi.xsl
associated to eventType StoredParcelGeoloc added
```

5.2.6.2 *Remove transform file associated to specified Event Type*

$rootService/Esper4FastData/eventTypes/{name}/transform

DELETE

URL :

http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/parcelStep/transform

Result

```
transformFile associated to eventType geolocation removed
```

5.2.6.3 *Get transform file associated to specified Event Types*

$rootService/Esper4FastData/eventTypes/{name}/transform
GET
URL :
http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/parcelStep/transform
Result

```
no transform file defined for eventType
```

5.2.6.4 *Get all transform files associated to Event Types*

$rootService/Esper4FastData/eventTypes/transform
GET
URL :
http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/parcelStep/transform
Result

```
transform file defined eventType Transform file
geolocalisation ../webapps/CEPManagerService/WEB-INF/resource/geolocalisation.xsl
StoredParcelGeoloc ../webapps/CEPManagerService/WEB-INF/resource/transformGeolocation2ngsi.xsl
```

5.2.6.5 *Save transform file associated to specified Event Type*

$rootService/Esper4FastData/eventTypes/transform/save
POST
URL :
http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/transform/save
Result

```
transform file saved ../webapps/CEPManagerService/WEB-INF/resource/eventtypes.transform
```

5.2.6.6 *Remove all transform file associated to specified Event Type*

$rootService/Esper4FastData/eventTypes/transform
DELETE
URL : http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/transform
Result

```
transforms deleted
```

5.2.6.7 *Reload transform file associated to specified Event Type*

$rootService/Esper4FastData/eventTypes /transform/reload POST URL: http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/transform/reload

DELETE
URL : http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/transform
Result

```
transform file reloaded
{geolocalisation=../webapps/CEPManagerService/WEB-
INF/resource/geolocalisation.xsl,
StoredParcelGeoloc=../webapps/CEPManagerService/WEB-
INF/resource/transformGeolocation2ngsi.xsl}
```

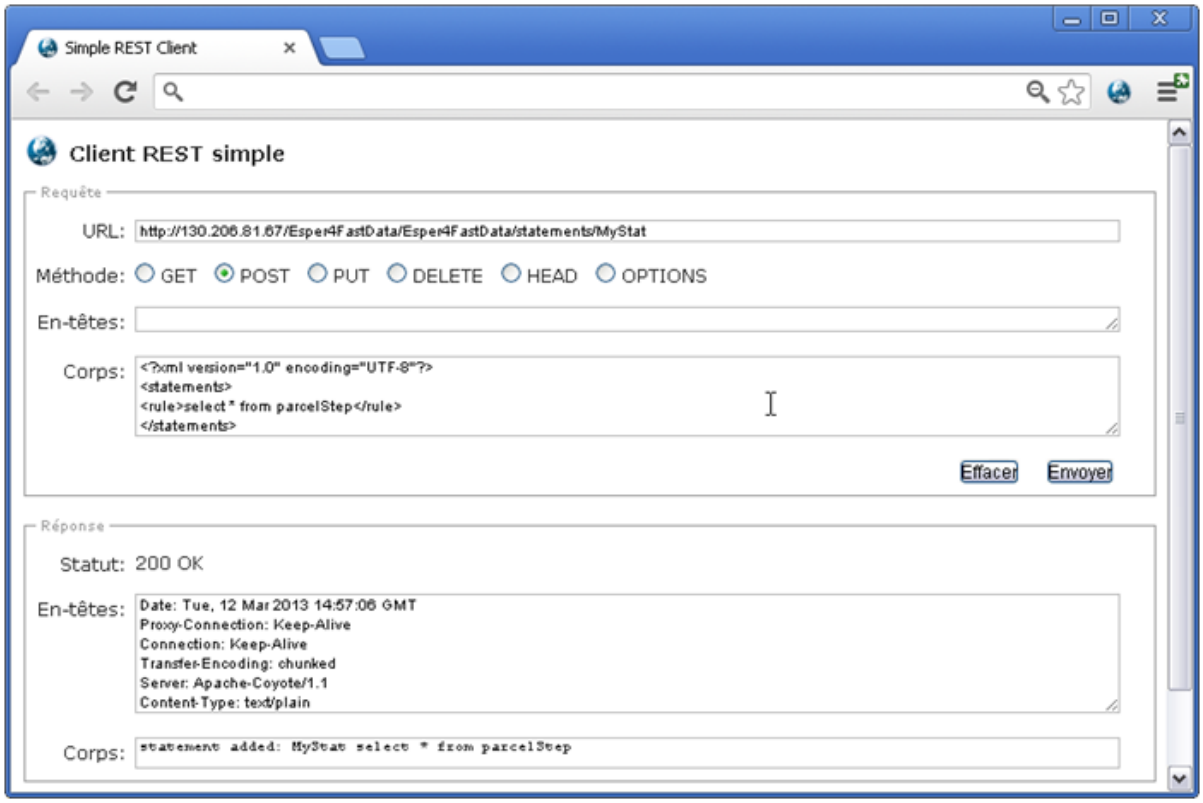
5.2.7 Statement methods

5.2.7.1 *add Statement POST request*

Add a statement to the CEP running Creates and starts an EPL statement. The CEP engine must be started The eventType used in statement must have been defined before adding statement The engine assigns a unique name to the statement. The returned statement is in started state. The statement name is optimally a unique name. If a statement of the same name has already been created, the engine assigns a postfix to create a unique statement name.
$rootService/Esper4FastData/statements/{name}
POST
URL : http://130.206.81.67/Esper4FastData/Esper4FastData/statements/mystat
Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<statements>
<rule>select * from availability</rule>
</statements>
```

Result statement added: mystat select * from availability



5.2.7.2 *get Statements Names*

Get all names of statements currently deployed in the CEP engine Returns the statement names of all started and stopped statements. This excludes the name of destroyed statements

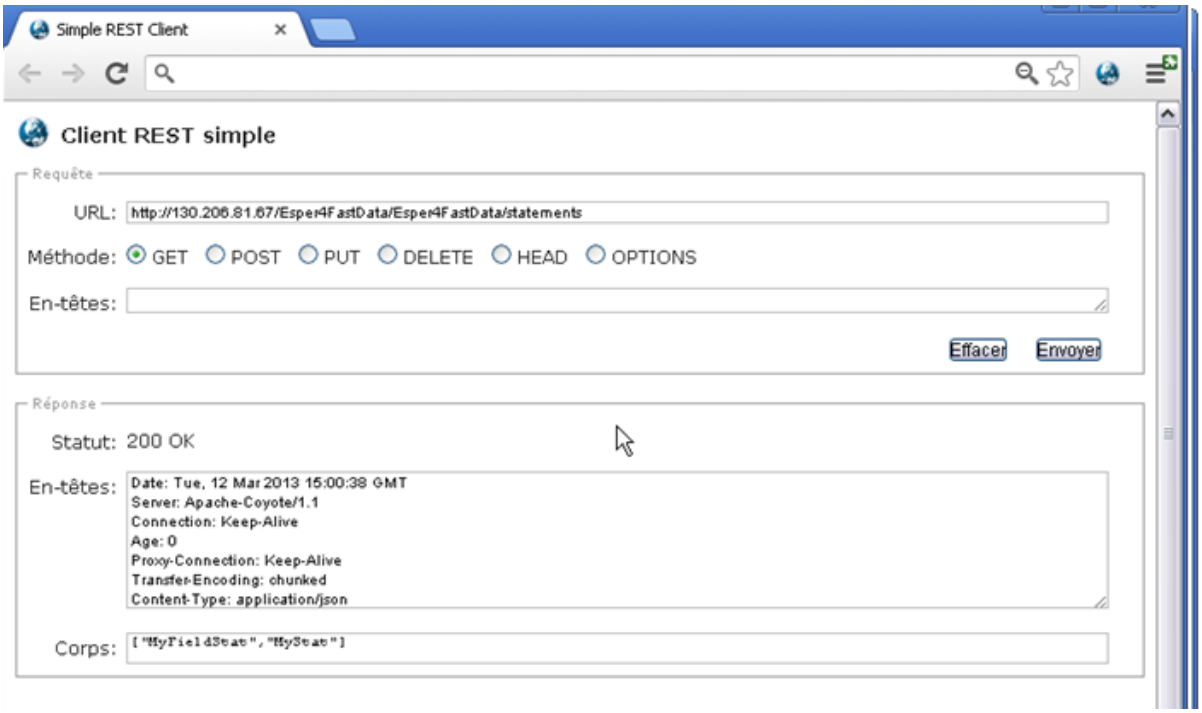
$rootService/Esper4FastData/statements/

GET

URL: http://130.206.81.67/Esper4FastData/Esper4FastData/statements/

Result

["mystat","Myfieldstat"]



