



EUROPEAN
COMMISSION

Community Research



Private Public Partnership Project (PPP)
Large-scale Integrated Project (IP)



fi-ware

D.3.3.2: FI-WARE Installation and Administration Guide

Project acronym: FI-WARE

Project full title: Future Internet Core Platform

Contract No.: 285248

Strategic Objective: FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform

Project Document Number: ICT-2011-FI-285248-WP3-D.3.3.2

Project Document Date: 2013-04-22

Deliverable Type and Security: Public

Author: FI-WARE Consortium

Contributors: FI-WARE Consortium

1.1 Executive Summary

This document describes the installation and administration process of each Generic Enabler developed within in the "Application and Services Ecosystem and Delivery Framework" 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.

EAB providing the GE "ServiceComposition" as software asset left the related work package and can't provide further support for the software and software related deliverables.

THALES contributed an optional component to the Mediator GE developed by TI. This optional component provides low-level (not end user) facilities to the Mediator GE for automatically solving certain dynamic service mediation issues. Only binaries are provided at this stage for the THALES component since its actual usage by the Mediator GE should be totally transparent to the end user or developer and not to be consumed by a third party (no Installation and Administration guide for the THALES component are reflected in this Installation and Administration guide). THALES may however add some information to the Installation and Administration guide in further releases, depending on the actual level of integration achieved in the final versions of the Mediator GE.

1.2 About This Document

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

1.3 Intended Audience

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

1.4 Chapter Context

The Generic Enablers for the Apps Chapter together can be used to build the core infrastructure for enabling a sustainable ecosystem of applications and services of future internet application domains, which foster innovation as well as cross-fertilization. In particular the Apps Generic Enablers supports unified description and publishing of services, offering of services in a store, matching demand and offering via marketplace capabilities, creating composed value added services and service networks, and monetization and revenue sharing, all in a complementary and harmonized business framework.

The concept of the Generic Enabler implies that there can be several possible implementations. There are various degrees of flexibility in the non-functional properties or functional profile of the Generic Enabler description. For example the Mediator GE has 2 different implementations. Not every GE has a RESTful Web interface. Especially the composition editors expose their functionality mainly through a User Interface. This case requires the interface to be described in an abstract way (e.g. what a user can do) and illustrated by screenshots of specific enabler implementations.

A couple of basic enablers are important to realize the vision of such a service business framework which enables new business models in an agile and flexible way:

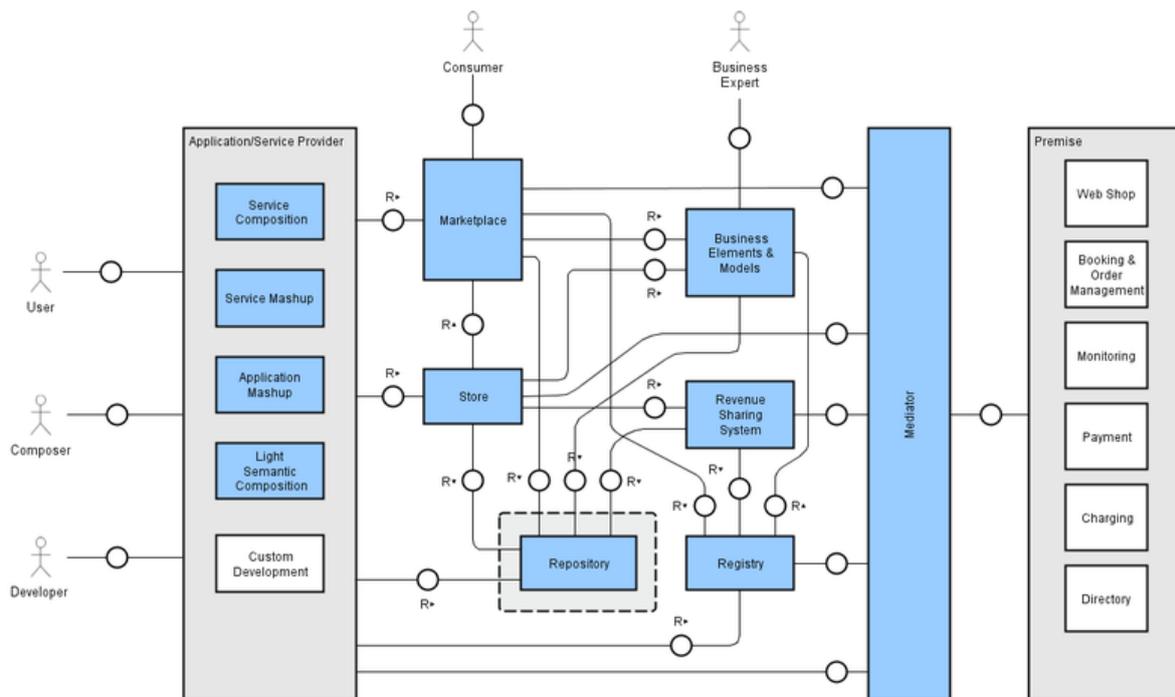
- **Repository** - defines a standard way of publishing service description in the Web in a scalable way.
- **Registry** - serves as a common database layer for run-time configuration and defines a common model and access interface.
- **Store** - allows to offer services for consumers as well as developers of future internet applications.
- **Marketplace** - defines a standard way to access market places in order to find and compare offerings from different stores and provides further functionality to foster the market for future internet applications and services in a specific domain.
- **Revenue Sharing System** - provides a common scheme and protocols for the calculation and distribution of revenues according to the agreed business models.
- **Composition** - to allow or to perform light semantic composition, furthermore composition of existing services to value added composite services and applications, which can be monetized in the Business Framework.
- **Mediator** - enables the interoperability between future internet services and applications and also allow to interface to existing enterprise systems.

This set of self-contained enablers represents only an initial starting point for a future business framework. It is expected that supplemental enablers (e.g. for contracting, quotation ...) will be developed outside the FI-WARE projects.

The Business Framework has been designed to inter operate with each other relying on Linked USDL as common uniform description format for services, which does not only focus on technical aspects of service but also covers business aspects as well as functional and non-functional service attributes. Linked USDL itself is not a Generic Enabler, since it is a data format and vocabulary specification. Nevertheless, it will be introduced as an Open Specification, which is used by different enablers in their provided and consumed APIs.

The Applications and Services Generic Enablers are named according to their main functionality. While the role names, introduced in the FI-WARE Vision (Aggregator, Gateway ...), are used to describe the stakeholders of the service ecosystem in an abstract way, the enablers names now are referring to concrete software components.

The following diagram gives an example of how the Generic Enablers can be combined to form a concrete architecture for a Service Business Framework.



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

<http://wiki.fi-ware.eu>

[Architecture of Applications and Services Ecosystem and Delivery Framework](#)

[Materializing Applications/Services Ecosystem and Delivery Framework in FI-WARE](#)

1.5 Structure of this Document

The document is generated out of a set of documents provided in the public FI-WARE wiki.

For the current version of the documents, please visit the public wiki at <http://wiki.fi-ware.eu/>

The following resources were used to generate this document:

D.3.3.2 Installation and Administration Guide front page

[Repository - Installation and Administration Guide](#)
[Marketplace - Installation and Administration Guide](#)
[Registry - Installation and Administration Guide](#)
[Application Mashup - Wirecloud - Installation and Administration Guide](#)
[Service Mashup - Mashup Factory - Installation and Administration Guide](#)
[Light Semantic Composition - Installation and Administration Guide](#)
[Service Composition - Installation and Administration Guide](#)
[Mediator - Installation and Administration Guide](#)

1.6 Typographical Conventions

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

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

1.6.1 Links within this document

The links within this document point towards the wiki where the content was rendered from. You can browse these links in order to find the "current" status of the particular content. Due to technical reasons not all pages that are part of this document can be linked document-local within the final document. For example, if an open specification references and "links" an API specification within the page text, you will find this link firstly pointing to the wiki, although the same content is usually integrated within the same submission as well.

1.6.2 Figures

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

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

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

1.6.3 Sample software code

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

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

1.7 Acknowledgements

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

1.8 Keyword list

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

1.9 Changes History

Release	Major changes description	Date	Editor
v1	First Version	2013-04-22	SAP
v2	Second Version of deliverable submission	2013-04-26	SAP
v3	preparation of submission	2013-05-16	SAP

1.10 Table of Contents

Contents

1.1	Executive Summary	2
1.2	About This Document	3
1.3	Intended Audience	3
1.4	Chapter Context	3
1.5	Structure of this Document	4
1.6	Typographical Conventions	5
1.6.1	Links within this document.....	5
1.6.2	Figures	5
1.6.3	Sample software code	5
1.7	Acknowledgements.....	6
1.8	Keyword list.....	6
1.9	Changes History	6
1.10	Table of Contents	6
2	Repository - Installation and Administration Guide	12
2.1	Repository Installation and Administration Guide.....	12
2.2	System Requirements.....	12

- 2.2.1 Hardware Requirements.....12
- 2.2.2 Operating System Support12
- 2.2.3 Software Requirements.....12
- 2.2.4 Performance.....12
- 2.3 Software Installation and Configuration.....14
 - 2.3.1 Install all required software packages14
 - 2.3.2 MongoDB Configuration14
 - 2.3.3 Application Server Configuration.....15
 - 2.3.4 Repository Configuration15
- 2.4 Sanity check procedures15
 - 2.4.1 End to End testing.....15
 - 2.4.2 List of Running Processes16
 - 2.4.3 Network interfaces Up & Open16
 - 2.4.4 Databases17
- 2.5 Diagnosis Procedures17
 - 2.5.1 Resource availability18
 - 2.5.2 Remote Service Access18
 - 2.5.3 Resource consumption.....18
 - 2.5.4 I/O flows18
- 3 Marketplace - Installation and Administration Guide19
 - 3.1 Marketplace Installation and Administration Guide19
 - 3.2 System Requirements.....19
 - 3.2.1 Hardware Requirements.....19
 - 3.2.2 Operating System Support19
 - 3.2.3 Software Requirements.....19
 - 3.3 Software Installation and Configuration.....20
 - 3.3.1 Database Configuration.....20
 - 3.3.2 Application Server Configuration.....20
 - 3.3.3 Marketplace Configuration.....20
 - 3.4 Sanity check procedures21
 - 3.4.1 End to End testing.....21
 - 3.4.2 List of Running Processes22
 - 3.4.3 Network interfaces Up & Open23
 - 3.4.4 Databases23
 - 3.5 Diagnosis Procedures24

- 3.5.1 Resource availability24
- 3.5.2 Remote Service Access24
- 3.5.3 Resource consumption.....24
- 3.5.4 I/O flows24
- 4 Registry - Installation and Administration Guide25
 - 4.1 Registry Installation and Administration Guide25
 - 4.2 System Requirements.....25
 - 4.2.1 Hardware Requirements.....25
 - 4.2.2 Operating System Support25
 - 4.2.3 Software Requirements25
 - 4.3 Software Installation and Configuration25
 - 4.3.1 MongoDB Configuration26
 - 4.3.2 Registry Configuration26
 - 4.4 Sanity check procedures26
 - 4.4.1 End to End testing.....26
 - 4.4.2 List of Running Processes26
 - 4.4.3 Network interfaces Up & Open27
 - 4.4.4 Databases27
 - 4.5 Diagnosis Procedures28
 - 4.5.1 Resource availability28
 - 4.5.2 Remote Service Access28
 - 4.5.3 Resource consumption.....28
 - 4.5.4 I/O flows28
- 5 Application Mashup - Wirecloud - Installation and Administration Guide29
 - 5.1 The Wirecloud Mashup Editor & Execution Engine GEs Installation29
 - 5.1.1 Requirements29
 - 5.1.2 Getting the source code29
 - 5.1.3 Database installation and configuration30
 - 5.1.4 Final steps33
 - 5.1.5 Running Wirecloud.....35
 - 5.2 Sanity check procedures36
 - 5.2.1 End to End testing.....37
 - 5.2.2 List of Running Processes41
 - 5.2.3 Network interfaces Up & Open42
 - 5.2.4 Databases42

