Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

D.5.3.2: FI-WARE Installation and Administration 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.5.3.2
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 describes the installation and administration process of each Generic Enabler developed within in the "Internet of Things Service Enablement" chapter. The system requirements for the installation of a Generic Enabler are outlined with respect to necessary hardware, operating system and software. Each GE has a section dedicated to the software installation and configuration process as well as a section, which describes sanity check procedures for the system administrator to verify that the GE installation was successful.

1.2 About This Document

The "FI-WARE Installation and Administration Guide" comes along with the software implementation of components, each release of the document referring to the corresponding software release (as per D.x.2), to facilitate the users/adopters in the installation (if any) and administration of components (including configuration, if any).

1.3 Intended Audience

The document targets system administrators as well as system operation teams of FI-WARE Generic Enablers from the FI-WARE project.

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.

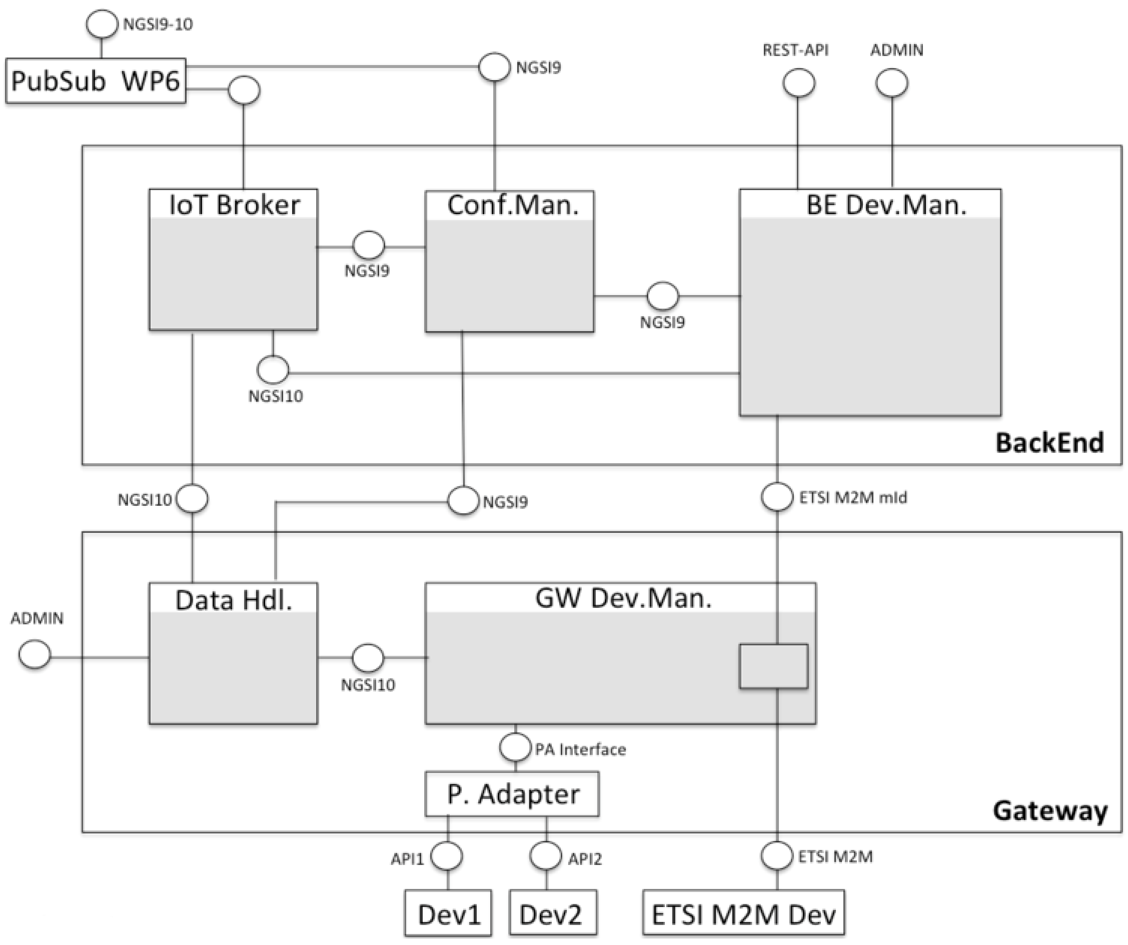
A key design statement is that, whenever present, IoT Gateways are not expected to be permanently connected to the Backend as per communications design or failures. Another relevant remark is that IoT Gateways are expected to be constrained devices in some scenarios. Therefore, light-weight implementations 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 theFI-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.5.3.2_Installation_and_Administration_Guide_front_page

Backend IoT Broker GE - Installation and Administration Guide

Backend Configuration Manager - Installation and Administration Guide

Gateway Data Handling GE - Installation and Administration Guide

Gateway Data Handling - SOL/CEP Complex Event Processor - Installation and Administration Guide

Gateway Data Handling - Esper4FastData Servlet - Installation and Administration Guide

Gateway Data Handling - Esper4FastData OSGi - Installation and Administration Guide

Gateway Data Handling - Esper4FastData Mobile - Installation and Administration 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 the following partners: Atos, Ericsson, NEC, Orange, Telefonica and Telecom Italia .

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 5
1.6.1 Links within this document 5
1.6.2 Figures 5
1.6.3 Sample software code 5
1.7 Acknowledgements 6
1.8 Keyword list 6
1.9 Changes History 6
1.10 Table of Contents 6
2 Backend IoT Broker GE - Installation and Administration Guide 10
2.1 Introduction 10
2.2 System Installation 10
2.3 System Administration 11
2.4 Sanity Check Procedures 12
2.4.1 End to End testing 12

2.4.2 List of Running Processes ... 14
2.4.3 Network interfaces Up & Open ... 14
2.4.4 Databases ... 14
2.5 Diagnosis Procedures ... 14
2.5.1 Resource availability ... 14
2.5.2 Remote Service Access ... 14
2.5.3 Resource consumption ... 14
2.5.4 I/O flows ... 14
3 Backend Configuration Manager - Installation and Administration Guide ... 15
3.1 Installation and Preparation ... 15
3.1.1 MySQL installation and startup ... 15
3.1.2 MySQL preparation ... 15
3.2 Starting iotConfigMgr ... 15
3.3 Log file ... 16
3.4 REST Interface ... 16
3.4.1 Modifying/Getting verbose level ... 16
3.4.2 Modifying/Getting debug level - depracated but still implemented ... 17
3.4.3 Modifying/Getting reads level ... 17
3.4.4 Modifying/Getting writes level ... 17
3.4.5 Modifying/Getting trace levels ... 17
3.4.6 Getting version of running executable ... 17
3.5 Sanity check procedures ... 18
3.5.1 End to End testing ... 18
3.5.2 List of Running Processes ... 18
3.5.3 Network interfaces Up & Open ... 18
3.5.4 Databases ... 18
3.6 Diagnosis Procedures ... 18
3.6.1 Resource availability ... 19
3.6.2 Remote Service Access ... 19
3.6.3 Resource consumption ... 19
3.6.4 I/O flows ... 19
4 Gateway Data Handling GE - Installation and Administration Guide ... 20
5 Gateway Data Handling - SOL/CEP Complex Event Processor - Installation and Administration Guide ... 21
5.1 Purpose and Audience ... 21
5.2 System Requirements ... 21
5.2.1 Hardware Requirements ... 21
5.2.2 Operating System Support ... 21

5.2.3 Software Requirements ... 21
5.3 Software Installation ... 22
5.3.1 Delivery Contents ... 22
5.3.2 Installation ... 22
5.4 Configuration ... 23
5.4.1 Configuration file ... 23
5.4.2 Configuration file validation ... 25
5.4.3 Configuration File Location ... 26
5.4.4 Command Line options ... 26
5.4.5 Plugin Parameters ... 26
5.5 Troubleshooting steps ... 28
5.6 Sanity check Procedures ... 28
5.6.1 End to End testing ... 28
5.6.2 List of Running Processes ... 29
5.6.3 Network interfaces Up & Open ... 29
5.6.4 Databases ... 31
5.7 Diagnosis Procedures ... 31
5.7.1 Resource availability ... 31
5.7.2 Remote Service Access ... 31
5.7.3 Resource consumption ... 31
5.7.4 I/O flows ... 32
6 Gateway Data Handling - Esper4FastData Servlet - Installation and Administration Guide ... 33
6.1 Introduction ... 33
6.2 System installation ... 33
6.3 System administration ... 33
6.4 Sanity check procedures ... 34
6.4.1 End to End testing ... 35
6.4.2 List of Running Processes ... 36
6.4.3 Network interfaces Up & Open ... 36
6.4.4 Databases ... 36
6.5 Diagnosis Procedures ... 36
6.5.1 Resource availability ... 36
6.5.2 Remote Service Access ... 36
6.5.3 Resource consumption ... 37
6.5.4 I/O flows ... 37
7 Gateway Data Handling - Esper4FastData OSGi - Installation and Administration Guide ... 38

