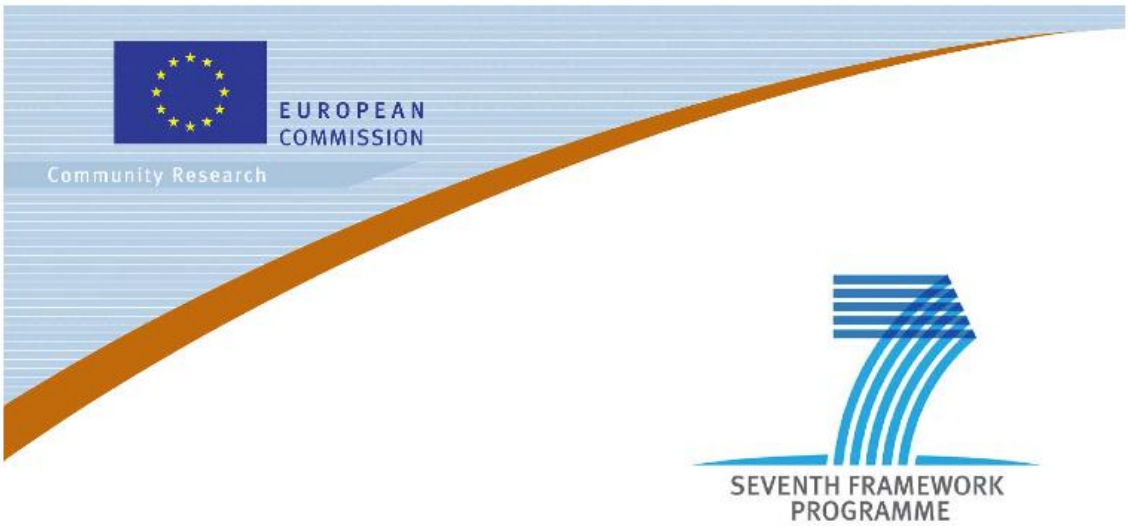


Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

D.5.3.3: 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.5.3.3
Project Document Date: 2014-08-27
Deliverable Type and Security: Private
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.3), 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.

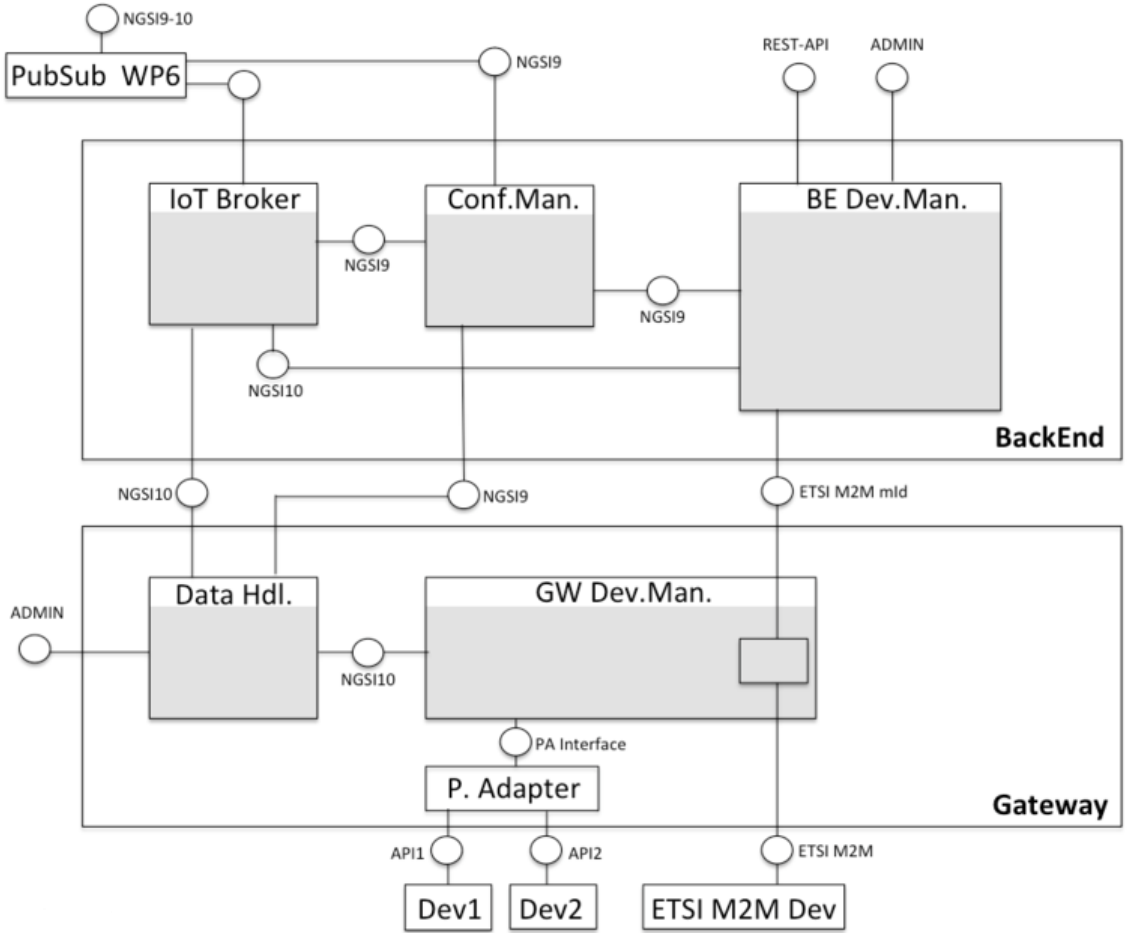
A key design statement is that, whenever present, IoT Gateways are not expected to be permanently connected to the Backend as per communications design or failures. Another relevant remark is that IoT Gateways are expected to be constrained devices in some scenarios. Therefore, light-weight implementations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

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.org/

The following resources were used to generate this document:

D.5.3.3_Installation_and_Administration_Guide_front_page

Backend IoT Broker - IoT Broker - Installation and Administration Guide

Publish/Subscribe Broker - Orion Context Broker - Installation and Administration Guide (This Data/Context Management chapter GE can be configured as Configuration Manager - Orion Context Broker)

Configuration Manager - IoT Discovery - Installation and Administration Guide

Backend Device Manager - IDAS - Installation and Administration Guide

Gateway Device Manager - OpenMTC - Installation and Administration Guide

Gateway Protocol Adapter - ZPA - Installation and Administration Guide

Gateway Protocol Adapter - EPCGE - Installation and Administration Guide

Gateway Protocol Adapter - MRCoap - Installation and Administration Guide

Gateway Data Handling - Esper4FastData - Installation and Administration Guide

Template Handler - Template Handler - 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 our wikis. 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
v1	First draft of deliverable submission generated	2014-07-22	TID
v0	First draft of deliverable submission generated	2014-08-27	TID

1.10 Table of Contents

2 Backend IoT Broker - IoT Broker - 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

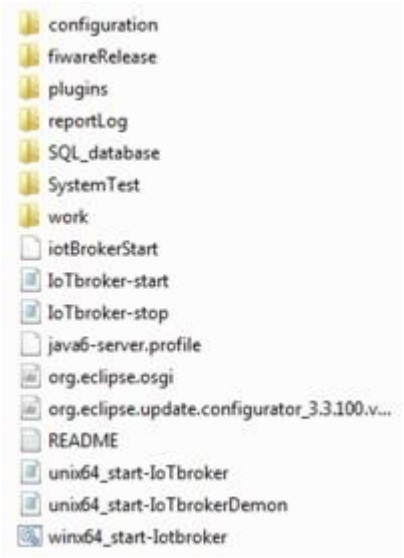
Minimum System Requirements:

- Processor: 1 CPU 1.2 GHZ
- RAM: 1 GB
- DISK Spacd:50 MB
- JAVA : Java 7
- Operating System: 32 or 64-bit version Windows or Linux

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:



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.ini* in the *configuration* folder. The *config.ini* 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 IMPORTANT that the path has / at the end). It is also possible to change the IoT Broker server port by changing the *tomcat.init.port* entry (the default value is 80). An example of config.ini is shown below:

```
 ##############################

 # Equinox settings

 ##############################

 eclipse.ignoreApp=true

 osgi.clean=true

 osgi.noShutdown=true

 osgi.bundles.defaultStartLevel=4

 osgi.java.profile=java6-server.profile

 osgi.java.profile.bootdelegation=override

 # Used for define the folder wher the log properties are
located

 bundles.configuration.location=.//configuration//configadmin

 # tomcat.init.port could be configured for changing the IoT
Broker Port
```

```
tomcat.init.port=8090

# Define the location of HSQLDB used for storing the
Subscription information

hsqldb.directory=.//SQL_database/database

# Remember to add \\ at the end of the path

dir.config=E:\\FI-
WARE_software_releases\\IoTBroker_FIWARE_3.3.3\\

ngsiclient.layer=connector

java.awt.headless=true

file.encoding=UTF-8

##############################

# Client bundles to install

##############################

......
```

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">./fiwareRelease/iotBroker/config/sc
hema/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">./fiwareRelease/iotBroker/config/s
chema/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="default_throttling">1000</entry>

 <entry key="default_duration">31536000000</entry>

 <entry key="hsqldb.username">NEC</entry>

 <entry key="hsqldb.password">neclab</entry>

 </properties>
```

In that file it is possible to modify the URL for connecting the IoT Broker with the Discovery component (ngsi9Uri) the pub/sub broker (pub_sub_addr) for the FI-WARE enviroment. In addition to that is possible to modify the address of the xsd files used for validate the incoming requests to the broker. 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*. Internally the IoT Broker use an hsqldb for storing the subscription information and the username and password could also be modified changing the hsqldb.username and hsqldb.password.

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_2.3.0-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
```

In the example above, the log level is selected in the first line. The possible log levels are:

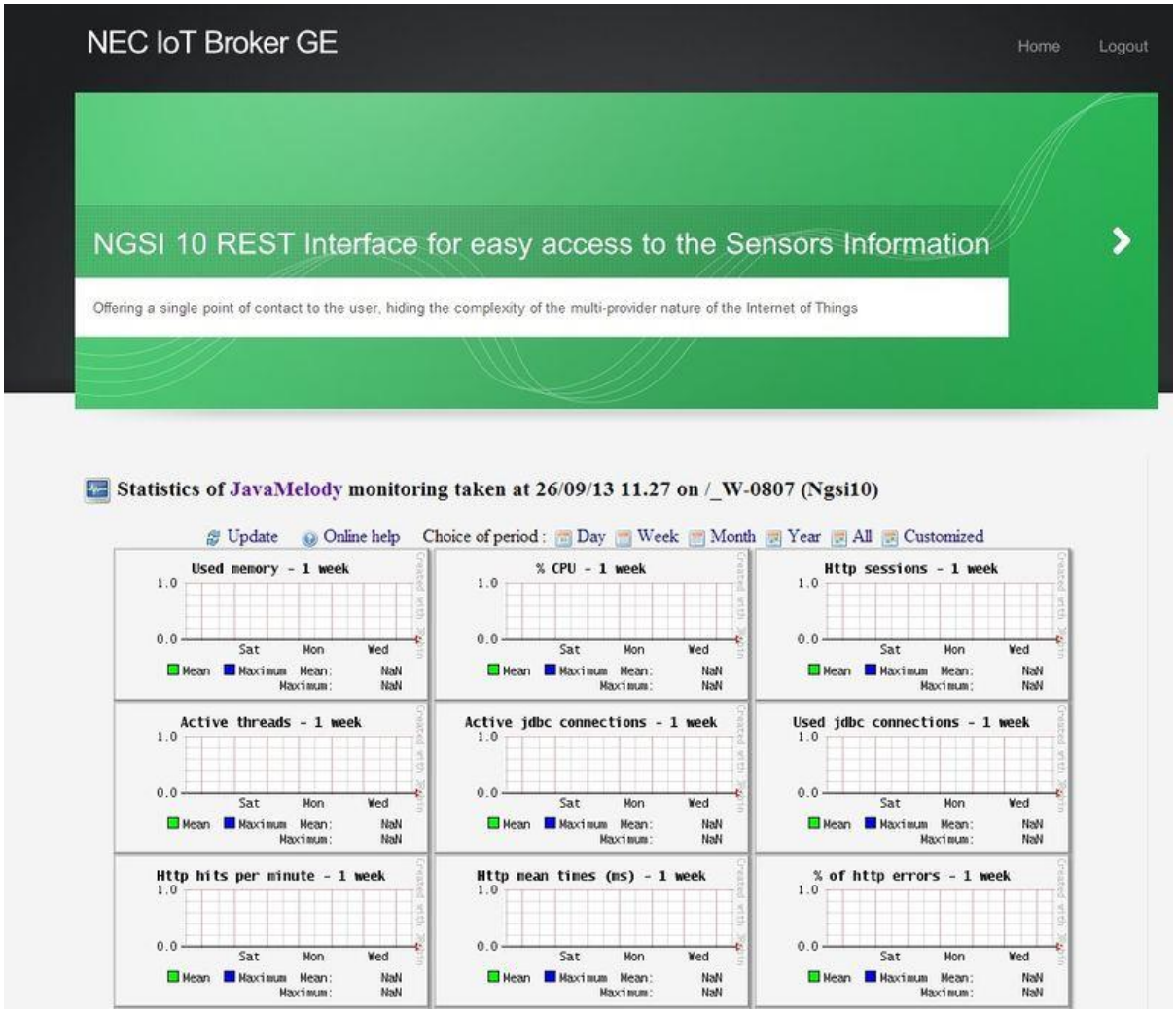
- log4j.rootLogger=INFO
- log4j.rootLogger=DEBUG

The IoT Broker has also a monitoring panel that can be accessible using the "Admin login" button in the index page. After that the administrator should use the ADMIN credential to login into the Monitoring page. The default username and password are: admin, admin. It is possible to customize the administrator login credential modifying the user.properties file in the \fiwareRelease\iotBroker\config\ directory. It is important to remember that the Monitoring panel is using an HTTPS connection based on SSL certificate. In the fiwareRelease\iotBroker\https\ folder there is already a dummy key, but for security reason the admin should generate a private key containing a valid certivicate. A simple way to generate one of these is to use Java's keytool utility located in the $JAVA_HOME/bin directory.

Example:

```
'keytool -genkey -alias admin -keyalg RSA -keystore
...\fiwareRelease\iotBroker\https\key.keystore'
```

An example of the Monitoring panel is shown in the picture below:



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://{IoT Broker IP}/ngsi10/sanityCheck/

If the response is:

```
<sanityCheck>
<name>IoT Broker GE</name>
<type>Sanity Check</type>
<version>Version: 2.3.0.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.

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://{IoT Broker IP}/ngsi10/contextEntities/Kitchen)

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

```
  <contextElementResponse>
       <contextElement>
              <entityId isPattern="false">
                     <id>Kitchen</id>
              </entityId>
       </contextElement>
       <statusCode>
              <code>500</code>
              <reasonPhrase>RECEIVER                    INTERNAL
ERROR</reasonPhrase>
              <details
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">Error I/O with:
                     http://localhost:8999</details>
       </statusCode>
  </contextElementResponse>
```

or

```
  <contextElementResponse>
       <contextElement>
              <entityId   isPattern="false">
                     <id>Kitchen</id>
              </entityId>
       </contextElement>
       <statusCode>
              <code>404</code>
              <reasonPhrase>CONTEXT          ELEMENT          NOT
FOUND</reasonPhrase>
       </statusCode>
  </contextElementResponse>
```

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.

HTPPS port (9443) should be accessible.

2.4.4 Databases

Embedded HSQLDB database on port 9001. http://hsqldb.org/. This database is embedded in the NEC IoT Broker and therefore requires no particular configuration or installation procedure (user and password could be changed in the config.xml files are explained in the installation section). For checking the status of the database it is possible to send an SQL query using the hsqldb driver on port 9001 with username NEC and password neclab.

The database log are in the folder SQL_database and the file is database.txt.

At start up the IoT Broker is starting automaticaly the database instance as shown from the IoT Broker logger:

```
 [Server@3b2d38f6]: [Thread[SpringOsgiExtenderThread-8,5,spring-
osgi-extender[1c57a236]-threads]]: checkRunning(false) entered

 [Server@3b2d38f6]: [Thread[SpringOsgiExtenderThread-8,5,spring-
osgi-extender[1c57a236]-threads]]: checkRunning(false) exited

 [Server@3b2d38f6]: Initiating startup sequence...

 [Server@3b2d38f6]: Server socket opened successfully in 1 ms.

 Jul 29, 2013 9:17:23 AM org.hsqldb.persist.Logger logInfoEvent

 INFO: checkpointClose start

 Jul 29, 2013 9:17:23 AM org.hsqldb.persist.Logger logInfoEvent

 INFO: checkpointClose end

 [Server@3b2d38f6]: Database [index=0, id=0,
db=file:E:\SQL_database\database, alias=linkdb] opened
sucessfully in 738 ms.

 [Server@3b2d38f6]: Startup sequence completed in 750 ms.

 [Server@3b2d38f6]: 2013-07-29 09:17:23.347 HSQLDB server 2.2.9
is online on port 9001

 [Server@3b2d38f6]: To close normally, connect and execute
SHUTDOWN SQL

 [Server@3b2d38f6]: From command line, use [Ctrl]+[C] to abort
abruptly
```

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

The minimum of RAM needed for sunning the IoT Broker is 256MB, however our recommandation is to have minimum 512MB of available RAM. Regarding the physical

memory, the IoT Broker binary are really light and need only 25MB of space on the HD, however the IoT Broker will use an internal database for storing subscription information, for that reason we recommend to have at least 100MB of space available on the disk.

2.5.2 Remote Service Access

- TCP connection to Pub/Sub Broker GE (via NGSI-10 on port 80)
- TCP connection to Data Handling GE (via NGSI-10 on port 80)
- TCP connection to Configuration Management Component (via NGSI-9 on port 80)

The port for accessing the different services could be also configured in the IoT Broker config file inside the fiwareRelease folder.

Please make sure port 80 is accessible.

2.5.3 Resource consumption

This section states the amount of resources that are abnormally high or low. It applies to RAM, CPU and I/O.

In the IDLE State the IoT Broker consumption is:

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

If the IoT Broker process uses more then 500MB and CPU usege % is more than 40% in the idle state, this can be consider abnormal.

2.5.4 I/O flows

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

3 Configuration Manager - IoT Discovery - Installation and Administration Guide

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

3.1 Introduction

Welcome to the Installation and Administration Guide for the Configuration Management GE Implementation (GEi) - IoT Discovery. 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. The main components for installation the GEi are 2 Java WAR files that are deployed on Apache Tomcat as server modules, and a backend MySQL database.

3.2 System Requirements

In order to deploy the Configuration Management Support GE the following software must be previously installed:

- Oracle Java J2SE v7 JRE/JDK
  - The GEi requires that Oracle Java 7 is installed.
  - If Ubuntu Linux is used, please follow the guide available here.
- Apache Tomcat 7.x
  - Tomcat can simply be installed by downloading the compressed file and unpackaging it in the required folder.
  - Alternatively, it can be installed for autorun at startup.
    - For Windows 7, download and run the *.msi file.
    - For Ubuntu, refer to the guide available here.
- MySQL Community Server 5.5
  - MySQL can be installed on Ubuntu by running the following command:

```
sudo apt-get install mysql-server
```

- MySQL Community Workbench 5.2
  - MySQL Workbench can be installed on Ubuntu by running the following command:

```
sudo apt-get install mysql-workbench
```

3.3 System Installation

The following steps below need to be performed to get the IoT Discovery GEi up & running.

3.3.1 Unpackage the GEi

The GEi is provided in a zip file. The content of the zip file is the follow:

- *semanticDBs.sql*: contains the SQL script for creating the tables for the Sense2Web Platform
- *IoTDiscoveryNgsiTest-v2.zip*: contains the Unit Test Suite for the NGSI-9 Server.
- *IoTDiscoveryS2WTest-v2.zip*: contains the Unit Test Suite for the Sense2Web Platform.
- *ngsi9.war*: The Java WAR file for the NGSI-9 Server.
- *S2W.war*: The Java WAR file for the Sense2Web Platform.

3.3.2 Setup MySQL Databases

The first task to perform is to create the databases and their respective tables for the **Sense2Web** platform module. This is done by importing the SQL script "**semanticDBs.sql**" provided in the GEi package, which should produce the following databases:

- "**resourcedb**"
- "**entitydb**"
- "**servicedb**"

3.3.2.1 *Option 1: Command-line Client*

Alternatively, to do this step directly from the command-line, connect to MySQL using:

```
mysql -u <username> -p
```

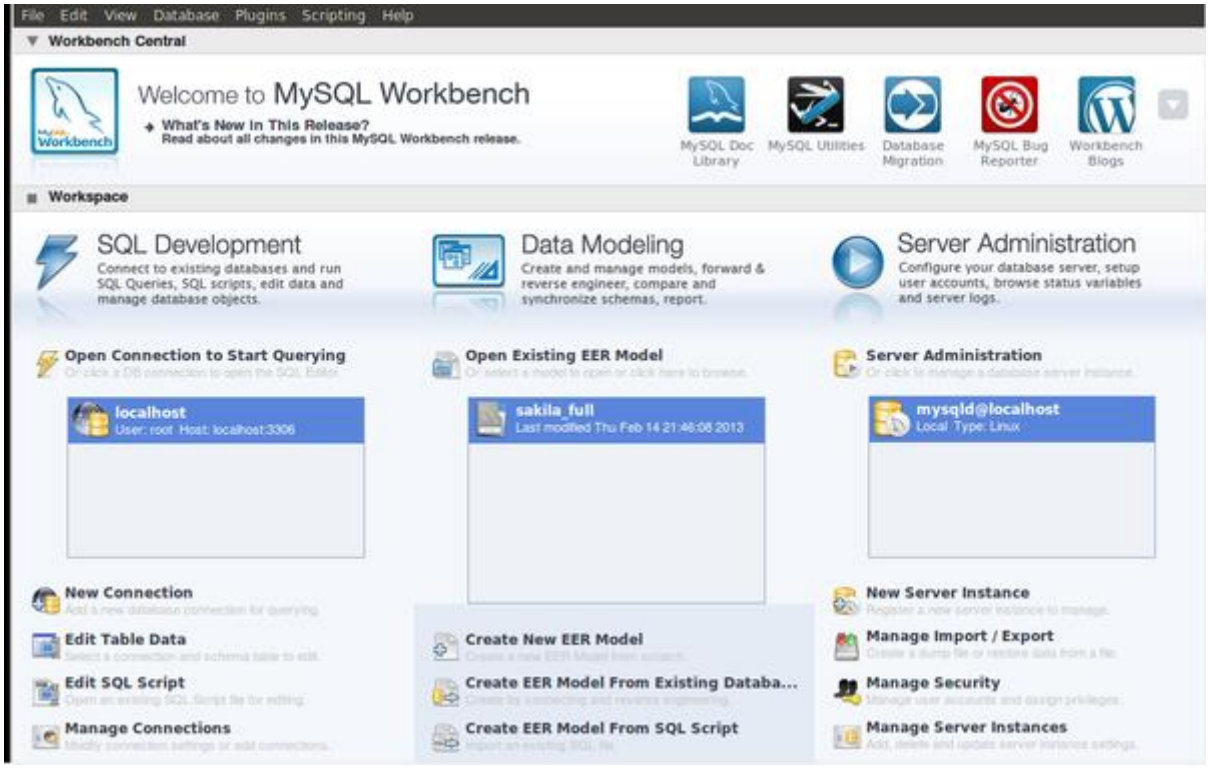
Enter the password, then run the script included in the GEi package using the command:

```
mysql < \. <scriptname>
```

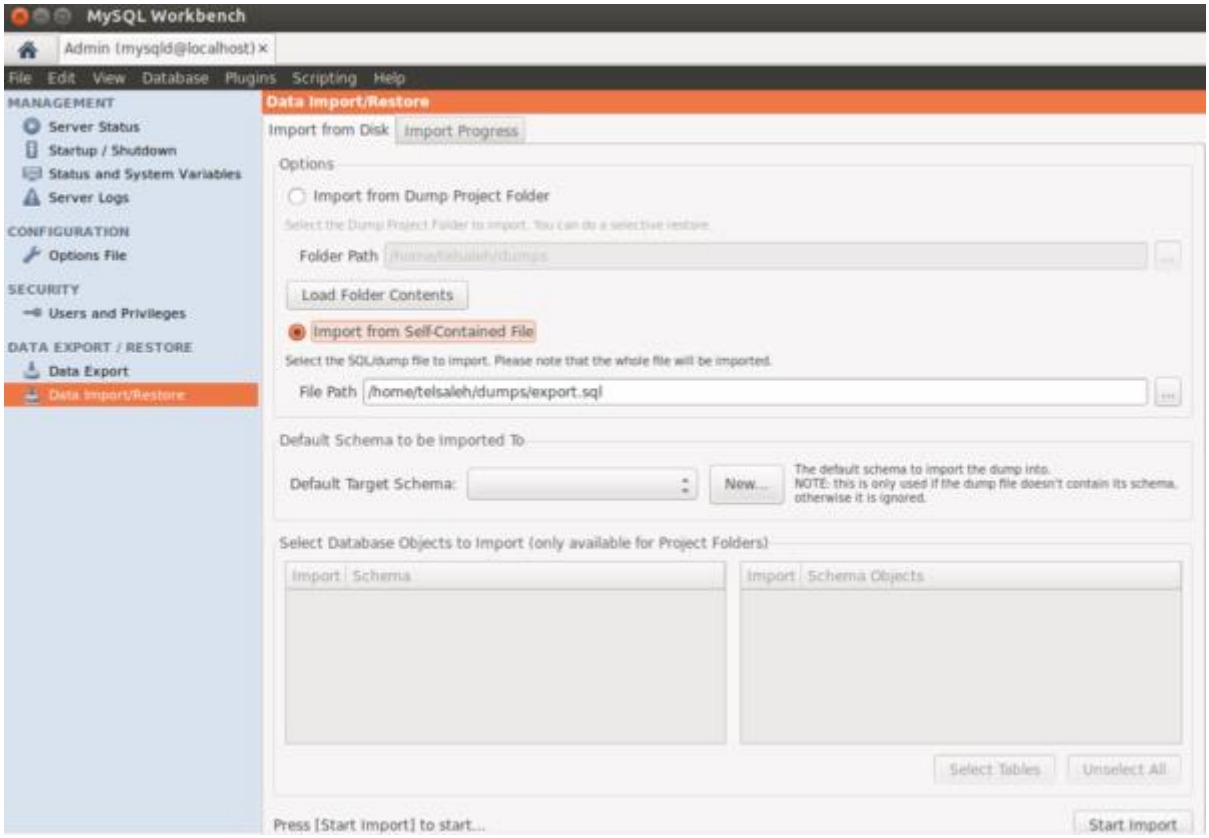
3.3.2.2 *Option 2: MySQL workbench*

To do this, run the MySQL workbench by entering the following command:

```
mysql-workbench
```



Access the running MySQL server instance under "Server Administration". Check that the server is running. Under "Data Import/Export", import and run the SQL script "**semanticDBs.sql**" file that is included in the GEi package.



3.3.3 Setup WARs for Tomcat

- Copy the Web Application Archive files **"S2W.war"** and **"ngsi9.war"** to

```
CATALINA_HOME\webapps\
```

whereby CATALINA_HOME is the root folder for Tomcat.

3.3.3.1 *ngsi9.war*

Nothing other than copying the **ngsi9.war** file to the **/webapps** folder is required.

3.3.3.2 *S2W.war*

- Before starting the Tomcat server, the context parameters (**context-param**) in the **web.xml** should be checked.
- To access and edit this file:
    - Extract the contents of **S2W.war** to the **"/webapps"** folder.
        - This can be done using a file archiver tool such as 7-Zip.
        - In Ubuntu, the command line below can be used:

```
jar -xvf  S2W.war
```

- In the extracted folder i.e. **"/S2W"**, go to **"/WEB-INF"** and open the **web.xml** using your preferred text editor.

Part of the **web.xml** containing the context parameters is shown below:

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

<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <!--    <servlet>

        <servlet-name>ssi</servlet-name>

        <servlet-class>

          org.apache.catalina.ssi.SSIServlet

        </servlet-class>

        <init-param>

          <param-name>buffered</param-name>

          <param-value>1</param-value>

        </init-param>

        <init-param>

          <param-name>debug</param-name>

          <param-value>0</param-value>

        </init-param>

        <init-param>

          <param-name>expires</param-name>

          <param-value>666</param-value>

        </init-param>

        <init-param>

          <param-name>isVirtualWebappRelative</param-name>

          <param-value>0</param-value>

        </init-param>
```

```
        <load-on-startup>4</load-on-startup>
    </servlet>-->
    <description>Linked Data Platform for the Internet of Things
        CCSR, University of Surrey 2013</description>
    <display-name>Sense2Web / IoTDiscovery</display-name>
    <context-param>
        <description>username for MySQL access</description>
        <param-name>username_mysql</param-name>
        <param-value>root</param-value>
    </context-param>
    <context-param>
        <description>password for MySQL access</description>
        <param-name>password_mysql</param-name>
        <param-value>root</param-value>
    </context-param>
    <context-param>
        <description>This value reflects the percentage between
the number of trained concepts and concepts that have been
folded in.
            The lower the value, the more frequent the engine
will likely be trained</description>
        <param-name>retrain_threshold</param-name>
        <param-value>0.1</param-value>
    </context-param>
    <context-param>
        <description>minimum number of descriptions in database
for training</description>
        <param-name>training_set_min</param-name>
        <param-value>10</param-value>
    </context-param>
```

```
    ...
</web-app>
```

- There are several parameters in the web.xml file that can be configured. These are:
    - *username_mysql* - this is the **username** for accessing the MySQL database.
    - *password_mysql* - this is the **password** for accessing the MySQL database.
    - *training_set_min* - this parameter sets the **minimum number** of descriptions stored in the repository that are required for the training process to be executed.
    - *retrain_threshold* - this threshold value (between 0 and 1) is the **ratio** of new concepts (i.e. samples received after the training process) to the the number of old concepts (i.e. samples used in the training process). The lower the value, the more likely the engine will be re-trained more frequently, which becomes more resource-intensive as the repository grows.

For terminologies relating to the parameters *training_set_min* and *retrain_threshold* such as "concept", please refer to the subsection on IoT Search Engine in the Open Specification of the Configuration Management GE.

3.3.3.3 *Run Tomcat*

- The last step is to run the Tomcat server. In a console terminal enter the following command:
    - Ubuntu:

```
sudo /CATALINA_HOME/bin/startup.sh
```

    - Windows:

```
 /CATALINA_HOME/bin/startup.bat
```

- **PLEASE NOTE**: if MySQL is not setup correctly, the Sense2Web Platform will not run. This means making sure the MySQL server is running, the tables are created, and the username and passwords in the web.xml are valid for access to your installation of MySQL.

3.4 System Administration

The GEi can be monitored by opening another terminal and entering the command in the directory:

```
 tail -f /CATALINA_HOME/logs/catalina.out
```

3.4.1 Logging

For accessing logs, they can be found in:

```
/CATALINA_HOME/logs/catalina.{date}.log

/CATALINA_HOME/logs/localhost.{date}.log
```

These can be used to check if deployment was successful, and also any issues with Servlets processing requests. The *date* value reflects the date of the log, which is created on a daily basis.

3.5 Sanity Check Procedures

The Sanity Check Procedure is the first step that a System Administrator will do to verify that the GEi 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. The first step is to check the databases, and then the end-to-end testing.

3.5.1 End to End testing

Using a web browser or REST client, contact the GEi via HTTP on:

- http://{serverRoot}/S2W/repository/getVersion/
- http://{serverRoot}/ngsi9/sanityCheck/

The response should be:

```
<sanityCheck>

<name>IoT Discovery</name>

<type>Sanity Check</type>

<version>Version: 3.2.3.SNAPSHOT</version>

</sanityCheck>
```

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

3.5.1.1 *Sense2Web*

3.5.1.1.1 Test 1

Verify that http://{serverRoot}/S2W/ can be reached and returns the following page:

3.5.1.1.2 Test 2

- The RESTful interface for the Sense2Web platform can be tested by sending a HTTP GET request using the following URL:

```
http://{serverRoot}/S2W/repository/lookup/iot-
a/resource/ALL
```

The result should be an RDF document with all the descriptions in the "**Resource**" repository.

3.5.1.2 *NGSI-9 Server*

3.5.1.2.1 Test 1

Verify that http://{serverRoot}/ngsi9 can be reached and returns the following web page:

IoT Discovery NGSI-9 Server v3.2.3

- Standard Operations:
  - URL: http://{serverRoot}/ngsi9/registerContext - Method: POST - Payload: registerContextRequest
  - URL: http://{serverRoot}/ngsi9/discoverContextAvailability - Method: POST - Payload: discoverContextAvailabilityRequest
  - URL: http://{serverRoot}/ngsi9/subscribeContextAvailability - Method: POST - Payload: subscribeContextAvailabilityRequest
  - URL: http://{serverRoot}/ngsi9/updateContextAvailabilitySubscription - Method: POST - Payload: updateContextAvailabilitySubscriptionRequest
  - URL: http://{serverRoot}/ngsi9/unsubscribeContextAvailability - Method: POST - Payload: unsubscribeContextAvailabilityRequest
- Convenience Operations:
  - URL: http://{serverRoot}/ngsi9/contextEntities/{EntityID} - Method: POST - Payload: registerContextRequest
  - URL: http://{serverRoot}/ngsi9/contextEntities/{EntityID} - Method: GET
  - URL: http://{serverRoot}/ngsi9/contextEntities/{EntityID}/attributes - Method: GET
  - URL: http://{serverRoot}/ngsi9/contextEntities/{EntityID}/attributes/attributeName - Method: GET
  - URL: http://{serverRoot}/ngsi9/contextEntities/{EntityID}/attributeDomains/{attributeDomainName} - Method: GET
  - URL: http://{serverRoot}/ngsi9/contextEntityTypes/{typeName} - Method: GET

CONTACT

Tarek Elsaleh

- Email: telsaleh AT surrey DOT ac DOT uk
- Centre of Communication Systems Research, University of Surrey, Guildford, Surrey GU2 7XH

This page contains information about the version of the GEi and which operations are supported on the current release.

3.5.1.2.2 Test 2

After that it is possible to test one of the supported NGSI-9 resources. For example let us try to send an HTTP GET to the *contextEntity/EntityId* resource. (http://{serverRoot}/ngsi9/contextEntities/Kitchen)

The response should be a **discoverContextAvailabilityResponse** response in XML, with the ERROR CODE:

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

If you get this error this means that the Entity that you are requesting is not available.

3.5.2 List of Running Processes

- java
- mysqld

3.5.3 Network interfaces Up & Open

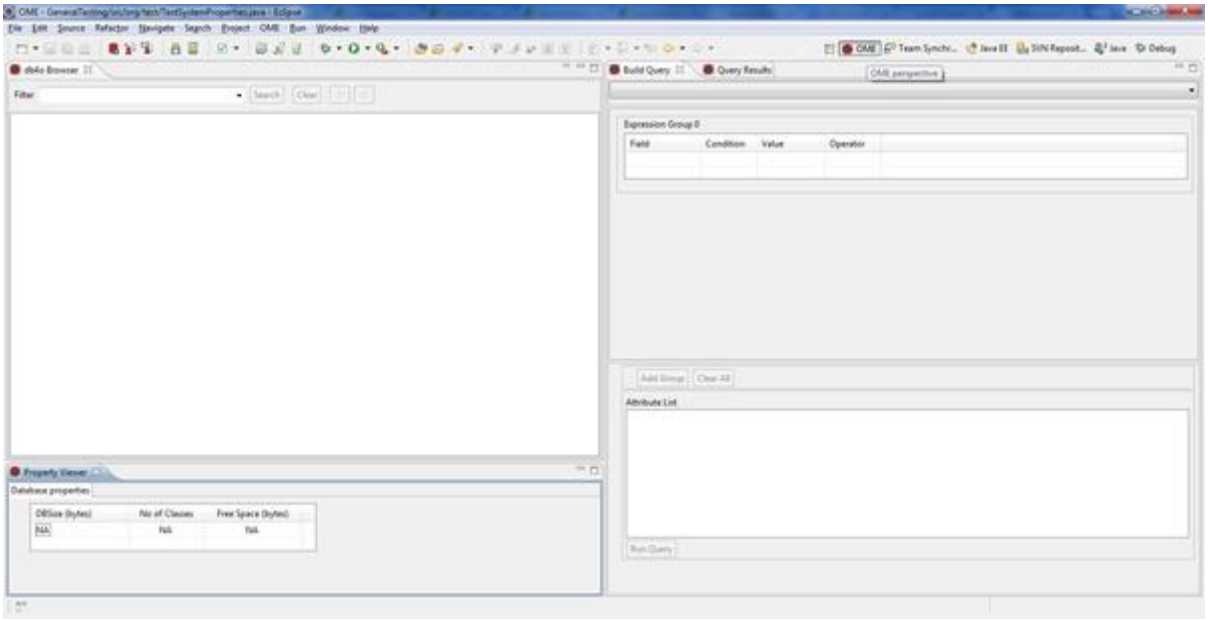
- TCP: 8080 (default)
- TCP: 3306

3.5.4 Databases

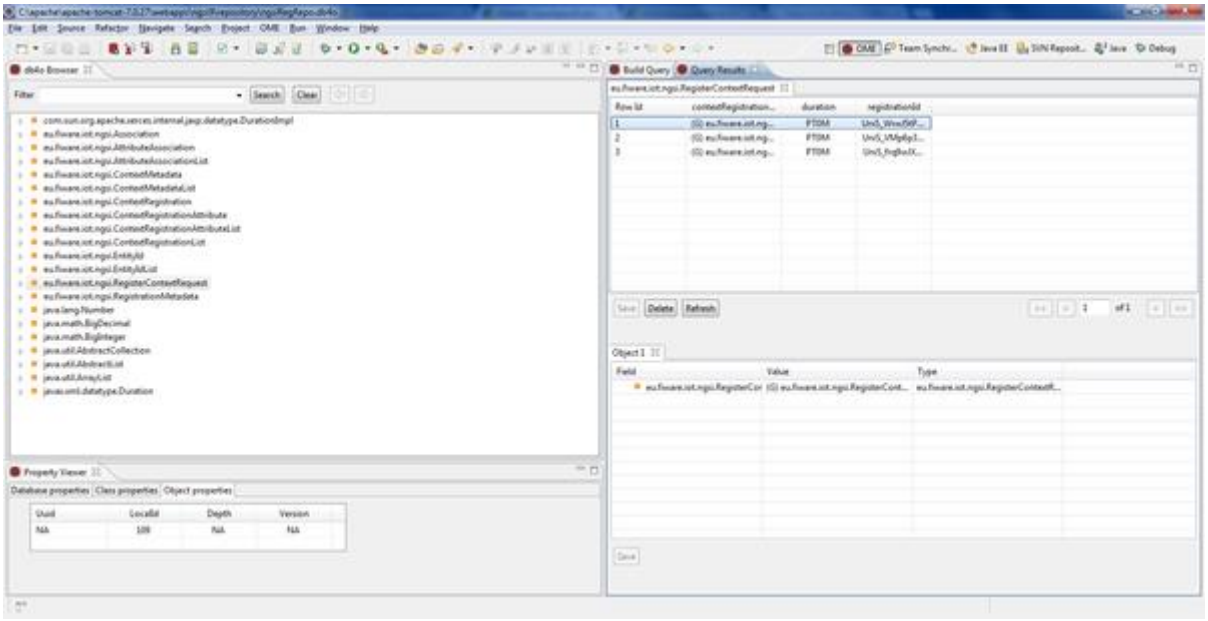
- db4o for the NGSI-9 Server.
- MySQL for the Sense2Web Platform.

3.5.4.1 *Check db4o*

The database will be created upon the first NGSI-9 registration. To check the database, you will need to use the Eclipse IDE. Download the db4o 8.0 package, unzip it, and go to **/ome** folder and follow the instructions in **readme.html** to install the Object Manager Enterprise plugin for Eclipse. Once this is done, go to the OME perspective in the IDE and you should have this view.



- In the menu, go to **OME->Connect/Disconnect DB**.
- Enter the location of the db4o file. This should point to the **/{CATALINA}/webapps/ngsi9/repository** folder.
- Once connected, you should see a list of objects in the db4o browser, after the first NGSI-9 registration has occurred.
- Look for RegisterContextRequest, right-click it, and in the context menu choose "**view all objects**".



- On the right-hand side in the Query Results tab, you can see the available registrations. If you choose one, the object structure of the registration will show below.

3.5.4.2 *Check MySQL*

To test that the script that was run during setup has been successful, enter the following commands in the mysql command line client.

3.5.4.2.1 Test 1

```
mysql> \G SELECT * FROM resourcedb.Triples LIMIT 10;

mysql> \G SELECT * FROM entitydb.Triples LIMIT 10;

mysql> \G SELECT * FROM servicedb.Triples LIMIT 10;
```

The result should present a table with the following columns. The tables will be empty when newly created, but will be populated as new description are registered. The table defines the mappings between RDF **triples** in the semantic repository.

```
+-----+-----+------+
| s   | p   | o    |
+-----+-----+------+
|  46 |  46 |  315 |
|  46 |  46 | 2793 |
```

```
|  46 | 171 |  305 |

|  48 |  46 |  315 |

|  48 |  46 | 2793 |

|  48 | 171 |  193 |

| 157 |  46 |   47 |

| 157 |  48 |   60 |

| 157 |  50 |   60 |

| 157 |  51 |   60 |

+-----+-----+------+

10 rows in set (0.00 sec)
```

3.5.4.2.2 Test 2

```
mysql> \G SELECT * FROM resourcedb.Nodes LIMIT 10;

mysql> \G SELECT * FROM entitydb.Nodes LIMIT 10;

mysql> \G SELECT * FROM servicedb.Nodes LIMIT 10;
```

The result should present a table with the following columns. The tables will be empty when newly created, but will be populated as new description are registered. The table defines the mappings between RDF **nodes** in the semantic repository.

```
+-----+----------------------+----------------------------------
------------------------------+------+----------+------+

| id  | hash                 | lex

+-----+----------------------+----------------------------------
------------------------------+------+----------+------+

| 109 |  5596883189078519408 | http://www.surrey.ac.uk/ccsr/IoT-
A/EntityModel.owl#            |      |          |    2 |

| 110 | -6430697865200335348 | http://www.w3.org/1999/02/22-rdf-
syntax-ns#type               |      |          |    2 |

| 111 | -2401239393240133156 | http://www.surrey.ac.uk/ccsr/IoT-
A/EntityModel.owl#entity     |      |          |    2 |

| 112 |  6454844767405606854 | http://www.w3.org/2000/01/rdf-
schema#label                |      |          |    2 |

| 113 |  7196782991867380714 |
```

```
| 114 | -7436251478343769787 | http://www.surrey.ac.uk/ccsr/IoT-
A/EntityModel.owl#hasentityID |      |          |    2 |

| 115 | -5872643041377611799 | http://www.surrey.ac.uk/ccsr/IoT-
A/EntityModel.owl#hasName     |      |          |    2 |

| 116 |  2122874005417209587 | http://www.surrey.ac.uk/ccsr/IoT-
A/EntityModel.owl#hasTag      |      |          |    2 |

| 117 | -8874962510547211933 | http://www.surrey.ac.uk/ccsr/IoT-
A/EntityModel.owl#hasURITag   |      |          |    2 |

| 118 | -8028859106695045829 | http://www.surrey.ac.uk/ccsr/IoT-
A/EntityModel.owl#hasTimeZone |      |          |    2 |

+-----+----------------------+----------------------------------
------------------------------+------+----------+------+
```

3.5.4.2.3 Test 3

```
mysql> \G SELECT * FROM resourcedb.Prefixes LIMIT 10;

mysql> \G SELECT * FROM entitydb.Prefixes LIMIT 10;

mysql> \G SELECT * FROM servicedb.Prefixes LIMIT 10;
```

The result should present a table with the following columns. The tables will be empty when newly created, but will be populated as new description are registered. The table defines the mappings between RDF **prefixes** in the semantic repository.

```


+----------------+-------------------------------------------------
---------------------+

| prefix         | uri      |

+----------------+-------------------------------------------------
---------------------+

| :                                                                |
http://www.surrey.ac.uk/ccsr/ontologies/VirtualEntityInstance.ow
l# |

|    EntityModel:           |    http://www.surrey.ac.uk/ccsr/IoT-
A/EntityModel.owl#                |

|     ResourceModel:      |     http://www.surrey.ac.uk/ccsr/IoT-
A/ResourceModel.owl#              |

|  owl2xml:                |  http://www.w3.org/2006/12/owl2-xml#
|
```

```
|  owl:                          |  http://www.w3.org/2002/07/owl#
|

|       protege:                                                  |
http://protege.stanford.edu/plugins/owl/protege#
|

| rdf:              | http://www.w3.org/1999/02/22-rdf-syntax-ns#
|

|  rdfs:                  |  http://www.w3.org/2000/01/rdf-schema#
|

|  swrl:                        |  http://www.w3.org/2003/11/swrl#
|

|  swrlb:                      |  http://www.w3.org/2003/11/swrlb#
|

+---------------+-----------------------------------------------
----------------------+

10 rows in set (0.01 sec)
```

3.5.4.2.4 Test 4

```
mysql> \G SELECT * FROM resourcedb.Quads LIMIT 10;

mysql> \G SELECT * FROM entitydb.Quads LIMIT 10;

mysql> \G SELECT * FROM servicedb.Quads LIMIT 10;
```

The result should present a table with the following columns. The tables will be empty when newly created, but will be populated as new description are registered. The table defines the mappings between RDF **quads** in the semantic repository.

```
+-----+-----+-----+------+

  g  | s   |  p  | o    |

+-----+-----+-----+------+

 23  |  46 |  46 |  315 |

 56  |  46 |  46 | 2793 |

 76  |  46 | 171 |  305 |

 12  |  48 |  46 |  315 |

 34  |  48 |  46 | 2793 |
```

```
  67   |   48 | 171 |  193 |
  31   | 157 |  46 |   47 |
  22   | 157 |  48 |   60 |
  78   | 157 |  50 |   60 |
  18   | 157 |  51 |   60 |
+-----+-----+-----+------+
10 rows in set (0.00 sec)
```

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

- RAM
  - recommended: 4GB
- HARD DISK
  - minimum: 5GB
  - recommended: 15GB
- CPU:
  - recommended: 2x2.8GHz

3.6.2 Remote Service Access

- TCP connection exposed for invocation from IoT GEs to ngsi-9 server on port 8080 (default).
- TCP connection exposed for invocation from Semantic GEs and Applications to S2W platform on port 8080 (default).