- 5.3 Diagnosis Procedures43
 - 5.3.1 Resource availability43
 - 5.3.2 Remote Service Access43
 - 5.3.3 Resource consumption.....43
 - 5.3.4 I/O flows43
- 6 Service Mashup - Mashup Factory - Installation and Administration Guide44
 - 6.1 Composition Editor Mashup Factory44
 - 6.2 System Requirements.....44
 - 6.2.1 Hardware Requirements.....44
 - 6.2.2 Operating System Support44
 - 6.2.3 Software Requirements.....44
 - 6.3 Software Installation and Configuration44
 - 6.3.1 Application Server Configuration.....45
 - 6.3.2 Composition Editor Configuration.....45
 - 6.3.3 Composition Execution Engine Configuration45
 - 6.4 Engine Status Information46
 - 6.4.1 Axis Web Services Listing46
 - 6.4.2 ActiveBPEL Engine Administration46
 - 6.5 Sanity check procedures46
 - 6.5.1 End to End testing.....46
 - 6.5.2 List of Running Processes47
 - 6.5.3 Network interfaces Up & Open47
 - 6.5.4 Databases48
 - 6.6 Diagnosis Procedures48
 - 6.6.1 Resource availability48
 - 6.6.2 Remote Service Access48
 - 6.6.3 Resource consumption.....48
 - 6.6.4 I/O flows48
- 7 Light Semantic Composition - Installation and Administration Guide49
 - 7.1 Light Semantic Composition Installation and Administration Guide.....49
 - 7.2 System Requirements.....49
 - 7.2.1 Hardware Recommendations49
 - 7.2.2 Software Requirements.....49
 - 7.3 Software Installation and Configuration.....49
 - 7.3.1 Configuration49

- 7.3.2 Compilation50
- 7.3.3 Deployment50
- 7.3.4 Development55
- 7.4 Troubleshooting.....57
 - 7.4.1 Activiti database creation.....57
 - 7.4.2 Tomcat58
 - 7.4.3 Development Eclipse58
- 7.5 Sanity check procedures58
 - 7.5.1 End to End testing.....58
 - 7.5.2 List of Running Processes59
 - 7.5.3 Network interfaces Up & Open59
 - 7.5.4 Databases59
- 7.6 Diagnosis Procedures60
 - 7.6.1 Resource availability60
 - 7.6.2 Remote Service Access60
 - 7.6.3 Resource consumption.....60
 - 7.6.4 I/O flows60
- 8 Service Composition - Installation and Administration Guide61
 - 8.1 General61
 - 8.2 Composition Editor Installation and Administration61
 - 8.3 Sanity check procedures61
 - 8.3.1 End to End testing.....61
 - 8.3.2 List of Running Processes62
 - 8.3.3 Network interfaces Up & Open62
 - 8.3.4 Databases62
 - 8.4 Diagnosis Procedures62
 - 8.4.1 Resource availability62
 - 8.4.2 Remote Service Access62
 - 8.4.3 Resource consumption.....62
 - 8.4.4 I/O flows62
- 9 Mediator - Installation and Administration Guide63
 - 9.1 Mediator Installation and Administration Guide.....63
 - 9.2 System Requirements.....63
 - 9.2.1 Hardware Requirements.....63
 - 9.2.2 Operating System Support63

- 9.2.3 Software Requirements63
- 9.3 Software Installation and Configuration64
 - 9.3.1 Application Server Configuration64
- 9.4 Sanity check procedures64
 - 9.4.1 End to End testing.....65
 - 9.4.2 List of Running Processes65
 - 9.4.3 Network interfaces Up & Open66
 - 9.4.4 Databases66
- 9.5 Diagnosis Procedures66
 - 9.5.1 Resource availability66
 - 9.5.2 Remote Service Access66
 - 9.5.3 Resource consumption.....67
 - 9.5.4 I/O flows67

2 Repository - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

2.1 Repository Installation and Administration Guide

The purpose of this document is to describe how to install and administrate the software necessary to run the Repository on a server. The Repository itself is a Java Web Application, packaged in a WAR file and relies on a MongoDB NoSQL database system.

2.2 System Requirements

This section covers the requirements needed to install and use the Repository.

2.2.1 Hardware Requirements

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

Resource	Requirement
CPU	1-2 cores with at least 2.4 GHZ
Physical RAM	1G-2GB
Disk Space	25GB The actual disk space depends on the amount of data being stored within the Repositories NoSQL database System.

2.2.2 Operating System Support

The Repository has been tested against the following Operating Systems:

- Ubuntu 11.04 and 11.11, 12.04 LTS
- Microsoft Windows 7

NOTE: This Installation Guide describes the installation process on a Linux based System.

2.2.3 Software Requirements

In order to have the Repository running, the following software is needed:

- MongoDB 2.x - mandatory
- Java 1.6.x - mandatory
- Application Server, Apache Tomcat 6.x - mandatory
- Repository Software - mandatory
- Mongo Shell - optional (JavaScript shell that allows you to execute commands on the internal data store of the Repository from the command line)

2.2.4 Performance

The charts depicted below give you a short impression how about the Repository GE performs under load. The tests were performed using [SoPeCo](#), developed within the Tools Chapter WP9, a framework for systematic performance evaluations of software systems.

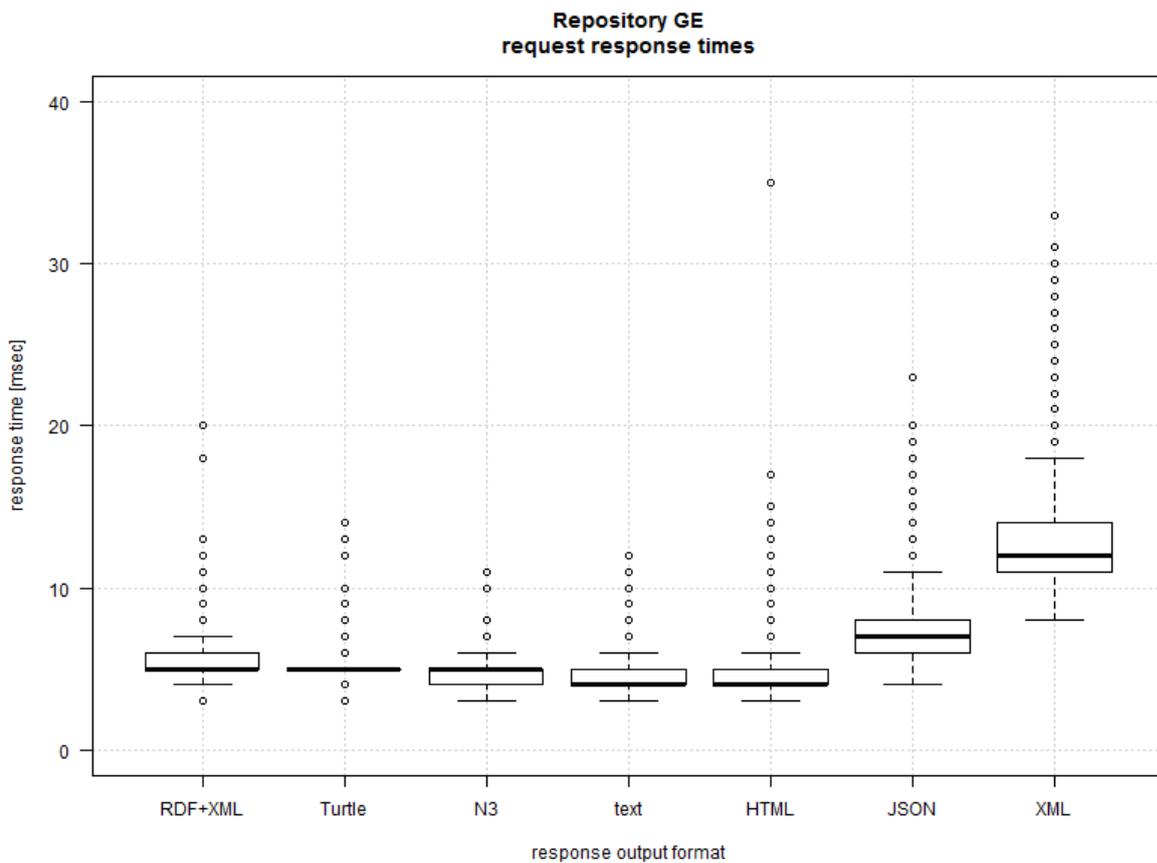
This hardware was used for running the Repository in this test:

Resource	Requirement
CPU	Intel(R) Core(TM)2 CPU 6600 @ 2.40GHz
Physical RAM	4047844 kB
Disk Space	200GB

Operating System: Linux version 3.5.0-23-generic (buldd@komainu) (gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)) #35~precise1-Ubuntu SMP Fri Jan 25 17:13:26 UTC 2013

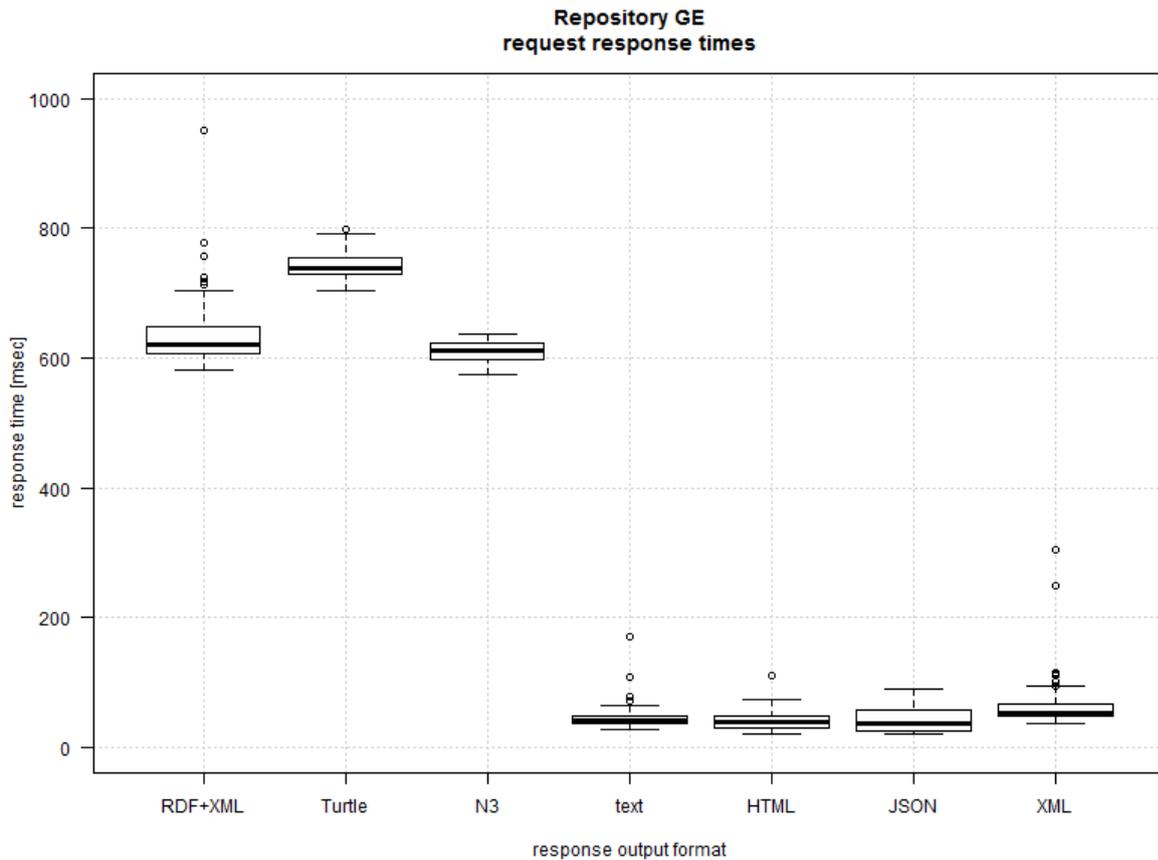
Test 1: Retrieving meta information of a single resource

In this test, meta information of a collection containing a single resource was requested in different file formats (n=1000 for each data type).