5.2.7.3 *get Statement*

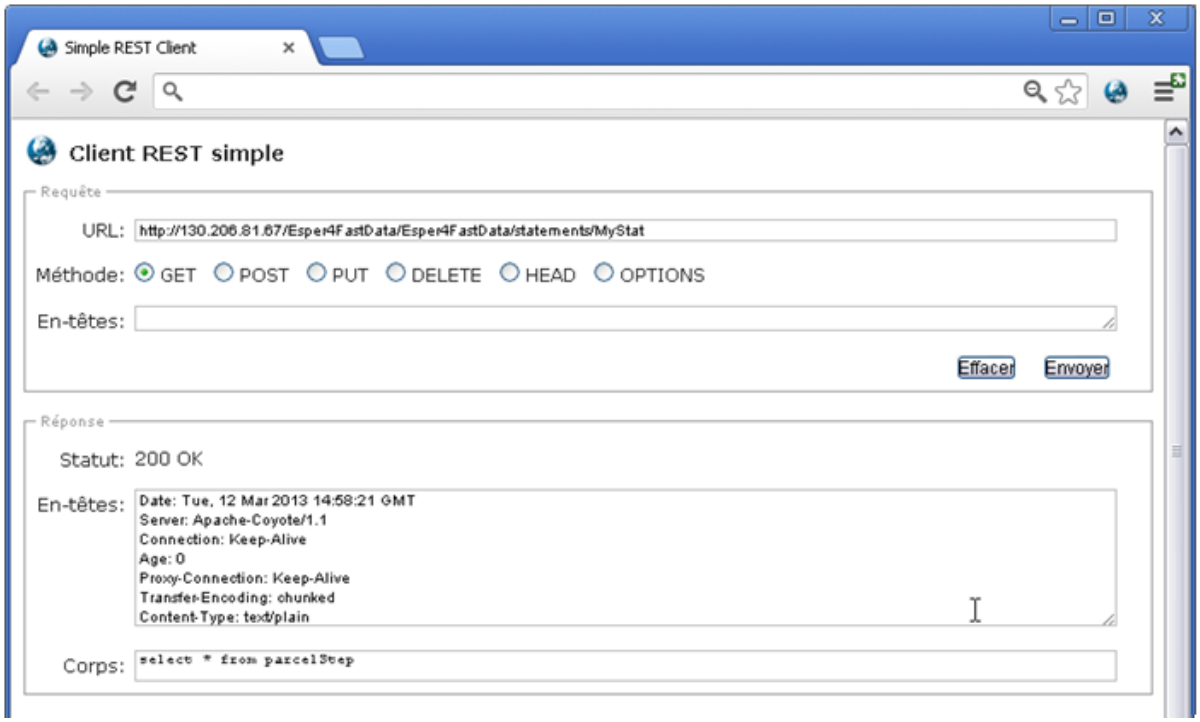
Get statement rule for a specific statement **$rootService/Esper4FastData/statements/{name}**

GET

URL: http://130.206.81.67/Esper4FastData/Esper4FastData/statements/mystat

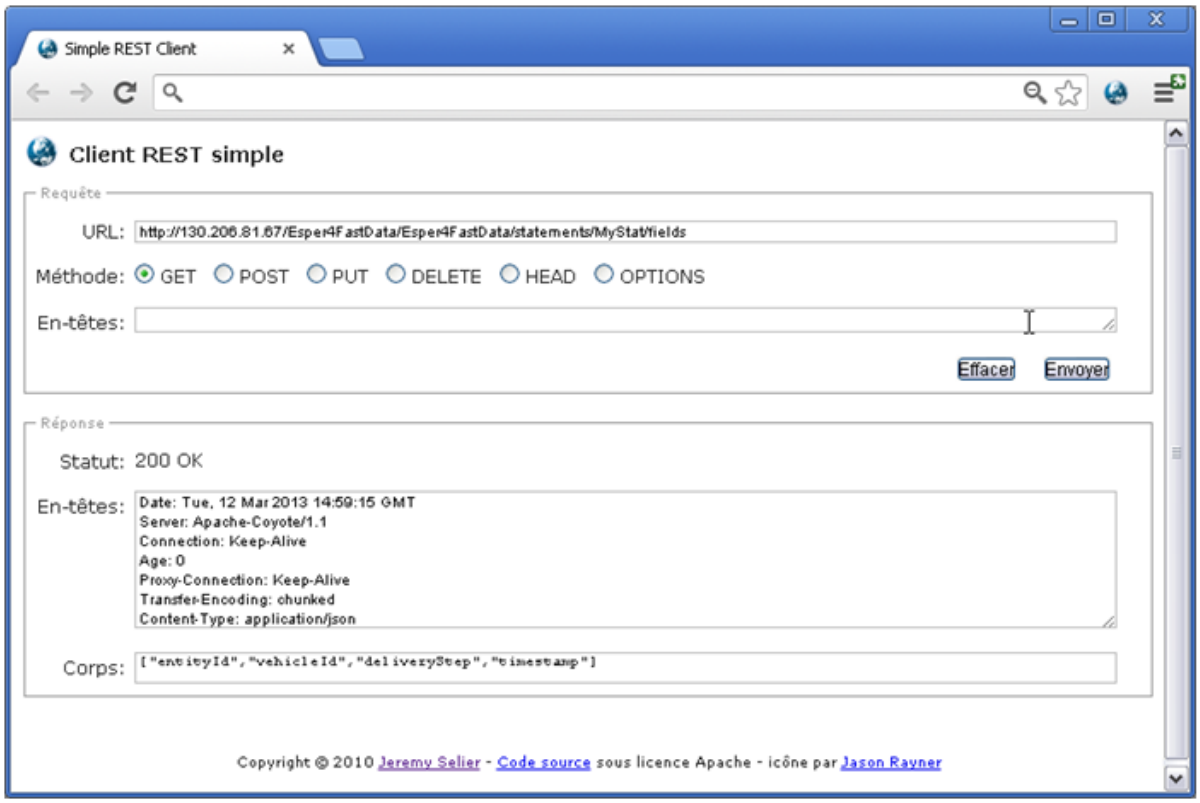
Result

select * from availability



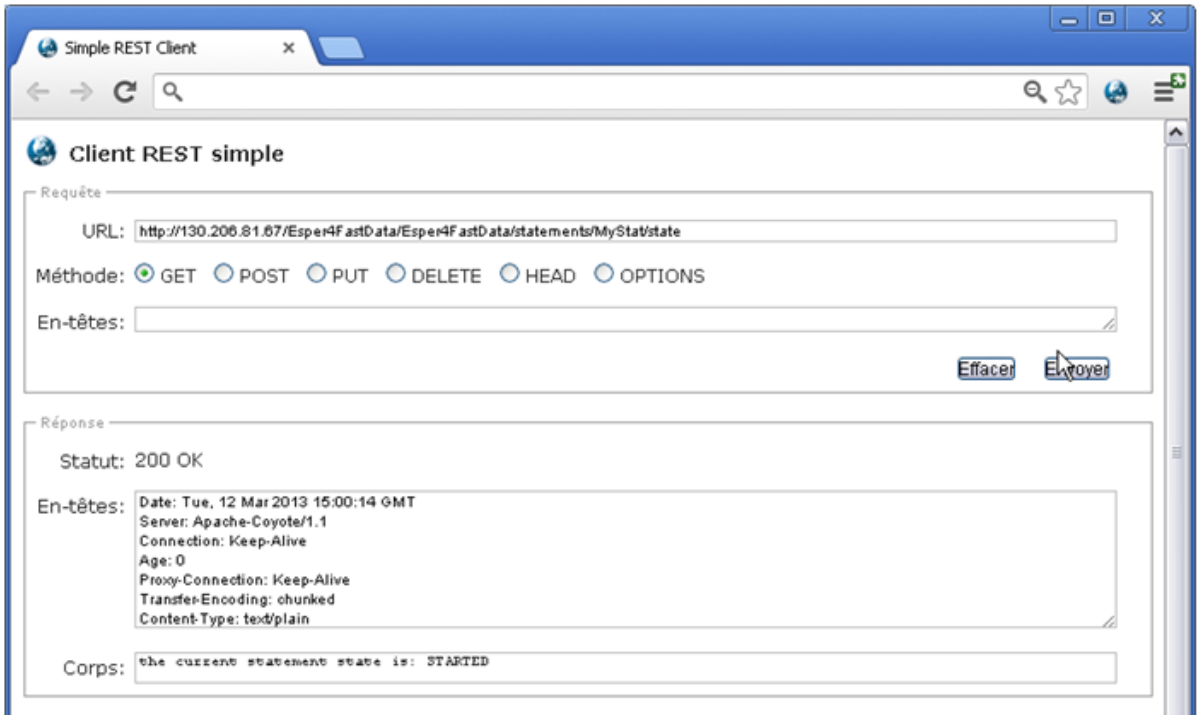
5.2.7.4 *getStatementFields*

Get attributes selected for a specific statement
$rootService/Esper4FastData/statements/{name}/fields
GET
URL: http://130.206.81.67/Esper4FastData/Esper4FastData/statements/mystat/fields
Result
["userType","userName","stream","phoneNumber","timeStamp"]