7.1 Introduction 38
7.2 System installation 38
7.3 System administration 38
7.4 Sanity check procedures 38
7.4.1 End to End testing 39
7.4.2 List of Running Processes 39
7.4.3 Network interfaces Up & Open 39
7.4.4 Databases 39
7.5 Diagnosis Procedures 39
7.5.1 Resource availability 39
7.5.2 Remote Service Access 40
7.5.3 Resource consumption 40
7.5.4 I/O flows 40
8 Gateway Data Handling - Esper4FastData Mobile - Installation and Administration Guide 41
8.1 Introduction 41
8.2 System installation 41
8.3 System administration 42
8.4 Sanity check procedures 43
8.4.1 End to End testing 44
8.4.2 List of Running Processes 44
8.4.3 Network interfaces Up & Open 45
8.4.4 Databases 45
8.5 Diagnosis Procedures 45
8.5.1 Resource availability 45
8.5.2 Remote Service Access 45
8.5.3 Resource consumption 45
8.5.4 I/O flows 45

2 Backend IoT Broker GE - Installation and Administration Guide

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

2.1 Introduction

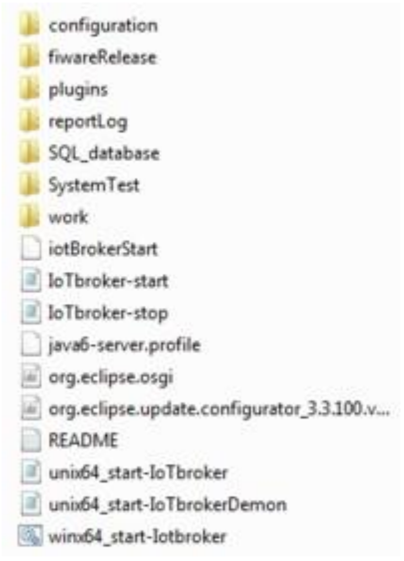
Welcome to the Installation and Administration Guide for the IoT Broker GE. The online documents are being continuously updated and improved, and will therefore be the most appropriate place to get the most up-to-date information on installation and administration.

2.2 System Installation

The IoT Broker is based on the OSGI framework. The following steps need to be performed to get the IoT Broker up & running:

1. Check if inside the system there is JavaSE 1.6 x64 installed (this is the minimum requirement).

2. The software will be provided as a zip file. The content of the zip file is the follow:



```
configuration
fiwareRelease
plugins
reportLog
SQL_database
SystemTest
work
iotBrokerStart
IoTbroker-start
IoTbroker-stop
java6-server.profile
org.eclipse.osgi
org.eclipse.update.configurator_3.3.100.v...
README
unix64_start-IoTbroker
unix64_start-IoTbrokerDemon
winx64_start-Iotbroker
```

One of the most important folders is the *fiwareRelease* folder containing the startup configuration of the IoT Broker. Before to start the IoT Broker you need to configure the path to this folder. For doing that you need to modify the *config.properties* in the *configuration* folder. The *config.properties* file is the OSGI Equinox configuration file. The only thing that needs to be changed is the *dir.config* entry; here you can set the path of the *fiwareRelease* folder (for using the default configuration it is advisible to copy the *fiwareRelease* folder into the user/home directory). It is also possible to change the IoT Broker server port by changing the *tomcat.init.port* entry (the default value is 80).

3. After that step, before to start the IoT Broker, it is advisable to have look into the *config.xml*. The default configuration is:

```
 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
 <!DOCTYPE                    properties                    SYSTEM
"http://java.sun.com/dtd/properties.dtd">
```

```
 <properties>
 <entry
key="schema_ngsi9_operation">${user.home}/fiwareRelease/iotBroke
r/config/schema/Ngsi9_Operations_v07.xsd</entry>
 <entry key="pathPreFix_ngsi9">ngsi9</entry>
 <entry key="ngsi9Uri">http://localhost:8999</entry>
 <entry key="pathPreFix_ngsi10">ngsi10</entry>
 <entry
key="schema_ngsi10_operation">${user.home}/fiwareRelease/iotBrok
er/config/schema/Ngsi10_Operations_v07.xsd</entry>
 <entry key="pub_sub_addr">http://localhost:8070</entry>
 <entry
key="path_config_folder">/fiwareRelease/iotBroker/config/</entry
>
 <entry
key="Ngsi9_10_dataStructure_v07">${user.home}/fiwareRelease/iotB
roker/config/schema/Ngsi9_10_dataStructure_v07.xsd</entry>
 </properties>
```

In that file it is possible to modify many things. By default, all pointers to xsd files refer to the *fiwareRelease* folder in the user/home directory. If your *fiwareRelease* folder is not in the user/home directory, you need to modify the values of *schema_ngsi9_operation*, *Ngsi9_10_dataStructure_v07*, *schema_ngsi10_operation*.

4. The last step is to run the IoT Broker. If you are on the windows machine, double-click on *winx64_start-Iotbroker*, otherwise on linux you need to run the *unix64_start-IoTbroker* script from the shell.

2.3 System Administration

The IoT Broker is using as logger system the pax logging: https://ops4j1.jira.com/wiki/display/paxlogging/Pax+Logging. The logs are stored in the *reportLog* folder that is part of the main folder. The logger configuration is in the *\IoTBroker_FIWARE_2.2.0-SNAPSHOT\configuration\configadmin\services* folder. It is possible to modify the logger level in the *org.ops4j.pax.logging* file:

```
log4j.rootLogger=INFO, ReportFileAppender, console
#Console Appender
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{ISO8601} | %-
5.5p | (%F:%M:%L) | %m%n
#Solve the digerest Tomcat logger errors
log4j.logger.org.apache.commons=WARN
log4j.logger.org.apache.commons.beanutils=WARN
log4j.logger.org.apache.struts=WARN
#File Appender
```

```
# ReportFileAppender - used to log messages in the report.log
file.

log4j.appender.ReportFileAppender=org.apache.log4j.FileAppender

log4j.appender.ReportFileAppender.File=.//reportLog//report.log

log4j.appender.ReportFileAppender.layout=org.apache.log4j.Patter
nLayout

log4j.appender.ReportFileAppender.layout.ConversionPattern= %-4r
[%t] %-5p %c %x - %m%n
```

2.4 Sanity Check Procedures

The Sanity Check Procedure is the first step that a System Administrator will do to verify that the IoT Broker GE is well installed and ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

For proceding with the Sanity Check you need to contact the IoT Broker via HTTP on: http://{IoTbroker_IP}/ngsi10/sanityCheck/

If the Response is:

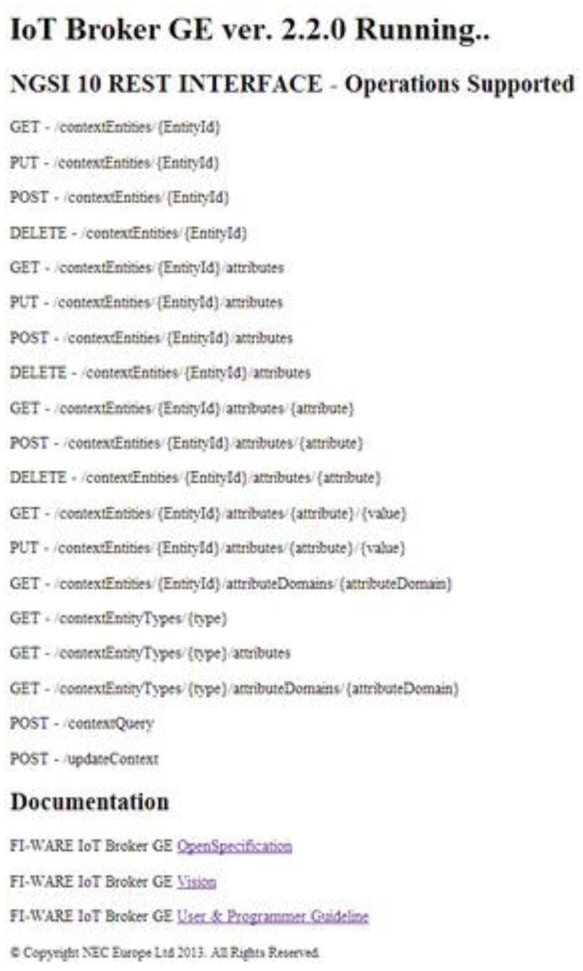
```
  <sanityCheck>

  <name>IoT Broker GE</name>

  <type>Sanity Check</type>

  <version>Version: 2.2.0.SNAPSHOT</version>

  </sanityCheck>
```

this means that the Sanity Check is passed and the IoT Broker is correctly deployed. If you get a JAVA INTERNAL ERROR or 405 METHOD NOT SUPPORTED it means that the IoT Broker is not correctly deployed.