Test 2: Retrieving meta information of 1000 resources

In this test, meta information of a collection containing 1000 different resources was requested in different file formats (n=50 for each data type).



2.3 Software Installation and Configuration

2.3.1 Install all required software packages

All the mandatory dependencies can be easily installed on a debian based Linux distribution using `apt-get`:

```
$ sudo apt-get install mongodb
$ sudo apt-get install java6-runtime
$ sudo apt-get install tomcat6 tomcat6-docs tomcat6-admin
```

2.3.2 MongoDB Configuration

The next step is to create the Repository internal database named e.g. "test". You may need to have root permissions to do that.

```
$ mongo
$ use test
```

As default the Database saves its data in `/var/lib/mongodb`. Since all the Resources you upload to the repository are stored there, the size of this folder can grow rapidly. If you want to relocate that folder, you have to edit `/etc/mongodb.conf`

1. `mongodb.conf`

1. Where to store the data.

```
dbpath=/var/lib/mongodb
```

2.3.3 Application Server Configuration

It is possible to use the Apache Tomcat Application server as is, that is, without any further configuration. However, it is recommended to allow incoming connections to the Repository only through HTTPS. This can be achieved by using a front-end HTTPS server that will proxy all requests to Repository, or by configuring the Application Server in order to accept only HTTPS/SSL connection, please refer to [this link](#) for more information.

2.3.4 Repository Configuration

Now you can deploy the Repository software to your Application Server. For that you have to copy the Repository WAR package into the "webapp" folder of Apache Tomcat. To install it on other Java Application Servers (e.g. JBoss), please refer to the specific application server guidelines.

2.4 Sanity check procedures

The Sanity Check Procedures are those activities that a System Administrator has to perform verify that an installation is ready to be tested. Therefore there is a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

2.4.1 End to End testing

Although one End to End testing must be associated to the Integration Test, we can show here a quick testing to check that everything is up and running. The first test step involves creating a new resource as well as the implicit creation of a collection. The second test step checks if meta information in different file formats can be obtained.

Step 1 - Create the Resource

Create a file named test.txt with arbitrary content.

```
curl -v -H "Content-Type: text/plain" -X PUT --data "@test.txt"
http://[SERVER_URL]:8080/FiwareRepository/v1/testCollection/testResource
```

You should receive a HTTP/1.1 201 as status code

Step 2 - Retrieve meta information

Test HTML Response:
open

```
http://[SERVER_URL]:8080/FiwareRepository/v1/testCollection/
```

in your web browser. You should receive meta information about the implicit created collection in HTML format.

Test Text Response:

```
curl -v -H "Content-Type: text/plain" -X GET
http://[SERVER_URL]:8080/FiwareRepository/v1/testCollection/
```

You should receive meta information about the implicit created collection in text format. You may use curl to also test the other supported content types (*application/json,application/rdf+xml,text/turtle,text/n3,text/html,text/plain,application/xml*)

2.4.2 List of Running Processes

You can execute the following command to check whether the Tomcat web server and the MongoDB database are running:

```
ps -ax | grep 'tomcat\|mongo'
```

The resulting output should show a message text similar to the following:

```
$ ps -ax | grep 'tomcat\|mongo'

 646 ?          Ssl      0:53 /usr/bin/mongod --config
/etc/mongodb.conf
 1100 ?          Sl       0:12 /opt/bitnami/java/bin/java -
Djava.util.logging.config.file=/opt/bitnami/apache-
tomcat/conf/logging.properties
                        -XX:MaxPermSize=512m -Xms256m -Xmx512m -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
                        -Djava.endorsed.dirs=/opt/bitnami/apache-
tomcat/endorsed -classpath /opt/bitnami/apache-
tomcat/bin/bootstrap.jar
                        -Dcatalina.base=/opt/bitnami/apache-tomcat -
Dcatalina.home=/opt/bitnami/apache-tomcat
                        -Djava.io.tmpdir=/opt/bitnami/apache-
tomcat/temp org.apache.catalina.startup.Bootstrap start
 3824 pts/3     S+       0:00 grep --color=auto tomcat\|mongo
```

2.4.3 Network interfaces Up & Open

To check the ports in use and listening, execute the command:

```
$ sudo netstat -ltp
```

The produced output must be somehow similar to the following:

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
State      PID/Program name
tcp        0      0 localhost:28017         *:*
LISTEN    646/mongod
tcp        0      0 localhost:27017         *:*
LISTEN    646/mongod
tcp6       0      0 [::]:8009              [::]:*
LISTEN    1100/java
```

2.4.4 Databases

The last step in the sanity check, once that we have identified the processes and ports, is to check the database to be up and accept queries. For that, we execute the following commands:

```
$ mongo test
$ show dbs
```

to connect to the mongo database on the server. It should show a message text similar to the following:

```
$ mongo test
MongoDB shell version: 2.0.4
connecting to: test

show dbs
local   (empty)
test    4.3625GB
```

2.5 Diagnosis Procedures

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

specific testing is out of the scope of this section. The following sections have to be filled in with the information or an “N/A” (“Not Applicable”) where needed.

2.5.1 Resource availability

The resource load of the Repository strongly depends on the number of concurrent requests received as well as free main memory and disk space:

- minimum available main memory: 256 MB
- minimum available hard disk space: 10 GB

2.5.2 Remote Service Access

N/A

2.5.3 Resource consumption

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

- the main memory consumption of the Tomcat application server should be between 48MB and 1024MB. These numbers can vary significantly if you use a different application server.

2.5.4 I/O flows

The only expected I/O flow is of type HTTP or HTTPS, on ports defined in Apache Tomcat configuration files, inbound and outbound. Requests interactivity should be low.

3 Marketplace - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

3.1 Marketplace Installation and Administration Guide

The purpose of this document is to describe how to install and administrate the necessary software on a server so that it can run the Marketplace.

The Marketplace itself is a Java Web Application, packaged in a WAR file and relays on a SQL database.

3.2 System Requirements

This section covers the requirements needed to install and use the Marketplace.

3.2.1 Hardware Requirements

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

Resource	Requirement
CPU	1-2 cores with at least 2.4 GHZ
Physical RAM	2G-4GB
Disk Space	10GB The actual disk space will depend on the amount of data being stored within the Marketplace.</ref>

3.2.2 Operating System Support

The Marketplace has been tested against the following Operating Systems:

- Ubuntu 11.04 and 11.11, 12.04 LTS
- Microsoft Windows 7

NOTE: This Installation Guide describes the isntalltion process on a Linux based System.

3.2.3 Software Requirements

In order to have the Marketplace running, the following software is needed:

- Database Manager - MySQL (5.5) Server - mandatory
- MySQL Client - mandatory
- Java 1.6.x - mandatory
- Application Server, Apache Tomcat 6.x - mandatory
- Luke - optional (development and diagnostic tool, which can access the Fi-Ware Marketplace search index)
- Marketplace Software

3.3 Software Installation and Configuration

All the mandatory dependencies can be easily installed on a debian based Linux distribution using `apt-get`:

```
$ sudo apt-get install mysql-server mysql-client
$ sudo apt-get install java6-runtime
$ sudo apt-get install tomcat6 tomcat6-docs tomcat6-admin
```

3.3.1 Database Configuration

The next step is to create the Marketplace internal database named "marketplace". You need to have administrator permissions in MySQL. This usually means that you have to use the MySQL root user with the password you chose during the installation process.

```
$ sudo /etc/init.d/mysql start
$ sudo mysqladmin -u root -p[MYSQL_ROOT_PWD] create marketplace
```

Running the Marketplace requires access to the MySQL tables defined in the Marketplace software package. The DB schema should contain several tables that can be created using the SQL Script from Marketplace software "ddl_mysql5.sql"

```
$ mysql -u root -padmin
use marketplace;

// ddl_mysql5.sql statements
create table Localuser (LOCALUSER_ID integer ...
...
...
...
```

3.3.2 Application Server Configuration

It is possible to use the Apache Tomcat Application server as is, that is, without any further configuration. However, it is recommended to allow incoming connection to the Marketplace only through HTTPS. This can be achieved by using a front-end HTTPS server that will proxy all requests to Marketplace, or by configuring the Application Server in order to accept only HTTPS/SSL connection, please refer to [this link](#) for more information.

3.3.3 Marketplace Configuration

Before you deploy the Marketplace software to your Application Server you have to configure the database.properties file according to your environment.

File Location: marketplace.war/WEB-INF/classes/properties/database.properties

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/marketplace
jdbc.username=[YOUR_DB_USER]
jdbc.password=[YOUR_DB_PASSWORD]
```

To enable the fulltext search to run properly, you have to specify a folder where the search indexes should be stored.

File Location: marketplace.war/WEB-INF/classes/properties/marketplace.properties

```
luceneIndexPath=[PATH_TO_INDEXES]
```

The Marketplace WAR package can now be installed by copying it into "webapp" folder of Apache Tomcat. To install it on other Java Application Servers (e.g. JBoss), please refer to their specific application server guidelines.

3.4 Sanity check procedures

The Sanity Check Procedures are those activities that a System Administrator has to perform to verify that an installation is ready to be tested. Therefore there is a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

3.4.1 End to End testing

Although one End to End testing must be associated to the Integration Test, we can show here a quick testing to check that everything is up and running. The first test step involves authenticating against the marketplace. The second test step tests if its possible to register a new user.

Step 1: Authenticate a user

To quickly check if the application is running open

```
http://[SERVER_URL]/FiwareMarketplace/v1/registration/stores/
```

in your web browser.

Try to login with the demo user account demo1234 (Password: demo1234). If you can authenticate the demo user successfully the Application Server is running and the Marketplace is deployed correctly.