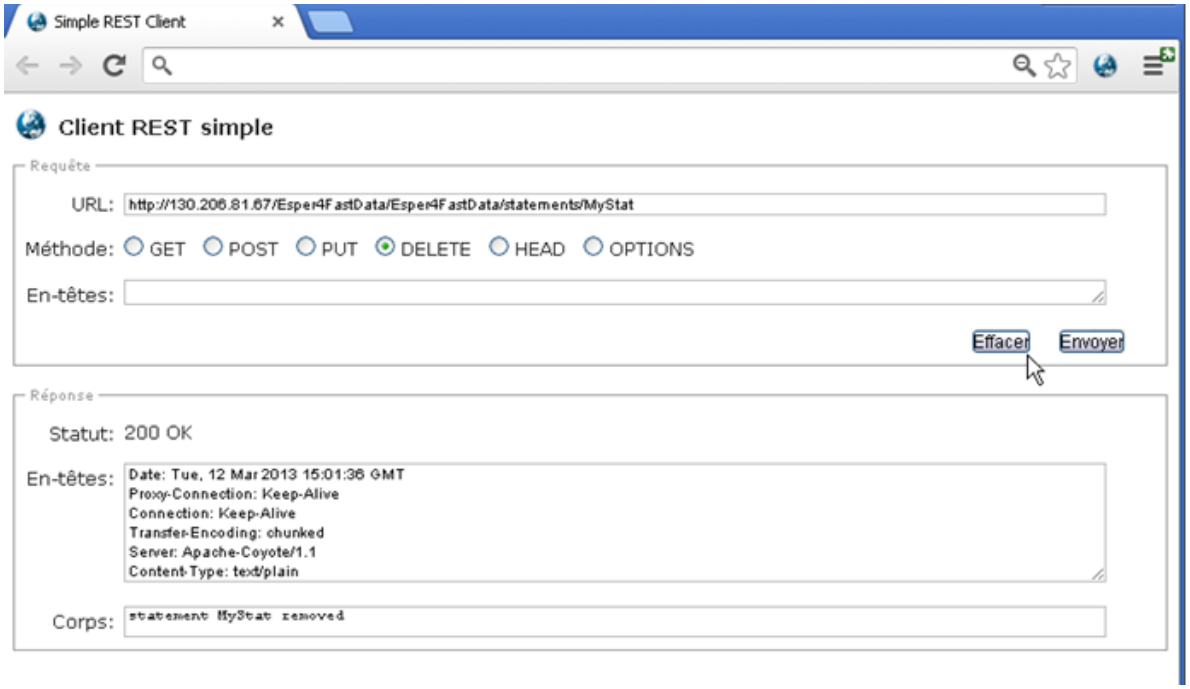
5.2.7.5 *getStatementState*

Get the state of a specific statement
$rootService/Esper4FastData/statements/{name}/state
GET
URL: http://130.206.81.67/Esper4FastData/Esper4FastData/statements/mystat/state
Result
the current statement state is: STARTED



5.2.7.6 *removeStatement*

Remove a specific statement **$rootService/Esper4FastData/statements/{name}**
DELETE
URL: http://130.206.81.67/Esper4FastData/Esper4FastData/statements/mystat
Result
statement mystat removed

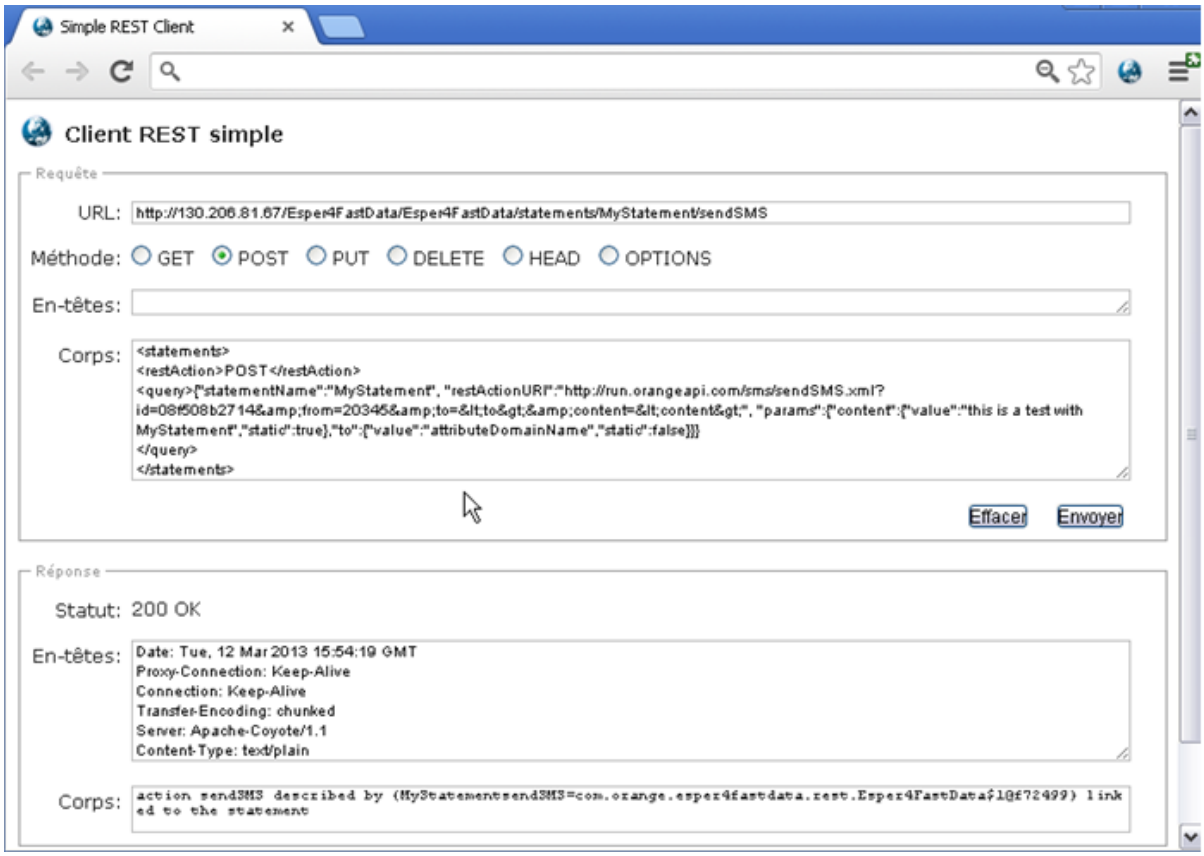


5.2.7.7 *Add Action to Statement*

$rootService/Esper4FastData/statements/{statName}/{actionName}
POST
URL:
http://130.206.81.67/Esper4FastData/Esper4FastData/statements/MyStatement/sendSms
Payload