3.6.3 Resource consumption

Tomcat Server:

- Process
    - java.exe - org.apache.catalina.startup.Bootstrap start
    - VM config:

```
-XX:MaxPermSize=512M -Xms2048M -Xmx2048M -Xss64M -
XX:+CMSClassUnloadingEnabled -
XX:+CMSPermGenSweepingEnabled
```

- Memory consumption
    - Committed: 2GB
- Threads
    - Live threads: 28
    - Peak: 29
    - Daemon threads: 27
- CPU:
    - idle: 0%
    - active: ~5% (max)

MySQL:

- Process
    - mysqld
- Memory consumption
    - Committed: 100MB
    - Typical: 9MB
- Threads
    - Live threads: 46
- CPU:
    - idle: 0%
    - active: ~2% (max)

3.6.4 I/O flows

The only expected I/O flow is of type HTTP, on the port set for the Tomcat Server (default is 8080).

4 Backend Device Manager - IDAS - Installation and Administration Guide

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

4.1 Backend Device Management GE Installation

This guide provides a step-by-step instructions to get IDAS BE Device Management GE installed and running on a machine, as well as a list of perquisites, requirements and possible troubleshooting that may occur during the installation. IDAS is fully a backend asset that is able to collect and store events from physical devices with or without the presence of intermediate gateways. It is also capable of forwarding commands to bidirectional devices (actuators). IDAS users are always M2M application developers and therefore this manual has no user's guide section but only the programmer's one.

From the software components point of view, IDAS is made of two assets: the IDAS M2M platform and the IoT-Agent, which plays the role of NGSI connector to external capable components.

Therefore, the installation and Administration of IDAS GEi will always mean tasks related to those both assets: IDAS M2M platform and the IoT-Agent.

4.1.1 Requirements

The instructions below were tested on the following environment, it may or may not be the same on other configurations.

4.1.1.1 *IDAS M2M Platform*

- A virtual machine with CentOS 5.8 installed.
- A minimum of 1024 Mb of RAM and 30 Gb of HDD are needed.
- Additional software packages (Java Runtime Environment, Tomcat, MongoDb and Monit) are needed but their installation procedure is described in the IDAS step-by-step installation guide detailed in the sections below.

4.1.1.2 *IoT Agent*

The requirements for a successful installation of the IoT Agent are the following ones:

- A virtual machine with CentOS 5.8 installed
- IDAS M2M platform installed. This includes the Mongo database and Monit tool.

- As per the requirement above, the virtual machine is assumed to fulfill all IDAS M2M installation requirements (see previous subsection).

4.1.1.3 *ETSI-M2M Agent*

The requirements for a successful installation of the ETSI-M2M Agent are the following ones:

- A virtual machine with CentOS 5.8 installed
- IDAS M2M platform installed. This includes the Mongo database and Monit tool.
- As per the requirement above, the virtual machine is assumed to fulfill all IDAS M2M installation requirements (see previous subsection).

4.1.2 Installation Procedure

4.1.2.1 *IDAS M2M Platform*

4.1.2.1.1 IDAS MongoDB installation

The first steps are:

- Create a mongodb group: groupadd -og 500 mongodb
- Create the export folder: mkdir /home
- Create the mongodb user:

```
useradd -d /home/mongodb -m -g mongodb -s /bin/ksh -u 500 mongodb
```

- Install the Mongo distribution downloaded from http://www.mongodb.org/downloads

```
cp mongo-x86_64-2.0.2.tgz /home/mongodb

tar xzf mongo-x86_64-2.0.2.tgz
```

- Make a link o the mongodb binaries for future installations:

```
ln -s /home/mongodb/mongodb-linux-x86_64-2.0.2/bin bin
```

Monit installation

- Download Monit software from http://rpmfind.net/linux/rpm2html/search.php?query=monit
- Install the RPM: su - ; rpm -i monit-5.2.5-1.el5.rf.x86_64.rpm

- Ensure proper working by adding "127.0.0.1 <host_name> localhost.localdomain localhost" line to the /etc/hosts

Installation of neoidas-monitbd-1.0-canis.release.rhel5.5.x86_64.rpm

- Edit a file with the environment vars: vi Entorno.sh

The content of the file can be (for one mongod and one mongoc in the same machine):

```
# the name of the replica set for arbitrer and for mongod export
PARAM_RS_MONGOA=paid

export PARAM_RS_MONGOD=idas

#port for mongod

export PARAM_MONGOD_PORT=27018

#port for rest, it must be PARAM_MONGOD_PORT + 1000 export
PARAM_REST_MONGOD_PORT=28018

#port for mongos

export PARAM_MONGOS_PORT=27017

#port for mongo config server

export PARAM_MONGOC_PORT=27019

#port for mongo arbitrier

export PARAM_MONGOA_PORT=27016

#yes if you want to start a mongod

export PARAM_MONGOD=YES

#yes if you want to start a mongo config server export
PARAM_MONGOC=YES

#yes if you want to start a mongod arbitrier export
PARAM_MONGOA=NO

#yes if you want to start a mongos

export PARAM_MONGOS=NO

#first mongodb config server

export PARAM_MONGOC1=mongodb2-1 #second mongodb config server
```

```
export PARAM_MONGOC2=mongodb1-2 #third mongodb config server

export PARAM_MONGOC3=mongodb2-2

export
PARAM_SERVERSDB=$PARAM_MONGOC1:$PARAM_MONGOC_PORT,$PARAM_MONGOC2
:$PARAM_MO NGOC_PORT,$PARAM_MONGOC3:$PARAM_MONGOC_PORT
```

- Now, we should install the .rpm in each machine:

```
su -

../entorno.sh

rpm -i neoidas-monitbd-1.0-canis.release.rhel5.5.x86_64.rpm
```

IDAS Database RPM Installation

- Choose the machine from where we are going to administrate the Database.
- With the root user, create the file mongodb_environment.sh

```
#mongodb install directory for remote machine export
PARAM_DIR_MONGO=/home/mongodb #mongodb install directory for
local machine

export PARAM_LOCAL_MONGO=/home/mongodb

#database name

export PARAM_BBDD=nIDAS

#port for mongod

export PARAM_MONGOD_PORT=27018

#port for rest, it must be PARAM_MONGOD_PORT + 1000 export
PARAM_REST_MONGOD_PORT=28018

#port for mongos

export PARAM_MONGOS_PORT=27017

#port for mongodb config server

export PARAM_MONGOC_PORT=27019
```

```
#port for mongodb arbiter

export PARAM_MONGOA_PORT=27016

#user for ssh withn the remote machine export
PARAM_MONGO_USR=mongodb

#name of the first replica set export PARAM_RS1=idas #name of
the seconds replica set export PARAM_RS2=paid

#first host for the first replica set

export PARAM_MONGO11=vmongodb1-1 #seconde host for the first
replica set

export PARAM_MONGO12=vmongodb1-2 #third host for the first
replica set

export PARAM_MONGO13=vmongodb1-3

#first host for the second replica set

export PARAM_MONGO21=vmongodb2-1 #second host for the second
replica set export PARAM_MONGO22=vmongodb2-2 #third host for the
second replica set export PARAM_MONGO23=vmongodb2-3

#first mongodb config server

export PARAM_MONGOC1=vmongodb2-1 #second mongodb config server

export PARAM_MONGOC2=vmongodb1-2 #third mongodb config server

export PARAM_MONGOC3=vmongodb2-2

# varible for start the mongos

export
PARAM_SERVERSDB=$PARAM_MONGOC1:$PARAM_MONGOC_PORT,$PARAM_MONGOC2
:$PARAM_MO NGOC_PORT,$PARAM_MONGOC3:$PARAM_MONGOC_PORT
```

- From the root account, configure the variables aend install the package:

```
su -

../entorno.sh

rpm -i neoidas-bd-1.0-canis.release.rhel5.5.x86_64.rpm
```

- Uninstall IDAS DB RPM (root user):

```
rpm -e neoidas-bd-1.0-canis.release.rhel5.5
```

- neoidas-bd-1.0-canis.release.rhel5.5.x86_64 installation creates the following scripts at /export/open/IDAS/scripts:

- start_mongos.sh : starts mongos. To wake up mongos in other machine just copy and run this script. - create_replica_set.sh: creates the replica set. - create_shard.sh: creates the shard with the replica set. - create_bbdd.sh: creates nIDAS database, its collections and its initial data sets. - provision_idas_test.js: initial data sets for IDAS tests. - regenerateID.sh: if a full shard fails, and the other stays alive, this script will check the sequences are correct. - backup.sh: script to store a data set backup. - backup_recover.sh: script to restore a backup of the data set to the current database. - backup_second.sh: copies data sets from a secondary node to re-stablish a failing node.

Automate the Database start

- Edit the /etc/inittab adding the following line:

```
# Run monit in standard runlevels
mu:2345:respawn:su - mongodb -c "monit -v -I -c
/home/mongodb/scripts/monit.conf -d 10 -p
/home/mongodb/run/monit.pid"
```

- Restart the init process: init q
- To get all process to be launched automatically, edit or modify the file /etc/init.d/borrapids.sh in the following way:

```
#! /bin/sh case "$1" in
start)
rm -rf /home/mongodb/datos/*/mongod.lock ;;
stop) ;;
restart) ;;
condrestart) ;;
status) ;;
*)
```

```
echo "Usage: $0 {start|stop|restart|condrestart|status}" exit 1

esac

exit
```

- Give run privilege: chmod +x /etc/init.d/borrapids.sh
- Create the necessary symbolic links (if not existent):

```
cd /etc/rc3.d

ln -s ../init.d/borrapids.sh S96borrapids cd /etc/rc5.d

ln -s ../init.d/borrapids.sh S96borrapids
```

4.1.2.1.2 IDAS GW installation

- Create gwidas group: groupadd -og 501 gwidas
- Create gwidas user: useradd -d /home/gwidas -m -g gwidas -u 501 gwidas
- Copy & Installa the installation package:

```
scp
develop@neoidasdes:/home/develop/JENKINS/workspace/BuildGatewayI
DAS/servers/dis                t/Release.rhel5.5-x86_64/GNU-Linux-
x86/package/neoidas-gw-*-

canis.*rpm .


rpm -i neoidas-gw-1.0-canis.release.rhel5.5.x86_64.rpm
```

- To avoid failures due to memory exhaustion, it is necessary to run as root user the following procedures:
- In the file /etc/profile replace "ulimit -S -c 0 > /dev/null 2>&1" by "ulimit -c unlimited >/dev/null 2>&1"
- In /etc/sysconfig/init file add the line: DAEMON_COREFILE_LIMIT='unlimited'
- Run:

```

```

```
echo 1 > /proc/sys/fs/suid_dumpable sysctl -p
```

- To install monit, download it from: scp neoidasfe01:/root/monit* .
- Install monit: rpm -i monit-5.2.5-1.el5.rf.x86_64.rpm
- Configure monit:

- Create as root user the folder: /var/monit Edit the file: /etc/monit.conf

```
configurando las siguientes líneas (en este ejemplo se ha supuesto que el servidor de Gateway se denomina IDASSERVER y que la instalación se ha realizado en el directorio por defecto /usr/local/gwidas):



set daemon 60

set logfile /var/log/monit set httpd port 2815 and

use address localhost # only accept connection from localhost allow localhost # allow localhost to connect to the server and

#

#

#

#

check file monit with path /var/log/monit

allow admin:monit # require user 'admin' with password 'monit' allow @monit # allow users of group 'monit' to connect (rw)

allow @users readonly # allow users of group 'users' to connect readonly

if size > 4 MB then

exec "/bin/bash -c 'F=/var/log/monit; /bin/rm $F.5; /bin/mv $F.4 $F.5; /bin/mv $F.3 $F.4; /bin/mv $F.2 $F.3; /bin/mv $F.1 $F.2; /bin/mv $F $F.1; monit -c /etc/monit.conf -p /var/run/monit.pid reload'"

check process IDASSERVER with pidfile /usr/local/gwidas/IDASSERVER/IDASSERVER.pid

start program = "/bin/sh -c '/usr/local/gwidas/bin/MonitGwIdas IDASSERVER'"
```

```
stop program = "/bin/sh -c 'kill -9 `cat /usr/local/gwidas/IDASSERVER/IDASSERVER.pid`'"

#include /etc/monit.d/*
```

- If monit needs to be restarted, run the following command:

```
/etc/init.d/monit start
```

4.1.2.1.3 IDAS SE installation

Service Logic Front-end

- Create the groups idas & dba: groupadd -g 101 idas groupadd -g 102 dba
- Create the folder export: mkdir /export
- Create the user manager: useradd -d /export/manager -m -g idas -G dba -s /bin/ksh -u 101 manager
- Set the access rights to manager user home folder: chmod 755 /export/manager
- Export vars to copy the installation packages:

```
export BASEPJ=/export/develop/JENKINS/workspace/Rebuild_RedHat_5.5/trunk export RELEASE=canis
```

- Copy the installation packages:

OBJ

```
scp develop@neoidasdes:$BASEPJ/SE.ss/packages.ss/neoidas.pkg/RPMS/*/*frontend*- $RELEASE.*.rpm .
```

- Export environment vars needed for the installation:

```
export PARAM_FRONTEND_ADDR=10.25.144.48

export PARAM_FRONTEND_BCUP=10.25.144.49

export PARAM_BACKEND_ADDR=10.25.144.50

export PARAM_BACKEND_BCUP=10.25.144.51
```

- If the installation is run with "sudo", the variables PARAM* need to be added to env_keep at /etc/sudoers:

```
Defaults env_keep = "PARAM_* COLORS DISPLAY HOSTNAME HISTSIZE
INPUTRC KDEDIR \ LS_COLORS MAIL PS1 PS2 QTDIR USERNAME \


LANG LC_ADDRESS LC_CTYPE LC_COLLATE LC_IDENTIFICATION \
LC_MEASUREMENT LC_MESSAGES LC_MONETARY LC_NAME LC_NUMERIC \
LC_PAPER LC_TELEPHONE LC_TIME LC_ALL

LANGUAGE LINGUAS \ _XKB_CHARSET XAUTHORITY"
```

- Install the dependencies of the application software:

```
requires: monit , libicu, libbz2.so.1, glibc >= 2.5,
libcrypto.so.6, libgcc >= 4.1.2 , libsasl2.so.2, libstdc++ >=
4.1.2 , libuuid.so.1, zlib
```

- To install monit, download it from: scp neoidasfe01:/root/monit* and install it:

rpm -i monit-5.2.5-1.el5.rf.x86_64.rpm

- Install the package with the aplication software:

```
sudo rpm -q neoidas-fe; if [ $? -eq 1 ]; then echo "First
installation"; else sudo rpm -e neoidas-fe-1.0; fi;

sudo rpm -Uvh -vv neoidas-fe-1.0-
$RELEASE.release.rhel5.5.x86_64.rpm --prefix
/export/open/IDAS_FE > listado_instalac.txt 2>&1
```

- To launch monit, run the following commands:

```
su - manager -c 'monit -v -c
/export/open/IDAS_FE/conf/monitIDAS.conf -d 10 -p
/export/open/IDAS_FE/run/monit.pid'

su - manager -c 'monit -v -c
/export/open/IDAS_FE/conf/monitIDAS.conf -p
/export/open/IDAS_FE/run/monit.pid start all'
```

Service Logic Back-end

- Create the groups idas & dba: groupadd -g 101 idas groupadd -g 102 dba
- Create the folder export: mkdir /export
- Create the user manager: useradd -d /export/manager -m -g idas -G dba -s /bin/ksh -u 101 manager
- Update access rights: chmod 755 /export/manager
- Export variables to copy installation packages:

```
export
BASEPJ=/export/develop/JENKINS/workspace/Rebuild_RedHat_5.5/trun
k export RELEASE=canis
```

- Copy installation packages:

```
scp
develop@neoidasdes:$BASEPJ/SE.ss/packages.ss/neoidas.pkg/RPMS/*/
*backend*- $RELEASE.*.rpm .
```

- Export environment variables needed for the installation:

```
export PARAM_FRONTEND_ADDR=10.25.144.48

export PARAM_FRONTEND_BCUP=10.25.144.49

export PARAM_BACKEND_ADDR=10.25.144.50

export PARAM_BACKEND_BCUP=10.25.144.51

export PARAM_DBTYPE= MONGODB

export PARAM_DBNAME= nIDAS

export  PARAM_SERVERSDB=  10.25.144.89:27019,  10.25.144.90:27019,
10.25.144.91:27019
```

If the installation is run with "sudo", the variable PARAM* need to be added to env_keep at /etc/sudoers:

```
Defaults
```

```
env_keep = "PARAM_* COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC
KDEDIR \ LS_COLORS MAIL PS1 PS2 QTDIR USERNAME \


LANG LC_ADDRESS LC_CTYPE LC_COLLATE LC_IDENTIFICATION \
LC_MEASUREMENT LC_MESSAGES LC_MONETARY LC_NAME LC_NUMERIC \
LC_PAPER LC_TELEPHONE LC_TIME LC_ALL

LANGUAGE LINGUAS \ _XKB_CHARSET XAUTHORITY"
```

- Install dependencies of the application software:

```
requires: monit , libicu, libbz2.so.1, glibc >= 2.5,
libcrypto.so.6, libgcc >= 4.1.2 , libsasl2.so.2, libstdc++ >=
4.1.2 , libuuid.so.1, zlib
```

- To install monit, download it from: scp neoidasfe01:/root/monit* and install it:

```
rpm -i monit-5.2.5-1.el5.rf.x86_64.rpm
```

- Install the package with the application software:

```
sudo rpm -q neoidas-be; if [ $? -eq 1 ]; then echo "First
installation"; else sudo rpm -e neoidas-be-1.0; fi;

sudo rpm -Uvh -vv neoidas-be-1.0-
$RELEASE.release.rhel5.5.x86_64.rpm --prefix
/export/open/IDAS_BE > listado_instalac.txt 2>&1;
```

- To Launch the monit, run the following commands:

```
su - manager -c 'monit -v -c
/export/open/IDAS_BE/conf/monitIDAS.conf -d 10 -p
/export/open/IDAS_BE/run/monit.pid'

su - manager -c 'monit -v -c
/export/open/IDAS_BE/conf/monitIDAS.conf -p
/export/open/IDAS_BE/run/monit.pid start all'
```

4.1.2.1.4 IDAS ADMIN REST API installation

Prerequisites

- As root user, download and install JRE 1.7.04 (or a newer version 1.7.xx):

```
rpm –i jdk-7u4-linux-x64.rpm --force
```

- Make a symbolic link to /usr/java/default if the RPM failed to do so:

```
ln –s /usr/java/default /usr/java/jdk1.7.0_04
```

- Download and install Tomcat 7.x in /opt/apache-tomcat-7.0.28.tar.gz

```
cd /opt

tar xzvf apache-tomcat-7.0.28.tar.gz

mv apache-tomcat-7.0.28 tomcat

chown –R manager tomcat/
```

- Delete the ROOT folder (where the REST API will be installed) /opt/tomcat/webapps/ROOT

```
cd /opt/tomcat/webapps/

rm ROOT/ -rf
```

Edit Tomcat configuration (/opt/tomcat/conf/server.xml) adding a connector to port 5371:

```
<!-- A "Connector" represents an endpoint by which requests are
received

blocking)

and responses are returned. Documentation at :

Java HTTP Connector: /docs/config/http.html (blocking & non-

Java AJP Connector: /docs/config/ajp.html

APR (HTTP/AJP) Connector: /docs/apr.html
```

```
Define a non-SSL HTTP/1.1 Connector on port 8080
-->
<Connector port="5371" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```

- Edit Tomcat logging configuration (/opt/tomcat/conf/logging.properties), disabling jersey logging by adding the following line:

```
com.sun.jersey = OFF
```

- Install the following script at /etc/init.d/tomcat:

```
#!/bin/bash
#
# tomcat
#
# chkconfig:
# description: Start up the Tomcat servlet engine.
# Source function library.
. /etc/init.d/functions
RETVAL=$?
CATALINA_HOME="/opt/tomcat"
export JAVA_HOME=/usr/java/jdk1.7.0_04/
export CATALINA_OPTS="-DENV_IDAS=generic"
export CATALINA_PID=/opt/tomcat/bin/catalina.pid case "$1" in
start)
if [ -f $CATALINA_HOME/bin/startup.sh ];
          then
            echo $"Starting Tomcat"
```

```
            $CATALINA_HOME/bin/startup.sh
fi
;; stop)
if [ -f $CATALINA_HOME/bin/shutdown.sh ]; then
30 de 55
            echo $"Stopping Tomcat"
            $CATALINA_HOME/bin/shutdown.sh
        fi
;; *)
        echo $"Usage: $0 {start|stop}"
        exit 1
        ;;
esac
exit $RETVAL
```

- Install the following script delete_pids_rest.sh, as root user, at /etc/init.d/:

```
#!/bin/bash
case "$1" in
start)
rm -rf /export/open/IDAS_REST/run/*.pid ;;
stop)
;;
restart)
;;
condrestart)
;;
```

```
status)

;;

*)

echo "Usage: $0 {start|stop|restart|condrestart|status}" exit 1

esac exit
```

- Generate the following symbolic links as root:

```
cd /etc/rc3.d

ln -s ../init.d/delete_pids_rest.sh ./S96delete_pids_rest cd
/etc/rc5.d

ln -s ../init.d/delete_pids_rest.sh ./S96delete_pids_rest
```

- Update access rights and format of the files delete_pids (root user):