2.4.1 End to End testing

The end-to-end test is able to send a query to the IoTbroker and receive the response.

First verify that http://IoTbroker_IP/ngsi10 can be reached and returns the following information:



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.

This page contains information about the status of the IoT Broker and which operations are supported on the current release.

After that it is possible to test one of the supported NGSI-10 resources. For example let us try to send an HTTP GET to the *contextEntity/EntityId* resource (http://{IoTbroker_IP}/contextEntities/Kitchen).

You should get back an XML response, with an ERROR CODE, similar to this:

```
<queryContextResponse>
  <errorCode>
  <code> 500 </code>
  <reasonPhrase>RECEIVER INTERNAL ERROR</reasonPhrase>
  </errorCode>
</queryContextResponse>
```

or

```
<queryContextResponse>
  <errorCode>
  <code> 404 </code>
  <reasonPhrase>CONTEXT ELEMENT NOT FOUND</reasonPhrase>
  </errorCode>
  </queryContextResponse>
```

If you get the first error this means that there is no comunication between the IoT Broker GE and the Config Managment GE. In case of the second error this means that the Entity that you are requesting is not available.

2.4.2 List of Running Processes

- Only JavaVM process

2.4.3 Network interfaces Up & Open

HTPP Standard port (80) should be accessible.

2.4.4 Databases

Embedded HSQLDB database. http://hsqldb.org/

2.5 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

2.5.1 Resource availability

- RAM = minimum 100MB
- HARD DISK = minimum 100MB

2.5.2 Remote Service Access

- TCP connection to Pub/Sub Broker GE (via NGSI-10)
- TCP connection to Data Handling GE (via NGSI-10)
- TCP connection to Configuration Management Component (via NGSI-9)

Please make sure port 80 is accessible.

2.5.3 Resource consumption

- Memory consumption = 20MB (idle state)
- CPU Usage = 0% (idle state)
- Thread runnning = 21 (idle state)

2.5.4 I/O flows

The only expected I/O flow is of type HTTP, on standard port 80 (Tomcat Server).

3 Backend Configuration Manager - Installation and Administration Guide

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

3.1 Installation and Preparation

The Configuration Management (CM) component of the Things Management GE is provided as an RPM package and thus easily installed using yum:

% yum install <RPM-file>

Apart from the external dependencies, and startup files, this RPM package consists of one executable only - 'iotConfigMgr', installed under /usr/bin/iotConfigMgr.

Internally, iotConfigMgr maintains its data on a MySQL database. The main advantages of this is that a huge amount of data can be supported (not limited on the amount of RAM in the system) and that the iotConfigMgr will not loose any of its data in case it needs to be restarted. Thus, in order for iotConfigMgr to work, MySQL must be installed and prepared.

3.1.1 MySQL installation and startup

```
% sudo yum install mysql-server.x86_64

% sudo service mysqld start
```

3.1.2 MySQL preparation

```
% mysql -u root

mysql> create database cm;

mysql> create user 'cm' identified by 'cm';

mysql> grant all on cm.* to 'cm'@'localhost' identified by
'cm';
```

As the MySQL commands suggest, the name of the database is 'cm', the username is 'cm' and its password is also 'cm'. These are also the default values for the corresponding command line options of iotConfigMgr. It is highly recommended to alter these values when preparing the database and of course use the three command line options to indicate the correct values wheb starting iotConfigMgr.

3.2 Starting iotConfigMgr

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

- -u (print the complete usage on screen)
- -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. At any time, iotConfigMgr can be killed (Unix standard: Ctrl-C) and restarted (without using the '-reset' option) and in the second run, the same data content will be available as in the first. If the option '-reset' is used when starting iotConfigMgr, all database table will be emptied at startup.

3.3 Log file

iotConfigMgr maintains a log file at /tmp/iotConfigMgrLog and the level of debugging is determined by the following command line options:

- -t <trace level>
- -v (verbose mode)
- -vv (verbose2 mode)
- -vvv (verbose3 mode)
- -vvvv (verbose4 mode)
- -vvvvv (verbose5 mode)
- -d (debug mode)
- -r (reads mode)
- -w (writes mode)

The '-t' option is followed by a comma-separated list of ranges of trace levels, from 0 to 255. The significance of each of the trace levels is explained in [User Guide]. The trace levels are used for specific actions, while the verbose levels are used for more general debug output. If you just want to know what the program is doing, on a higher level, the verbose levels whould be used, whereas if something fails, a specific trace level should be set.

The five levels of verbose mode is pretty self explanatory - the more v:s, the more verbose output.

The '-d' option is depracated and is not used.

The '-r' and '-w' options are to be set if monitorization of incoming and outgoing packages is desired.

All these debug levels can be modified and viewed using the REST interface of iotConfigMgr.

3.4 REST Interface

3.4.1 Modifying/Getting verbose level

- curl --request DELETE <host>:<port>/log/verbose
- curl --request GET <host>:<port>/log/verbose

- curl --request PUT <host>:<port>/log/verbose/off
- curl --request PUT <host>:<port>/log/verbose/[0-5]

3.4.2 Modifying/Getting debug level - depracated but still implemented ...

- curl --request DELETE <host>:<port>/log/debug
- curl --request GET <host>:<port>/log/debug
- curl --request PUT <host>:<port>/log/debug/on
- curl --request PUT <host>:<port>/log/debug/on

3.4.3 Modifying/Getting reads level

- curl --request DELETE <host>:<port>/log/reads
- curl --request GET <host>:<port>/log/reads
- curl --request PUT <host>:<port>/log/reads/on
- curl --request PUT <host>:<port>/log/reads/on

3.4.4 Modifying/Getting writes level

- curl --request DELETE <host>:<port>/log/writes
- curl --request GET <host>:<port>/log/writes
- curl --request PUT <host>:<port>/log/writes/on
- curl --request PUT <host>:<port>/log/writes/on

3.4.5 Modifying/Getting trace levels

- curl --request DELETE <host>:<port>/log/trace
- curl --request DELETE <host>:<port>/log/trace/t1
- curl --request DELETE <host>:<port>/log/trace/t1-t2
- curl --request DELETE <host>:<port>/log/trace/t1-t2,t3-t4
- curl --request GET <host>:<port>/log/trace
- curl --request PUT <host>:<port>/log/trace/t1
- curl --request PUT <host>:<port>/log/trace/t1-t2
- curl --request PUT <host>:<port>/log/trace/t1-t2,t3-t4

3.4.6 Getting version of running executable

- curl --request GET <host>:<port>/version

3.5 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

3.5.1 End to End testing

Login to any host from where the iotConfigMgr host is visible and issue the command:
% curl <iotConfigMgr host>:1025/version

Expected Output: <ngsi><key>version</key><value>0.0.1</value></ngsi>

Note that the port (1025 is the default port) depends on how iotConfigMgr was started. iotConfigMgr is configurable in this sense ...

3.5.2 List of Running Processes

- iotConfigMgr
- mysqld

3.5.3 Network interfaces Up & Open

- TCP:1025 (depending on command line options for iotConfigMgr)
- TCP:3306

As stated before, the port used is 1025, unless iotConfigMgr has been started with the command line option '-port' changing the default port number.

3.5.4 Databases

As always, iotConfigMgr is pretty configurable, but if the default values are used:

MySQL(database='cm', user='cm', password='cm')

Simple test:

```
% mysql -user cm -p cm
password> cm
mysql> SELECT * FROM entity;
```

3.6 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

3.6.1 Resource availability

Whatever MySQL needs for a small amount of data. The iotConfigMgr will adapt to current conditions.

3.6.2 Remote Service Access

IoT devices will connect directly to the iotConfigMgr. The input needed to connect is the host where the iotConfigMgr is running and the port where iotConfigMgr listens for connections. The default port value is 1025, but that can be changed using the '-port' command line option at starting iotConfigMgr. The iotConfigMgr itself connect to the Pub/Sub Broker and the host and port for this connection is defined when starting iotConfigMgr (options -psBrokerHost, -psBrokerPort).

3.6.3 Resource consumption

Whatever MySQL needs for a small amount of data. The iotConfigMgr will adapt to current conditions.