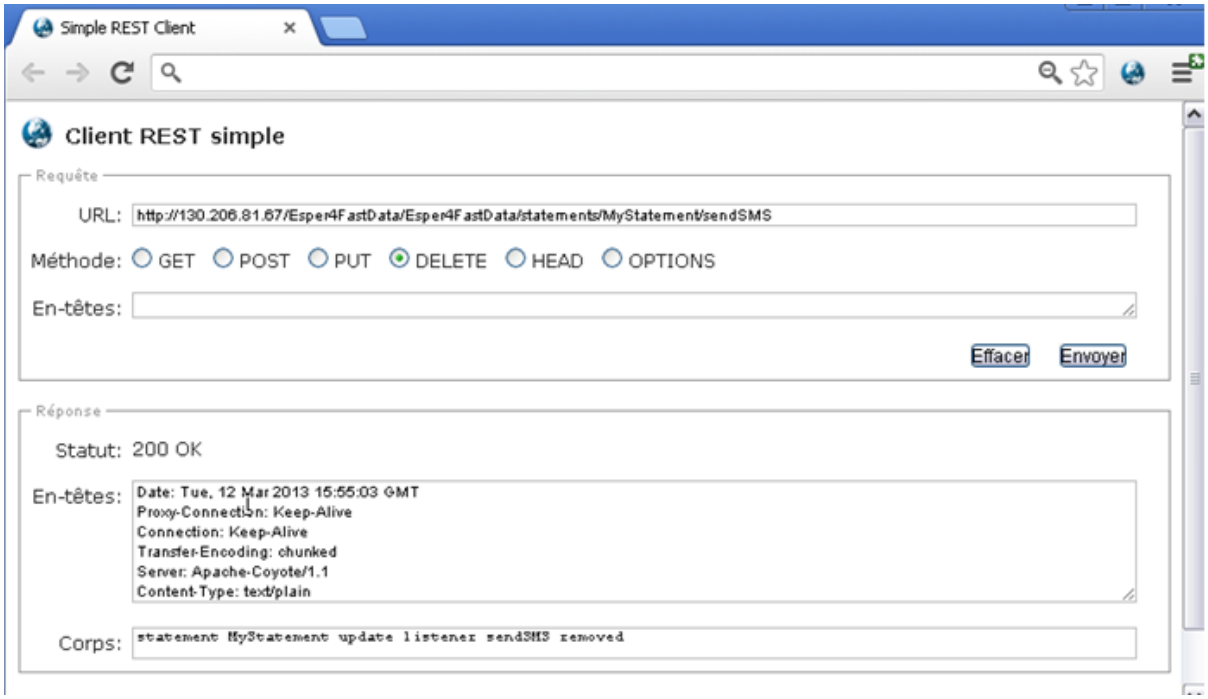
```
<statements>
<restAction>POST</restAction>
<query>{"statementName":"MyStatement",
"restActionURI":"http://run.orangeapi.com/sms/sendSMS.xml?id=08f
508b2714&from=20345&to=<to>&content=<content>",
"params":{"content":{"value":"this is a test with
MyStatement","static":true},"to":{"value":"attributeDomainName",
"static":false}}}
</query>
</statements>
```

Result
action sendSMS described by {MyStatementsendSMS=com.orange.esper4fastdata.rest.Esper4FastData$1@183e984} linked to the statement



5.2.7.8 *Remove Action to Statement*

$rootService/Esper4FastData/statements/{statName}/{actionName}
DELETE
URL:
http://130.206.81.67/Esper4FastData/Esper4FastData/statements/MyStatement/sendSms
Result
statement MyStatement update listener sendSMS removed



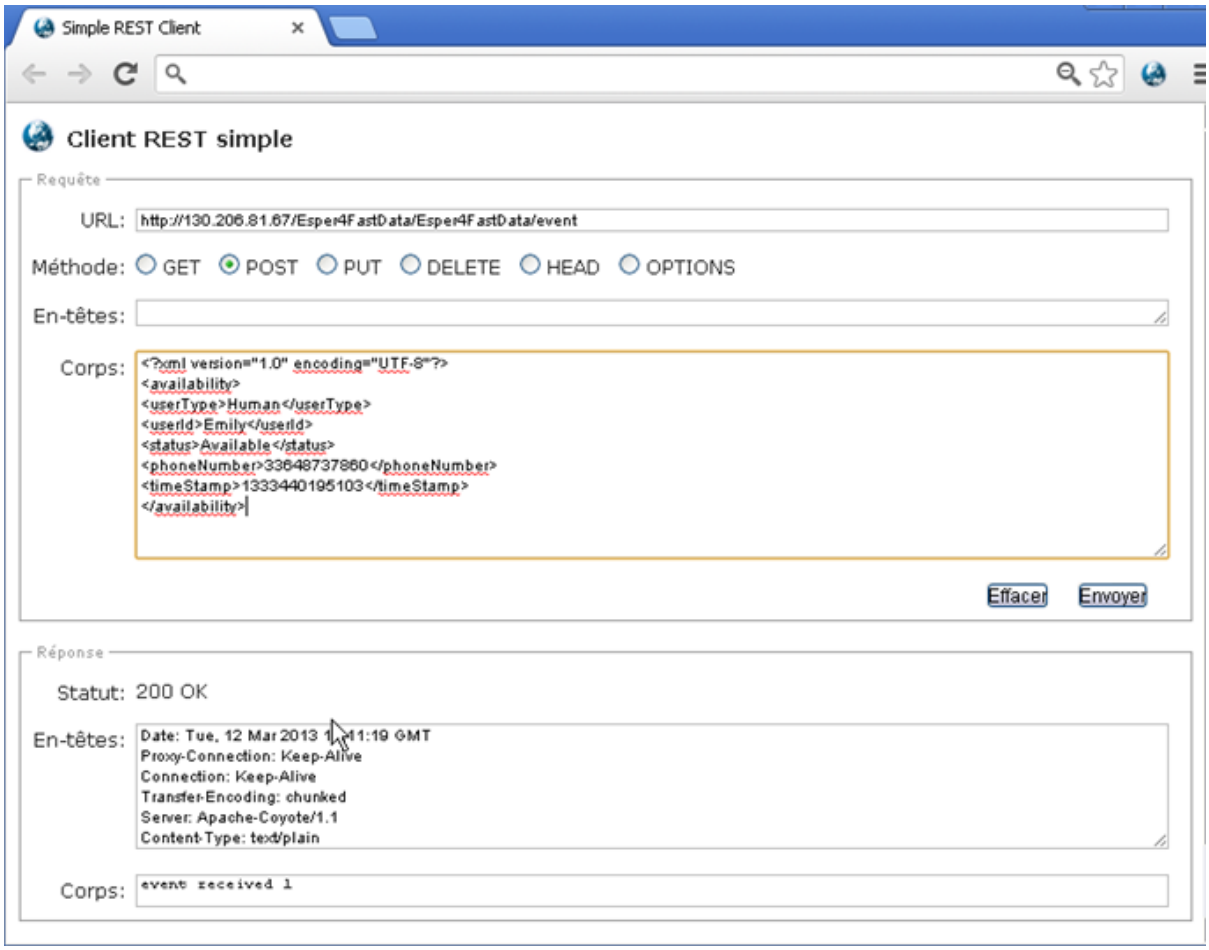
5.2.8 Send Event methods

5.2.8.1 *send Event*

Send an event within the CEP
$rootService/Esper4FastData/event
POST
URL: http://130.206.81.67/Esper4FastData/Esper4FastData/event
Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<availability>
<userType>Human</userType>
<userId>Emily</userId>
<status>Available</status>
<phoneNumber>33648737860</phoneNumber>
<timeStamp>1333440195103</timeStamp>
</availability>
```

Result
event received



5.2.9 Use Case from end to end

5.2.9.1 *Add Event Type*

Add Event Type availability, customerlocation, geolocation, stateorder and pollution
To add an event Type, use the google chrome Client REST simple with parameters:
POST: http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/availability
Payload: Event Type availability.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified">
  <xs:element name="availability" type="availability" />
  <xs:complexType name="availability">
    <xs:sequence>
      <xs:element name="userType" type="xs:string" minOccurs="1"
maxOccurs="1" />
```

```
      <xs:element name="userId" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="status" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="phoneNumber" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="timeStamp" type="xs:time" minOccurs="1"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

POST:
http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/customerlocation
Payload: Event Type CustomerLocation.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified">
  <xs:element name="customerlocation" type="customerlocation" />
  <xs:complexType name="customerlocation">
    <xs:sequence>
      <xs:element name="customerId" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="address" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="latitude" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="longitude" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="phoneNumber" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="timeStamp" type="xs:time" minOccurs="1"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

POST: http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/geolocation
Payload: Event Type geolocation.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified">
  <xs:element name="geolocation" type="geolocation" />
  <xs:complexType name="geolocation">
    <xs:sequence>
      <xs:element name="userType" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="userId" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="latitude" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="longitude" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="phoneNumber" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="timeStamp" type="xs:time" minOccurs="1"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

POST: http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/stateorder
Payload: Event Type stateorder.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified">
  <xs:element name="stateorder" type="stateorder" />
  <xs:complexType name="stateorder">
    <xs:sequence>
      <xs:element name="customerPhoneNumber" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="factoryId" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="status" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="timeStamp" type="xs:time" minOccurs="1"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

POST: http://130.206.81.67/Esper4FastData/Esper4FastData/eventTypes/pollution
Payload: Event Type pollution.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified">
  <xs:element name="pollution" type="pollution" />
  <xs:complexType name="pollution">
    <xs:sequence>
      <xs:element name="sensorId" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="pollutionMessage" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="pollutionType" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="pollutionLevel" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="latitude" type="xs:string" minOccurs="1"
maxOccurs="1" />
      <xs:element name="longitude" type="xs:string"
minOccurs="1" maxOccurs="1" />
      <xs:element name="timeStamp" type="xs:time" minOccurs="1"
maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

5.2.9.2 *Add Statements*

Add statements statPollution and statAlert.
To add a statement, use the google chrome Client REST simple with parameters:

- add Statement statPollution

POST: http://130.206.81.67/Esper4FastData/Esper4FastData/statements/statPollution
Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<statements>
<rule>select * from pollution</rule>
</statements>
```