```
chmod 755 /etc/init.d/delete_pids* dos2unix
/etc/init.d/delete_pids*
```

- Copy the installation packages:

```
scp
develop@neoidasdes:$BASEPJ/SE.ss/packages.ss/neoidas.pkg/RPMS/*/
*rest*- $RELEASE.*.rpm .
```

- Define the environment variables:

```
export PARAM_FRONTEND_ADDR=10.95.11.236 export
PARAM_FRONTEND_BCUP=10.95.10.96 export
PARAM_BACKEND_ADDR=10.95.8.153 export
PARAM_BACKEND_BCUP=10.95.29.82 export PARAM_GW_ADDR=10.95.11.117

export PARAM_GW_BCUP=10.95.31.110

export PARAM_DBTYPE=MONGODB

export PARAM_DBNAME=nIDAS

export PARAM_MONGOS_PORT=27021
```

```
export                PARAM_SERVERSDB=vmongodb1-1:27019,vmongodb1-
2:27019,vmongodb1-3:27019 export PARAM_TOMCATPORT=5371
```

REST API Installation

- Install the REST interface package:

```
rpm -Uvh -vv neoidas-rest-1.0-canis.release.rhel5.5.x86_64.rpm -
-prefix /export/open/IDAS_REST/
```

- Modify, as root, the /etc/inittab, adding at the end of this file the following line:

```
mp:2345:respawn:su    -    manager    -c    "monit    -I    -c
/export/open/IDAS_REST/conf/monitIDAS.conf     -d     10     -p
/export/open/IDAS_REST/run/monit.pid"
```

4.1.2.2 *IoT Agent*

4.1.2.2.1 iotAgent-2.3.0.tar.gz installation

As root user, execute the following steps:

- Create export directory (if not existent): mkdir /export
- Download iotAgent-v23.tgz to export directory and uncompress it: tar xzvf iotAgent-v23.tgz
- This last command creates a directory (/export/INS_iotAgent) with the SW of iotAgent

4.1.2.2.2 EntityId pattern file (optional)

If you want to associate a pattern in sensor identifier with a EntityID type, you must create the file /etc/iotAgent/entityPatterns.conf. Otherwise all EntityID types will be set to “Sensor”.

As root user, execute the following steps:

```
cd /etc

mkdir iotAgent

vi entityPatterns.conf
```

In this file you can define the patterns, an example may be:

```
OUTSMART.NODE       Node
```

```
OUTSMART.AMMS       AMMS

OUTSMART.RG         Regulator
```

4.1.2.2.3 MongoDB user "iota" creation

As root user, execute the following steps:

- Go to the directory of mongoDB installation and execute mongo shell, for example:

```
cd /home/mongodb/bin

./mongo

In mongo shell, type:

use iota

db.addUser("iota", "iota")
```

- An example of this user creation may be:

```
[root@localhost bin]# ./mongo

connecting to: test

> use iota

switched to db iota

> db.addUser("iota", "iota")

{

        "user" : "iota",

        "readOnly" : false,

        "pwd" : "447d4cfa9f732ce7f69f35751febfd79",

        "_id" : ObjectId("5194e15f57714ad7c98e707c")

}

>
```

4.1.2.3 ***ETSI-M2M Agent***

4.1.2.3.1 etsim2magent-3.3.3.tar.gz installation

As root user, execute the following steps:

- Create export directory (if not existent): mkdir /export
- Download etsim2magent-3.3.3.tar.gz to export directory and uncompress it: tar xzvf etsim2magent-3.3.3.tar.gz
- This last command creates a directory (/export/INS_etsim2mAgent) with the SW of etsim2mAgent

4.1.3 Administration Procedures

4.1.3.1 *IDAS*

In order to stop or start a process, the scripts (installed by neoidas-monitbd-1.0-canis.release.rhel5.5.x86_64.rpm) should be used:

```
cd /home/mongodb/scripts
./stop_mongod.sh
```

Once neoidas package has been installed, all running nodes might be updated. For instance, if we would like to remove mongoc:

```
cd /home/mongodb/scripts; mv mongoc.monitrc mongoc.monitrc.seg
./reload_monit.sh
```

This makes the process to be out of the monitoring. However, to stop the process we need to run "stop".

The status of the processes can be obtained by executing: ./status_monit.sh

NOTE: All events aand alarms are sent to the mongodb user. If we want them to be received in other mail account, the easiest is to redirect mails to that new account:

```
cd
echo usuario@mail > .forward
```

Another way to get the information of the Database processes is through the monit supervision. At each host, we can run the following command:

```
/home/mongodb/scripts/status_monit.sh o lo que es lo mismo monit
-v        -c        /home/mongodb/scripts/monit.conf        -p
/home/mongodb/run/monit.pid status
```

Finally, the easiest way of monitoring the database processes is to use the web control offered by the monit itself. A third host can be authorized to access and collect the status of monit and its processes by using the mongodb user account.

```
cd /home/mongodb/scripts vi monit.conf set httpd port 2811 and

#use address localhost # only accept connection from localhost
allow localhost # allow localhost to connect to the server and
allow 10.91.91.10

:wq reload_monit.sh
```

Over the web control interface processes can be monitored, stopped, started, etc. from a single management host for all cluster machines. Access to the port 2811 (or the one configured alternatively) has to be granted.

4.1.3.2 *IoT Agent*

4.1.3.2.1 iotAgent process start

As root user, execute the following steps:

```
cd /export/INS_iotAgent

./start_iotAgent.sh
```

You can check that iotAgent has been successfully started, executing

```
ps -ef | grep iotAgent

And will appear a proccess

root      29665      1   0 07:20 pts/1     00:00:00 ./iotAgent -
vvvvv -ipbroker 0.0.0.0 -portbroker 1026 -pidpath ./iotAgent.pid
```

4.1.3.2.2 iotAgent start with monit supervision

As root user, execute the following steps:

```
cd /export/INS_iotAgent

./start_monit_IOTA.sh
```

You can check that iotAgent an his monit process have been successfully started, executing

```
ps -ef | grep iotAgent

And will appear two process

root       3729      1   0 18:20 ?          00:00:00 monit -v -c
/export/INS_iotAgent/monitIOTA.conf        -d        10        -p
/export/INS_iotAgent/monit.pid

root         3731         1    0  18:20 ?                00:00:00
/export/INS_iotAgent/iotAgent   -vvvvv   -ipbroker   0.0.0.0   -
portbroker 1026  -pidpath /export/INS_iotAgent/iotAgent.pid
```

4.1.3.3 *ETSI-M2M Agent*

4.1.3.3.1 etsim2mAgent process start

As root user, execute the following steps:

```
cd /export/INS_etsim2mAgent

./start_etsim2mAgent.sh
```

You can check that etsim2mAgent has been successfully started, executing

```
ps -ef | grep etsim2mAgent

And will appear a proccess

root     12010     1  0 Apr23 ?        00:00:03 ./etsim2mAgent -
t 0-255 -vvvvv -ipidas 0.0.0.0 -portidas 8002 -sclbase /m2m -
apikey xdaxpdfggq8hnrb -pidpath /tmp/etsim2mAgent.pid
```

4.2 Sanity Check Procedures

The Sanity Check Procedure is the first step that a System Administrator will do to verify that the GEI 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.

IDAS

For proceding with IDAS Sanity Check you can execute the curl:

```
curl -v --request GET http://0.0.0.0:5371/m2m/v2/services/Service_Dummy
```

If IDAS is well installed you must receive a response like:

```
* About to connect() to 0.0.0.0 port 5371
*   Trying 0.0.0.0... connected
* Connected to 0.0.0.0 (0.0.0.0) port 5371
> GET /m2m/v2/services/Service_Dummy HTTP/1.1
>     User-Agent:     curl/7.15.5     (x86_64-redhat-linux-gnu) libcurl/7.15.5 OpenSSL/0.9.8b zlib/1.2.3 libidn/0.6.5
> Host: 0.0.0.0:5371
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Server: Apache-Coyote/1.1
< Content-Type: application/json
< Transfer-Encoding: chunked
< Date: Fri, 28 Jun 2013 10:12:57 GMT
* Connection #0 to host 0.0.0.0 left intact
* Closing connection #0
{"exceptionId":"svc_1006","exceptionText":"Resource      SERVICE Service_Dummy does not exist"}
```

If there is a problem with IDAS installation, you will receive:

```
About to connect() to 0.0.0.0 port 5371 (#0)
*   Trying 0.0.0.0... ConexiÃ³n rehusada
* couldn't connect to host
* Closing connection #0
curl: (7) couldn't connect to host
```

IoT Agent

IotAgent Sanity Check can be made executing the curl:

```
curl --request POST  http://0.0.0.0:1029/ngsi10/queryContext  --
header  'Content-Type:  application/xml'  $CURL_VERBOSE  --data
"<queryContextRequest>


</queryContextRequest>"
```

With a right installation of iotAgent you must receive a response like:

```
<queryContextResponse>
<errorCode>
  <code>400</code>
  <reasonPhrase>Query Context Request failed</reasonPhrase>
  <details><queryContextResponse>
<errorCode>
  <code>400</code>
  <reasonPhrase>No entities</reasonPhrase>
</errorCode>
</queryContextResponse>
</details>
</errorCode>
```

```
</queryContextResponse>
```

When there is a problem with iotAgent installation, you will receive:

```
curl: (7) couldn't connect to host
```

4.2.1 End to End testing

The end-to-end test is able to send a discovery request to the GEI and receive the response.

IDAS + IoT Agent

The following curl command sends a request to the IoT-Agent that will progress in turn the request to IDAS.

```
curl --request POST http://0.0.0.0:1029/ngsi10/updateContext --header 'Content-Type: application/xml' $CURL_VERBOSE --data "<updateContextRequest><contextElementList><contextElement><entityId type=\"Sensor\" isPattern=\"\"><id>SENSOR_E2E</id></entityId><contextAttributeList><contextAttribute><name>service</name><type></type><contextValue>Service_E2E</contextValue></contextAttribute><contextAttribute><name>PropertyE2E</name><type>urn:x-ogc:def:phenomenon:IDAS:1.0:PropertyE2E</type><contextValue>58</contextValue></contextAttribute></contextAttributeList></contextElement></contextElementList><updateAction>update</updateAction></updateContextRequest>"
```

If the IoT-Agent and IDAS work in the right way, request is received by IDAS which search the service in database and response 404 “Not Found” because Service_E2E is not found in IDAS SERVICE mongo collection.

```
<updateContextResponse>
<contextResponseList>
  <contextElementResponse>
    <contextElement>
      <entityId type="Sensor" isPattern="">
        <id>SENSOR_E2E</id>
      </entityId>
```

```
      <contextAttributeList>
        <contextAttribute>
          <name>service</name>
          <type></type>
          <contextValue>Service_E2E</contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>PropertyE2E</name>
          <type>urn:x-
ogc:def:phenomenon:IDAS:1.0:PropertyE2E</type>
          <contextValue>58</contextValue>
        </contextAttribute>
      </contextAttributeList>
    </contextElement>
    <statusCode>
      <code>404</code>
      <reasonPhrase>Not Found</reasonPhrase>
    </statusCode>
  </contextElementResponse>
</contextResponseList>
</updateContextResponse>
```

In the event the IoT-Agent is not able to progress the request to IDAS, you will get a 500 “Internal Error”.

```
<updateContextResponse>
<contextResponseList>
  <contextElementResponse>
    <contextElement>
      <entityId type="Sensor" isPattern="">
```

```
        <id>SENSOR_E2E</id>
      </entityId>
      <contextAttributeList>
        <contextAttribute>
          <name>service</name>
          <type></type>
          <contextValue>Service_E2E</contextValue>
        </contextAttribute>
        <contextAttribute>
          <name>PropertyE2E</name>
          <type>urn:x-
ogc:def:phenomenon:IDAS:1.0:PropertyE2E</type>
          <contextValue>58</contextValue>
        </contextAttribute>
      </contextAttributeList>
    </contextElement>
    <statusCode>
      <code>500</code>
      <reasonPhrase>Internal Error</reasonPhrase>
    </statusCode>
  </contextElementResponse>
</contextResponseList>
</updateContextResponse>
```

If there is a communication problem with iotAgent you will receive:

```
curl: (7) couldn't connect to host
```

4.2.2 List of Running Processes

IDAS

- Apache Tomcat
- MongoDB

IoT Agent

- MongoDB

4.2.3 Network interfaces Up & Open

IDAS

- TCP: 5371

IoT Agent

- TCP :1029

4.2.4 Databases

IDAS

- MongoDB

As first step connect to mongodb and select nIDAS database

```
cd /home/mongodb/bin

./mongo

> use nIDAS
```

Then make a query test, such as:

```
> db.PHENOMENON.find({"name" : "longitude"})
```

If IDAS is correctly, result will be similar to this one:

```
{    "_id" :    ObjectId("51dd22d1a78e2571911af932"),    "name" :
"longitude",              "quantityType" :              "urn:x-
ogc:def:phenomenon:IDAS:1.0:longitude",  "translation" :  [     {
"language" : "es",      "translation" : "Longitud geografica" },
{       "language" : "en",      "translation" : "Longitude" } ],
"units" : [ ] }
```

IoT Agent

- MongoDB

As first step connect to mongodb and select iota database

```
cd /home/mongodb/bin

./mongo

> use iota
```

Then make a query test, such as:

```
db.system.users.find()
```

If iotAgent is correctly, result will be similar to this one:

```
{ "_id" : ObjectId("51dd2dbdc1afd1045fb9b9b5"), "user" : "iota",
"readOnly" : false, "pwd" : "xxxxxxxx" }
```

4.3 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.

4.3.1 Resource availability

This section states the amount of available resources in terms of RAM and hard disk that are necessary to have a healthy enabler. This means that bellow these thresholds the enabler is likely to experience problems or bad performance.

IDAS M2M Platform MongoDB and Apache Tomcat need an amount of available resources. In addition there must be 1 Gigabyte of hard disk available necessary for RPMS, binaries, and traces of IDAS processes.

IoT Agent

MongoDB needs an amount of available resources. In addition there must be 200 Megabytes of hard disk available necessary for iotAgent bianary and traces.

4.3.2 Remote Service Access

This section states what enablers talk with the Backend Device Management GE and give the parameters to characterise such connection. The administrator will use this information to verify that such links are available.

IDAS M2M Platform

Administration API REST will be available in port 5371 of the host in which IDAS is installed.

Sensor devices will communicate with IDAS through port 8002.

IoT Agent
NGSI API of iotAgent will be available in port 1029.

This value con be changed using the '-port' command line option at starting iotAgent.

4.3.3 Resource consumption

This section states the amount of resources that are abnormally high or low. It applies to RAM, CPU and I/O.

IDAS M2M Platform
In IDAS processes a RAM usage higher than 100 Megabytes, and CPU usage higher than 70% can be considered abnormal.

IoT Agent
For iotAgent a RAM usage higher than 50 Megabytes, and CPU usage higher than 50% can be considered abnormal.

4.3.4 I/O flows

This section states what a normal I/O flow are in terms of flows to be configured at firewalls.

IDAS M2M Platform
Input flows:

- IDAS listen to connections in the ports 5371, 8002 , and will receive HTTP POST messages.

Output flows:

- IDAS response to HTTP messages received in the listen ports.

IoT Agent Input flows:

- The IoT Agent listen to connections in the port 1029 , and will receive HTTP POST whit NGSI messages.

Output flows:

- The Iot Agent will send NGSI messages to contextBroker, which IP and port con be set with '–ipbroker' and '-portbroker' command line option at starting.

5 Gateway Device Manager - OpenMTC - Installation and Administration Guide

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

5.1 Introduction

This document describes how to run the Fraunhofer FOKUS OpenMTC software as well as the corresponding NGSI 9/10 interface components.

5.2 System Requirements

5.2.1 Reference Platform

Although the OpenMTC software package is designed to run on any platform that supports the NodeJS framework and the MongoDB database software, we define a reference platform for the purpose of this document. If you experience any issues or errors in the following steps, please make sure to repeat those steps on a machine that is configured exactly as follows before seeking support.

The OpenMTC reference platform consists of the following:

- Ubuntu Linux 12.04 64Bit
- Intel x86_64 CPU clocked at 2GHz or higher
- at least 1GByte of physical RAM
- at least 10GByte of free Hard Disk Space
- single partition layout
- NodeJS 0.8.16
- MongoDB 2.0.4
- GNU make 3.81
- Python 2.7
- GNU screen
- werkzeug
- jinja2
- lxml

- python-iso8601
- python-six

5.2.2 Prerequisites

Before running the software, the following dependencies must be installed manually:

- NodeJS framework
- MongoDB
- GNU make
- GNU screen

On the reference platform described above, the MongoDB and GNU make packages can be installed through following command:

```
$ sudo apt-get install mongodb make screen python-werkzeug python-jinja2 python-lxml python-iso8601 python-six python-setuptools
```

followed by:

```
$ sudo easy_install flask-restful
```

However, the required NodeJS version is currently not yet available in Ubuntu's official software repositories. It must therefore be installed manually as described below.

5.2.2.1 *Installing NodeJS*

To build and install NodeJS from the NodeJS source distribution, a C compiler as well as the OpenSSL development headers are required. The following command will install these packages on the reference platform:

```
$ sudo apt-get install gcc libssl-dev g++ make
```

Then, the actual NodeJS package must be downloaded, compiled and installed by issuing the following commands:

```
$ wget -qO - http://nodejs.org/dist/v0.8.16/node-v0.8.16.tar.gz | tar -xz

$ cd node-v0.8.16

$ ./configure && make && sudo make install
```

5.3 Installation

The OpenMTC software distribution is fully self contained and does not need to be installed to a specific location. To prepare the software distribution, please follow these steps:

- Either unpack the distribution archive or checkout the source code from the repository.
- Change to the distribution directory: `cd openmtc`

5.4 Running the Software

The OpenMTC software is now ready to run. The individual components can be run directly from the distribution package.

To run the OpenMTC GSCL:

```
$ ./gscl
```

To run the OpenMTC NGSI 9/10 interface components:

```
$ ./openmtc-ngsi_9_10
```

When the software is running you should be able to access the root of the MTC resource tree with a web browser.

By default, the software is configured to expose the resource tree at this URLs:

- GSCL: `http://localhost:5000/m2m`
- NGSI9: `http://localhost:5050/ngsi9`
- NGSI10: `http://localhost:5050/ngsi10`

A simple, human readable greeting page is available at these URLs:

- GSCL: `http://localhost:5000/`

5.5 Configuration

Note that for the basic operation described in this document, no further configuration is neccessary. Things should "just work". If they don't, you may have encoutenred an error in our software. Please notify us in this case.

This is an overview of common configuration options applicable to both NSCL and GSCL. The general mode of operation used in the OpenMTC software is that when a configuration value applies to both SCLs but different values are desirable, reasonable defaults are set in the common configuration which are subsequently overriden in the GSCL or NSCL configuration.

5.5.1 Database Configuration

The following configuration configure how the OpenMTC backend database is accessed.

```
   db.driver = './db/mongoose'; //This controls which actual
database driver is to be used.

                                  //At present, `'./db/mongoose'`
is the only supported driver.
```

```
   db.host = 'localhost'; //The hostname or IP of the machine
running the database server.

                           //For the default configuration, we
assume that it is running on the same host as the OpenMTC
software.

   db.port = 27017; //The TCP port the database server is
listening on for incoming connection. 27017 is the default for
mongodb.

   db.dropDB = false; //When set to `true`, the backend database
will be erased and re-initialized on every startup.

                       //Useful for reproducable debugging.

   db.database = 'm2m'; //The name of the database to use.

                         //The default for this option "m2m" is
overridden in the NSCL configuration and GSCL configuration to
be "m2m_nscl" and "m2m_gscl" respectively.
```

5.5.2 SCL Configuration

The following options control the runtime behaviour of the SCL.