Step 2: Register a new user

Post the following body information:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><user
username="demoCompany"><company>demoCompany</company><email>demo1213
@sap.com</email><password>demoCompany</password></user>
```

to this URL:

```
[SERVER_URL]/FiwareMarketplace/v1/registration/userManagement/user
```

a new user should be created by the system, the http response should have the status 201 and you should be able to login using your new created user.

3.4.2 List of Running Processes

You can execute the following command to check that the Tomcat web server and the MySQL database are running:

```
ps -ax | grep 'tomcat\|mysql'
```

It should show a message text similar to the following:

```
$ ps -ax | grep 'tomcat\|mysql'

 689 ?          S          0:00 /bin/sh
/opt/bitnami/mysql/bin/mysqld_safe --defaults-
file=/opt/bitnami/mysql/my.cnf --port=3306
--socket=/opt/bitnami/mysql/tmp/mysql.sock --
datadir=/opt/bitnami/mysql/data --log-
error=/opt/bitnami/mysql/data/mysqld.log
--pid-file=/opt/bitnami/mysql/data/ip-10-234-
150-94.pid --lower-case-table-names=1
1055 ?         S1         0:01 /opt/bitnami/mysql/bin/mysqld.bin --
defaults-file=/opt/bitnami/mysql/my.cnf --basedir=/opt/bitnami/mysql
--datadir=/opt/bitnami/mysql/data --plugin-
dir=/opt/bitnami/mysql/lib/plugin --user=mysql --lower-case-table-
names=1
--log-error=/opt/bitnami/mysql/data/mysqld.log
--pid-file=/opt/bitnami/mysql/data/ip-10-234-150-94.pid
--socket=/opt/bitnami/mysql/tmp/mysql.sock --
port=3306
1100 ?         S1         0:07 /opt/bitnami/java/bin/java -
Djava.util.logging.config.file=/opt/bitnami/apache-
tomcat/conf/logging.properties
-XX:MaxPermSize=512m -Xms256m -Xmx512m -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
-Djava.endorsed.dirs=/opt/bitnami/apache-
tomcat/endorsed -classpath /opt/bitnami/apache-
tomcat/bin/bootstrap.jar
-Dcatalina.base=/opt/bitnami/apache-tomcat -
Dcatalina.home=/opt/bitnami/apache-tomcat
-Djava.io.tmpdir=/opt/bitnami/apache-
tomcat/temp org.apache.catalina.startup.Bootstrap start
```

```
1533 pts/2      S+          0:00 grep --color=auto tomcat\|mysql
```

3.4.3 Network interfaces Up & Open

To check whether the ports in use are listening, execute the command:

```
$ sudo netstat -ltp
```

The expected results must be somehow similar to the following:

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
State      PID/Program name
tcp        0      0 localhost:mysql        *:*
LISTEN    1055/mysqlld.bin
tcp6       0      0 [::]:8009              [::]:*
LISTEN    1100/java
```

3.4.4 Databases

The last step in the sanity check, once that we have identified the processes and ports is to check the database that has to be up and accept queries. For that, we execute the following commands:

```
$ mysql -u [DB_USER] -p
$ use marketplace;
$ show tables;
```

It should show a message text similar to the following:

```
+-----+
| Tables_in_marketplace |
+-----+
| localuser              |
| rating                 |
| ratingcategory        |
| ratingcategoryentry   |
| ratingobject          |
| ratingsystem          |
| service                |
```

```

| store |
+-----+
8 rows in set (0.00 sec)

```

3.5 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator has to take to locate the source of an error in a GE. Once the nature of the error is identified by these tests, the system admin can 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. The following sections have to be filled in with the information or an “N/A” (“Not Applicable”) where needed.

3.5.1 Resource availability

The resource load of the Marketplace strongly depends on the number of concurrent requests received as well as on the free main memory and disk space:

- minimum available main memory: 256 MB
- minimum available hard disk space: 2 GB

3.5.2 Remote Service Access

N/A

3.5.3 Resource consumption

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

- the main memory consumption of the Tomcat application server should be between 48MB and 1024MB. These numbers can vary significantly if you use a different application server.

3.5.4 I/O flows

The only expected I/O flow is of type HTTP or HTTPS, on ports defined in Apache Tomcat configuration files), inbound and outbound. Requests interactivity should be low.

4 Registry - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

4.1 Registry Installation and Administration Guide

The purpose of this document is to describe how to install and administrate the necessary software on a server so that it can run the Registry.

The Registry itself is a nodejs Web application, packaged in a NPM file. It relies on a MongoDB NoSQL database system.

4.2 System Requirements

This section covers the requirements needed to install and use the Registry.

4.2.1 Hardware Requirements

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

Resource	Requirement
CPU	1-2 cores with at least 2.4 GHZ
Physical RAM	1G-2GB
Disk Space	25GB The actual disk space will depend on the amount of data being stored within the NoSQL database System.</ref>

4.2.2 Operating System Support

The Registry has been tested against the following Operating Systems:

- Ubuntu 11.04 and 11.11, 12.04 LTS

NOTE: This Installation Guide describes the installation process on a Linux based System.

4.2.3 Software Requirements

In order to have the Registry running, the following software is needed:

- MongoDB 2.x - mandatory
- Nodejs 0.8.8 or higher - mandatory
- Registry Software - mandatory
- NPM 1.1.59 - nodejs package manager - mandatory
- Mongo Shell - optional (JavaScript shell that allows you to execute commands on the internal data store of the Registry from the command line)

4.3 Software Installation and Configuration

All the mandatory dependencies can be easily installed on a Debian based Linux distribution using `apt-get`:

```
$ sudo apt-get install mongodb
$ sudo apt-get install nodejs npm
```

The Registry application can be installed from github. All dependencies should be resolved automatically.

```
$npm install https://github.com/fi-ware/registry/tarball/master
```

4.3.1 MongoDB Configuration

The next step is to create the Registry internal database named e.g. "test". You may need to have root permissions to do that.

```
$ mongo
$ use test
```

The Database saves its data in `/var/lib/mongodb` per default. Since all the Registry entries are stored there, the size of this folder can grow rapidly. If you want to relocate that folder, you have to edit `/etc/mongodb.conf`

1. `mongodb.conf`
1. Where to store the data.

```
dbpath=/var/lib/mongodb
```

4.3.2 Registry Configuration

4.4 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.

4.4.1 End to End testing

To quickly check if the application is running open

```
http://SERVER_URL:TOMCAT_PORT/registry/v1/
```

in your web browser. If you see a directory page with an empty list "[]", the Registry is deployed correctly. Press Ctrl-F5 to clear the cache and reload the page. You may also use a browser extension such as "Advanced REST client", "Dev HTTP Client", or "Postmaster" to put register entries onto the server.

4.4.2 List of Running Processes

You can execute the following command to check that the Tomcat web server and the MongoDB database are running:

```
$ ps ax |grep node\|mongo
5739 pts/2    S+      0:00 grep --color=auto node\|mongo
```

It should show something similar to the following:

```
$ $ ps ax |grep 'node\|mongo'
1021 ?        Ssl    24:29 /usr/bin/mongod --config
/etc/mongod.conf
5670 pts/0    Sl+    0:00 node app.js
5741 pts/2    S+     0:00 grep --color=auto node\|mongo
```

4.4.3 Network interfaces Up & Open

To check the ports in use and listening, execute the command:

```
$ sudo netstat -ltp
```

The expected results must be something similar to the following:

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
State      PID/Program name
tcp        0      0 localhost:28017          *:*                     LISTEN
646/mongod
tcp        0      0 localhost:27017          *:*                     LISTEN
646/mongod
tcp        0      0 *:5000                 *:*                     LISTEN
5670/node
```

4.4.4 Databases

The last step in the sanity check, once that we have identified the processes and ports is to check the database that has to be up and accept queries. For that, we execute the following commands:

```
$ mongo test
$ show dbs
```

to connect to the mongo database on the server. It should show something similar to the following results:

```
$ mongo test
MongoDB shell version: 2.0.4
connecting to: test

show dbs
local    (empty)
registry 0.20312GB
```

4.5 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. The following sections have to be filled in with the information or an “N/A” (“Not Applicable”) where needed. Do not delete section titles in any case.

4.5.1 Resource availability

The resource load of the Registry strongly depends on the number of concurrent requests received as well as free main memory and disk space:

- minimum available main memory: 256 MB
- minimum available hard disk space: 10 GB

4.5.2 Remote Service Access

N/A

4.5.3 Resource consumption

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

- the main memory consumption of the nodejs application server should be between 10MB and 15MB. These numbers can vary greatly if you use a different application server.

4.5.4 I/O flows

The only expected I/O flow is of type HTTP or HTTPS, on ports defined in the configuration files), inbound and outbound. Requests interactivity should be low.

5 Application Mashup - Wirecloud - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

5.1 The Wirecloud Mashup Editor & Execution Engine GEs Installation

This page contains the Installation and Administration Guide for the Wirecloud Mashup Platform, a reference implementation of the Application Mashup Generic Enabler, based on the [Wirecloud](#) Open Source project. The corresponding [online documentation](#) is continuously updated and improved, and provides the most appropriate source to get the most up-to-date information on installation and administration. Both wirecloud users and developers have the option to create tickets through a [jira instance](#).

5.1.1 Requirements

In order to get Wirecloud up and running, the following software is needed:

- A Database Manager (MySQL, PostgreSQL, Sqlite3...)
- Python 2.5, 2.6 or 2.7. Python 3 and other versions are not supported.
- Django 1.3
- South 0.7.3+
- lxml
- BeautifulSoup
- django-compressor 1.2
- rdflib 3.1.0+

Most of software above can be easily installed both in [Debian Wheezy](#) and [Ubuntu Oneiric](#) using `apt-get`:

```
$ sudo apt-get install python-django python-django-south python-beautifulsoup python-lxml python-pip
```

However, the last two software packages have to be installed using `pip`:

```
$ sudo pip install django-compressor>=1.2 rdflib
```

NOTE: Wirecloud also depends on the Marketplace, Store and Repository GEs. In order to run properly, these GEs must be correctly installed already.

5.1.2 Getting the source code

The Wirecloud source code is available from the [GitHub Wirecloud repository](#).

To get the latest release of the code, you can choose between two options (you will need 'root' privileges) :

- Go to the Wirecloud repository on GitHub and click on the ZIP button to download the repository as a zip file, or just click on this [link](#). Then unzip the file on `/opt/wirecloud`
- Or use a [GIT](#) client to get the latest development version via Git:

```
# cd /opt
# git clone git://github.com/Wirecloud/wirecloud.git
```

5.1.3 Database installation and configuration

To set up the database engine, it is necessary to create a configuration file called `local_settings.py` in the `/opt/wirecloud/src` directory with the following content:

```
DATABASES = {
    'default': {
        'ENGINE': '',
        'NAME': '',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
    }
}
```

The database connection must be defined in that configuration file by modifying the **DATABASE** setting. You can use any of the [database engines supported by Django](#). By default Wirecloud is configured and prepared to use a SQLite database. However, this database is just recommended for giving Wirecloud a try. In order to make use of this configuration, you only have to make sure that you have already installed the SQLite module for python:

```
$ sudo apt-get install python-pysqlite2
```

The following examples show how to configure SQLite and PostgreSQL databases.

5.1.3.1 *SQLite*

Setting up a SQLite database can be just accomplished within seconds by using the following parameters into the `local_settings.py` file:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': '<dbfile>',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
    }
}
```

```

        'PORT': '',
    }
}

```

where `<dbfile>` is the full path to the database file.

Remember to have the `python-pysqlite2` module installed:

```
$ sudo apt-get install python-pysqlite2
```

Finally, please take into account that SQLite database is **not recommended for production purposes**. It is only useful for evaluation purposes.