result statement added: mystat select * from pollution

- add Statement statAlert

POST: http://130.206.81.67/Esper4FastData/Esper4FastData/statements/statAlert
Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<statements>
<rule>select * from pollution (pollutionMessage='Alerte')
</rule>
</statements>
```

result statement added: select * from pollution (pollutionMessage='Alerte')

- check statement state is STARTED

POST:
http://130.206.81.67/Esper4FastData/Esper4FastData/statements/statPollution/state

- check Fields selected:

POST:
http://130.206.81.67/Esper4FastData/Esper4FastData/statements/statPollution/fields
Payload
sensorId, pollutionMessage, pollutionType, pollutionLevel, latidtude, longitude, timestamp

5.2.9.3 *Add Action to Statements*

- statPollution: linked a REST getMethod, which open a file and write all params inside
- POST
- URL: http://130.206.81.67/Esper4FastData/Esper4FastData/statements/statPollution/getMethod
- Payload

```
<statements>
<restAction>POST</restAction>
<query>
{"statementName":"statPollution",
"restActionURI":"http://localhost/CEPManagerService/CEPManager/getMethod?sensorId=<id>&pollutionMessage=<msg>&pollutionType=<type>&pollutionLevel=<lev>",
"params":{"id":{"value":"sensorId","static":false},"lev":{"value":"pollutionLevel","static":false},"type":{"value":"pollutionType","static":false},"msg":{"value":"pollutionMessage","static":false}}}
</query>
</statements>
```

- statAlert : linked a REST sendSms Method, which send an alert message to the administrator
- POST
- URL: http://130.206.81.67/Esper4FastData/Esper4FastData/statements/statAlert/sendSms
- Payload

```
<statements>
<restAction>POST</restAction>
<query>
{"statementName":"statAlert",
"restActionURI":"http://run.orangeapi.com/sms/sendSMS.xml?id=08f508b2714&from=20345&to=<to>&content=Alerte Pollution de <type> avec level=<level>","params":{"to":{"value":"33648737860","static":true},"level":{"value":"pollutionLevel","static":false},"type":{"value":"pollutionType","static":false}}}
</query>
</statements>
```

5.2.9.4 *Send Events*

Send pollution events: Mesure or Alerte

To send events, use the google chrome Client REST simple with parameters:

- URL: http://130.206.81.67/Esper4FastData/Esper4FastData/event
- Method: POST
- Payload:

```
<?xml version="1.0" encoding="UTF-8"?>
<pollution>
<sensorId>sensor1</sensorId>
<pollutionMessage>Mesure</pollutionMessage>
<pollutionType>SO2</pollutionType>
<pollutionLevel>1000</pollutionLevel>
<longitude>2.36165071681415 9</longitude>
<latitude>48.91714399001996</latitude>
<timeStamp>1333440195103</timeStamp>
</pollution>
```

Alert event:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<pollution>
<sensorId>sensor1</sensorId>
<pollutionMessage>Alerte</pollutionMessage>
<pollutionType>SO2</pollutionType>
<pollutionLevel>1000</pollutionLevel>
<longitude>2.361650716814159</longitude>
<latitude>48.91714399001996</latitude>
<timeStamp>1333440195103</timeStamp>
</pollution>
```

6 Gateway Data Handling - Esper4FastData Mobile - User and Programmers Guide

You can find the content of this chapter as well in the wiki of fi-ware.

6.1.1 Introduction

6.1.1.1 *Events overview*

Basically, an event is just « something that happens ».

In philosophy, events are objects in time, or instantiations of properties in objects, whereas in computing, they are action that are usually initiated outside the scope of a program and that is handled by a piece of code inside the program.

Complex Event Processing(CEP) is a usual solution for the following questions :

- How to handle massively growing data volumes ?
- how to make meanings of all events flowing through your system at the speed of your business ?
- How to preserve flexibility ?

6.1.1.2 *Value-added of a CEP system*

It takes subsequent action in real time, delivers high-speed processing of many events, operates across all the layers of an organization. It also filters the most meaningful events, merges data from many events and has to deal with event privacy and subscribing.

6.1.1.3 *The Esper CEP library*

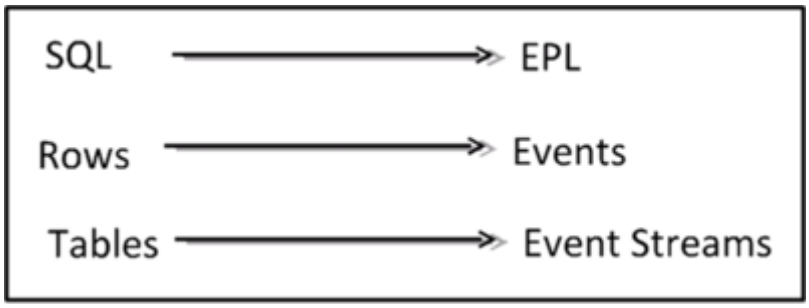
CEP Mobile Manager engine is based upon the open-source Esper library. There is a very active community of developpers behind it.

Here are some technical facts about it :

- Is available under GPL v2
- Is available as a Java library (jar)
- Has been adapted to Android
- Has tiny footprint

6.1.1.4 *Esper Concepts*

Esper is like a database that has been turned upside-down, every database concept having its counterpart in the CEP world :



The Esper library is based on the Event Processing Language(EPL), that looks roughly like SQL. But the "querying" model is completely different, because CEP provides a continuous model of querying. This could be named "real-time data mining", in contrary to "store-now query-ater" model, which is equivalent to analyzing historical data.

Despite the syntax similarities between EPL and SQL, CEP is definitely not a database replacement.

Esper uses events windows to store event streams, which can be compared to tables in the database world. These windows have a predefined size, in terms of data volume, or retention time. They are in fact sliding windows that contain the events flowing THROUGH the EPL statement. This is a key point to understand the difference between a database and a CEP :

- In database world, one can instanciate an SQL statement to SYNCHRONOUSLY retrieve particular data
- In the CEP world it's the data that ASYNCHRONOUSLY flows THROUGH an EPL statement, which is trigerred only if properties criteria are fulfiled