```
   scl.host = 'localhost';

   scl.id = 'openmtc';

   // internal port

   scl.port = 2000;

   // mId config

   scl.mid.port = '14000';

   // dIa config

   scl.dia.port = '15000';
```

```
   scl.dia.network = '10.0.0.1/24';

   //SSL configuration

   //When these two config keys are present, SSL will be used
for incoming connections

   //Filenames must be specified either absolute or relative to
the project directory.

   scl.mid.ssl.cert = './openmtc-GSCL/keys/mid-cert.pem'; //SSL
certificate

   scl.mid.ssl.key = './openmtc-GSCL/keys/mid-key.pem';    //SSL
certificate key
```

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. 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.

5.6.1 End to End testing

To quickly verify if the software is running correctly, one can use a browser to access the human readable greeting page that is provided by the applicationwhich is available at http://{serverRoot}:5000.

Likewise, the `curl` command (or again a browser) can be used to retrieve the `sclBase` resource of the M2M resource tree, which is exposed at http://{serverRoot}:5000/m2m. The output of this should look roughly like the following:

```
 $ curl http://localhost:5000/m2m

   {

   "sclBase": {

     "creationTime": "2013-06-11T15:23:25.804Z",

     "lastModifiedTime": "2013-06-11T15:23:25.804Z",

     "searchStrings": {

       "searchString": [

         "GSCL Base"
```

```
        ]
      },
      "aPocHandling": "SHALLOW",
      "accessRightID": "/m2m/accessRights/sclBaseAccessRight",
      "sclsReference": "/m2m/scls",
      "applicationsReference": "/m2m/applications",
      "containersReference": "/m2m/containers",
      "groupsReference": "/m2m/groups",
      "accessRightsReference": "/m2m/accessRights",
      "subscriptionsReference": "/m2m/subscriptions",
      "discoveryReference": "/m2m/discovery"
    }
}
```

5.6.2 List of Running Processes

Under normal conditions, the following processes should be running:

- The NodeJS process running the OpenMTC GSCL

```
$ pgrep -a node
10477 node ./gscl
```

- The MongoDB process

```
$ pgrep -a mongo
6267 /usr/bin/mongod --bind_ip 127.0.0.1 --journal
```

- The Python web container that hosts the NGSI interface parts:

```
$ pgrep -af ngsi
10536 /usr/bin/python2.7 openmtc_ngsi/wsgi_flask.py
```

5.6.3 Network interfaces Up & Open

By default, the software accepts connections on the following ports:

- mId interface: 5000 tcp

- mIa interface: 4000 tcp
- NGSI9/10: 5050 tcp

In addition, the MongoDB database will by default accept connections on port 27017(tcp). When hosting the database on the same system as the GE (as describe din this guide), internal (loopback) communication for this port will suffice.

5.6.4 Databases

The default installation of this GE uses the MongoDB database system which by default accepts connections on localhost:27017 (tcp).

To test if the database is running, the following simple command can be used:

```
$ echo "db.version()" | mongo --quiet

2.2.2
```

The test is sucessful when the MongoDB software version is displayed and the command returns successfully (exit code 0). It is sufficient that the database is up and running for the GE to work. An uninitialized database will automatically be initialized by the software.

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 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.

5.7.1 Resource availability

A typical deployment should run well with the following amount of resources available:

- RAM: >= 1GByte
- CPU: >= 1GHz
- HD: >= 10 GByte

Note that it is possible to run the system with significantly less harddisk space by starting the MongoDB software with the --smallfiles commandline parameter. This can however impact performance in a negative way. Disk usage heavily depends on how much data is stored. In any case note that in its default configuration, MongoDB behaves rather lavish when preallocating disk space.

5.7.2 Remote Service Access

Other GEs are foreseen to access the system via NGSI9/10 (port 5050 tcp). The Backend Device Management GE is foreseen to access the mId interface on port 5000 (tcp), while ETSI M2M compliant devices will access the dIA interface on port 4000 (tcp).

5.7.3 Resource consumption

The following resource consumption is considered abnormal:

- CPU: >=40% over prolonged periods of time without outside activity.
- RAM: >=1Gbyte

5.7.4 I/O flows

All expected I/O is HTTP, on ports 5000 (mId), 4000 (dIa) and 5050 (NGSI9/10).

6 Gateway Protocol Adapter - ZPA - Installation and Administration Guide

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

6.1 Introduction

The ZPA is an implementation of the Protocol Adapter GE which allows the IoT Gateway to communicate with ZigBee devices. This is its Installation and Administration Guide which allows two types of installations: the NGSI standalone installation and the installation together with the Gateway Device Management GE by Ericsson. N.B. The installation together with the Gateway Device Management GE by Ericsson is still available but it is probable that will in the future it be dropped in favor of the NGSI standalone installation (because of the withdrawal of Ericsson from the project).

In the following sections both types of installation are described.

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.

6.2 System Requirements

6.2.1 Reference Platform

Although the ZPA software package is designed to run on any platform that supports the OSGI framework, we define a reference platform for the purpose of this document. In particular the ZPA reference platform is based on using Ubuntu Linux 12.04 as the Operation System.

6.3 System Installation

6.3.1 NGSI standalone installation

In this case the ZPA is a set of bundles that run on an OSGi knopflerfish framework.

The repository that contains the files for this type of installation is the 'Files' tool that you may find at this link https://forge.fi-ware.eu/frs/?group_id=23 (Access Restricted to PPP Programme members). In this page refer to the package "IoT-ZPA" release 3.3.3.

Download from the repository "ZPAKnoplerfishOsgi_3.3.3.zip" and unzip it. Into the directory <knopflerfish home>/osgi edit the file telecomitalia.xargs and put the actual values in the following lines:

```
-Dit.telecomitalia.zgd.gateway-root= http:// <ip_gal>: <port>
```

where <ip_gal> is the IP where the GAL is installed (for the GAL installation see the following steps) and <port> is the port on which it listens

```
-Dit.telecomitalia.zgd.local-address = <ip_protocoladapter>
```

where <ip_protocoladapter> is the local IP where the OSGi framework is installed

Into the directory <knopflerfish home>/osgi/jars/telecomitalia_jars there are all the bundles needed for the standalone installation for the ZPA.

Into the directory <knopflerfish home>/osgi/ the file startTelecomItalia and then start it:

```
sh startTelecomItalia.sh or startTelecomItalia.bat
```

When Knopflerfish has started please type:

```
ps
```

in the console to check which bundles are running.

You should see at least:

- it.telecomitalia.osgi.zigbee.pa
- Generic Device Access Library

In order to run the ZigBee gateway on a PC it's needed:

- install the GAL (Gateway Abstraction Layer). The GAL is a sw that implements the ZigBee specification ZigBee Gateway Device (see http://www.zigbee.org/Standards/ZigBeeNetworkDevices/Overview.aspx)
- acquire and install an USB dongle. The USB dongle certified is "FlexKey" and it is produced by FlexGrid (http://www.flexgrid.it). It is important to install a proper firmware that is able to work with the GAL, and the current release is called "Black Box_combo_StdSecLinkKey_31_08_12.bin".
- acquire some ZigBee nodes. At the moment the GE is certified with the "ePlug" smart plugs produced by 4-noks (http://www.4-noks.it/intellygreen/) and with the "FlexPlug" smart plugs produced by CEDAC (http://www.cedac.com). But in principle it is possible to use any smart plug ZigBee certified implementing the OnOff Cluster and the Metering Cluster.

The FlexKey USB dongle and the FlexPlug smart plugs may be acquired by CEDAC (http://www.cedac.com), while the ePlug smart plugs may be acquired by 4-noks (http://www.4-noks.it/intellygreen/).

The next step is the installation of the component GAL following these instructions:

- Go to 'Files' tool (https://forge.fi-ware.org/frs/?group_id=23 - Access Restricted to PPP Programme members), go to IoT ZPA package at release 3.3.3 and download and unzip the file GAL5.3.4.zip
- Insert the FlexKey USB dongle in the USB port of the PC
- If needed, install the drivers for the USB dongle from the site http://www.ftdichip.com/Drivers/VCP.htm (the file "CDM 2.08.24 WHQL Certified.zip" shall be downloaded).
- Select "MyComputer" icon on the desktop, right click the mouse, select "Properties" and then "Device Manager". Verify which COM port is used under "Ports" label (for example COM6).
- Edit the file "start CEDAC.bat" found in the directory GAL5.3.4 by replacing the port COM with the value of the COM port subtracted by 1 (for example if the port is COM6, the path in the .bat file shall be ttyS5)
- Start the GAL by launching "start CEDAC.bat"

Finally follow the instructions to set up the smart plugs:

- First of all, be sure that the node is "virgin", that is, it has no joined any previous networks.

To be sure, a "reset-to-factory default" could be done: keep the button of the FlexPlug pressed when the node is powered on/plugged in: you will see the red and blue lights blinking for few seconds and then a stable red led. When done, unplug the node and you're concluded this step.

- When Knopflerfish has started and the process is running you can open a browser and type the following url:

http://localhost:8080/zpa/openNwk a message will be shown in the browser: "The ZigBee network will be open for 60 seconds" You have 60 seconds to do the next procedure (described in the next step)

- After the permitJoin flag has been activated, just plug the node and keep pressed the button:

after few seconds it will start blinking (blue led): at that point you can release the button since the node has started the association procedure and it will soon join the network (the flexplug will stop blinking when it joins the network).

6.3.2 Installation together with the Gateway Device Management GE

In this case the ZPA is a set of bundles that run on the OSGi framework of the Ericsson Gateway (since the Ericsson withdrew from the project, the Ericsson Gateway will be substituted from OpenMTC of Fraunhofer).

```
The prerequisites are:
```

- the Gateway Device Management GE has already been installed (the instructions were in the section “System Installation” of the Ericsson Gateway Device Management)
- inside the system there is JavaSE 1.6 installed (this is the minimum requirement).

The files for this type of installation may be found in the 'Files' tool at this link https://forge.fi-ware.org/frs/?group_id=23 (Access Restricted to PPP Programme members), scroll until the IoT ZPA package, release 3.3.3.

Download from the 'Files' tool repository the file telecomitalia.xargs. This file contains the properties and the list of bundles to install in the OSGI framework. Edit this file and put the actual values in the following lines:

```
-Dit.telecomitalia.zgd.gateway-root= http://<ip_gal>:<port>
```

where <ip_gal> is the IP where the GAL is installed (for the GAL installation see the following steps) and <port> is the port on which it listens

```
-Dit.telecomitalia.zgd.local-address = <ip_protocoladapter>
```

where <ip_protocoladapter> is the local IP where the OSGi framework is installed

-Dorg.osgi.service.http.port=8080

Download from the 'Files' tool repository the file telecomitaliaStart.xargs. This file contains the list of bundles to start in the OSGI framework.

Place these files telecomitalia.xargs and telecomitaliaStart.xargs into the directory <knopflerfish home>/osgi.

Edit the file props.xargs into <knopflerfish home>/osgi and comment out the line -Foscar.repository.url=http://www.knopflerfish.org/releases/3.2.0/repository.xml

Edit init.xargs into the directory <knopflerfish home>/osgi, comment out the following lines

-install bundlerepository/bundlerepository_all-3.1.2.jar -start bundlerepository/bundlerepository_all-3.1.2.jar

Then, always in init.xargs, add the following lines:

after the line: "-initlevel 6" add the line: "-install io/io_all-3.0.0.jar"

after the line: "-initlevel 7" add the line: "-install kf_metatype/kf_metatype_all-3.0.1.jar"

after the line: "-start desktop/desktop_all-3.1.6.jar" add the line: "-start io/io_all-3.0.0.jar"

Download from the 'Files' tool repository the file ZPAKnoplerfishOsgi_3.3.3.zip, unzip it and obtain the directory ZPAKnoplerfishOsgi_3.3.3. In this directory there are all the bundles needed for the installation for the ZPA. Since the installation of the Gateway Device Management GE has already been done, remove from this directory two bundles: generic.device.access-1.37-SNAPSHOT.jar and generic.device.access.services-1.37-SNAPSHOT.jar. Place this folder and its content (telecomitalia_jars) into <knopflerfish home>/osgi/jars/

Download from the 'Files' tool repository the file startTelecomItalia and then start it:

```
sh startTelecomItalia.sh or startTelecomItalia.bat
```

When Knopflerfish has started please type:

```
ps
```

in the console to check which bundles are running.

You should see at least:

- it.telecomitalia.osgi.zigbee.pa
- Generic Device Access Library

In order to run the ZigBee gateway on a PC it's needed:

- install the GAL (Gateway Abstraction Layer). The GAL is a sw that implements the ZigBee specification ZigBee Gateway Device (see http://www.zigbee.org/Standards/ZigBeeNetworkDevices/Overview.aspx)
- acquire and install an USB dongle. The USB dongle certified is "FlexKey" and it is produced by FlexGrid (http://www.flexgrid.it). It is important to install a proper firmware that is able to work with the GAL, and the current release is called "Black Box_combo_StdSecLinkKey_31_08_12.bin".
- acquire some ZigBee nodes. At the moment the GE is certified with the "ePlug" smart plugs produced by 4-noks (http://www.4-noks.it/intellygreen/) and with the "FlexPlug" smart plugs produced by CEDAC (http://www.cedac.com). But in principle it is possible to use any smart plug ZigBee certified implementing the OnOff Cluster and the Metering Cluster.

The FlexKey USB dongle and the FlexPlug smart plugs may be acquired by CEDAC (http://www.cedac.com), while the ePlug smart plugs may be acquired by 4-noks (http://www.4-noks.it/intellygreen/).

The next step is the installation of the component GAL following these instructions:

- Go to 'Files' tool (https://forge.fi-ware.org/frs/?group_id=23), go to IoT ZPA package at release 3.3.3 and download and unzip the file GAL_5.3.4-3.3.3.zip
- Insert the FlexKey USB dongle in the USB port of the PC
- If needed, install the drivers for the USB dongle from the site http://www.ftdichip.com/Drivers/VCP.htm (the file “CDM 2.08.24 WHQL Certified.zip” shall be downloaded).
- Select “MyComputer” icon on the desktop, right click the mouse, select “Properties” and then “Device Manager”. Verify which COM port is used under “Ports” label (for example COM6).
- Edit the file “start CEDAC.bat” found in the directory GAL5.3.4 by replacing the port COM with the value of the COM port subtracted by 1 (for example if the port is COM6, the path in the .bat file shall be ttyS5)
- Start the GAL by launching “start CEDAC.bat”

Finally follow the instructions to set up the smart plugs:

- First of all, be sure that the node is “virgin”, that is, it has no joined any previous networks.

To be sure, a “reset-to-factory default” could be done: keep the button of the FlexPlug pressed when the node is powered on/plugged in: you will see the red and blue lights blinking for few seconds and then a stable red led. When done, unplug the node and you’re concluded this step.

- When Knopflerfish has started and the process is running you can open a browser and type the following url:

http://localhost:8080/zpa/openNwk a message will be shown in the browser: "The ZigBee network will be open for 60 seconds" You have 60 seconds to do the next procedure (described in the next step)

- After the permitJoin flag has been activated, just plug the node and keep pressed the button:

after few seconds it will start blinking (blue led): at that point you can release the button since the node has started the association procedure and it will soon join the network (the flexplug will stop blinking when it joins the network).

6.4 System Administration

In order to stop or start a process, these instructions and scripts should be used:

```
ps  -ef | grep java

kill -9 <pid java process>

cd /root/out

rm SDTOUT.log

cd ../knopflerfish_osgi_3.2.0/osgi

rm -rf  fwdir

./startTelecomItalia.sh
```

In order to troubleshoot the installation you may use the command

```
log
```

in the OSGi console which allows to get the OSGi logs

6.5 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.

6.5.1 End to End testing

An end-to-end test allows to actuate (switch on-off) and read the power data of a smart plug.

Go to 'Files' tool (https://forge.fi-ware.org/frs/?group_id=23), go to IoT ZPA package at release 3.3.3 and download for both types of installations it.telecomitalia.osgi.client.zigbee.pa_1.1.0.-SNAPSHOT.jar [N.B. this jar contains the client that was developed in release 1.1 but can be used with the current release too] with these commands:

```
*  Install  file:  it.telecomitalia.osgi.client.zigbee.pa_1.1.0.-
SNAPSHOT.jar

*    Start        it.telecomitalia.osgi.client.zigbee.pa_1.1.0.-
SNAPSHOT.jar
```

If this bundle is already active it is possible to access to the Protocol Adapter with the following url:

```
http://<protoadapter_ip>:8080/zpa
```

You should get back this title:

```
The ZigBee Protocol Adapter is running!
```

At this URL: http://<protoadapter_ip>:8080/zpa/ the servlet uploaded shows a list of smart plugs discovered and when you check one of these the related details (included the actuation command) are shown. So you can execute the action to switch on/off the device selected: if an electrical device is connected to the smart plug it is possible to turn off or on the device and read its value of power. If you refresh the browser it is possible to see the changes of status of the devices.

6.5.2 List of Running Processes

Installation together with the Gateway Device Management GE

- 7/active Eclipse Jobs Mechanism
- 7/active Common Eclipse Runtime
- 7/active Equinox Util Bundle
- 7/active Noelios Restlet Engine - Extension - Net
- 7/active Noelios Restlet Engine
- 7/active mToolkit Instrumentation Agent RPC Bundle
- 7/active luminis Configuration Admin command line client
- 7/active Http Service Registry Extensions
- 7/active Http Services Servlet
- 7/active Servletbridge Http Service
- 7/active Initial Provisioning
- 7/active Wire Admin Service
- 7/active Pax ConfMan - Properties Loader
- 7/active OPS4J Pax Web - Extender - Whiteboard
- 7/active ZigBee GAL Launcher and Service
- 7/active Home Automation Core Service Lib
- 7/active ZigBee Network Manager
- 7/active jGAL Service
- 7/active Equinox

- 7/active Generic Device Access Library
- 7/active it.telecomitalia.osgi.zigbee.pa

Standalone installation

- 7/active Eclipse Jobs Mechanism
- 7/active Common Eclipse Runtime
- 7/active Equinox Util Bundle
- 7/active Noelios Restlet Engine - Extension - Net
- 7/active Noelios Restlet Engine
- 7/active mToolkit Instrumentation Agent RPC Bundle
- 7/active luminis Configuration Admin command line client
- 7/active Http Service Registry Extensions
- 7/active Http Services Servlet
- 7/active Servletbridge Http Service
- 7/active Initial Provisioning
- 7/active Wire Admin Service
- 7/active Pax ConfMan - Properties Loader
- 7/active OPS4J Pax Web - Extender - Whiteboard
- 7/active ZigBee GAL Launcher and Service
- 7/active Home Automation Core Service Lib
- 7/active ZigBee Network Manager
- 7/active jGAL Service
- 7/active Equinox
- 7/active Generic Device Access Library
- 7/active it.telecomitalia.osgi.zigbee.pa
- 7/active it.telecomitalia.osgi.client.zigbee.pa

Test to check if the ports are opened

Type the following command to check the ports status

```
netstat -na |grep 8080
```

```
netstat -na |grep 9000

netstat -na |grep 9100
```

6.5.3 Network interfaces Up & Open

Usually the GAL and OSGi framework are on the same PC but it is possible that the GAL runs in a different PC. In this case the TCP, UDP port 9000, 8080 and 9100 should be open.

6.5.4 Databases

N/A

6.6 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.

6.6.1 Resource availability

The minimum HW requirements for the OSGi platform are:

- 16 MB flash for boot image (could work on 8M as well)
- 16 MB flash for file system
- 64 MB RAM (can work on 32 MB as well)
- gt 300 Mhz CPU

The minimum software requirements are:

- Java 1.6

6.6.2 Remote Service Access

The ZigBee devices will connect directly to the ZPA over a radio interface IEEE 802.15.4. The northbound interface of the ZPA allows to communicate with the Gateway Device Management GE using an internal java interface

6.6.3 Resource consumption

There are currently no data on resource consumption for the ZPA. Usually it runs correctly on medium level PC

6.6.4 I/O flows

On the port 9000 where the GAL is listening there is a REST traffic between the bundles of the ZPA and the ZigBee FlexKey USB dongle. This interface is defined in the ZigBee specification ZigBee Gateway Device

7 Gateway Protocol Adapter - EPCGE - 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 "EPC Generic Enabler" Installation and Administration Guide. This online document is continuously updated and improved to provide the most up-to-date informations. This guide provides a step-by-step instruction to get EPCGE installed and running on the Fiware testbed, as well as a list of requirements and software to set up on the machine.

7.1.1 EPCGE overview

EPCGE is a web service that is able to track an item or a product identified by a tag. The tag stores a unique EPC (Electronic Product Code) that allows the traceability of the product which is done through a set of recorded events thanks to the RFID (Radio Frequency IDentification) system.

7.2 System installation

This EPC GE is composed of 2 software components to install on the testbed : the "web application" and the "resolver". The "web application" allows the user first, to enter an EPC then, to get all the recorded data related to this product and finally, displays the path of traceability on Google Maps. The "resolver" handles the user's demand by requesting other services like the ONS (Object Naming Service) and the DS (Discovery Service) and retrieves all the recorded events from the EPCIS (EPC Information Service).

7.2.1 Prerequisite

Java, Glassfish and Mysql Server are required and must be installed. To do that, perform the followings steps :

1. Download and extract Java JDK 1.7.0 under /usr/lib/jvm/ directory.
2. Download and install the server container environment glassfish-3.1.1.zip and unzip the file in a <glassfish-directory>.
3. Download and set up the Mysql Server mysql.
4. Set up JAVA_HOME, GLASSFISH_HOME and PATH in /etc/bash.bashrc by adding, for instance :
    - export JAVA_HOME=/usr/lib/jvm/jdk1.7.0

- export GLASSFISH_HOME=<glassfish-directory>
- export PATH=$PATH:$JAVA_HOME/bin:$GLASSFISH_HOME/bin

5. Set up JAVA_HOME, AS_JAVA and GLASSFISH_HOME in /etc/environment by adding, for instance :
   - JAVA_HOME="/usr/lib/jvm/jdk1.7.0"
   - AS_JAVA="/usr/lib/jvm/jdk1.7.0"
   - GLASSFISH_HOME=<glassfish-directory>
6. If needed, edit the file *my.cnf* in the /etc/mysql/ directory and change the value of *wait_timeout* and *interactive_timeout*.

7.2.2 EPC GE

1. Download and extract the EPCGE-3.2.1.zip file. The directory EPCGE-3.2.1 will be automatically created and it will contain files concerning the web application, the resolver, the local epcis and glassfish :
   - copy *EPCGE-TESTBED.war* file to the <glassfish-directory>/glassfish/domains/domain1/autodeploy/ directory. For instance,<glassfish-directory> =/root/glassfish3/
   - copy *epcisrest-0.0.3.war* file to the <glassfish-directory>/glassfish/domains/domain1/autodeploy/ directory.
   - copy *domain.xml* file to the <glassfish-directory>/glassfish/domains/domain1/config/ directory.
   - copy the *ApplicationInterface* directory to <ApplicationInterface-directory> (for instance, ApplicationInterface-directory=/root/wingsTestbed/ApplicationInterface)

7.2.3 Databases

The EPCGE access to remote databases for the ONS, DS and EPCIS services. But, we still need a local database to store readers and their coordinates.

1. Set up a MySQL database for the local EPCIS repository to use. Perform the following steps:
   - Log into the MySQL Command Line Client as root

   mysql -u root -p

   - Create the database *epcis*

mysql> CREATE DATABASE epcis;

- Create a user *epcis* with a password *epcis* that can access to the newly created database *epcis*.

mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON epcis.* TO epcis IDENTIFIED BY 'epcis';

- Create the database schema by running *epcis_schema.sql* :

mysql> USE epcis;

mysql> SOURCE <EPCGE-directory>/epcis_schema.sql;

- Populate the tables with the readers and their coordinates by running *readpoint.sql* and *coordreadpoint.sql* :

mysql> SOURCE <EPCGE-directory>/readpoint.sql;

mysql> SOURCE <EPCGE-directory>/coordreadpoint.sql;

7.2.4 Running EPCGE

Once all the softwares have been installed, you must in the order:

- check that Mysql Server is running :
  - status mysql

```
mysql start/running, process 17078
```

- if Mysql is stopped, you must launch it :
  - sudo start mysql

```
mysql start/running, process 17078


NOTE: if you  start or restart (stop then start) Mysql Server, you must start or restart (stop then start) the web application !!
```

- start the EPCGE Resolver which is a RMI application.
  - cd <ApplicationInterface-directory>
  - ./launch.sh

```
binding...