5.1.3.2 *PostgreSQL*

For production purposes, PostgreSQL database is a much better choice. To do so, the following parameters must be set in `local_settings.py`:

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': '<dbname>',
        'USER': '<dbuser>',
        'PASSWORD': '<dbpassword>',
        'HOST': '',
        'PORT': '',
    }
}

```

where `<dbname>` represents the name of the database, and `<dbuser>` is the name of the user with privileges on the database.

First install the object-relational database system:

```
$ sudo apt-get install postgresql
```

And then the python interface to the PostgreSQL database `python-psycopg2`:

```
$ sudo apt-get install python-psycopg2
```

Afterwards you have to create the project Database. We assume that your user has super administrator permissions in PostgreSQL. This usually means that you have to login as the `postgres` user (i.e. `$ sudo su postgres`).

Both the PostgreSQL database and its user can be created with the following commands:

```

$ createuser <dbuser> [-P]
$ createdb --owner=<dbuser> <dbname>

```

If you want to create a password protected user you must use the `-P` option.

If you want to create a database called 'wirecloud' and a user called 'wc_user' with privileges on this database, you should write the following:

```
$ createuser wc_user [-P]
$ createdb --owner=wc_user wirecloud
```

Finally, it is also needed to allow local connections to the database, i.e. from the computer you are installing Wirecloud. To do so, add the following rules to the beginning of the `/etc/postgresql/X.X/main/pg_hba.conf` file. In other words, the following two rules **MUST** be the first two rules of the file:

```
# TYPE      DATABASE          USER              CIDR-ADDRESS
METHOD
local      wirecloud        wc_user
trust
local      test_wirecloud   wc_user
trust # only necessary for testing Wirecloud
```

Reload `pg_hba.conf` in PostgreSQL server with the following command:

```
$ sudo service postgresql reload
```

And finally, restart PostgreSQL and check if your user has access using this command:

```
$ psql wirecloud -U wc_user
```

5.1.3.3 *Database population*

Before running Wirecloud, it is necessary to populate the database. This can be achieved by using this command:

```
# python manage.py syncdb --migrate
```

This command creates some tables and asks you if you want to create a Django superuser. This user is required to login into Wirecloud and to be able to perform administrative tasks; please respond yes. An example of the command output, where user/password are admin/admin, is the following:

```
...

You just installed Django's auth system, which means you don't have
any superusers defined.
Would you like to create one now? (yes/no): yes
Username (leave blank to use 'wirecloud'): admin
E-mail address: admin@c.com
Password: ***** (admin)
Password (again): ***** (admin)
```

Finally, whenever the wirecloud code is updated, the database must be migrated (and this is one of those times):

```
# python manage.py migrate
```

Note: It is strongly recommended to perform a full database backup before starting to migrate wirecloud to a new version.

5.1.4 Final steps

Make sure the `/opt/wirecloud/src/catalogue/media`, `<wirecloud>/src/deployment/gadgets` and `<wirecloud>/src/deployment/tmp` directories exist and the server has sufficient permissions to write on them. To do so:

```
# chgrp -R www-data /opt/wirecloud/src/deployment/tmp
/opt/wirecloud/src/deployment/gadgets \
    /opt/wirecloud/src/catalogue/media
# chmod g+wrX -R /opt/wirecloud/src/deployment/tmp
/opt/wirecloud/src/deployment/gadgets \
    /opt/wirecloud/src/catalogue/media
```

The `/opt/wirecloud/src/settings.py` file allows you to set several options in Wirecloud. If `DEBUG` is `False` you will need to collect Wirecloud static files using the following command and answering 'yes' when asked:

```
$ python manage.py collectstatic
```

If you use the [runserver command](#) (not recommended for production) you will have to call it with the `--insecure` switch in order to make it serve the static files when not debugging. In addition, you should serve the static files with a fast performance http server like [Nginx](#) or [Apache](#). Django has documentation for this [topic](#).

Finally, you can compress css and javascript code files for better performance using the following command:

```
$ python manage.py compress
```

Note: Don't forget to rerun the `collectstatic` and `compress` commands each time the wirecloud code is updated, this include each time an add-on is added or remove.

5.1.4.1 *Integration with Django "sites" framework*

Wirecloud uses the hostname provided by the http request when building internal URLs. This behaviour is usually good for normal use.

However, when the [sites framework](#) is installed, Wirecloud make use of it to obtain the domain to use when building internal urls. This is very useful when the hostname of the request doesn't match the public name of the Wirecloud server.

5.1.4.2 *Installing the Wirecloud Pub Sub add-on*

The development of the pub sub add-on is carried out at [\[github.com\]](#). You can always find the latest information about how to install and use it on the main page of the repository.

Newer versions of the pub sub add-on can be installed directly using pip, nevertheless it's strongly recommended to install it using the following steps to be able to perform proper testing:

```
$ cd /opt
$ git clone https://github.com/conwetlab/wirecloud-pubsub.git
$ cd wirecloud/src
$ ln -s /opt/wirecloud_pubsub/wirecloud_pubsub .
```

Since `wirecloud_pubsub` uses `django.contrib.static` functionalities, you should add it to your **INSTALLED_APPS** in `settings.py`:

```
INSTALLED_APPS = (
    ...
    'wirecloud_pubsub',
    ...
)
```

You also have to add it to your **WIRECLOUD_PLUGINS**:

```
WIRECLOUD_PLUGINS = (
    ...
    'wirecloud_pubsub.plugins.PubSubPlugin',
    ...
)
```

As last step, add a **DEFAULT_SILBOPS_BROKER** setting with the URL of the broker to use:

```
DEFAULT_SILBOPS_BROKER =
'http://pubsub.server.com:8080/silbops/CometAPI'
```

Don't forget to run the `collectstatic` and `compress` commands on your Wirecloud installation:

```
$ ./manage.py collectstatic
$ ./manage.py compress
```

5.1.4.3 *NGSI proxy*

Wirecloud comes with a javascript library that allows widgets and operators to connect to NGSI-9/10 servers. This support comes for free when installing Wirecloud, except for the subscribe operations. These operations requires what is called NGSI proxy, this proxy is a facade that receives NGSI notifications and passes them to the final targets: Widgets and Operators. This NGSI proxy doesn't need to be installed in the same machine as Wirecloud and can be shared with other Wirecloud instances.

You can install a NGSI proxy following those steps:

```
$ apt-get install nodejs npm
$ ln -s /usr/bin/nodejs /usr/bin/node
$ git clone git://github.com/conwetlab/ngsijs.git
```

```
$ cd ngsijs/ngsi-proxy
$ npm install
```

After this, you can run the NGSI proxy issuing the following command:

```
$ npm run start
```

5.1.5 Running Wirecloud

We recommend running Wirecloud based on an Apache Web Server. However, it is also possible to run it using the Django internal web server, just for testing purposes.

5.1.5.1 *Running Wirecloud using the Django internal web server*

Please note:

- Be aware that this way of running Wirecloud should be used for evaluation purposes. Do not use it in a production environment.

To start Wirecloud, type the following command:

```
$ python manage.py runserver 0.0.0.0:8080 --insecure
```

Then, go to http://computer_name_or_IP_address:8080/ where `computer_name_or_IP_address` is the name or IP address of the computer on which Wirecloud is installed, and use the username and password you provided when populating the database to sign in on the platform.

5.1.5.2 *Integrating Wirecloud with Apache*

If you choose to deploy Wirecloud in Apache, the `libapache2-mod-wsgi` module must be installed (and so does Apache!). To do so, type the following command:

```
$ sudo apt-get install apache2 libapache2-mod-wsgi
```

Then you have to create a `django.wsgi` file anywhere, but we recommend to have it inside your wirecloud installation (i.e. `/opt/wirecloud/src/apache/django.wsgi`):

```
import os
import sys
path = '<path_to_wirecloud/src>'
if path not in sys.path:
    sys.path.insert(0, path)
os.environ['DJANGO_SETTINGS_MODULE'] = 'settings'
import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```

Please, pay attention that you set the right path to the wirecloud/src directory (i.e. `/opt/wirecloud/src`)

Finally, add the following lines in the main *virtualhost* to the Apache's *sites-available* configuration file, usually located in `/etc/apache2/sites-available/default`:

```
<VirtualHost *:80>
    ...
    ### Wirecloud ###
    WSGIScriptAlias / <path_to_django_wsgi>
    WSGIPassAuthorization On
    Alias /static <path_to_wirecloud/src/static>
    <Location "/static">
        SetHandler None
        <IfModule mod_expires.c>
            ExpiresActive On
            ExpiresDefault "access plus 1 week"
        </IfModule>
        <IfModule mod_headers.c>
            Header append Cache-Control "public"
        </IfModule>
    </Location>
    <Location "/static/cache">
        <IfModule mod_expires.c>
            ExpiresDefault "access plus 3 years"
        </IfModule>
    </Location>
    ...
</VirtualHost>
```

Again, pay special attention to the paths to the `django wsgi` file and the `/opt/wirecloud/src/static` directory.

Once you have the site enabled, restart Apache

```
# apache2ctl graceful
```

and go to http://computer_name_or_IP_address/ to get into Wirecloud.

5.2 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.

5.2.1 End to End testing

Please note that the following information is required before carrying out this procedure:

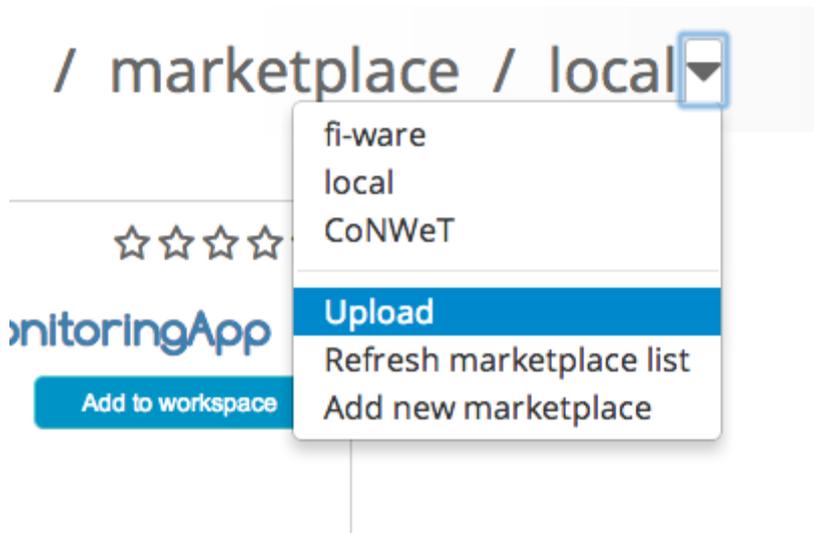
- `computer_name_or_IP_address` is the name or IP address of the computer on which Wirecloud is installed.
- Valid credentials for the Wirecloud instance to test (e.g. user: admin / password: admin, as stated in the "Database population" section of this guide).
- The following files:
 - http://conwet.fi.upm.es/docs/download/attachments/1278018/CoNWET_weather-example_1.0.1.wgt
 - http://conwet.fi.upm.es/docs/download/attachments/1278018/CoNWET_wms-viewer-geowidget_0.1.wgt
 - http://conwet.fi.upm.es/docs/download/attachments/1278018/CoNWET_weather-mashup-example_1.0.wgt

To quickly check if the application is running, follow these steps:

1. Open a browser and type http://computer_name_or_IP_address/ in the address bar.
2. The following user login form should appear:

A user login form with a light blue background. It contains two input fields: "Username" and "Password". To the right of the "Password" field is a blue button labeled "LOG IN". Below the "Password" field is a blue link labeled "Forgot password?".A form titled "New to Wirecloud?". It contains two input fields: "Username" and "Email". To the right of the "Email" field is a blue button labeled "SIGN UP".