6.1.1.5 ***Basic EPL statements and sliding events windows***

6.1.1.5.1 *Filter-free statement*

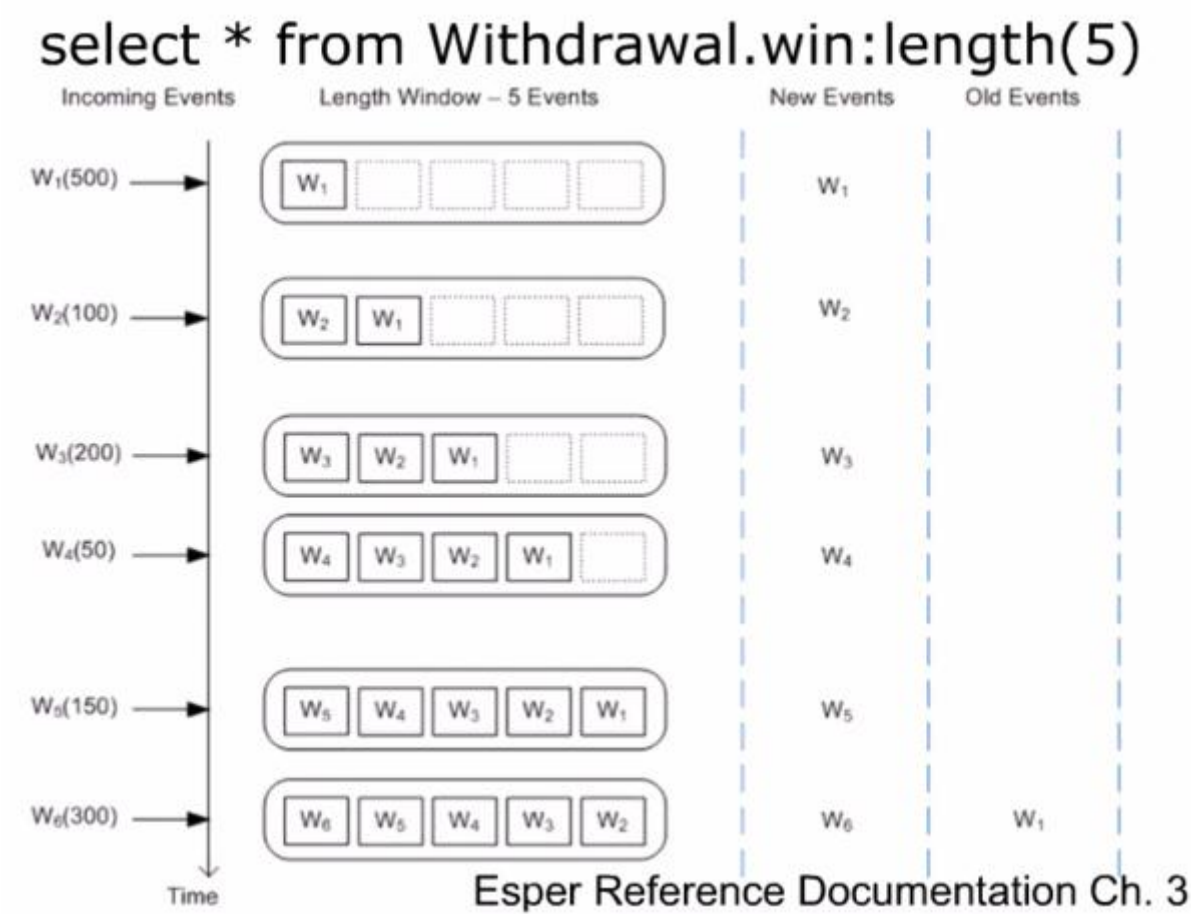
A length window instructs the engine to only keep the last N events for a stream. The next statement applies a length window onto the Withdrawal event stream. The statement serves to illustrate the concept of data window and events entering and leaving a data window:

select * from Withdrawal.win:length(5)

The size of this statement's length window is five events. The engine enters all arriving Withdrawal events into the length window. When the length window is full, the oldest Withdrawal event is pushed out the window. The engine indicates to listeners all events entering the window as new events, and all events leaving the window as old events.

While the term insert stream denotes new events arriving, the term remove stream denotes events leaving a data window, or changing aggregation values. In this example, the remove stream is the stream of Withdrawal events that leave the length window, and such events are posted to listeners as old events.

The next diagram illustrates how the length window contents change as events arrive and shows the events posted to an update listener :

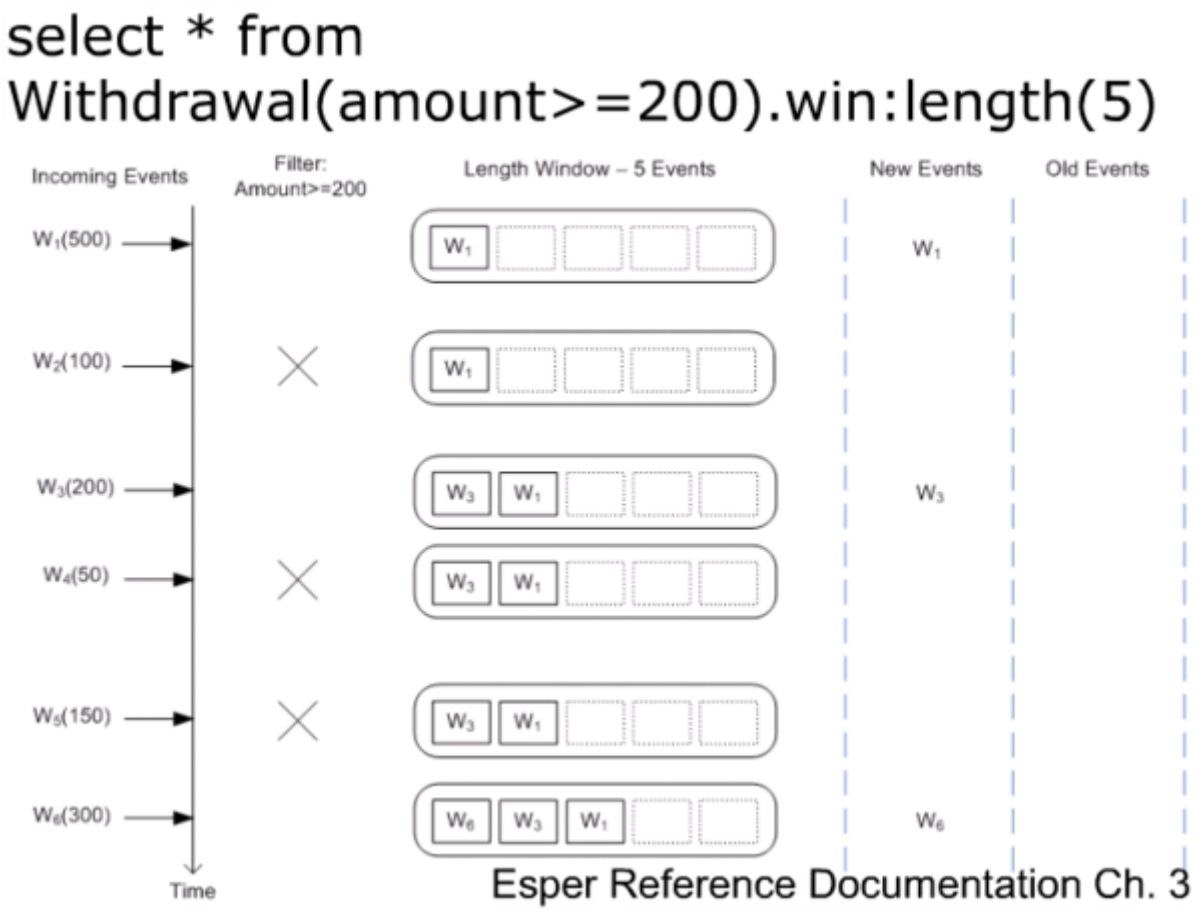


6.1.1.5.2 *Input filter statement*

Filters to event streams allow filtering events out of a given stream before events enter a data window. The statement below shows a filter that selects Withdrawal events with an amount value of 200 or more.

select * from Withdrawal(amount>=200).win:length(5)

With the filter, any Withdrawal events that have an amount of less then 200 do not enter the length window and are therefore not passed to update listeners :