server started at address : //localhost:1099/epcinterface
```

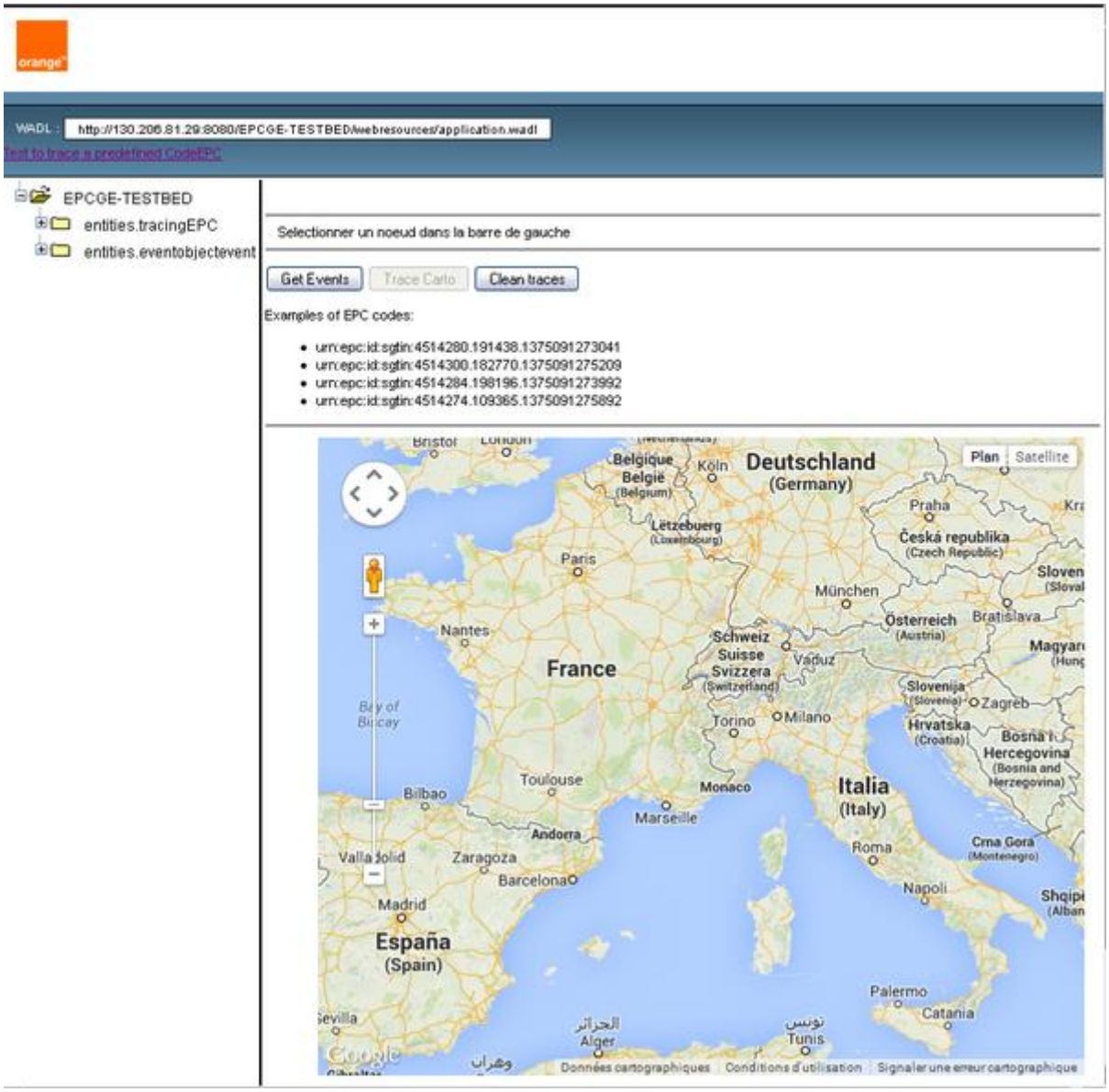
- When the EPCGE Resolver is launched, it starts 2 processes :*orange.applicationlevel.io.serveurRMI* and *rmiregistry*. You can check that by entering the following command :
    - ps -cafe | grep rmi

```
root     19422 19421 TS   19 Sep03 ?        00:00:42 rmiregistry

root      19423 19421 TS    19 Sep03 ?          00:00:50 java -
Djava.rmi.server.hostname=localhost                              -
Djava.security.policy=./resources/no.policy

-
Djava.rmi.server.codebase=file:///root/wingsTestbed/ApplicationI
nterface/build/classes/ orange.applicationlevel.io.serveurRMI
```

- start the web application :
    - <glassfish-directory>/bin/asadmin start-domain

```
Waiting for domain1 to start ..............................

Successfully started the domain : domain1

domain  Location: /root/glassfish3/glassfish/domains/domain1

Log                                                          File:
/root/glassfish3/glassfish/domains/domain1/logs/server.log

Admin Port: 4848

Command start-domain executed successfully.
```

- access to EPCGE at <serveraddress:port>/EPCGE-TESTBED
    - the url is http://{Base_URL}/EPCGE-TESTBED/. The welcome page looks like :



.

7.2.5 Stopping EPCGE

- stop glassfish : <glassfish-directory>/bin/asadmin stop-domain

```
Waiting for the domain to stop ......

Command stop-domain executed successfully.
```

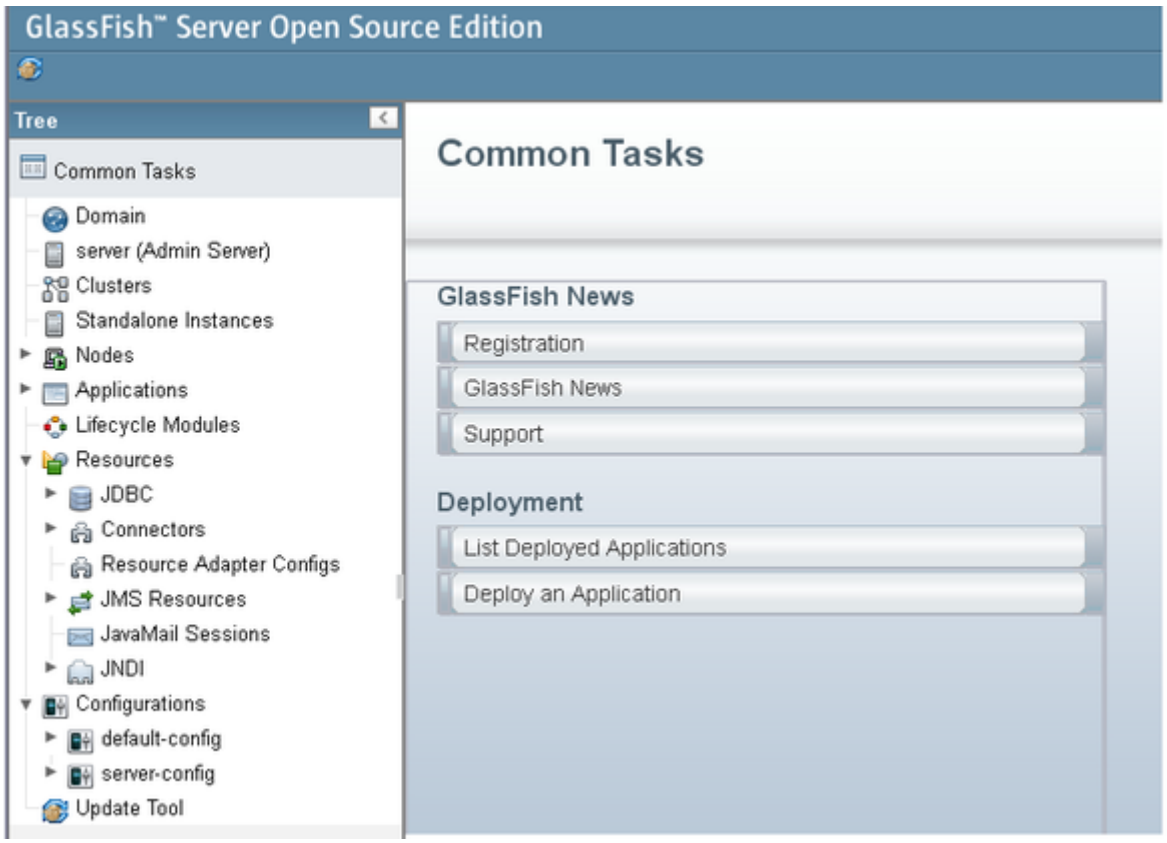
- kill the <pid> of *orange.applicationlevel.io.serveurRMI* and if needed the <pid> of *rmiregistry*
- stop Mysql Server if needed : stop mysql

```
mysql stop/waiting
```

7.3 System administration

- There is a property file *<glassfish-directory>/domains/domain1/applications/EPCGE-TESTBED/WEB-INF/classes/epcge.properties* which must be properly initialized :
    - Glassfish directory : *glassfish-install-dir=<glassfish-directory>*
    - Readpoint file directory : *readpoint-file-dir=<glassfish-directory>/glassfish/domains/domain1/applications/EPCGE-TESTBED/*
    - Once you've modified these properties, you must restart glassfish.
- If problems occur, you can have a quick look at *server.log* under <glassfish-directory>/glassfish3/glassfish/domains/domain1/logs/.
- You can access to the Administration console of glassfish http://<glassfish-directory>:4848/ to check that the web applications have been deployed.

Click on "List Deployed Application":



Check that applications are enabled:

Setting up epcge-resolver init script

- create and edit a file *epcge-resolver* under /etc/init.d/ :

```
#!/bin/sh -e
# File epcge-resolver
echo -n "Welcome to ..."
echo "epcge-resolver Service"
start()
{
        cd /root/wingsTestbed/ApplicationInterface/
        ./launch.sh&
        echo "Start epcge-resolver"
        echo
}
stop()
{
        pkill rmiregistry
        echo "Kill rmiregistry & Stop epcge-resolver"
        echo
}
```

```
restart()
{
        stop;
        sleep 1;
        start;
        echo "Restart epcge-resolver"
        echo
}
case $1 in
start)
        start;;
stop)
        stop;;
restart)
        restart;;
*)
        echo          "Usage:          /etc/init.d/epcge-resolver
{start|stop|restart}"
        echo
        exit 1
        ;;
esac
exit 0
```

- make the init script file executable : sudo chmod a+x /etc/init.d/epcge-resolver
- configure epcge-resolver for autostart on ubuntu boot : sudo update-rc.d epcge-resolver defaults
- to run the service : service epcge-resolver start
- to stop the service : service epcge-resolver stop

Setting up epcge-glassfish init script

- create and edit a file *epcge-glassfish* under /etc/init.d/ :

```
#!/bin/sh -e
# File epcge-glassfish
echo -n "Welcome to ..."
echo "epcge-glassfish Service"
start()
{
        cd /root/glassfish3/bin/
        ./asadmin start-domain
        echo "Start epcge-glassfish"
        echo
}
stop()
{
        cd /root/glassfish3/bin/
        ./asadmin stop-domain
        echo  "Stop epcge-glassfish"
}
restart()
{
        stop;
        sleep 1;
        start;
        echo "Restart epcge-glassfish"
        echo
}
```

```
case $1 in

start)

        start;;

stop)

        stop;;

restart)

        restart;;

*)

        echo          "Usage:          /etc/init.d/epcge-glassfish
{start|stop|restart}"

        echo

        exit 1

        ;;

esac

exit 0
```

- make the init script file executable : sudo chmod a+x /etc/init.d/epcge-glassfish
- configure epcge-resolver for autostart on ubuntu boot : sudo update-rc.d epcge-glassfish defaults
- to run the service : service epcge-glassfish start
- to stop the service : service epcge-glassfish stop

7.4 Sanity check procedures

The Sanity Check Procedure is the first step that a System Administrator will do to verify that the EPCGE 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.

1. Open a web browser and go to <serveraddress:port>:/EPCGE-TESTBED/webresources/application.wadl. For instance, access to http://{Base_URL}/EPCGE-TESTBED/webresources/application.wadl should display a WADL description file like this:

```
<application xmlns="http://research.sun.com/wadl/2006/10">
   <doc                    xmlns:jersey="http://jersey.java.net/"
jersey:generatedBy="Jersey: 1.8 06/24/2011 12:17 PM"/>
   <resources              base="http://130.206.81.29:8080/EPCGE-
TESTBED/webresources/">
       <resource path="entities.eventobjectevent">
           <method id="create" name="POST">
               <request>
                   <representation mediaType="application/xml"/>
                   <representation
mediaType="application/json"/>
               </request>
           </method>
           <method id="edit" name="PUT">
               <request>
                   <representation mediaType="application/xml"/>
                   <representation
mediaType="application/json"/>
               </request>
           </method>
           <method id="findAll" name="GET">
               <response>
                   <representation mediaType="application/xml"/>
                   <representation
mediaType="application/json"/>
               </response>
           </method>
           <resource path="{id}">
```

```
                <param
xmlns:xs="http://www.w3.org/2001/XMLSchema"             name="id"
style="template" type="xs:long"/>

                <method id="remove" name="DELETE"/>

                <method id="find" name="GET">

                    <response>

                        <representation
mediaType="application/xml"/>

                        <representation
mediaType="application/json"/>

                    </response>

                </method>

            </resource>

            <resource path="{from}/{to}">

                <param
xmlns:xs="http://www.w3.org/2001/XMLSchema"             name="to"
style="template" type="xs:int"/>

                <param
xmlns:xs="http://www.w3.org/2001/XMLSchema"          name="from"
style="template" type="xs:int"/>

                <method id="findRange" name="GET">

                    <response>

                        <representation
mediaType="application/xml"/>

                        <representation
mediaType="application/json"/>

                    </response>

                </method>

            </resource>

            <resource path="count">

                <method id="countREST" name="GET">

                    <response>
```

```
                        <representation mediaType="text/plain"/>
                    </response>
                </method>
            </resource>
        </resource>
        <resource path="entities.tracingEPC">
            <method id="traceDefaultEPC" name="GET">
                <response>
                    <representation mediaType="text/plain"/>
                </response>
            </method>
            <resource path="{codeEPC}">
                <param
xmlns:xs="http://www.w3.org/2001/XMLSchema"       name="codeEPC"
style="template" type="xs:string"/>
                <method id="traceEPC" name="GET">
                    <response>
                        <representation
mediaType="application/xml"/>
                        <representation
mediaType="application/json"/>
                    </response>
                </method>
            </resource>
        </resource>
    </resources>
</application>
```

7.4.1 End to End testing

The end-to-end testing is done by tracing the predefined EPC Code : *urn:epc:id:sgtin:4514280.191438.1375091273041*

1. Open a web browser and go to <servername:port>/EPCGE-TESTBED/webresources/entities.tracingEPC/urn:epc:id:sgtin:4514280.191438.1375091273041. For instance, the request http://{Base_URL}/EPCGE-TESTBED/webresources/entities.tracingEPC/urn:epc:id:sgtin:4514280.191438.1375091273041 should answer with an xml file like this:

```
<EPCISEvent>

<action>ADD</action><bizLoc>urn:unicaen:iotatester:xxxx:ssl:_not_specified_</bizLoc>

<bizStep>urn:unicaen:iotatester:bizstep:xxxx:tester</bizStep><bizTrans/>

<disposition>urn:unicaen:iotatester:disp:xxxx:tester</disposition> <EPCClass/>

<epcs>urn:epc:id:sgtin:4514280.191438.1375091273041</epcs>

<eventTime>2013-07-29T11:47:53.041+02:00</eventTime>

<insertedTime>2013-07-29T11:47:53.041+02:00</insertedTime>
<parentID/><quantity/>

<readPoint>urn:unicaen:iotatester:xxxx:ssl:18</readPoint><type>Object</type></EPCISEvent>


<EPCISEvent><action>ADD</action><bizLoc>urn:unicaen:iotatester:xxxx:ssl:_not_specified_</bizLoc>

<bizStep>urn:unicaen:iotatester:bizstep:xxxx:tester</bizStep><bizTrans/>

<disposition>urn:unicaen:iotatester:disp:xxxx:tester</disposition> <EPCClass/>

<epcs>urn:epc:id:sgtin:4514280.191438.1375091273041</epcs>

<eventTime>2013-07-29T11:47:54.204+02:00</eventTime>
```

```
<insertedTime>2013-07-29T11:47:54.204+02:00</insertedTime>
<parentID/><quantity/>

<readPoint>urn:unicaen:iotatester:xxxx:ssl:10</readPoint><type>O
bject</type></EPCISEvent>



<EPCISEvent><action>ADD</action><bizLoc>urn:unicaen:iotatester:x
xxx:ssl:_not_specified_</bizLoc>

<bizStep>urn:unicaen:iotatester:bizstep:xxxx:tester</bizStep><bi
zTrans/>

<disposition>urn:unicaen:iotatester:disp:xxxx:tester</dispositio
n> <EPCClass/>

<epcs>urn:epc:id:sgtin:4514280.191438.1375091273041</epcs>

<eventTime>2013-07-29T11:47:54.405+02:00</eventTime>

<insertedTime>2013-07-
29T11:47:54.405+02:00</insertedTime><parentID/><quantity/>

<readPoint>urn:unicaen:iotatester:xxxx:ssl:5</readPoint><type>Ob
ject</type></EPCISEvent>



</ePCISEvents>
```

7.4.2 List of Running Processes

- MySQL Server
- Glassfish application server

7.4.3 Network interfaces Up & Open

HTTP port 8080 must be reachable.

7.4.4 Databases

- MySQL

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

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

7.5.2 Remote Service Access

Through REST, port 8080, HTTP protocol.

HTTP application root is http://{Base_URL}/EPCGE-TESTBED/

7.5.3 Resource consumption

The whole EPC GE processes should not consume more than 500MB of RAM and 70% CPU. Values beyond these thresholds should be considered as abnormal.

7.5.4 I/O flows

Port 8080,1099 HTTP

8 Gateway Protocol Adapter - MRCoap - Installation and Administration Guide

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

8.1 Fundamentals

The MR CoAP Protocol Adapter is an implementation of the Protocol Adapter GE which allows the IoT Gateway to communicate with Moterunner devices over 6LoWPAN and CoAP. This is its Installation and Administration. The MR CoAP Adapter is provided as a NGSI standalone installation.

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.

8.2 Prerequisites

8.2.1 Hardware and software requirements

The minimum HW requirements are:

- 16 MB flash for file system
- 64 MB RAM (can work on 32 MB as well)
- gt 300 Mhz CPU
- One or more Iris motes

The minimum software requirements are:

- Java 1.7

8.2.2 Wireless connection to Moterunner devices

The moterunner devices will connect directly to the protocol adapter over a radio interface IEEE 802.15.4.

8.2.3 Installation

You will need the following software packages:

- Moterunner Platform Version Beta11.0 or higher (with 6LoWPAN assembly)
- Several Java libraries (see below)

The moterunner platform is available from http://www.zurich.ibm.com/moterunner/. It is availabe for four platform (Linux 32-bit, Linux 64 bit, MAC OSX, Windows). Please click on Downloads, accept the license agreement and then download the moterunner version for your platform. On Linux (the preferred and best supported platform) unzip the moterunner-xx.zip with your favourite zip tool (e. g. unzip) and start the setup file (./setup)

Next please install Java. The software has been tested with Java 7, but should also work with later version. On Ubuntu you can do this via:

```
sudo apt-get install openjdk-7-jdk openjdk-7-source openjdk-7-demo openjdk-7-doc openjdk-7-jre-headless openjdk-7-jre-lib
```

On a redhat based system you can use:

```
su -c "yum install java-1.7.0-openjdk"
```

In case you want to install the JDK from scratch please download from:

```
http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
```

Furthermore the software has the following external dependencies:

- Californium & Element connector

Source: ETH Zurich
Download:
https://github.com/mkovatsc/maven/tree/master/snapshots/ch/ethz/inf/vs/californium/0.18.2-SNAPSHOT
https://github.com/mkovatsc/maven/tree/master/snapshots
Tested version: 0.18.2 (Californium), 0.1-20131126.100751 (element connector)

- Apache Commons CLI

Source: Apache
Download: http://commons.apache.org/proper/commons-cli/
Tested Version: 1.2

- Apache HTTP Components

Source: Apache
Download: http://hc.apache.org/
Tested Version: 4.3.1

Download the external dependencies into a specific libraries directory.

8.2.4 Moterunner Platform

First you need to ensure that the moterunner platform is running and the corresponding 6LoWPAN tunnel has been established. In the following we will describe a basic setup for an one mote scenarion. For more sophisticated setup and deployment scenarios please refert to the moterunner documentation.

The virtual tunnel has to be started, which will create a IPv6 prefix plus adding any mote added to the network with their unique ID as an IPv6 interface identifier, this works currently only for Linux/OSX.

```
export PATH=$HOME/moterunner/linux/bin:$PATH
```

Creating the tunnel with the network prefix: fc00:0db8:0005:0000. Suggested to be in a separated Terminal for a better overview (ctrl+shift+t)..

```
sudo ~/moterunner/examples/6lowpan/tunnel/tunnel --sonoran

sudo                                                    bash
~/moterunner/examples/6lowpan/tunnel/route_setup_linux.sh

cd $HOME/moterunner/examples/6lowpan/apps/coap-sensor/
```

First the wireless mote has to be loaded correctly with the assemblies, therefore put the mote on the programming board (mib520). Please make sure, that the wireless mote is set off, when it's on the programming board. Before you can run this tutorial make sure you flashed the Iris mote with the Mote Runner firmware

```
/moterunner/firmware/iris.srec.
```

Mount the Iris mote onto the MIB520 programming board but do **not** yet attach it to the PC. The next command monitors the serial ports of your PC and will tell you the port names of the attached Programming board. Now run the next command and after that attach the USB cable of the MIB520 to the PC.

```
lip-enumerate -w --iris -t30s
```

Start the Moterunner and create the mote, which is connected with the programming board. Mrsh

```
mote-create –p {MIB520_MR_PORT}
```

The command before will create a mote, usually it shows the unique id after it has been successfully created. When it is not shown, then press the hard-reset button on the mib520 board, and type

```
network-list
```

until the unique id is shown, in the column for “Uniqueid”, instead of the {MIB520_MR_PORT} and the State is “on”. Retry the hard reset, until the unique id is shown, continue after it is shown. If the 6lowpan application is loaded already, the next step can be bypassed.

```
a0 moma-load 6lowpan
```

Now load the sensor application, surely if it is not loaded already, onto the mote with the following command

```
a0 moma-load sensor-1.0
```

If the sensor, or any other, application has to be updated, delete the sensor application (moma-delete) first and then load it again on it.

If everything went well, type the next command, to exit moterunner.

```
quit
```

Wait until it’s finished. Unplug the mote and plug the gateway mote on the programming board

Now you need to setup the gateway mote:

```
mote-create –p {MIB520_MR_PORT}
```

The command before will create a mote, usually it shows the unique id after it has been successfully created. When it is not shown, then press the hard-reset button on the mib520 board, and type

```
```

```
network-list
```

until the unique id is shown, instead of the {MIB520_MR_PORT}. Retry the hard reset, until the unique id is shown, continue after it is shown. The next commands will initiate v6lowpan script for the gateway setup.

```
source ../../lib/js/v6lowpan.js

a0 v6lowpan-setup
```

Wait until its finished put the sensor board MTS400 onto the wireless sensor mote on which the sensor application is loaded on and make it on. The console should view that the sensor mote connected to the gateway mote, by showing the unique id number, when the following command is typed.