3.6.4 I/O flows

Input flows:

- The ConfMan listen to connections in the port especified with -port command line argument.

Output flows:

- The ConfMan will forward registrations to endpoind specified in -psbHost and -psbPort command line arguments.

4 Gateway Data Handling GE - Installation and Administration Guide

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

- Gateway Data Handling - SOL/CEP Complex Event Processor - Installation and Administration Guide
- Gateway Data Handling - Esper4FastData Servlet - Installation and Administration Guide
- Gateway Data Handling - Esper4FastData OSGi - Installation and Administration Guide
- Gateway Data Handling - Esper4FastData Mobile - Installation and Administration Guide

5 Gateway Data Handling - SOL/CEP Complex Event Processor - Installation and Administration Guide

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

5.1 Purpose and Audience

The purpose of this document is to describe how to install and administrate the SOL/CEP Complex Event Processor (referred to as the "Application", the "Software" or "SOL/CEP")

The Guide assumes basic Unix system administration knowledge.

The text 'Not Used' indicates that a particular feature cannot be used for the current release.

5.2 System Requirements

The Application will run correctly when installed on a system having the requirements listed in the sections below.

5.2.1 Hardware Requirements

The following table contains the minimum resource requirements for running the Repository:

Resource	**Requirement**
CPU	1-2 cores with at least 2.0 GHZ
Physical RAM	50MB
Disk Space	4GB The actual size will depend on the amount of logging and configuration of the Application.

5.2.2 Operating System Support

The Application has been tested on Debian Linux 6.0, Linux kernel 2.6.32-5-amd64.

This Installation Guide describes the installation process as based on mentioned operating system.

5.2.3 Software Requirements

Assuming a base installation of Debian was performed, no additional software components need to be installed. The unit tests depend on the presence of the `netcat` application.

In this documentation, the `sudo` command is used to perform `root`-level commands. The installation and use of this command is at the discretion of the system administrator. When used, the `solcep` user must be a valid 'sudoer'. Configuring this is outside the scope of this manual.

5.3 Software Installation

The software is delivered as an archive called `solcep.tar.gz`

For these steps, a functioning machine with a Debian 6.0 base installation is assumed, as well

as **root** access rights, or a valid `sudo`.

5.3.1 Delivery Contents

The delivery contains the following files.

- `Manifest.txt` - A file containing this enumeration.
- `Copyright.txt` - Copyright notice
- `solcep` - Standalone SOL/CEP binary
- `solcep.conf.xml` - Configuration file
- `solcep_ctrl` - Used for Debian service management integration.
- `channels.dolce` - Dolce complex event specification.
- `plugins/libUDPListenerPlugin.so.1.0` - Listener that accepts events in the Simple Event Format on a specified UDP port.
- `plugins/libUDPEmitterPlugin.so.1.0` - Emitter that generates complex events in the Simple Event Format on a specified UDP port.
- `plugins/libSimpleDecoderPlugin.so.1.0` - Decodes events specified in the Simple Event Format.
- `plugins/libSimpleEncoderPlugin.so.1.0` - Encodes complex events to the Simple Event Format.
- `UnitTest/channel1Event.evt` - Channel Test event #1
- `UnitTest/channel3Event.evt` - Channel Test event #2
- `UnitTest/channel5Event.evt` - Channel Test event #3

5.3.2 Installation

5.3.2.1 *SOL/CEP*

- Create a new user and home directory. It is suggested the user be called `solcep`.

```
sudo adduser solcep
```

- Unpack the contents into this directory.

```
tar -xvzf /location/of/solcep.tar.gz
```

- Ensure the execution bit of `solcep` is enabled.

```
chmod +x solcep
```

5.3.2.2 *Server Infrastructure*

- Copy `solcep_ctrl` into `/etc/init.d`
- Make sure the execution bit is set.

```
chmod +x solcep_ctrl
```

- Edit the `SOLCEP_HOME` and `SOLCEP_BIN` variables, so they match the `solcep` executable and the directory. Also set any command line options.
- Register the Application with the Debian service infrastructure.

```
sudo update-rc.d solcep_ctrl defaults
```

- Edit the `solcep.conf.xml` Configuration file.
- SOL/CEP is now set up on the system and will be started at boot time.
- It can be manually started now

```
/etc/init.d/solcep_ctrl start.
```

- The `/etc/init.d/solcep_ctrl` command can be executed at any time to control the Application.

The Sanity check procedures and the Diagnosis procedures can be followed for initial tests.

5.4 Configuration

Note that in it's current version, the Application needs to be restarted for the changes to be applied. This is also true for **any changes in the Dolce complex event specification**.

This is achieved with the `/etc/init.d/solcep_ctrl restart` command.

5.4.1 Configuration file

The Configuration File is shipped with the application, as mentioned in the previous sections.

To understand the configuration file, it helps to understand the structure of the Application. The Application is divided into the Event Collector, Complex Event Detector and Complex Event Publisher modules.

The Event Collector listens and decodes events. It allows the configuration of multiple listeners each with an associated decoder by means of the **<listeners>** and **<decoder>** tags.
The Complex Event Publisher encodes and emits complex events. It too, allows the configuration of multiple emitters, each with an associated encoder, using the **<emitters>** and **<encoder>** tags.
Multiple Complex Event Detector engines can be configured, with can have associated listeners and emitters, by means of the **<acceptedListeners>** and **<requiredEmitters>** tags.

The next section explains the structure in more detail. Also note that the structure coincides almost entirely with the REST-based Administration interface described in the Open Specifications

5.4.1.1 *<solcep>*

This is the top level of the configuration file.

<comment> Not used. Free text comments for administrative purposes
<date> Not used. Free text for administrative purposes
<serverRoot> Must be an absolute, existing path, without a trailing '/'. It indicates the base path for log files and configuration files.

5.4.1.2 *<coordinator>*

General configuration options

<portRange>, declares a range of ports SOL/CEP can use. The *start* attribute indicates the beginning of the range, with the size of the range specified in the *size* attribute. SOL/CEP will automatically find and allocate TCP/IP ports for the internal communications between the different modules within this port range.
<admin>
<port> The port on which the REST administration interface listens.
<logger>
<port> Port where the logger listens to log messages.
<file> The name of the log file. This must be a valid file name, relative to the **<serverRoot>**.
<level> The log level.

5.4.1.3 *<eventCollector>*

This sections declares the listeners and their decoders used to capture and transform events to be injected into the complex event detector engine.

<listeners> Declares all the listeners.
<listener> Declares a single listener, identified by *id*.
<enabled> If set to *1*, the listener is enabled.
<plugin> Describes the plugin used for the Listener.
<name> Must be the exact name of the configured plugin in the *plugin* directory.
<comment> Descriptive text for the plugin.
<params> Plugin parameters. Depend on the plugin manufacturer. At the end of this section an list provided for the different plugins.
<param-1> First plugin parameter. It is listed here for illustrative purposes.
<param-N> N-th plugin parameter. It is listed here for illustrative purposes.
<decoderRef> The decoder that the listener uses. It must reference a valid decoder by its *id* as described in the *<decoders>* section.

<decoders> Declares the decoders that can be used by the listeners.
<decoder> Declares a decoder, identified by *id*.
<enabled> If set to *1*, the decoder is enabled.
<plugin> Describes the plugin used for the decoder. The same structure applies as for those in the *<listeners>* section above. See also the plugin descriptions for an overview.

5.4.1.4 *<complexEventDetector>*

This section specifies the Complex Event Detectors. It consists of a number of socalled Detection Specifications, each of which in fact is a separate detection engine accepting events and emitting complex events.

<detectionSpecifications> Starts the detection specification section.
<detectionSpecification> Declares a single detection specification. The *id* attribute represents the file name of the Dolce specification, as located in *<serverRoot>*, without the `.dolce` extension. For example, if *id* is set to `test`, then a Dolce specification file named `test.dolce` is expected to be found.
<enabled> If set to *1*, the detection engine associated to the specification is enabled.
<acceptedListeners> Specifies the listeners from which events are accepted.
<listenerRef> The listener in question. It must reference a valid listener by its *id* as described in the *<listeners>* section above.
<requiredEmitters> Specifies the emitters to which complex events are published.
<emitterRef> The emitter in question. It must reference a valid emitter by its *id* as described in the *<emitters>* section below.

5.4.1.5 *<complexEventPublisher>*

This sections declares the emitters and their encoders used to publish the complex events generated by the complex event detector engines.

<emitters> Marks the top level of the emitters section.
<emitter> Declares a single emitter, identified by *id*.
<enabled> If set to *1*, the emitter is enabled.
<plugin> Describes the plugin used for the emitter. The same structure applies as for those in the *<listeners>* section above. See also the plugin descriptions for an overview.
<encoderRef> The encoder that the emitter uses. It must reference a valid encoder by its *id* as described in the *<encoders>* section.