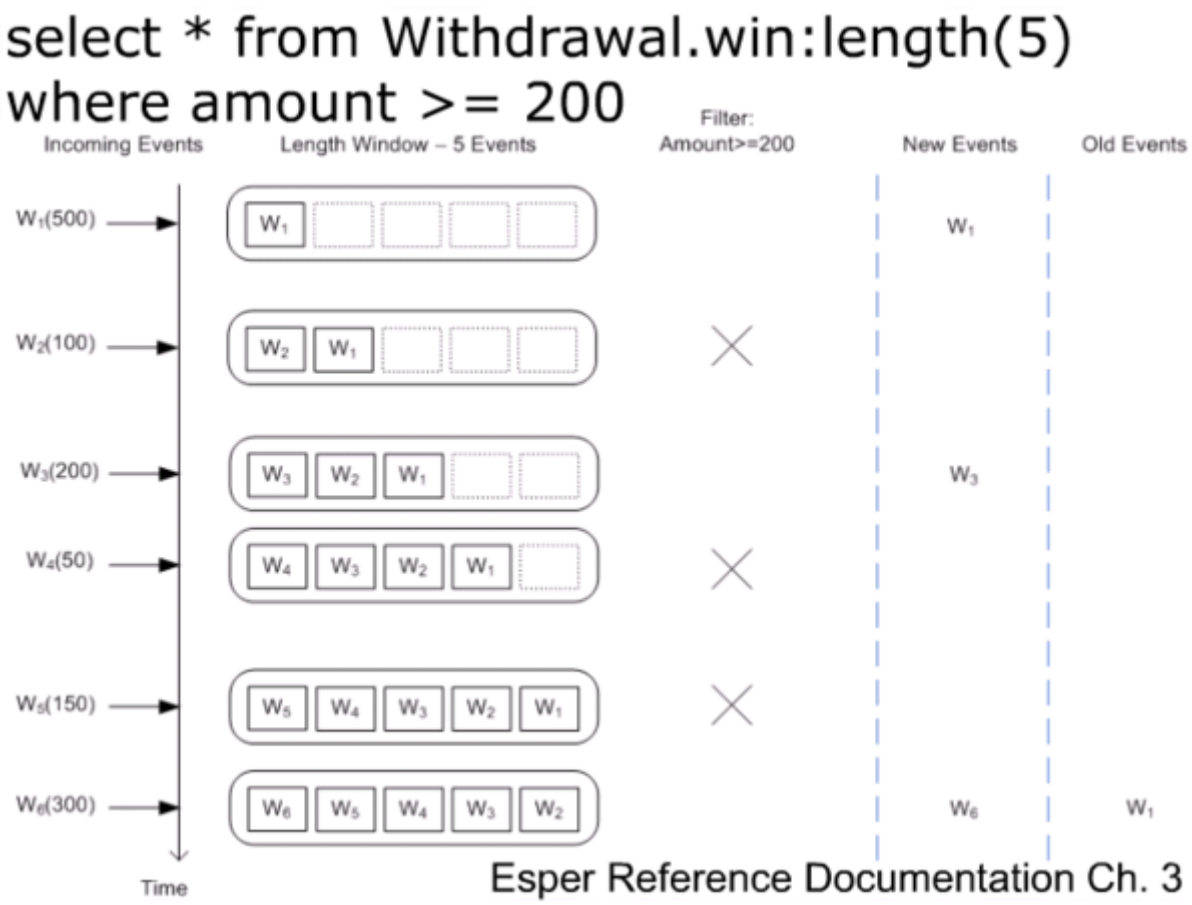
6.1.1.5.3 *Output filter statement*

The where-clause and having-clause in statements eliminate potential result rows at a later stage in processing, after events have been processed into a statement's data window or other views.

The next statement applies a where-clause to Withdrawal events.

select * from Withdrawal.win:length(5) where amount >= 200

The where-clause applies to both new events and old events. As the diagram below shows, arriving events enter the window however only events that pass the where-clause are handed to update listeners. Also, as events leave the data window, only those events that pass the conditions in the where-clause are posted to listeners as old events.



6.1.1.6 *Detailed EPL Reference*

See http://esper.codehaus.org/esper-4.6.0/doc/reference/en-US/html/epl_clauses.html

6.1.1.7 *Glossary of terms*

- Event: anything that happens, or is contemplated as happening
- Event Object, event message: an object that represents, encore or records an event, generally for the purpose of computer processing
- Event Type: a class of event objects. All events must be instances of an event type. An event has the structure defined by its type. Event types should be defined with a XML Schema Definition (XSD file)
- Event attribute or event property: a component of the structure of an event. An event attribute can have a simple or complex data type.
- Event processing: computing that performs operations on events, including reading, creating, transforming and deleting events.
- Timestamp: a time value attribute of an event. The time in which the event was created:creation time or observed:arrival time.
- Complex event processing (CEP): computing that performs operations on complex events, including reading, creating, transforming or abstracting them.
- Statement or event processing rules: a prescribed method for processing event. Event processing rules are described in Event Processing Language

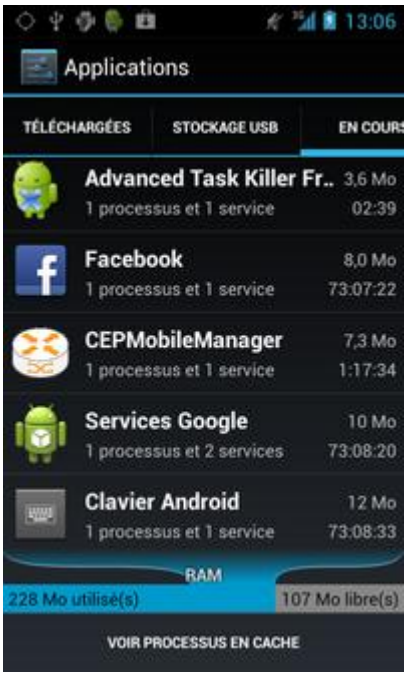
- Event Processing Language: a high level computer language for defining the behavior of event processing agents

6.1.2 User and Programmer Guide

This component is a part of the Data Handling Generic Enabler and is a Rest Web service project using standard interface NGSI 9/10 to communicate with the others components/GE. The following sections explains how to use the Esper4FastData Mobile or CEPMobileManager as a user or developer

6.1.2.1 *Accessing the Esper4FastData Mobile*

CEPMobileManager project allows to manage a single shared Complex Event Processing engine on mobile phone. The methods to manage the CEP engine can be accessed and tested thanks to an Android graphic user interface (GUI). In the Android Menu, go to Settings/Applications/Running in order to diplay running processes."CEPMobileManager" should be available in the list. The CEPMobileManager is approachable in the Android Application List.



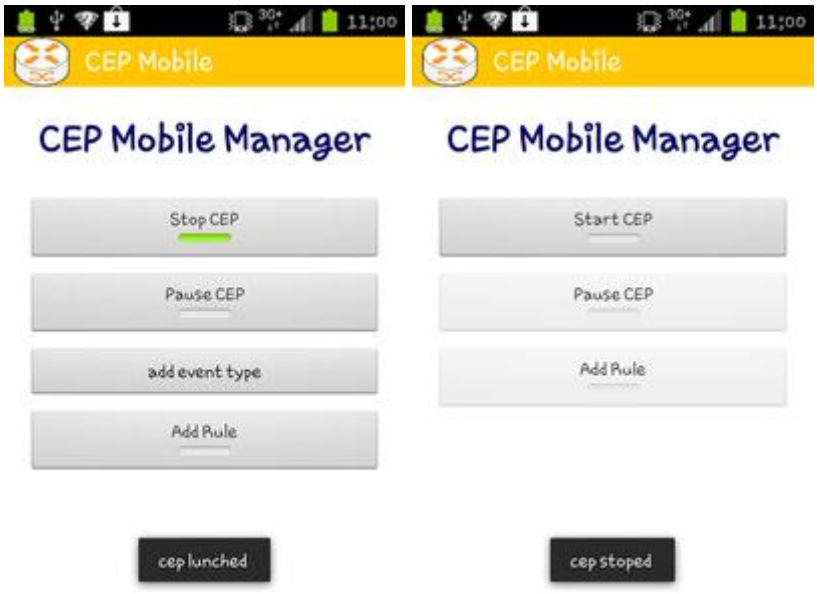
6.1.2.2 *CEP Engine methods*

- start: start the CEP engine if it is not already started

the engine must be started to call all CEPManager methods: adding eventType, rule (statements)…