```
v6lowpan-connect
```

You are now ready to run the protocol adapter.

8.2.5 Adapter

The MR CoAP protocol adapter is a standalone java application. Before startup you need to configure the location of moterunner and the external dependencies in either run.sh or run.bat (Linux / Windows). On Windows run.bat looks as follows:

```
REM directory with external dependencies (californium 18,
apache-commons-cli, apache-webelements)

set libs="c:\temp\*"


REM Moterunner base directory

set moterunner=c:\moterunner


REM Settings

set ngsi9=htpp://localhost:80/post.php

set ngsi10=htpp://localhost:80/post.php

set mrnetwork=fc00:db8:5::
```

```
set mrgateway=0000:0000:0000:0001

set webserverport=8001

REM ------ END OF CONFIGURATION ------
```

As one can see you need to add the ngsi9 and ngsi10 endpoints, the network of moterunner and the gateway mote (which is usually :::1).

then start the system with

./run.sh

or by executing run.bat on windows.

8.3 System Administration

In order to stop or start a process, these instructions and scripts should be used:

```
ps  -ef | grep java

kill -9 <pid java process>
```

on Windows please use the task manager.

then restart with

```
./run.sh
```

or

```
run.bat
```

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.

8.4.1 End to End testing

If the protocol adapter is already active it is possible to access to the Protocol Adapter with the following url:

```
http://protoadapter_ip:8080/fiMRCoAPAdapter
```

You should get back this title:

```
The Moterunner Protocol Adapter is up and running!
```

At this URL: http://protoadapter_ip:8080/fiMRCoAPAdapter/ the servlet uploaded shows a list of discovered moterunner devices and when you check one of these the related details are shown. If you refresh the browser it is possible to see the changes of status of the devices.

8.4.2 List of Running Processes

In a correct setup environment you should have the following processes running:

- sonoran
- java running mrcoapadapter.jar

8.4.3 Network interfaces Up & Open

There is a permanently opened interface for incoming connections created by Tomcat on its port, 8080 by default.

8.4.4 Databases

N/A

8.5 Diagnosis Procedures

8.5.1 Resource availability

During our tests, the Tomcat installation with all three running components needed approximately:

- 300 MB of hard disk space, including the necessary IBM moterunner framework
- 400-500 MB of RAM

On a Core i5 2540M CPU, run under Windows 7 64 Bit, the CPU usage for the Tomcat process has not exceeded ~20%.

8.5.2 Remote Service Access

The communication to other GEs is carried out via HTTP. The mote registers itself at a NGSI9 network participant and the measured values are transmitted to a NGSI10 compliant preconfigured network participant.

However, the MRCoAP adapter is not the component where those values should be accessible.

Ergo, the adapter only accesses other services via HTTP but it's not in its scope to offer an interface to be contacted by other and does not act as a remote service.

8.5.3 Resource consumption

The devices do not allocate resources dynamically. There was no huge peaks or spikes in consumption. The Tomcat server startup is however a procedure taking a few minutes and using the CPU at a level of approximately 20%

8.5.4 I/O flows

There is a proprietary radio connection between the motes and its gateway.

For the northbound interaction, the incoming communication is carried out via HTTP via the configured Tomcat port, which is 8080 by default.

The adapter queries the sensor motes via the gateway's ip configured ip address. The port for these outgoing UDP requests is always 1024.

As outgoing requests the measured sensor values are sent to a NGSI9/NGSI10 server via HTTP. The server's URL and port are configured as startup parameters.

9 Gateway Data Handling - Esper4FastData - Installation and Administration Guide

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

9.1 Introduction

Welcome to the Espr4FastData Installation and Administration Guide, which covers the installation of Espr4FastData version 3.3.3

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

The Espr4FastData software provides Complex Event Processing (CEP) features and is compliant with the OMA-NGSI standard. It exposes a REST API that makes remote administration easier. 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 easily test the main available features of the enabler.

9.2 System installation

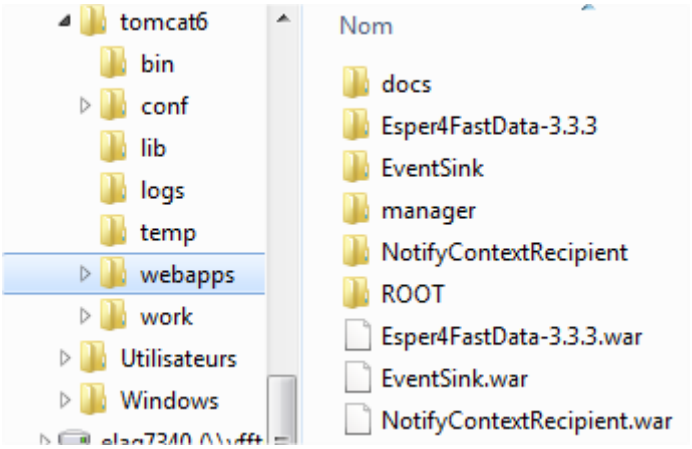
9.2.1 Generic procedure

Espr4FastData runs in a servlet container environment, like Tomcat. It does not depend on a particular operating system flavor. The following steps cover the installation on localhost.

Steps

- Get Java version 6 if needed.
- Get Apache Tomcat servlet container version 6 if needed
- Get the Espr4FastData-3.3.3.zip latest build. It contains war application files, the "Espr4FastData-soapui-project-3.3.3.xml" test file, and the "Espr4FastDataSanityCheck-3.3.3.jar" to perform sanity check.
- Unpack the Espr4FastData-3.3.3.zip file.
- Sanity check and SoapUI tests are configured to run on port 80: configure Tomcat to operate on TCP port 80 (by editing the tomcat server.xml file) then run it.
- Copy the extracted .war files to the {Tomcat6Root}/webapps directory: the event sink "war" application simulates an event recipient, whereas the notify context recipient "war" application simulates a NGSI-10 event subscriber. Both are needed for testing and tutorials.

- The "war" files should be automatically deployed in new directories, as shown below on a Windows operating system (but it could be Linux or any Java-compatible OS as well):



- Check wadl access : http://localhost/Espr4FastData-3.3.3/application.wadl

9.2.2 CloudEdge procedure using CloudEdgeVMInstaller

The installation of Espr4FastData using "CloudEdgeVMInstaller" on the Technicolor Cloud Edge should be done from the Unix/Linux command line. In any case, one needs the "curl" command to be available.

9.2.2.1 *Overview*

At first, please note that this bundle features:

- "CloudEdgeVMInstaller" bash software that makes the installation and deployment of Technicolor Cloud Edge box images easier
- an Espr4FastData-3.3.3 VM image
- a ZPA-1.2 GE image VM images

The provided software is structured very simply, as follows:

```
./CloudEdgeVMInstaller--
                        |
                        |--- /sh ---
                        |              |--- ce_install_images.sh
(the vm installer)
                        |           |
```

```
                          |                |--- cereset.sh (factory
reset then reboot the box)
                          |            |
                          |              |--- config.sh (config file
that must be edited)
                          |
                          |--- /tmp (hold temporary file)
                          |
                          |--- /vm_images (contains VM images you
want deploy)
```

9.2.2.2 *CloudEdgeVMInstaller installation and preliminary configuration*

Untar the package using

```
tar -zxvf CloudEdgeVMInstaller.tar.gz
```

in the directory of your choice.

Then you must set the CEVI_ROOT environment variable to the root directory of the CloudEdgeVMInstaller software.

Example:

```
export CEVI_ROOT="/usr/local/CloudEdgeVMInstaller"
```

After that, configure a local FTP server on your host and allow "anonymous" access.

Finally edit the CloudEdgeVMInstaller/sh/config.sh file in order to set its parameters according to your own configuration.

Example file:

```
#!/bin/bash


# Example


export localFtpDirectory="/srv/ftp" # location of your local
"anonymous" ftp directory
```

```
export ftpAddress="ftp://161.105.138.61" # Address of your ftp server as seen from outside your host.


export    cloudEdgeRootAddress="http://161.105.138.54:8080"    # Address of the cloud edge box, as seen from inside your host.

export ceUser="smithj" # User of the cloud edge box. Usually it's "smithj" so you should not change it.

export cePassword="secret" # Password of the cloud edge box. Usually it's "secret" so you should not change it.
```

9.2.2.3 *Virtual Machines (VMs) images installation*

If you need to factory-reset your box (in case you want to delete previous remaining VMs) then run the "cereset.sh" shell script. It will reset then reboot the box. This operation is not harmful to the VM managing software. So you can do this safely for the purpose of deleting existing VMs.

All VM file images are stored in your vm_images directory and MUST comply with the following standard:

- rootfs-ImageName.tar.gz for the image name.
- cloudedge-ImageName-vm.tar for the accompanying manifest file.

Many VMs images are allowed in the vm_images directory at the same time.

To install a VM, run the "ce_install_images.sh" shell script with the image name as an argument. It is normal to wait between automated installation steps. The programmed pauses allow the box to "breath" a little bit. They avoid unwanted box freezes caused by CPU-memory overload. Example: ce_install_images.sh Espr4FastData-3.3.3

9.3 System administration

The Espr4FastData engine uses the Apache log4j logging system. Log4j is configurable through its log4j.properties configuration file, located in `{Tomcat6Root}\Espr4FastData\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] (Espr4FastData.java:pause:795)
CEP engine pause

2013-04-26 14:46:57,020 [DEBUG] (Espr4FastData.java:pause:795)
CEP engine resume

2013-04-26 14:46:57,036 [DEBUG] (Espr4FastData.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] (Espr4FastData.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.
```

9.4 Sanity check procedures

The described sanity check software is not compliant with the Technicolor Cloud Edge box.

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. The sanity check is achieved with the `Espr4FastDataSanityCheck-3.3.3.jar` executable, which MUST be installed on the same host (but anywhere in the directory tree) as the "war" file. In the specific case of the FiWare testbed, it is pre-installed by default in the `/root` directory. In any case, this "jar" file is available in the downloaded `Espr4FastData-3.3.3.zip` archive.

Run Espr4FastDataSanityCheck-3.3.3.jar thanks to the following command:

```
java -jar Espr4FastDataSanityCheck-3.3.3.jar
```

The answer should be:

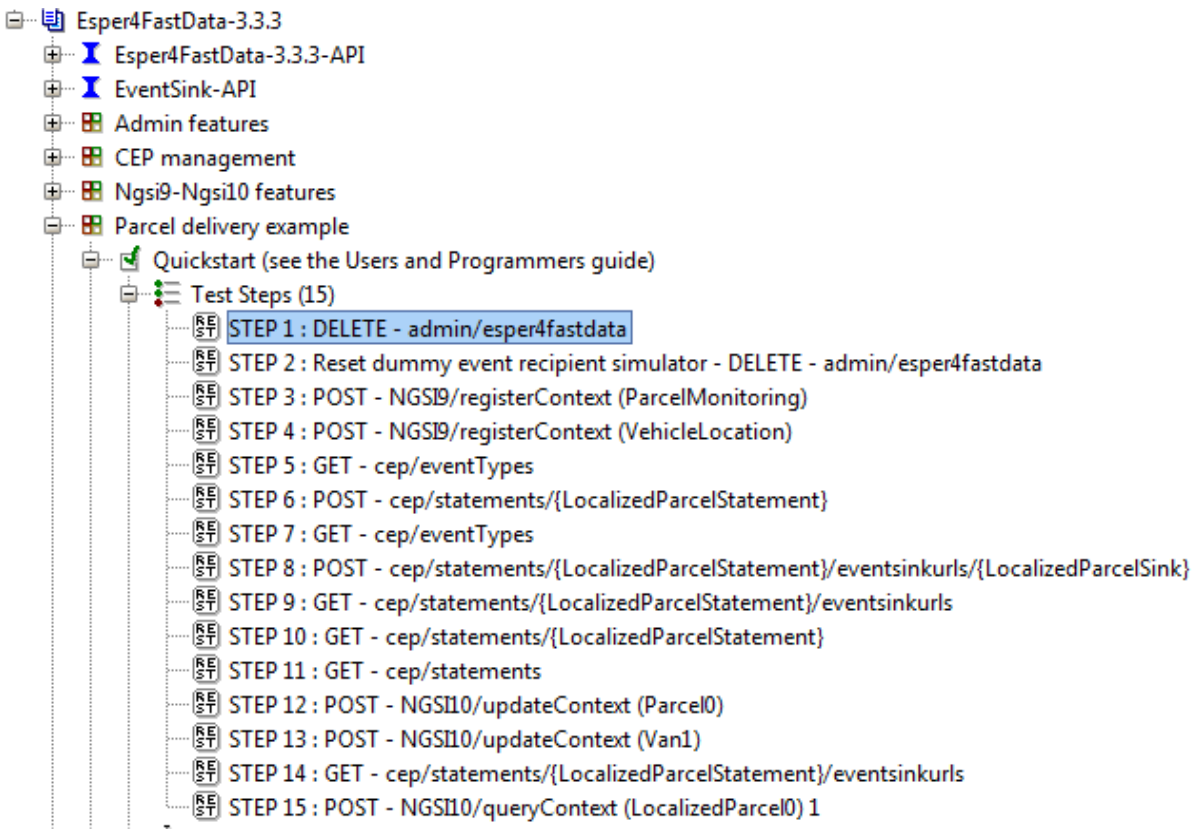
```
SANITY CHECK TEST HAS BEEN SUCCESSFULLY PASSED !
```

9.4.1 End to End testing

This end to end testing procedure is compliant with both the generic installation and the Technicolor Cloud Edge box installation.

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

1. Depending from where you are running the Espr4FastData GEi, ensure that the proxy menu is properly configured (disabled or active): File->Preferences->Proxy Settings.
2. Open the `Espr4FastData-soapui-project-3.3.3.xml` file which is available in the downloaded `Espr4FastData-3.3.3.zip` archive.
3. Open a web browser at `http://{ServerRoot}/Espr4FastData-3.3.3/application.wadl`. This should display a WADL description file like this.
4. Open the `Parcel delivery example` (within the `Quickstart` TestSuite) and run it. This tests the most important features in Espr4FastData. A snapshot of the TestSuite is illustrated below:



The result of the last REST query (`queryContext` on LocalizedParcel0) MUST return the following XML, otherwise the end-to-end testing should be considered as failed:

```
<queryContextResponse>
   <contextResponseList>
      <contextElementResponse>
         <contextElement>
            <entityId isPattern="false" type="LocalizedParcel">
               <id>LocalizedParcel0</id>
            </entityId>
            <contextAttributeList>
               <contextAttribute>
                  <name>vehicleId</name>
                  <type>xs:string</type>
                  <contextValue>Van1</contextValue>
```

```
                </contextAttribute>
                <contextAttribute>
                   <name>deliveryStep</name>
                   <type>xs:string</type>
<contextValue>DeliveryInProgress</contextValue>
                </contextAttribute>
                <contextAttribute>
                   <name>latitude</name>
                   <type>xs:double</type>
                   <contextValue>48.848103</contextValue>
                </contextAttribute>
                <contextAttribute>
                   <name>longitude</name>
                   <type>xs:double</type>
                   <contextValue>2.393302</contextValue>
                </contextAttribute>
             </contextAttributeList>
             <domainMetadata>
                <contextMetadata>
                   <name>timestamp</name>
                   <type>xs:string</type>
                   <value>20130301T140634+0100</value>
                </contextMetadata>
             </domainMetadata>
          </contextElement>
          <statusCode>
             <code>200</code>
             <reasonPhrase>OK</reasonPhrase>
```

```
          </statusCode>
       </contextElementResponse>
   </contextResponseList>
   <errorCode>
       <code>200</code>
       <reasonPhrase>OK</reasonPhrase>
   </errorCode>
</queryContextResponse>
```

9.4.2 List of Running Processes

Tomcat must be running.

9.4.3 Network interfaces Up & Open

HTTP port 80 must be reachable.

9.4.4 Databases

No database server is required.

9.5 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.

9.5.1 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

9.5.2 Remote Service Access

Through REST, port 80, HTTP protocol.

HTTP application root is http://{ServerName}/Espr4FastData-3.3.3

9.5.3 Resource consumption

150 MBytes of RAM

9.5.4 I/O flows

Port 80, HTTP

10 Template Handler - Template Handler - Installation and Administration Guide

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

10.1 Fundamentals

Template Handler is built on two different technologies and combines their strengths: NGSI and BPMN business modeling. Their respective specifications can be found below:

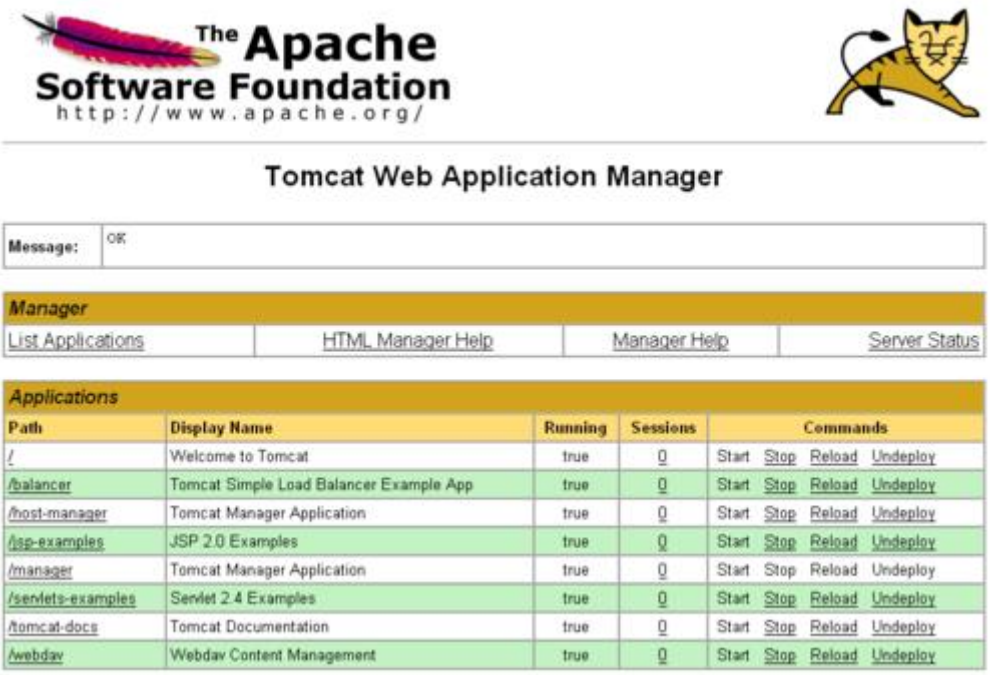
- NGSI Context Management v1.0
- BPMN v2.0
- NGSI RestFUL Binding for the mapping between HTTP and NGSI, v1.0

Both technologies strongly depend on XML files, for questions concerning these have a look into the

- XML Specification by W3C

10.2 Prerequisites

- You will need Java at least in version 6. The download and installation help are provided by Oracle.
    - To verify the correctness of the installation open a console and type "java". The possible command line options should appear.
- Tomcat at least in version 6. The distribution depends on the platform you are using so please download the version for your environment and have a look at the setup guide
    - To verify the installation of Tomcat browse to http://localhost:8080/manager/html. You should see a list of running applications like below



- You need a tool to modify existing zip files in order to configure your Template Handler instance. For Linux and Windows a 7-zip can be downloaded. The Windows installation is done by a simple installation wizard.
    - To verify the installation open "7-zip File Manager". A GUI should appear if you have a graphical desktop system.

10.3 Downloading Template Handler

You need the following files to install Template Handler:

- Template Handler Package

After unzipping, there are two subfolders:

"/deploy" contains three .war files needed in the next steps

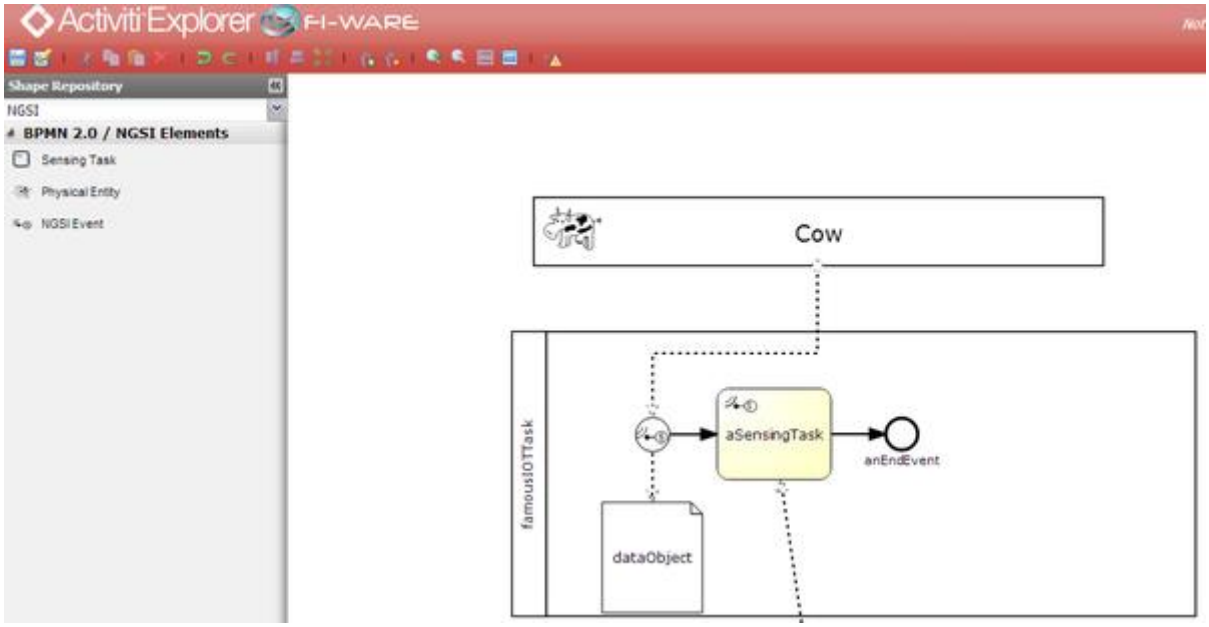
"/tests" contains a bunch of .xml files, one of them is used below to verify the installation

10.4 About FiWare Template Handler

FiWare Template Handler consists of three different components:

10.4.1 Modeler

The modeler allows users to model BPMN processes and enriches BPMN with the NGSI specific extensions.