3. Enter the credentials and click on the "log in" button.
4. Click on the *Marketplace* tab.
5. Open the local catalogue uploader view using the path selector as depicted in the following figure:



- 6. Upload CoNWeT_weather-example_1.0.1.wgt file using the "Adding widgets from packages" form.



- 7. The new widget should be now available on the local catalogue.

/ marketplace / local ▾

Weather Widget ☆☆☆☆☆

Example

widget **Add to workspace**



Vendor Name
CoNWeT

Description
Example of widget for the Wirecloud Platform.

8. Follow steps 6 and 7 using the CoNWeT_wms-viewer-geowidget_0.1.wgt and CoNWeT_weather-mashup-example_1.0.wgt files.
9. All the widgets and mashups should be now available on the local catalogue.

/ marketplace / local ▾

Web Map Service ☆☆☆☆☆

widget **Add to workspace**



Vendor Name
CoNWeT

Description
This widget gives users the ability to handle different map layers coming from several sources, showing all of them on the same map. It has a bunch of both events and slots that

Weather Widget ☆☆☆☆☆

Example

widget **Add to workspace**



Vendor Name
CoNWeT

Description
Example of widget for the Wirecloud Platform.

Weather Example ☆☆☆☆☆

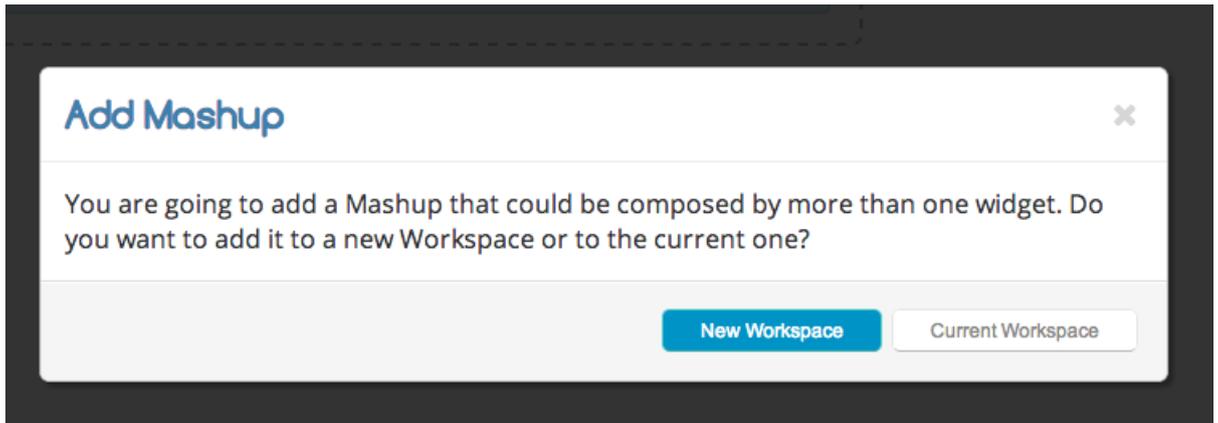
Mashup **Add to workspace**

No image available

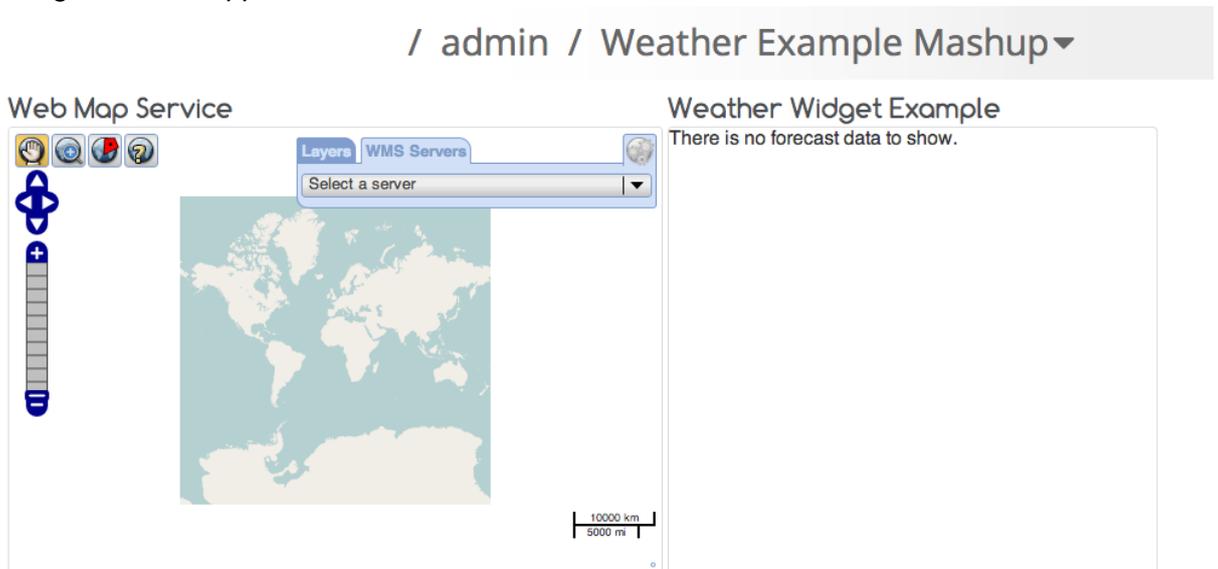
Vendor Name
CoNWeT

Description
Wirecloud Mashup composed of: Weather Widget Example, Web Map Service

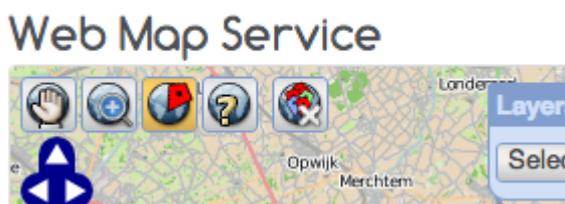
10. Click on the "Add to Workspace" button of the "Weather Example Mashup".
11. Click on the "New workspace" button.



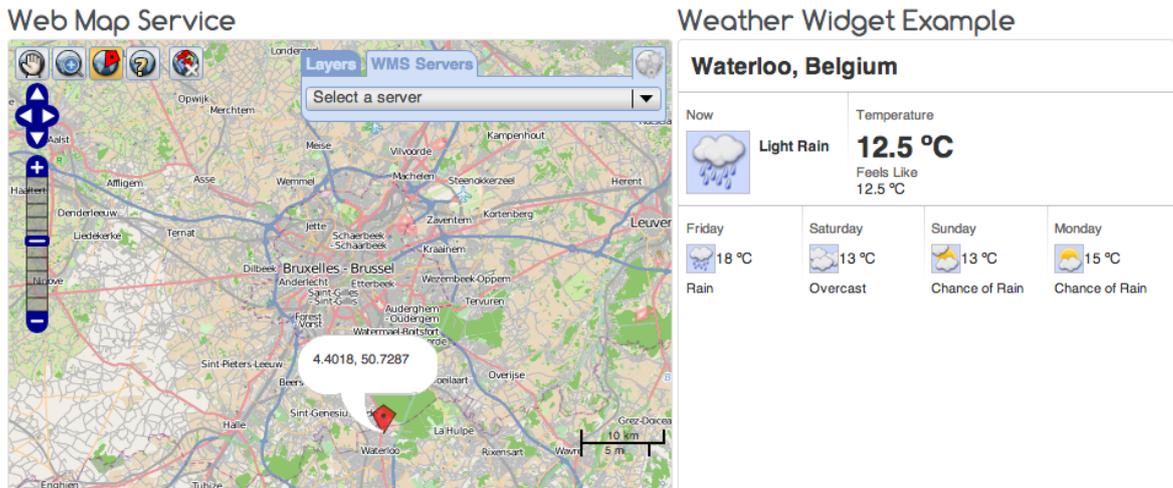
- The view should automatically change to the "Weather Example Mashup" view and widgets should appear in it.



- Select the pin tool in the "Web Map Service" widget clicking.



- And click the desired location. The "Weather Widget Example" should update the forecast info.



By performing this sequence of steps, you will check that the Wirecloud Mashup platform is running and correctly deployed, and its database has been properly set up and populated.

5.2.2 List of Running Processes

We need to check that the Apache web server and the Postgres database are running. Wirecloud uses a python interpreter, but it will not be listed as it runs embedded into apache2. If we execute the following command:

```
ps -ewF | grep 'apache2\|postgres' | grep -v grep
```

It should show something similar to the following:

```
$ ps -ewF | grep 'apache2\|postgres' | grep -v grep
postgres 1631      1  0 25212  9452   0 Jul03 ?          00:00:19
/usr/lib/postgresql/9.1/bin/postgres -D /var/lib/postgresql/9.1/main
-c config_file=/etc/postgresql/9.1/main/postgresql.conf
postgres 1702  1631  0 25208   3784   0 Jul03 ?          00:00:47
postgres: writer process
postgres 1703  1631  0 25208   1452   0 Jul03 ?          00:00:39
postgres: wal writer process
postgres 1704  1631  0 25462   2964   0 Jul03 ?          00:00:16
postgres: autovacuum launcher process
postgres 1705  1631  0 17370   1660   0 Jul03 ?          00:00:18
postgres: stats collector process
root      3811      1  0 50067  10848   0 13:13 ?          00:00:00
/usr/sbin/apache2 -k start
```

```

www-data 3818 3811 0 68663 39820 0 13:13 ? 00:00:00
/usr/sbin/apache2 -k start
www-data 3819 3811 0 68687 39448 0 13:13 ? 00:00:00
/usr/sbin/apache2 -k start
www-data 3822 3811 0 68901 40160 0 13:13 ? 00:00:00
/usr/sbin/apache2 -k start

```

5.2.3 Network interfaces Up & Open

To check the ports in use and listening, execute the command:

```
$ sudo netstat -ltp
```

The expected results must be something similar to the following:

```

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
State      PID/Program name
tcp        0      0 localhost:postgresql    *:*
LISTEN    1631/postgres
tcp        0      0 *:http                  *:*
LISTEN    3811/apache2

```

or these ones in case the machine is configured to use IPv6:

```

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
State      PID/Program name
tcp        0      0 localhost:postgresql    *:*
LISTEN    1631/postgres
tcp6       0      0 [::]:http               [::]:*
LISTEN    3811/apache2

```

5.2.4 Databases

The last step in the sanity check, once that we have identified the processes and ports, is to check the different databases that have to be up and accepting queries. If we execute the following command:

```
$ psql -U wc_user wirecloud
```

It should show a message text similar to the following:

```
psql (9.1.4)
```

```
Type "help" for help.
```

```
wirecloud=>
```

5.3 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.