- stop: stop the running CEP engine

a stopped engine has no more event Types and rules defined

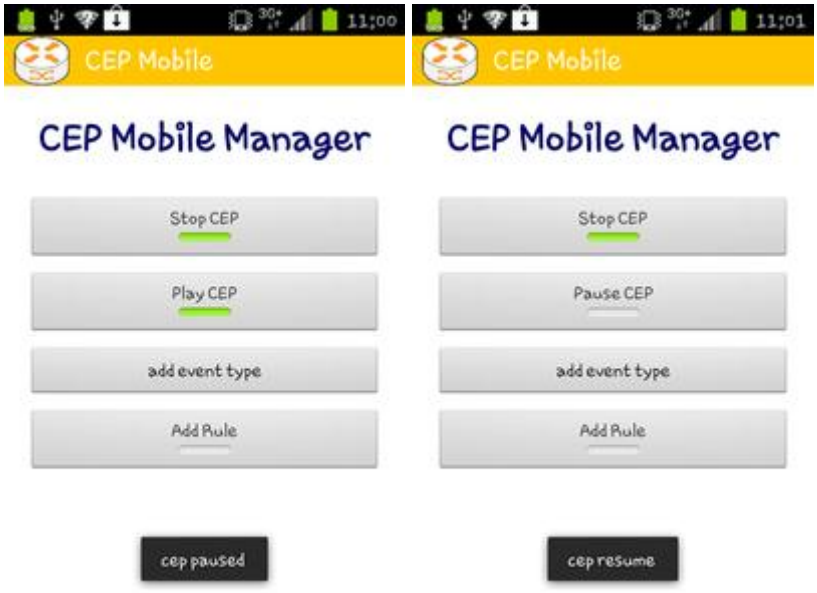


- pause: pause the running CEP engine

Paused engine keeps eventTypes and rules defined

- resume: resume the paused CEP engine

Paused engine keeps eventTypes and rules defined

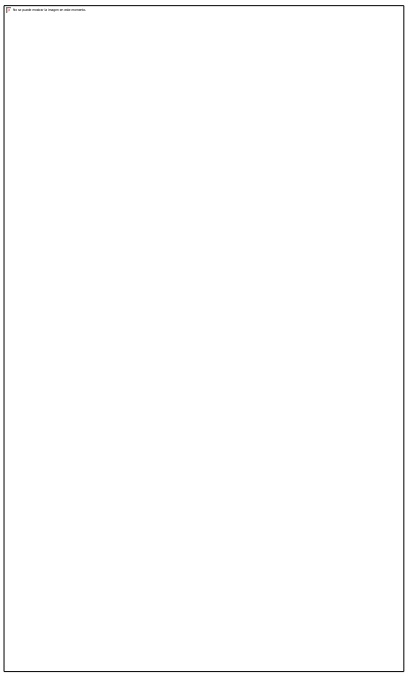
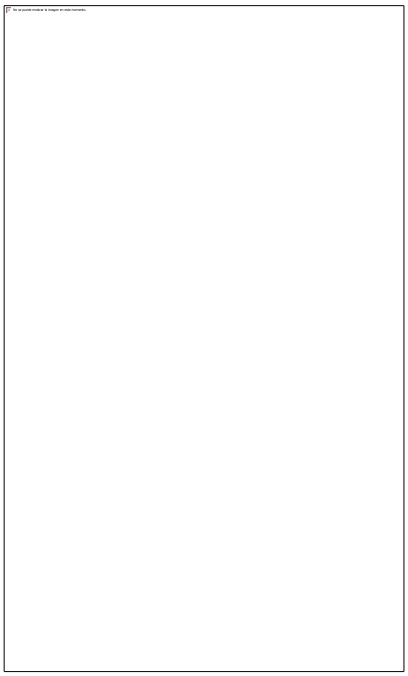


6.1.2.3 *Rule/Statement methods*

- Add Rule: add a rule to the CEP running

Creates and starts an EPL rule. The CEP engine must be started The eventType used in rule must have been defined before adding rule The engine assigns a unique name to the rule. The returned rule is in started state. The rule name is optimally a unique name. If a rule of the same name has already been created, the engine assigns a postfix to create a unique statement name.

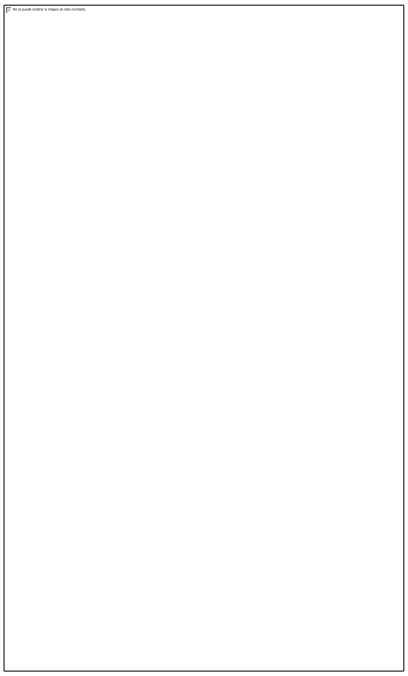
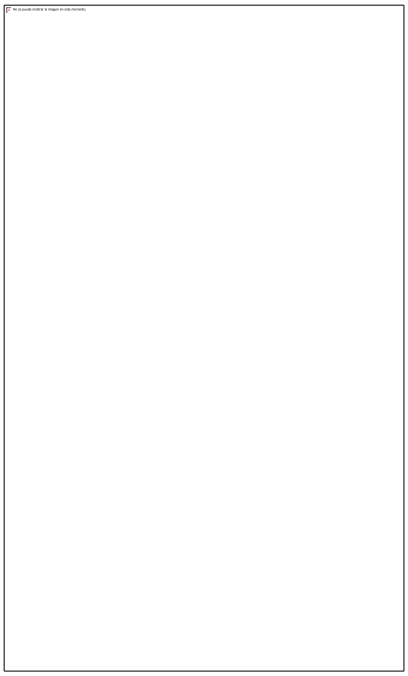
- Delete Rule: delete a specific rule

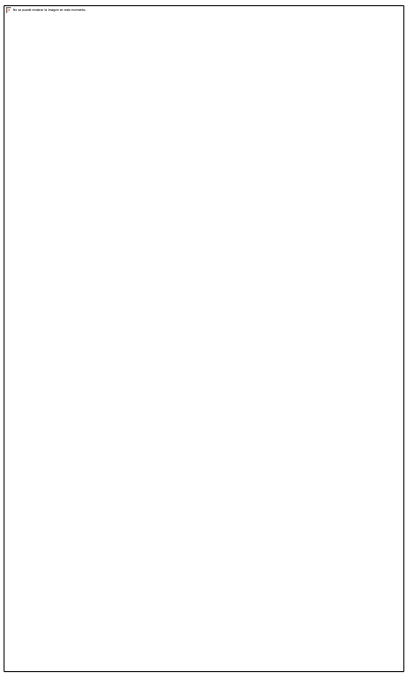
- all Rules: get all names of rules currently deployed in the CEP engine

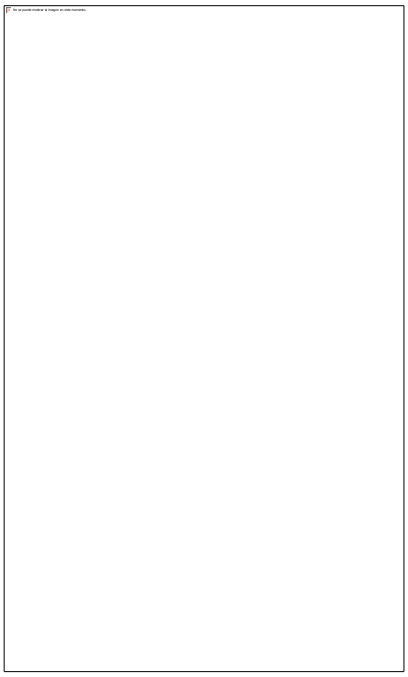
Returns the rule names of all started and stopped rules. This excludes the name of destroyed rules

- Rule Text: get the rule text for the named rule

- pause Rule: pause a specific rule
- Resume Rule: resume a specific rule





6.1.2.4 *EventType methods*

- add Event Type: add an event type to the CEP engine
- send Event: send an event within the CEP

7 Gateway Data Handling - SOL CEP User and Programmers Guide

You can find the content of this chapter as well in the wiki of fi-ware.

7.1 Introduction

SOL/CEP (Smart Object Lab Complex Event Processor) is driven using a domain specific language called Dolce, which stands for **D**escripti**o**n **L**anguage for **C**omplex **E**vents.

This enabler is meant to be only used by programmers. Therefore, no user guide is provided.

7.1.1 Addressed Topics

- *Introduction* to the language. Explains the concepts and how to create complex events from simple events.
- *Types, operators and expressions*, addressing data types, constants, expressions and dedicated complex event functions.
- *Program structure* explores in more detail the construction of complex events, their interaction with the simple events, as well as functionality such as sliding time and tuple windows.

7.2 Programmers Guide

7.2.1 Configuration

The Dolce complex event detection specification is presented to SOL/CEP by means of a text file that must be configured.

7.2.1.1 ***Configuration file***

In order for the changes to take place, the configuration file must be edited.

The Application does not specify a standard location for the configuration file.

At startup, it looks for a configuration file named `solcep.conf.xml` in the same directory as from where the application is started.

The location can be overridden by specifinyg `-c <configFile>` as a command line option.

7.2.1.2 ***Applying the changes***

The Application needs to be restarted for the changes to be applied. This is achieved with the `/etc/init.d/solcep_ctrl restart` command.

Events are described according to the Dolce language specification and placed in a text file, referenced from the configuration file.

Events are consumed and produced using adapters that accept a variety of protocols and formats, also driven by the configuration file. The configuration is out of scope for this document.

7.2.2 Restrictions

SOL/CEP is a first proof of concept of how Dolce can be implemented. The current implementation is a subset of the specification, and the goal is to have a full reference implementation in the second release of FI-WARE.

SOL/CEP implements the following features of Dolce.

- Event specification
- Complex Event detection
- The `int` data type
- The `and` and `or` event operators
- Time windows (non-recurring)
- Event filtering, allowing for different channels