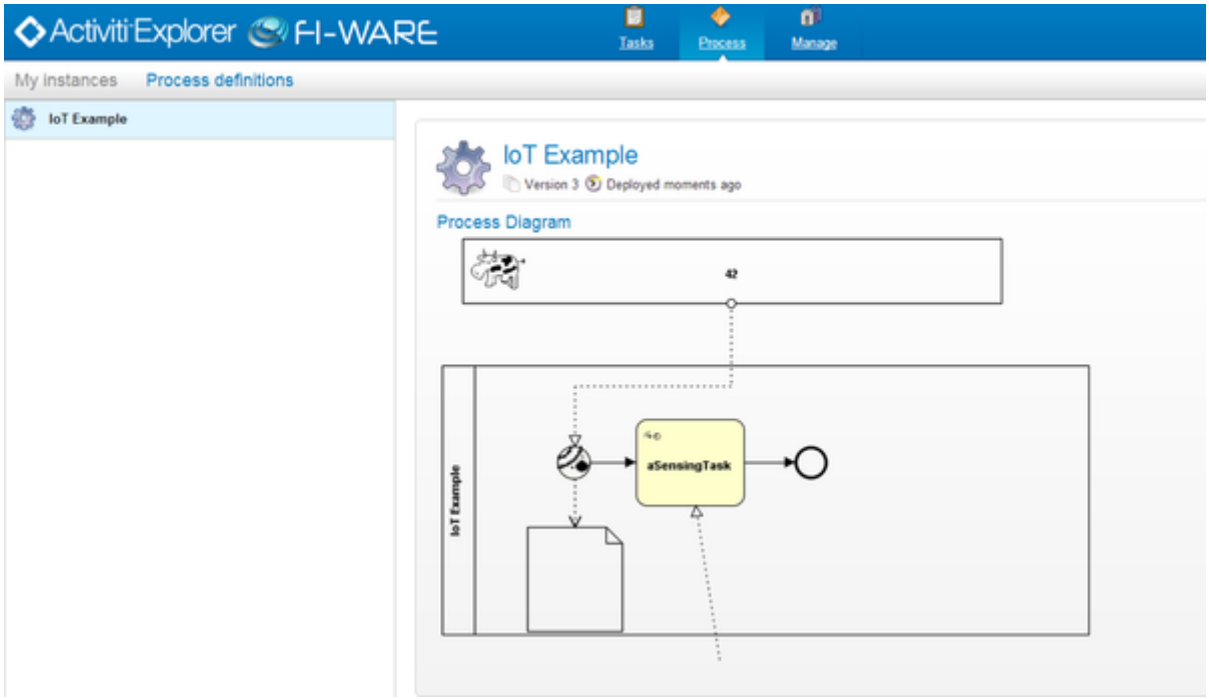


10.4.2 NGSI server

The NGSI server is a limited implementation of the centrale instance described in the NGSI specification. It follows the NGSI10 RESTful binding. The implementation is capable to process context queries, context updates and handle subscriptions.

10.4.3 Execution Environment

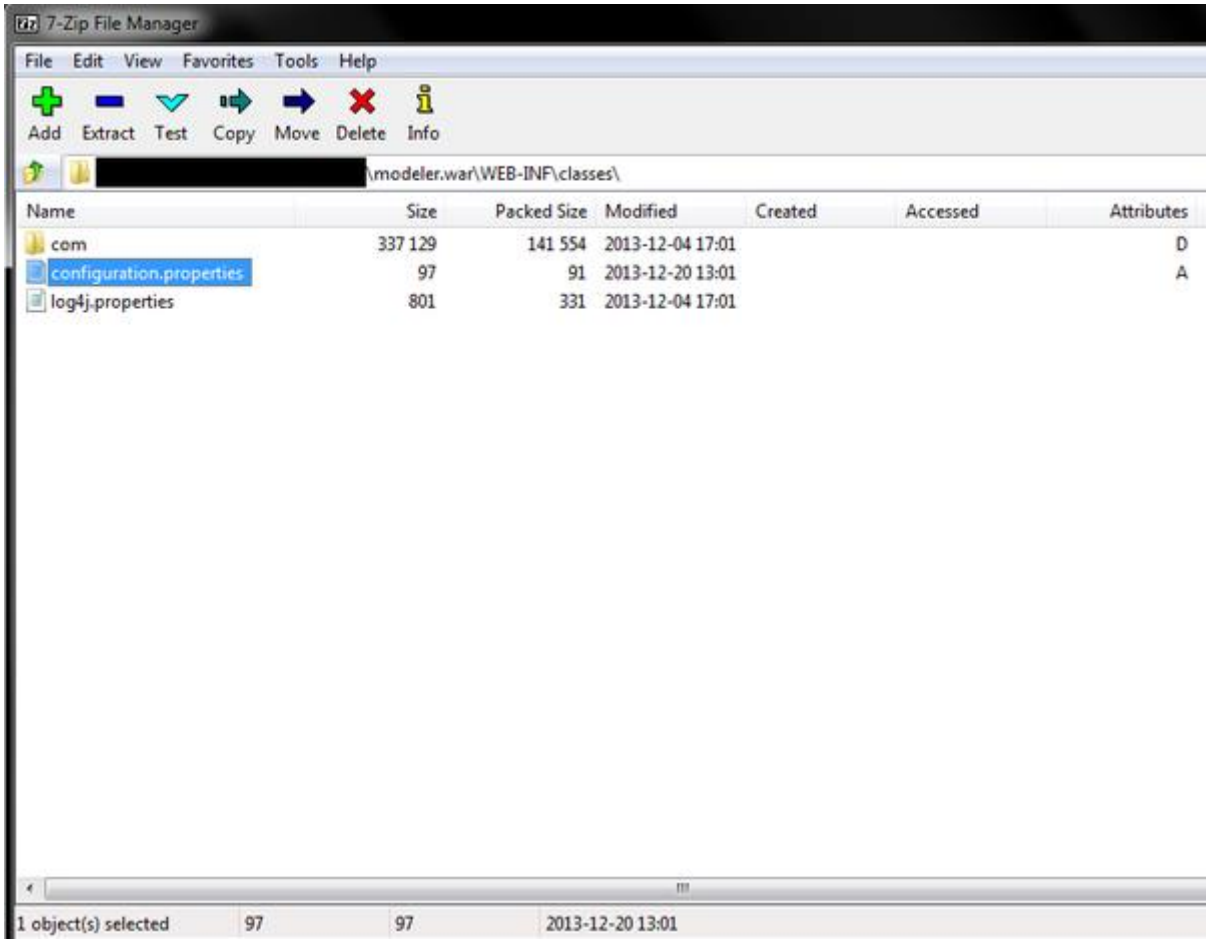
Activiti is an environment for BPMN process execution. It has been extended for the execution of NGSI sensing tasks to request values from the NGSI server. Additionally, start events may trigger business processes on receipt of NGSI notifications.



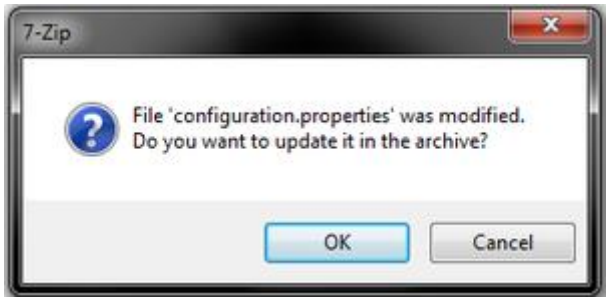
10.5 Deploying FiWare Template Handler

10.5.1 Modeler

- Open 7-zip. A file system navigator will appear as shown below. Browse to the location where you extracted the files and then under "/deploy/modeler.war" and double-click on it. Then double-click through the directory structure to WEB-INF/classes/configuration.properties.



- Double-clicking the file will open a text editor.
- Set for "fileSystemRootDirectory" an existing path on your system, e.g. "c:/repository". The created models will be saved in this directory.
    - Under Unix please make sure the user running the tomcat process has the permissions set to write in the folder
- Set for "host" the host and port under which tomcat is reachable. It must be accessible from the outside, i.e. use the actual ip address within your network and not localhost.
- Save the file in the editor and close the editor. A 7-zip message will appear asking whether the modifications should be applied to the orginal file, as shown below. Answer with ok.



- Deploy modeler.war to the server.

- Browse http://localhost:8080/manager/html
- Go to "Deploy WAR file"
- Click "Choose File"
- A file selection dialog opens
- Choose modeler.war
- Confirm
- Click on "Deploy"

10.5.2 NGSI server

- Deploy NGSI10.war to the server as described with the modeler

10.5.3 Execution Environment

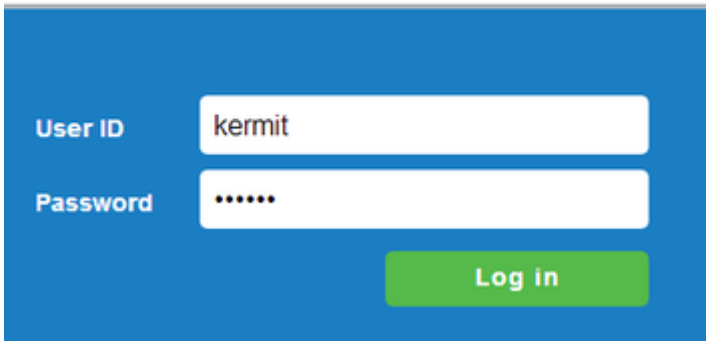
- Open activiti-webapp-explorer2-5.9.war with 7-zip as described for the modeler and modify activiti-webapp-explorer2-5.9\WEB-INF\applicationContext.xml. Write for "contextSubscriptionReferenceURL" the path where the deployment is accessible later, e.g. http://192.168.178.42:8080/activiti-explorer2/. It should be reachable from other participants, so do not use 'localhost' as host. The trailing slash is necessary.
- Set for ngsiServerURL the NGSI server's deployment path, e.g. http://localhost:8080/NGSI10/. If you are not sure go again to http://localhost:8080/tomcat/manager/html and have a look in the appearing list.

10.6 Sanity check procedures

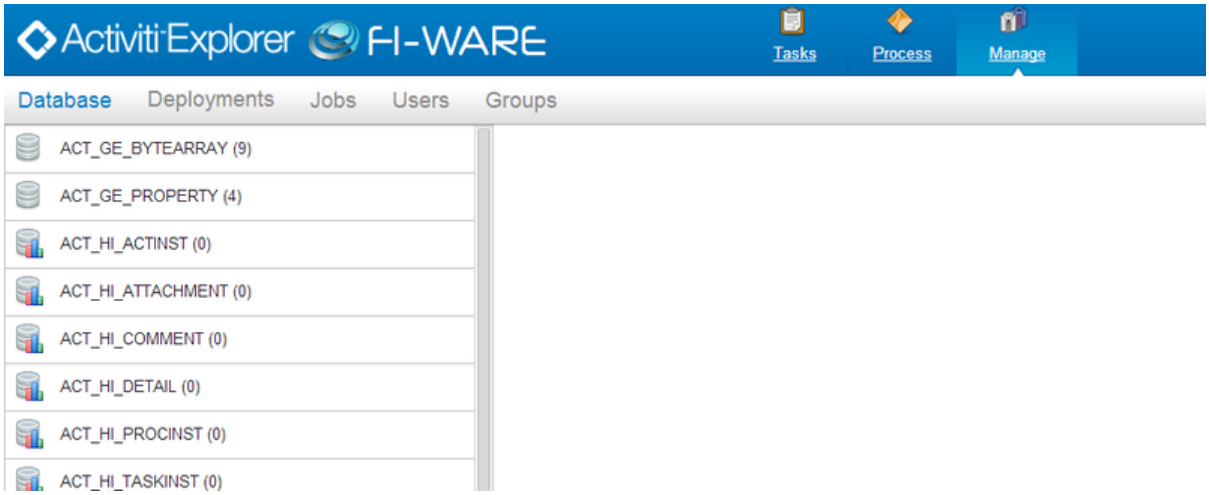
10.6.1 End to End testing

- **Modeler:** To verify your installation visit http://localhost:8080/modeler/p/explorer in the browser and test whether the start window appears, see below.

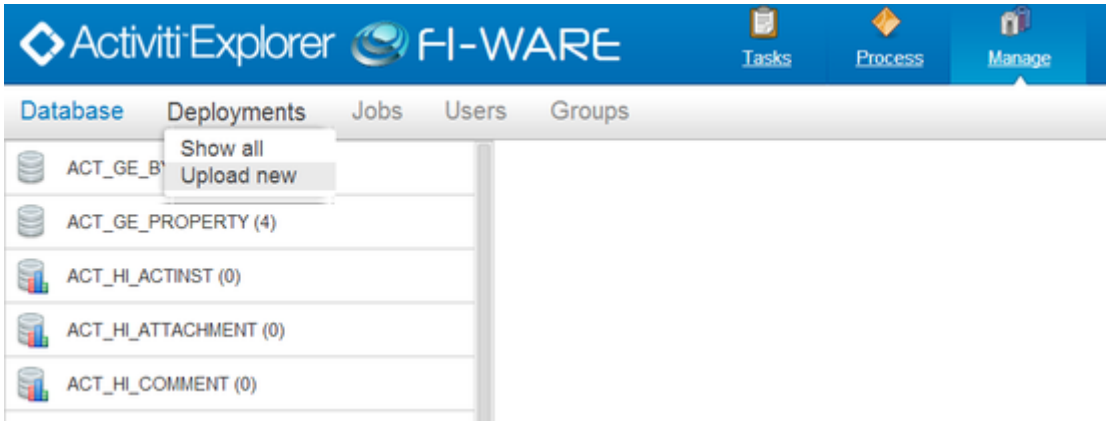
- **NGSI server:** To verify your installation visit http://localhost:8080/NGSI10/queryContext The following line should appear: "405 Method Not Allowed. Allowed: POST."
- **Execution environment:** To verify the installation, browse to http://localhost:8080/activiti-webapp-explorer2-5.9 and see whether you can log in with “kermit” / “kermit”, see below.



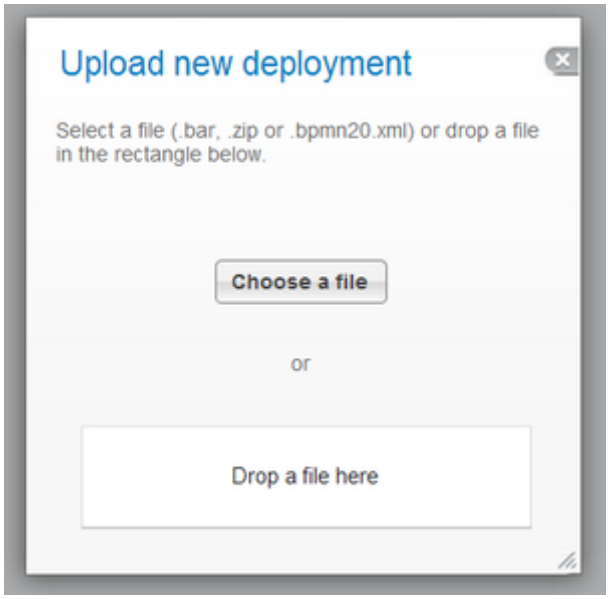
- To have a simple test whether the **components work together**, browse to activiti and add a new deployment. Select ngsiStartProcessDuration35Interval10.bpmn20.xml from the "test" folder and execute it as seen in the pictures below.
- On the start page click "Manage".



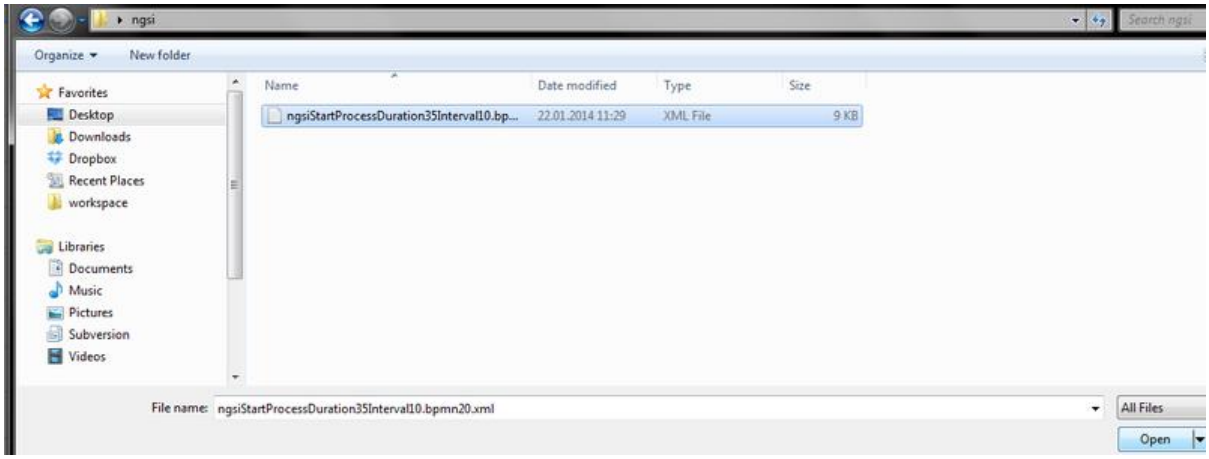
- Click "Deployments" and afterwards "Upload new".



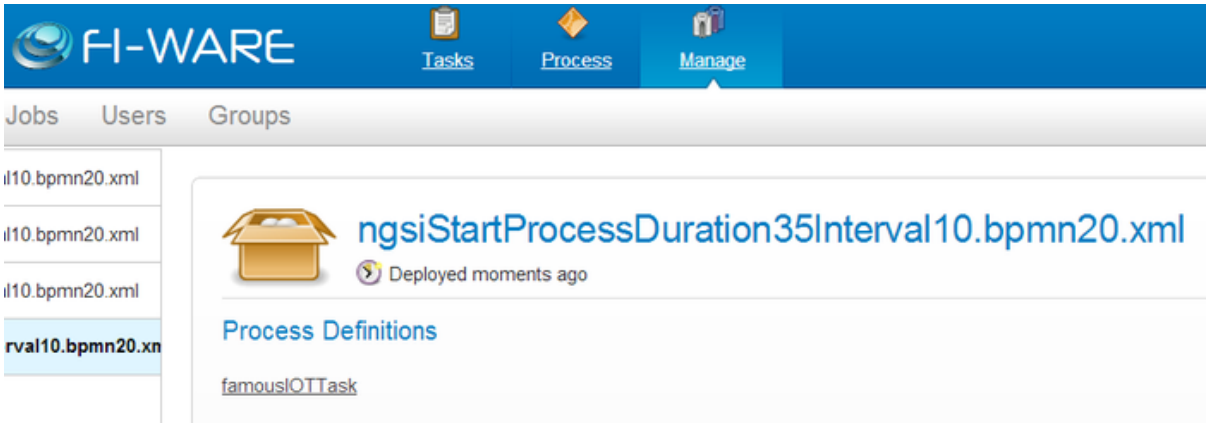
- Click "Choose a file".



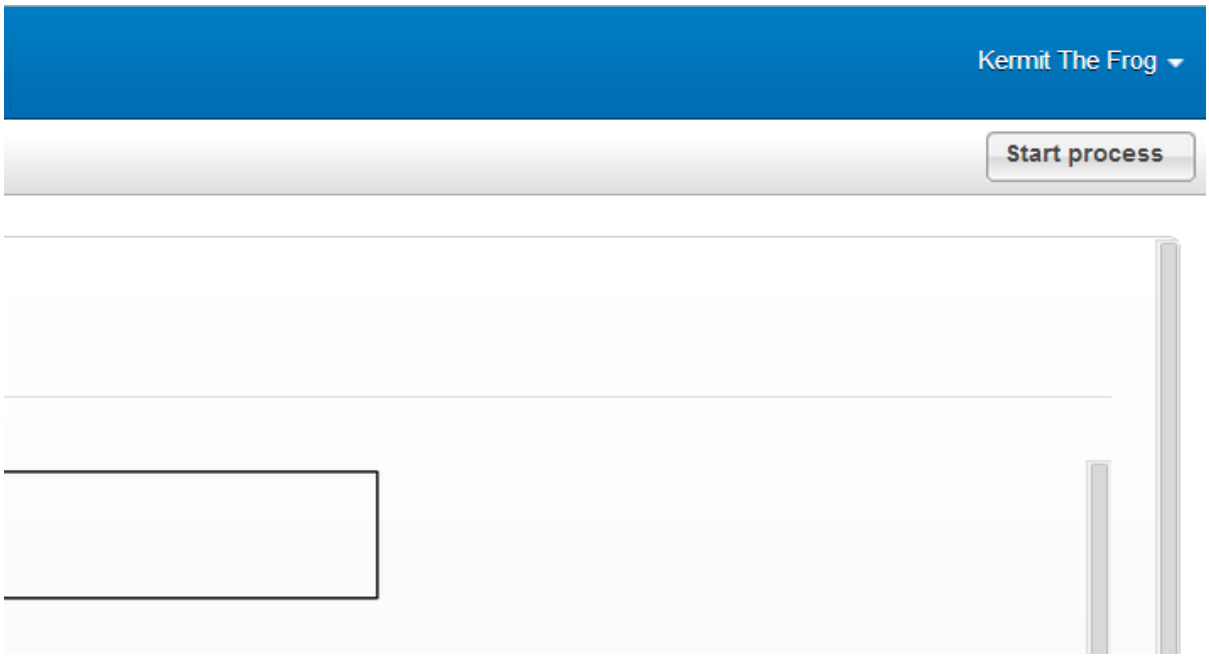
- Select ngsiStartProcessDuration35Interval10.bpmn20.xml.



- Click on the task's name under "Process Definitions".



- Click on "Start Process".



- The business process engine now requests a subscription from the NGSI server. It desires a notification about a context attribute's value every ten seconds. Due to the

NGSI start event within the process definition, this notification triggers the business process to execute.

- To see whether everything works, check the tomcat logs. They can be found in [tomcat's install dir}/logs/catalina-{yyyy-mm-dd}.log, with {yyyy-mm-dd} being the current date.
- There is a sensing task producing some output in this log starting with “Return value for execution with id”. This should appear three times and stop after that, since the model defines the notification length to be 35 seconds.

10.6.2 List of Running Processes

Since Tomcat is a single running server component, only one process called "java" will appear in the list.

10.6.3 Network interfaces Up & Open

TCP port 8080 is used by the applications.

10.6.4 Databases

N/A

10.7 Diagnosis Procedures

10.7.1 Resource availability

During our tests, the Tomcat installation with all three running components needed approximately:

- 1.5 GB of hard disk space
- 500-800 MB of RAM

On a Core i5 2540M CPU, run under Windows 7 64 Bit, the CPU usage for the Tomcat process has not exceeded ~20%.

10.7.2 Remote Service Access

At the moment there are no connections to other enablers. If there were enabler using the services provided by Template Handler, this would be HTTP communication. The incoming part would be the standard end user port, i.e. 8080 by default. The outgoing communication depends on the potential other enablers.

10.7.3 Resource consumption

During our tests the RAM and CPU consumption have always been below the maximum values aforementioned. Even if we processed a large process model, the RAM usage was always steady between 500-800 MB of RAM. The CPU usage - as expected - had their peaks when Tomcart started up (<20%), but after Tomcat was running, it never really spiked again, regardless of the process model being processed.

10.7.4 I/O flows

The interaction between the user and Template Handler as well as the communication between the components is carried out via HTTP.

All user interaction is done via the configured Tomcat port, which is 8080 by default.

In addition, the execution environment component uses the same port to receive notifications from the NGSI server component.