<encoders> Declares the encoders that can be used by the emitters.
<encoder> Declares a single encoder, identified by *id*.
<enabled> If set to *1*, the encoder is enabled.
<plugin> Describes the plugin used for the encoder. The same structure applies as for those in the *<listeners>* section above. See also the plugin descriptions for an overview.

5.4.2 Configuration file validation

In general, the Application does not do any validation on the XML file. In this release, it is the responsibility of the Administrator to make sure the file is correctly specified.

Especially, the following must be taken into account:

- The Application does not check if the configured ports are already in use. The Administrator must use `netstat` to find this out.
- The Application does not check if the listeners are referring to existing decoders, nor if emitters are referring to existing encoders. Likewise, the

required emitters and accepted listeners by the detection specifications are not cross-checked.

In any case, anomalies and failures will be reported in the log file at run time.

5.4.3 Configuration File Location

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. (This means that if the Application is in `/home/solcep`, and it is started from `/tmp`, the configuration file is expected in `/tmp`.)

The location can be overridden using command line options; in fact, this is the recommended procedure.

5.4.4 Command Line options

SOL/CEP accepts the following command line options. Note that command line options override any

setting specified in the configuration file.

`-c <configFile>` Specifies an alternative configuration file.

`-l <logLevel>` Indicates the log level. The higher the level, the more information is displayed.

0 - no logging

1 - log fatal errors

2 - as in level 1, also log normal errors

3 - as in level 2, also log warnings

4 - as in level 3, also log information

`-d` Duplicates the information written to the log file to the console.

`-v` Show version information.

NOTE: Command line options must be added to `/etc/init.d/solcep_ctrl` for them to take effect at startup.

5.4.5 Plugin Parameters

This section describes the plugin parameters used for each plugin. All error codes generated by the plugins are recored to the log file.

5.4.5.1 *UDPListenerPlugin*

This plugin listens to a UDP port for incoming events. It does not check for the correctness or conflicts of the port number - it will load, but may not function correctly in such cases.

Parameters

Name: - port
Type: - integer
Default value: - none.

Description: - The UDP port to which to listen for incoming events.
Behaviour: - The plugin will fail to load if this parameter is not specified.

Name: - maxPacketSize
Type: - integer
Default value: - 1024
Description: - The maximum size of a packet that can be handled.
Behaviour: - Packets bigger than this value may not be handled correctly.

Error codes

100 - Cannot create connection

101 - Cannot listen to the connection.

5.4.5.2 *UDPEmitterPlugin*

This plugin emits complex events to a UDP port specified in the parameters. It does not check for the correctness or conflicts of the port number - it will load, but may not function correctly in such cases.

Parameters

Name: - port
Type: - integer
Default value: - none.
Description: - The UDP port to which events are emitted.
Behaviour: - The plugin will not work correctly if this value is not specified.

Name: - address
Type: - string
Default value: - none.
Description: - The IP Address to which to send the data. *NOTE*: Literal IP addresses must be used.
Behaviour: - The plugin will not work correctly if this value is not specified.

Error codes

100 - Cannot create connection

101 - Invalid port configured.

102 - Send error.

5.4.5.3 *SimpleDecoderPlugin*

This plugin decodes events in the Simple Event Format and converts them to the internal SOL/CEP format.

Parameters

None.

Error codes

100 - Cannot decode the event.

5.4.5.4 ***SimpleEncoderPlugin***

This plugin encodes internal SOL/CEP complex events into the Simple Event Format.

Parameters

None.

Error codes

100 - Cannot decode the internal SOL/CEP event.

5.5 Troubleshooting steps

Normally, an Application malfunction is not even due to lack of resources, but a matter of configuration.

The following steps should help troubleshooting the application.

- Make sure that SOL/CEP is up and running. This can be verified by issuing the command:

`/etc/init.d/solcep_ctrl status`. Alternatively, do a `ps -e | grep solcep`.

- Make sure that the log file referred to in the configuration file exists and is writeable by the Application.
- Also make sure that the log level is set to at least 1, and not overridden by a lower level using a command line option, either manual or in the `/etc/init.d/solcep_ctrl` file.
- Make sure that the Dolce complex event specification referred to in the configuration file exists and is readable by the Application.
- Inspect the log file for any error messages.
- It is possible that the logging system is not up and running due to earlier errors. These earlier errors are always written to the console.
  -
    - If problems persist, then augment the log level and manually restart the Application: `/etc/init.d/solcep_ctrl restart`.
  -
    - Closely look at any messages written to the console.

5.6 Sanity check Procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. It is a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

5.6.1 End to End testing

The following steps are to be performed to ensure the Application is working correctly.

It is assumed that all installation steps have been followed correctly. If anything is not

working correctly, it is recommended to consult the diagnosis procedures

especially the troubleshooting section.

- Make sure that SOL/CEP is up and running. This can be verified by issuing the command:

```
/etc/init.d/solcep_ctrl status.
```

- Start a 'server' on the machine that listens to the port as configured for the *emitter plugin*, as specified in the plugin parameters section. This can best be done in a different terminal window.

```
nc -l -u -p<emitter-port>
```

- Launch the following two commands within 30 seconds of each other. Make sure that the port corresponds to that configured for the *listener plugin*, as specified in the plugin parameters section.

```
netcat -q 0 -u localhost <listener-port> < UnitTest/channel1Event.evt
netcat -q 0 -u localhost <listener-port> < UnitTest/channel3Event.evt
netcat -q 0 -u localhost <listener-port> < UnitTest/channel5Event.evt
```

- In the terminal window with the 'server', a message should appear, corresponding to the name of the complex event specified in the Dolce detection specification file.

5.6.2 List of Running Processes

Listing

```
ps -e | grep sol
```

Result

```
0  S   1000   1506   1250   1   80    0  -  77054  -          tty2
00:00:00 solcep
```

5.6.3 Network interfaces Up & Open

To check the ports in use and listening, *two* commands should be executed.

List the TCP/IP listening sockets.

```
$ sudo netstat -ltp
```

Result

```
 Active Internet connections (only servers)

 Proto  Recv-Q  Send-Q  Local  Address   Foreign  Address   State
PID/Program name
```

```
 tcp         0       0 *:33066         *:*                LISTEN
705/rpc.statd


 tcp         0       0 *:sunrpc        *:*                LISTEN
687/portmap


 tcp         0       0 *:5554          *:*                LISTEN
1760/solcep  <------


 tcp         0       0 *:5555          *:*                LISTEN
1760/solcep  <------


 tcp         0       0 *:ssh           *:*                LISTEN
1428/sshd


 tcp         0       0 localhost:smtp *:*                 LISTEN
1202/exim4
```

List the UDP listening sockets.

$ sudo netstat -lup

Result

```



 Proto  Recv-Q  Send-Q  Local  Address   Foreign  Address    State
PID/Program name


 udp                 0             0   *:bootpc               *:*
1359/dhclient


 udp                 0             0   *:sunrpc               *:*
687/portmap


 udp                0            0   *:881                    *:*
705/rpc.statd


 udp                 0             0   *:29201                *:*
1760/solcep <--------


 udp                 0             0   *:43154                *:*
705/rpc.statd
```

The ports are expected to reflect those specifed in the configuration file.

5.6.4 Databases

SOL/CEP does not use databases.

5.7 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in the Application. It is to be considered a first line of support diagnosis; once identified , it can be passed onto a higher level for specific analysis. This however, is out of the scope for this section.

5.7.1 Resource availability

The Application will function correctly if the system adheres to the minimal requirements.

5.7.2 Remote Service Access

The REST administration interface is a HTTP service located on the port specified in the configuration file.

Assuming that the port is configured to *8000*, the following example commands can be issued to test if it is working correctly.

```
curl -i localhost:<adminPort>/solcep/coordinator/logger/level
```

The following output should be returned

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 24

<name>solcep.log</name>
```

Other commands are:

```
curl -i localhost:8000/solcep/coordinator/logger/name
curl -i localhost:8000/solcep/eventCollector/listeners
curl -i localhost:8000/solcep/complexEventDetector/detectionSpecifications
```

See the REST Admin specification for more information about the supported RESTful commands.

5.7.3 Resource consumption

The Application needs at least 10 MB of free memory to work correctly, as well as 100 MB of free disk space. Within these margins, the Application should function satisfactorily.

CPU

The CPU load of the Application depends on the amount of events that are being processed. It is to be noted that SOL/CEP uses multi-threading, which means that potentially 100% of the machine's processing capacity can be used.

Currently, stress testing is performed in our lab, so no hard performance and resource figures can be given up to this moment.

Networking and I/O

The Application will normally be configured to receive and emit events on network ports. Depending on the amount of events to be processed, the network can become stressed. The Application uses network based message passing (with currently no option for inter process message passing), which also adds to the stress of the network.

Resource consumption strongly depends on the load, especially on the number of concurrent requests.

5.7.4 I/O flows

The expected I/O flow is of type TCP/IP and UDP on the port specified in the configuration file, both

inbound and outbound.

—•—

6 Gateway Data Handling - Esper4FastData Servlet - Installation and Administration Guide

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

6.1 Introduction

Welcome to the "Esper4FastData Servlet" Installation and Administration Guide.

This online document is continuously updated and improved to provide the most up-to-date informations.

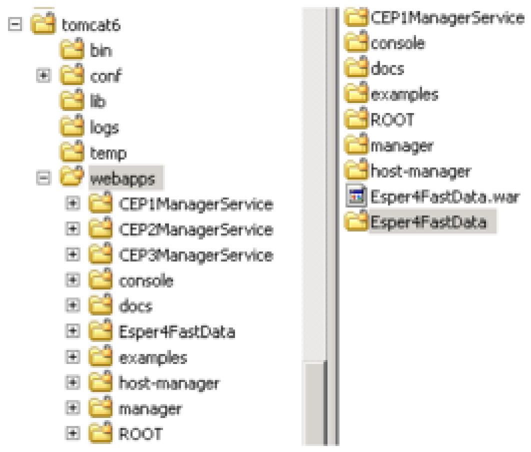
The "Esper4FastData Servlet" software provides CEP features. It exposes REST API in order to be remotely administrated. This REST API can easily be tested with the SoapUI testing tool. There is a provided SoapUI project that runs a test suite, in order to test easily the main available functionalities of the enabler.

6.2 System installation

The "Esper4FastData Servlet" runs in a servlet container environment, like Tomcat.

This installation procedure should be performed with an up and running Tomcat 6 server on a Windows operating system.

1. Download the Esper4FastData-2.2.3_Servlet.zip file. It contains both the .war application and the Esper4FastDataLocal-soapui-project.xml test file.
2. Extract the Esper4FastData.war file that is inside the Esper4FastData-2.2.3_Servlet.zip file.
3. Copy the .war file to the {Tomcat6Root}\webapps directory. The file should be automatically deployed in a directory with the same name, as shown below:



6.3 System administration

The Esper4FastData engine uses the Apache log4j logging system. Log4j is configurable through its log4j.properties configuration file, located in {Tomcat6Root}\Esper4FastData\WEB-INF\classes.

Here is the configuration file. By default, logs are available on the standard output:

```
log4j.rootLogger=DEBUG, stdout


log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d         [%-5p]
(%F:%M:%L) %m%n
```

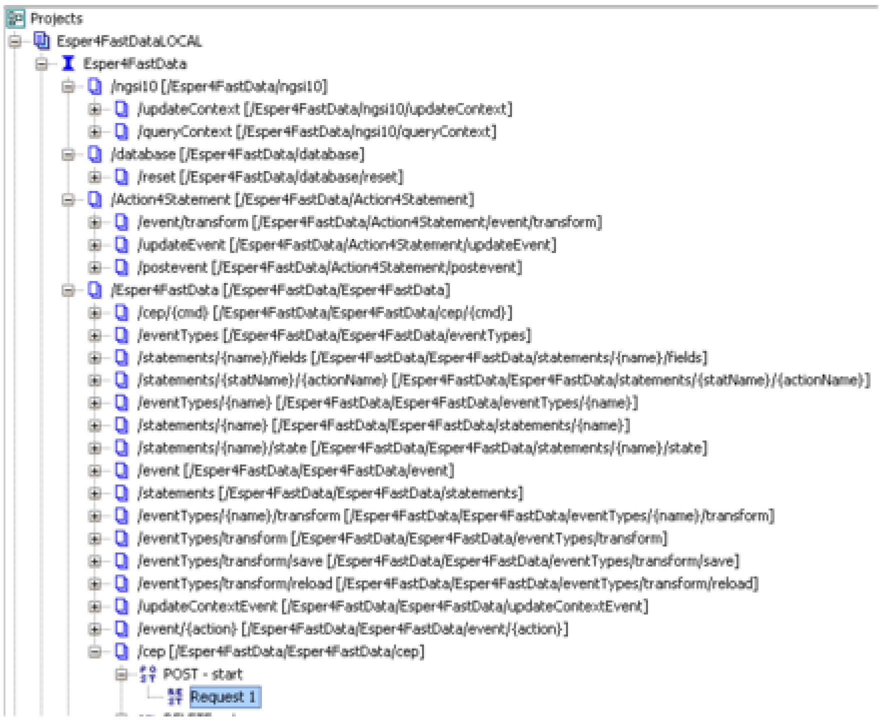
Log sample:

```
2013-04-26 14:46:57,004 [DEBUG] (Esper4FastData.java:pause:795)
CEP engine pause

2013-04-26 14:46:57,020 [DEBUG] (Esper4FastData.java:pause:795)
CEP engine resume

2013-04-26  14:46:57,036  [DEBUG]  (Esper4FastData.java:stop:764)
stop CEP engine

2013-04-26                 14:46:57,036                 [DEBUG]
(TimerServiceImpl.java:stopInternalClock:100)  .stopInternalClock
Stopping internal clock daemon thread

2013-04-26                 14:46:57,239                 [DEBUG]
(MetricScheduleService.java:clear:49)    Clearing    scheduling
service

2013-04-26                 14:46:57,239                 [DEBUG]
(FilterServiceImpl.java:destroy:52) Destroying filter service

2013-04-26                 14:46:57,239                 [DEBUG]
(SchedulingServiceImpl.java:destroy:49)   Destroying   scheduling
service

2013-04-26                 14:46:57,239                 [DEBUG]
(SchedulingMgmtServiceImpl.java:destroy:38)           Destroying
scheduling management service

2013-04-26 14:47:07,020 [DEBUG] (Esper4FastData.java:start:744)
start CEP engine

2013-04-26           14:47:07,036            [INFO          ]
(EPServiceProviderImpl.java:doInitialize:393)       Initializing
engine URI 'default' version 4.6.0

2013-04-26                 14:47:07,036                 [DEBUG]
(MetricReportingPath.java:setMetricsEnabled:38)          Metrics
reporting has been disabled, this setting takes affect for all
engine instances at engine initialization time.
```

6.4 Sanity check procedures

The Sanity Check Procedure is the first step that a System Administrator will do to verify that the Data Handling GE is well installed and ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation. SoapUI is used for the sanity check.

1. Ensure that the proxy is disabled in SoapUI: File->Preferences->Proxy Settings. This setting is mandatory to get SoapUI working properly on localhost.
2. Open the Esper4FastDataLocal-soapui-project.xml file available in the downloaded Esper4FastData-2.2.3_Servlet.zip
3. Open a web browser and go to http://127.0.0.1/Esper4FastData/application.wadl. This should display a WADL description file like this.
4. Contact the Esper4FastData servlet via HTTP POST on: http://127.0.0.1/Esper4FastData/Esper4FastData/cep. The location of this resource within the SoapUI project is documented in the snapshot below:



```
Projects
  Esper4FastDataLOCAL
    Esper4FastData
      /ngsi10 [/Esper4FastData/ngsi10]
        /updateContext [/Esper4FastData/ngsi10/updateContext]
        /queryContext [/Esper4FastData/ngsi10/queryContext]
      /database [/Esper4FastData/database]
        /reset [/Esper4FastData/database/reset]
      /Action4Statement [/Esper4FastData/Action4Statement]
        /event/transform [/Esper4FastData/Action4Statement/event/transform]
        /updateEvent [/Esper4FastData/Action4Statement/updateEvent]
        /postevent [/Esper4FastData/Action4Statement/postevent]
      /Esper4FastData [/Esper4FastData/Esper4FastData]
        /cep/{cmd} [/Esper4FastData/Esper4FastData/cep/{cmd}]
        /eventTypes [/Esper4FastData/Esper4FastData/eventTypes]
        /statements/{name}/fields [/Esper4FastData/Esper4FastData/statements/{name}/fields]
        /statements/{statName}/{actionName} [/Esper4FastData/Esper4FastData/statements/{statName}/{actionName}]
        /eventTypes/{name} [/Esper4FastData/Esper4FastData/eventTypes/{name}]
        /statements/{name} [/Esper4FastData/Esper4FastData/statements/{name}]
        /statements/{name}/state [/Esper4FastData/Esper4FastData/statements/{name}/state]
        /event [/Esper4FastData/Esper4FastData/event]
        /statements [/Esper4FastData/Esper4FastData/statements]
        /eventTypes/{name}/transform [/Esper4FastData/Esper4FastData/eventTypes/{name}/transform]
        /eventTypes/transform [/Esper4FastData/Esper4FastData/eventTypes/transform]
        /eventTypes/transform/save [/Esper4FastData/Esper4FastData/eventTypes/transform/save]
        /eventTypes/transform/reload [/Esper4FastData/Esper4FastData/eventTypes/transform/reload]
        /updateContextEvent [/Esper4FastData/Esper4FastData/updateContextEvent]
        /event/{action} [/Esper4FastData/Esper4FastData/event/{action}]
        /cep [/Esper4FastData/Esper4FastData/cep]
          POST - start
            Request 1
```

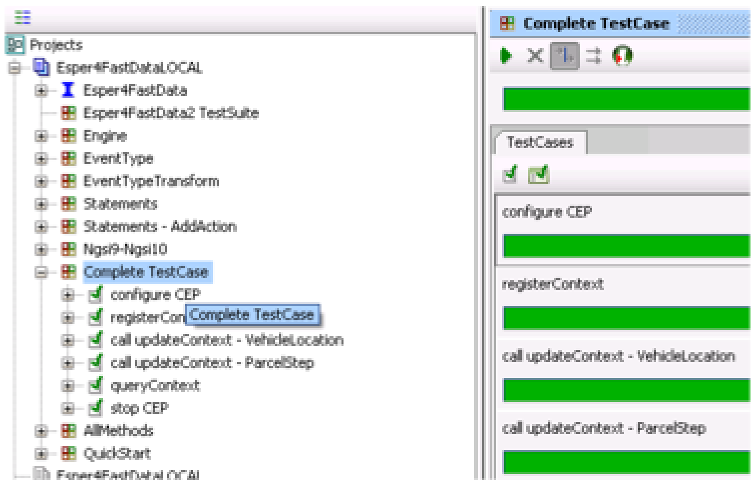
The answer should be

```
<data                               contentType="text/plain"
contentLength="76"><![CDATA[engine        with         database
../webapps/Esper4FastData/WEB-INF/resource/

started now]]></data>
```

6.4.1 End to End testing

The end-to-end testing is done with SoapUI.

In the aforementioned SoapUI project, double-click on the complete TestCase and run it. This tests all the possible available features in the DataHandling GE. No HTTP error code should be returned, otherwise the end-to-end test should be considered as failed.



6.4.2 List of Running Processes

Tomcat must be running.

6.4.3 Network interfaces Up & Open

HTTP port 80 must be reachable.

6.4.4 Databases

The Esper4FastData Servlet configuration of DataHandling GE hosts its own embedded database. No other database is required.

6.5 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

6.5.1 Resource availability

Resources requirements can vary a lot depending on the servlet container that is used and the context.

For Tomcat 6:

- 100 MBytes of available disk space at least
- 512 MBytes of RAM at least
- 1Ghz or more, single or multi-core CPU

6.5.2 Remote Service Access

Through REST, port 80, HTTP protocol.

HTTP application root is http://{ServerName}/Esper4FastData/

6.5.3 Resource consumption

150 MBytes of RAM

6.5.4 I/O flows

Port 80, HTTP

7 Gateway Data Handling - Esper4FastData OSGi - Installation and Administration Guide

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

7.1 Introduction

Welcome to the "Esper4FastData OSGi" Installation and Administration Guide.

This online document will be updated and improved to provide the most up-to-date informations.

The "Esper4FastData OSGi" software provides CEP features. It exposes SOAP API in order to be remotely administrated. This SOAP API can easily be tested with the SoapUI testing tool.

The project is delivered with a SOAPUI project that has been generated from the WSDL file, this SOAPUI project does also contain a TestSuite used to executes the main methods with the right parameters already filled in.

7.2 System installation

Requirements

- Linux OS
- JavaSE >= 1.6
- Esper4knopflerfish

Installation steps

- Download the esper4knopflerfish.zip
- Unzip the file
- Go into the unzipped folder and grant execution rights to start.sh (chmod u+x start.sh)

7.3 System administration

Running the software:

```
nohup ./start.sh > log.txt &
```

Stopping the software:

Get the PID of the process with the following command:

```
ps -ef | grep osgi
```

Stop the process with the following command:

```
kill -9 PidNumber
```

The log is available in the esper4knopflerfish directory. The file name is log.txt.

7.4 Sanity check procedures

Call the following in a web browser:http://{ServerRoot}:8084/axis2/services/ManagerService?wsdl When the CEP bundle is deployed and correctly registered in the axis bundle, it displays a WSDL that allows the sanity check to be considered as passed.

7.4.1 End to End testing

Requirements

- SoapUI
- Reachable WSDL
- The SoapUI project esper-soapui-workspace.xml.zip

Steps

- Download and install SoapUI
- Unzip the esper-soapui-workspace.xml.zip file
- Open the project from SOAPUI

Run all the pre-built tests provided in the EsperOSGITestSuite section.

7.4.2 List of Running Processes

There is a single java process running:

```
java   -jar   framework.jar   -create   --xargs   init.xargs   -
Forg.osgi.framework.system.packages.extra........
```

7.4.3 Network interfaces Up & Open

Input:TCP Port 8084

Output:HTTP requests

7.4.4 Databases

No database is needed

7.5 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

7.5.1 Resource availability

Resources requirements may vary a lot depending on the context.

100 MBytes of disk space at least 512 MBytes of RAM at least 1 GHz or more CPU

7.5.2 Remote Service Access

Through SOAP Web Services only, HTTP protocol on port 8084. The access URI follows this pattern:

http://{ServerRoot}:8084/axis2/services/ManagerService?wsdl

7.5.3 Resource consumption

- 40 Mbytes of disk space
- 5 Mbytes of RAM

7.5.4 I/O flows

- SOAP protocol inside HTTP
- POST and GET HTTP requests toward event consumers

8 Gateway Data Handling - Esper4FastData Mobile - Installation and Administration Guide

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

8.1 Introduction

Welcome to the "Esper4FastData Mobile" Installation and Administration Guide.

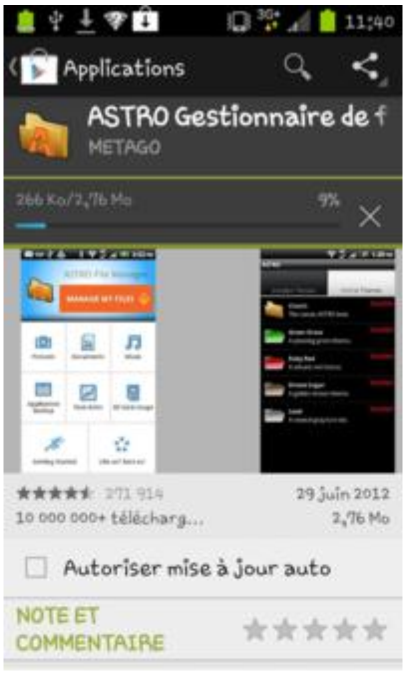
This online document is continuously updated and improved to provide the most up-to-date informations.

The "Esper4FastData Mobile" software suite provides CEP features for Android devices. It is composed of three items:

- An Android CEP software that aims to distribute CEP features among a fleet of mobile devices.
- An intermediate Google AppEngine software that provides a unique endpoint for many CEP through a REST API, and translate HTTP request into GCM, which is the standard used to communicate with Android devices.
- A CEP fleet management application that communicate toward the AppEngine software. This software is not provided. It could typically be a web GUI, but also any kind of CEP fleet management application. It's up to software developpers to build the application that fits their needs.

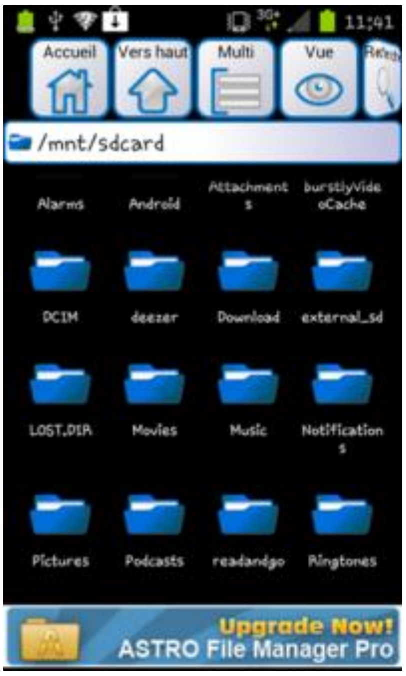
8.2 System installation

- Install Astro file manager application from Google Play store so you can browse file on the local Android filesystem

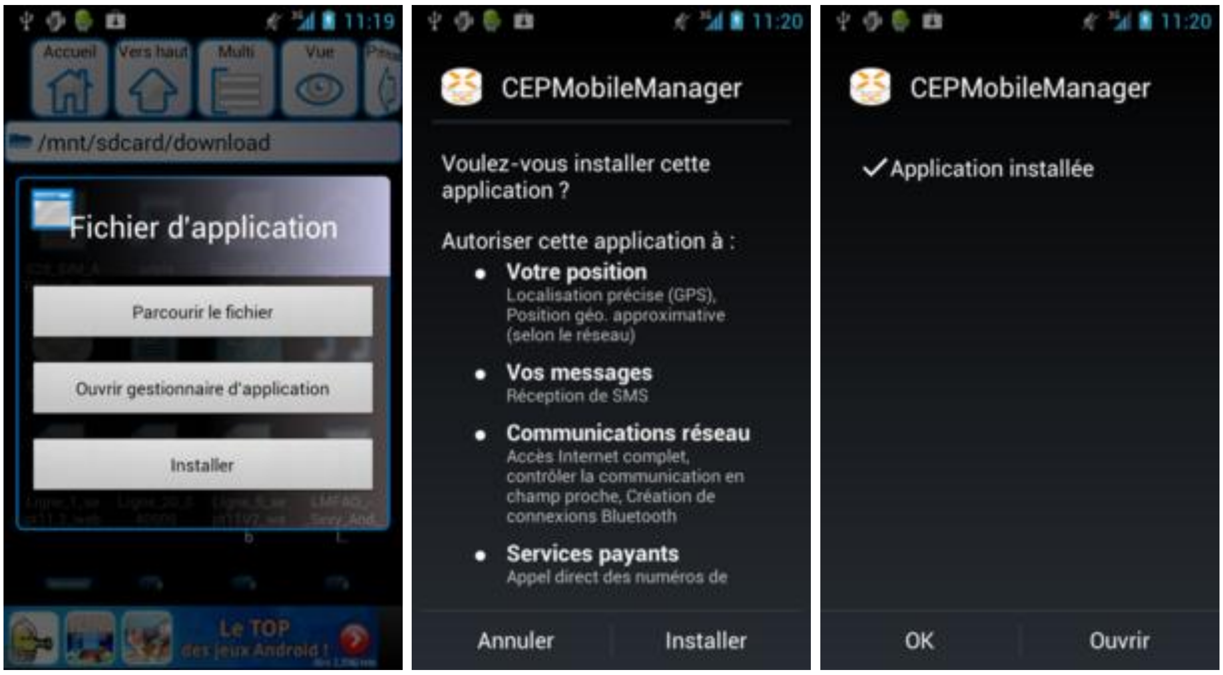


- Connect the Android device using usb to a windows pc
- Select the "enable USB storage memory" mode on the target Android device
- Upload the Esper4FastData-1.3.3_Mobile.apk on the Android device

- Change the phone state to "disable USB storage memory"
- Disconnect the USB wire
- Open Astro file manager and browse the file system to find CEPMobileManager.apk



- Click on it to install, which makes it visible in the Android application list



Voulez-vous installer cette application ?

Autoriser cette application à :

- **Votre position**
  Localisation précise (GPS), Position géo. approximative (selon le réseau)
- **Vos messages**
  Réception de SMS
- **Communications réseau**
  Accès Internet complet, contrôler la communication en champ proche, Création de connexions Bluetooth
- **Services payants**
  Appel direct des numéros de

✓Application installée

8.3 System administration

For the time being, the application needs some system level restart to work properly. After using it, the real way to stop it before starting it again is described as follows:

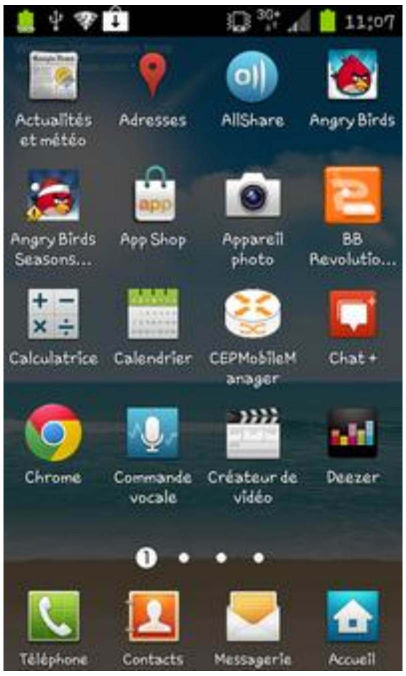
- Go to "Managing Applications"->"Downloaded" tab->"CEPMobileManager" application
- Select "Clear Data" then "Force Stop"

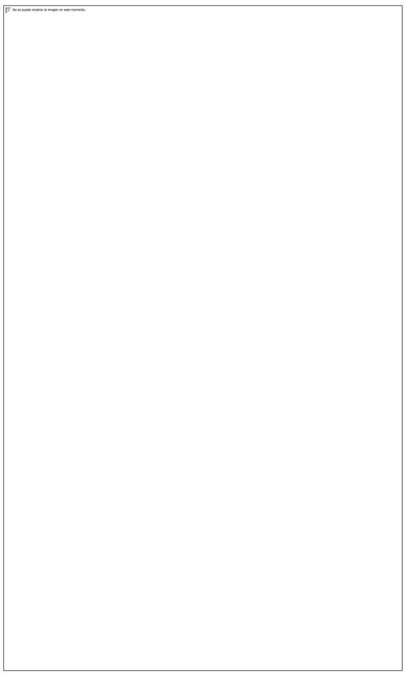
8.4 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

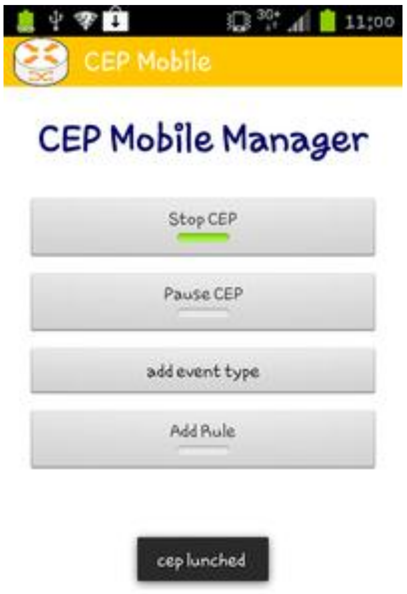
The CEPMobileManager GUI software allows managing a DataHandling Complex Event Processing engine on mobile phone.

- Go to the Android application list to run "CEPMobileManager".





- Go to Settings/Applications/Running in order to diplay running processes. "CEPMobileManager" should be available in the list.
- Click on the "Start CEP" button.
- If it becomes green, then sanity check is passed



8.4.1 End to End testing

Requirements

No particular requirements or tools needed for this alpha release.

Steps

The following tests describe the most important steps to insure the CEP is up and running, with its main features working: stop/start, defining a new event type then setting a processing rule.

After passing the sanity check, on the screen above:

- Stop the CEP by clicking on the "Stop CEP" button.
- Start it again the same way.
- Freeze it by clicking the "Pause button". This GUI tests the real CEP state before displaying feedback. Test is succesful if the button becomes green: the CEP is now still.
- Unfreeze CEP by clicking on this same button again, which in turn becomes gray again.
- Fed a dummy event type to the CEP by clicking on the "Add event type" type button. If a pop-up appears that displays: "event type added succesfully", then the test is passed.
- Set a dummy rule statement by clicking on the "Add rule" button. If the button turns green, the test is succesfully passed.
- Delete the rule by clicking the same button again. If it turns grey, the test is passed.

8.4.2 List of Running Processes

The "CEPMobileManager" process must be running

8.4.3 Network interfaces Up & Open

3G is required.

Note: WiFi is not reliable enough for GCM messaging due to TCP timeouts in many routers.

8.4.4 Databases

No database needed.

8.5 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

8.5.1 Resource availability

The Android system needs 30MB of available memory for the CEP to run properly.

The minimum version of Android is 2.3.3

8.5.2 Remote Service Access

Direct access possible through GCM.

Recommended access through HTTP REST via AppEngine middleware.

8.5.3 Resource consumption

The available RAM should never be lower than 20MB before CEP launching.

The CEPMobileManager application consumes 10MB of memory.

8.5.4 I/O flows

- Input: GCM packets
- Output: HTTP requests