5.3.1 Resource availability

Wirecloud runs fine with a minimum of 512 MB of available RAM (1024 MB recommended) and 10 GB of hard disk space. Nevertheless memory usage strongly depends on the number of concurrent users. According to normal usage patterns taken from the log history, memory usage exceeding 256 MB per user are to be considered abnormally high. Wirecloud is not CPU-intensive and thus CPU usages over 5% per user is considered abnormal. Wirecloud is i/o-intensive and performances below 12 http requests per second are considered abnormal. The results from monitoring the testbed instance usage shows that the aforementioned ranges remains valid.

5.3.2 Remote Service Access

N/A

5.3.3 Resource consumption

Resource consumption strongly depends on the load, especially on the number of concurrent users logged in.

- The main memory consumption of the Apache Web server should be between 64 MB and 1024 MB.
- Postgresql should consume a small amount of memory, not more than 64 MB.

5.3.4 I/O flows

The only expected I/O flow is of type HTTP, on port defined in Apache Web Server configuration files.

6 Service Mashup - Mashup Factory - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

6.1 Composition Editor Mashup Factory

This document is intended as an installation and an administration guide for usage of the Service Mashup GE implementation named Mashup Factory.

6.2 System Requirements

This section covers the requirements needed to install and use the Service Mashup Editor and Execution Environment.

6.2.1 Hardware Requirements

The following table contains the minimum resource requirements for running the Composition Editor:

Resource	Requirement
CPU	1-2 cores with at least 2.4 GHZ
Physical RAM	2G-4GB
Disk Space	10GB The actual disk space will depend on the amount of data being stored within the Composition Editor.

6.2.2 Operating System Support

The Composition Editor has been tested against the following Operating Systems:

- Ubuntu 10.04.4 LTS

NOTE: This Installation Guide describes the installation process on a Linux based System.

6.2.3 Software Requirements

In order to have the Composition Editor running, the following software is needed:

- Java 1.6.x - mandatory
- Application Server, Apache Tomcat 6.x - mandatory
- Composition Editor Software

6.3 Software Installation and Configuration

The Composition Editor Mashup Factory will be hosted in a Virtual Machine in the Testbed.

The GE instance owner (DT) is in charge of the installation of the package.

The Composition Editor GE (DT) is packaged as several .war archives. The following steps will take you through the binary distribution installation on Unix/Linux systems.

1. Download the Composition Editor GE binary distribution from the repository.
2. Copy the .war archives from the binary distribution to the webapps directory of an Apache Tomcat 6 instance.
3. Extract the exist.zip archive from the binary distribution and copy the directory 'exist' and its content to the webapps directory of the Apache Tomcat 6 instance.
4. Restart the tomcat server

6.3.1 Application Server Configuration

It is possible to use the Apache Tomcat Application server as is, without any further configuration. However, it is recommended to allow incoming connection to the Composition Editor only through HTTPS. This can be achieved by using a front-end HTTPS server that will proxy all requests to Composition Editor, or by configuring the Application Server in order to accept only HTTPS/SSL connection, please refer to [this link](#) for more information.

The Execution Engine of the Mashup Factory is a 3rd party BPEL engine that is no longer available to the public. The following description was extracted from the 3rd party BPEL engine installation guide. This documentation was provided by www.activebpel.org. There is no responsibility about correctness by Mashup Factory.

The Execution Engine Mashup Factory will be hosted in a Virtual Machine in the Testbed. The GE instance owner (DT) is in charge of the installation of the package.

Run the script install.bat (Windows) or install.sh (Unix). It copies the contents of lib into \$CATALINA_HOME/shared/lib and creates the directory \$CATALINA_HOME/bpr, where BPEL process .bpr archives are deployed.

```
% cd activebpel
activebpel% install.sh
```

When running install.sh on Unix, you may need root privileges.

6.3.2 Composition Editor Configuration

Edit the Tomcat users file 'conf/tomcat-users.xml' and add your desired users with the role 'authUser'.

6.3.3 Composition Execution Engine Configuration

There are a few server parameters that are configurable via a small XML file named aeEngineConfig.xml, which is found in the directory \$CATALINA_HOME/bpr. Configuration changes made via the BpelAdmin configuration page, normally found at http://SERVER_URL:TOMCAT_PORT/BpelAdmin/config.jsp, are saved to this file.

6.4 Engine Status Information

When the servlet engine (for example, Tomcat) is running, there are two Web pages available that display information about the server: the Axis Web services listing and the ActiveBPEL engine administration page.

6.4.1 Axis Web Services Listing

The page at http://SERVER_URL:TOMCAT_PORT/active-bpel/services is generated by Axis (modify the host and port to suit your installation). It lists all of the Web services that are available. This list is independent of the ActiveBPEL engine; the engine hands Web services to Axis to deploy. Each service lists its available operations and has a "wsdl" link next to its name. Clicking on that link causes Axis to generate the WSDL for that Web service and send it to your browser. (If your browser does not display anything, then you can use your browser's "Save Link As..." feature to save the WSDL to your computer and look at it that way.) The format for each WSDL link is `.../services/ServiceName?wsdl` so you can type the link in directly if you desire.

6.4.2 ActiveBPEL Engine Administration

The Web page at http://SERVER_URL:TOMCAT_PORT/BpelAdmin/ is the ActiveBPEL engine administration page (again, modify the host and port to suit your installation). From that page you can see and edit the engine's configuration parameters and click the links on the left to see information about the deployed and active BPEL processes.

The engine configuration parameters are described in the online help available from the BpelAdmin pages.

Clicking on "Deployed Processes" displays a list of the BPEL processes deployed within the engine.

Clicking on "Active Processes" displays by default a list of the BPEL processes that are running, have completed, or faulted. Click on "Running" or "Complete" in "Process Filter Selections" to limit the process display to those states.

Clicking on "Receive Queue" displays a list of queued receives.

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

To quickly check if the application is running open

```
http://SERVER_URL:TOMCAT_PORT/mamsCore/mamsCore.jsp
```

in your web browser (Firefox mandatory). If a user login pops up, the Application Server is running and the Composition Editor is deployed correctly. To quickly check if the application is running open

```
http://SERVER_URL:TOMCAT_PORT/BpelAdmin
```

in your web browser. If the ActiveBpel console pops up, the Application Server is running and the Composition Engine is deployed correctly.

6.5.2 List of Running Processes

You can execute the following command to check that the Tomcat web server is running:

```
ps -ef | grep tomcat
```

It should show something similar to the following:

```
$ ps -ef | grep tomcat

root      5311      1  0 00:12 ?          00:04:09 /usr/lib/jvm/java-6-
sun/bin/java -XX:MaxPermSize=512M
-Xmx1500m -Xdebug -
Xrunjdwp:transport=dt_socket,address=2875,suspend=n,server=y
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -
Djava.util.logging.config.file=/usr
/local/tomcat6/conf/logging.properties -
Djava.endorsed.dirs=/usr/local/tomcat6/endorsed -classpath
:/usr/local/tomcat6/bin/bootstrap.jar -
Dcatalina.base=/usr/local/tomcat6 -Dcatalina.home=/usr/local
/tomcat6 -Djava.io.tmpdir=/usr/local/tomcat6/temp
org.apache.catalina.startup.Bootstrap start
root      13618   5880  0 10:25 pts/0      00:00:00 grep tomcat
```

6.5.3 Network interfaces Up & Open

To check the ports in use and listening, execute the command:

```
$ sudo netstat -ltp
```

The expected results must be something similar to the following:

```
Active Internet connections (only servers)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address
State	PID/Program name			
tcp6	0	0	localhost:8009	[::]:*
LISTEN	8423/java			

6.5.4 Databases

MashupFactory uses an XML database which is deployed as a webapp in the tomcat container (included in the binary distribution). To check whether the database is up and running open the page http://SERVER_URL:TOMCAT_PORT/exist in a web browser.

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 Resource load of the Composition Editor depends on the number of concurrent requests received as well as free main memory and disk space:

- minimum available main memory: 256MB
- minimum available hard disk space: 2GB

6.6.2 Remote Service Access

The administrative console for remotely maintaining service compositions can be accessed at http://SERVER_URL:TOMCAT_PORT/exist/admin.

The administrative console for remotely maintaining service compositions execution can be accessed at http://SERVER_URL:TOMCAT_PORT/BpelAdmin.

6.6.3 Resource consumption

Resource consumption depends on the load, especially on the number of concurrent requests. The main memory consumption of the Tomcat application server should be between 256MB and 1024MB. These numbers can vary greatly if you use a different application server.

6.6.4 I/O flows

The only expected I/O flow is of type HTTP or HTTPS, on ports defined in Apache Tomcat configuration files), inbound and outbound. Requests interactivity should be low.

7 Light Semantic Composition - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

7.1 Light Semantic Composition Installation and Administration Guide

The purpose of this document is to provide the essential steps for the installation and configuration of the Light Semantic Composer that will make run this component from the initial installation.

The audience of the document is the system administrators that will have to install and configure this Generic Enabler.

7.2 System Requirements

7.2.1 Hardware Recommendations

- RAM:4GB
- Disk: 20GB

7.2.2 Software Requirements

- Java JDK 6 (tested with version 1.6.0_32) – Note: Current version does not work with Open JDK.
- Maven 3.X (tested with version 3.0.4).
- Tomcat 6.X or above (tested with version 6.0.35, current version does not work in Tomcat 7.X).
- Postgresql 8.3.X or above (tested with version 8.3).

7.3 Software Installation and Configuration

Checkout COMPEL code from SVN repository:

<https://forge.fi-ware.eu/scmrepos/svn/fi-ware-review/trunk/FI-WARE/Apps/>

Checked-out COMPEL directory in local filesystem that is referred as \$COMPEL hereafter

7.3.1 Configuration

Configure the following modules. Change Tomcat <host>:<port> in following configuration files:

- Compel module:
 - \$COMPEL/Compel/src/main/resources/compel.properties: modify Tomcat <host>:<port>^[1] for Oryx URL, ontologies^[2] and DTC endpoint
- KBClient module:
 - \$COMPEL/KBClient/src/main/resources/sesame.properties: modify Tomcat <host>:<port> for sesame.server property

- \$COMPEL/KBCClient/src/test/resources/ServiceDescriptions directory: modify Tomcat <host>:<port> in each test service description in (*.posm files) rdfs:isDefinedBy property pointing at the service wsdl url.
- activiti-webapp-explorer2:
 - \$COMPEL/activiti/activiti-webapp-explorer2/src/main/resources/yourbpm.properties: modify Tomcat <host>:<port> for compel.root and bpmnservice.endpoint properties.

In the following we refer to the server and port where COMPEL is deployed as \$COMPEL_SERVER and \$COMPEL_PORT

Later configuration can be modified by changing the corresponding property files in \$TOMCAT_HOME/webapps/ directory for each deployed war modules that requires to be configured: compel-<version>/ and KBCClient.jar/WEB_INF/classes/sesame.properties for each module that use this library.

1. ↑ Avoid using localhost as host. Default port is 8080 (Tomcat default port)
2. ↑ Not all ontologies are hosted locally in Tomcat.

7.3.2 Compilation

Compile modules:

```
$COMPEL> mvn -Dmaven.test.skip=true clean install
```

War files created for some modules in their target directories.

7.3.3 Deployment

Activiti Database setup

- Start PostgreSQL as root:
 - [LINUX]: > *sudo /etc/init.d/postgresql start.*
 - [Windows]: start Postgresql as Windows service or using Postgresql script provided by the installation

- Create activiti user:

```
createuser -U postgres -e -P -E activiti
```

when prompted:

Enter password for new role: activiti

Enter it again: activiti

Shall the new role be a superuser? (y/n) n

Shall the new role be allowed to create databases? (y/n) y

Shall the new role be allowed to create more new roles? (y/n) y

```
Output: CREATE ROLE activiti ENCRYPTED PASSWORD
```

```
'md54fae7683b3e8809f364b0026c885af8c' NOSUPERUSER CREATEDB
```

```
CREATEROLE INHERIT LOGIN;
```

- Create activiti database:

```
createdb -U postgres -O activiti -E utf8 -e activiti Output: CREATE DATABASE activiti OWNER activiti ENCODING 'utf8'; CREATE DATABASE
```

- Populate Activiti database:

It is done automatically when you start up the Activiti context.

NOTE: If you set up a different activiti user password (default: activiti) or the postgresql service url is different from the standard one, configure

`$YOUR_BPM/activiti/activiti-webapp-explorer2/src/main/resources/db.properties`

And recompile activiti-webapp-explorer2 module.

Deploy the following required YourBPM modules (war files) in `$TOMCAT_HOME/webapps` directory:

- `$COMPEL/activiti/activiti-webapp-explorer2/target/ActivitiCompelExplorer2.war`
- `$COMPEL/Compel/target/compel-1.0.war`
- `$COMPEL/DTC/DesignTimeComposer/target/DesignTimeComposer.war`
- `$COMPEL/BPMN20/BPMN20Module/target/BPMN20Module.war`
- `$COMPEL/third party dependencies/Knowledge Base/Software/Sesame/openrdf-sesame.war`
- `$COMPEL/third party dependencies/Knowledge Base/Software/Sesame/openrdf-workbench.war`

Deploy optional test services (war files) in `$TOMCAT_HOME/webapps` directory:

- `$COMPEL/MockServices/AirCongo/target/AirCongo.war`
- `$COMPEL/MockServices/BrusselAirlines/target/brusselAirlinesService.war`
- `$COMPEL/MockServices/firstHotelService/target/fisrtHotelService.war`
- `$COMPEL/MockServices/HostelWorld/target/HostelWorldService.war`
- `$COMPEL/MockServices/HotelFind/target/HotelFindService.war`

Deploy optional domain specific ontologies in

`$TOMCAT_HOME/webapps/ROOT/ontologies/`^[11] directory:

- `$COMPEL/third party dependencies/Knowledge Base/Ontologies/pizza.owl`
- `$COMPEL/third party dependencies/Knowledge Base/Ontologies/wine.owl`
- `$COMPEL/third party dependencies/Knowledge Base/Ontologies/wp7-businessregistration-fc.wsml.rdf.xml`
- `$COMPEL/third party dependencies/Knowledge Base/Ontologies/wp7-io.wsml.rdf.xml`
- `$COMPEL/third party dependencies/Knowledge Base/Ontologies/Turismo.owl.xml`

Launch Tomcat:

`$TOMCAT_HOME/bin>startup.sh`

Deploy COMPEL ontologies. Access this URL in a Web browser:

[http://<\\$COMPEL_SERVER>^{\[2\]}.<\\$COMPEL_PORT>/openrdf-workbench/repositories/NONE/create?type=owlim-lite](http://<$COMPEL_SERVER>[2].<$COMPEL_PORT>/openrdf-workbench/repositories/NONE/create?type=owlim-lite)

New Repository

Type: OWLIM-Lite ▾

ID: WebN1

Title: Web N+1

Storage folder: storage

Ruleset: OWL-Horst ▾

Base URL: http://www.atosresearch.eu/webn1

Entity index size: 200000

No Persistence: False ▾

Imported RDF files('; ' delimited):

Default namespaces for imports('; ' delimited):

This action will create the KB repository named WebN1.

- Select WebN1 repository in the Sesame Workbench left menu, in Repositories option

For each ontology in below table, follow the following procedure:

- Select menu Add
- Fulfill the form with the following data:
 - Base URI: Ontology_URI
 - Context: <[Ontology_URI]>
 - Select option "Select the file containing the RDF data you wish to upload"
 - Choose File: browse to the ontology RDF file in \$YOUR_BPM/third party dependencies/Knowledge Base/Ontologies/ directory
 - Click on upload

Ontology table:

Ontology	
Ontology File	Ontology_URI
\$COMPEL/third party dependencies/Knowledge Base/Ontologies/OntoBPMN.owl	http://dkm.fbk.eu/index.php/BPMN_Ontology
\$COMPEL/third party dependencies/Knowledge Base/Ontologies/POSM.rdf.owl	http://www.wsmo.org/ns/posm/0.2
\$COMPEL/third party dependencies/Knowledge Base/Ontologies/bridge.owl	http://bridge-webn1.atosresearch.eu/bridge
\$COMPEL/third party dependencies/Knowledge Base/Ontologies/integra.owl	http://bridge-webn1.atosresearch.eu/bridge

Deploying Semantic Web Service (SWS) descriptions
 \$YOUR_BPM> mvn install -f KBClient/pom.xml

The execution of KBClient tests will deploy the SWS descriptions once that Sesame KB is configured and running.

Access Activiti/YourBPM using Firefox browser at

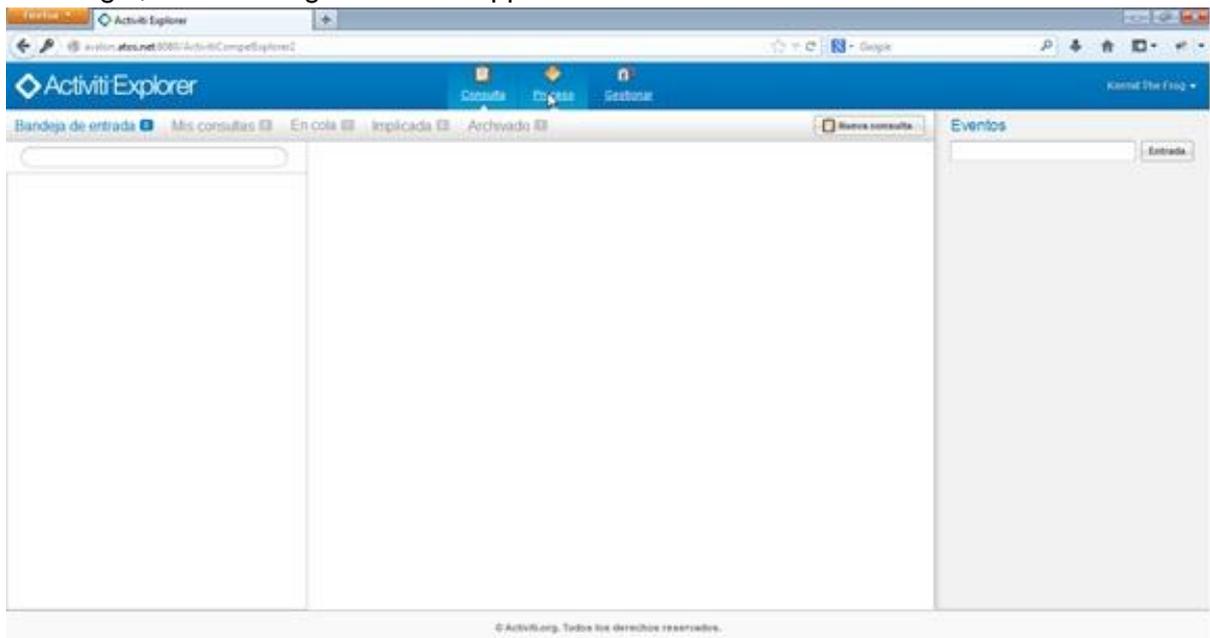
[http://<\\$COMPEL_SERVER>:<\\$COMPEL_PORT>/ActivitiCompelExplorer2](http://<$COMPEL_SERVER>:<$COMPEL_PORT>/ActivitiCompelExplorer2)



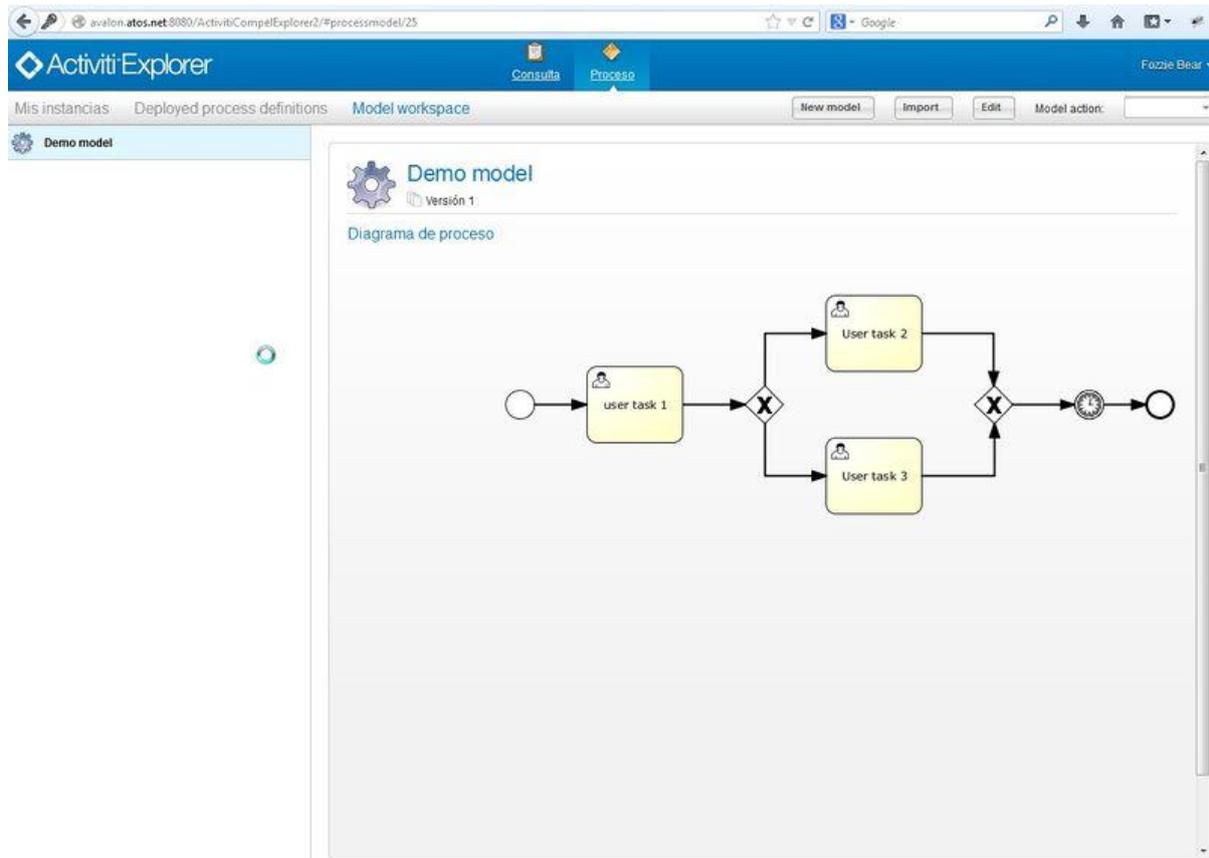
There are three users in the Activiti engine to access to the environment integrated with Compel Editor.

- kernel/kelnet --> Role: admin
- Gonzo/gonzo --> Role: manager
- fozzie/fozzie --> Role: user

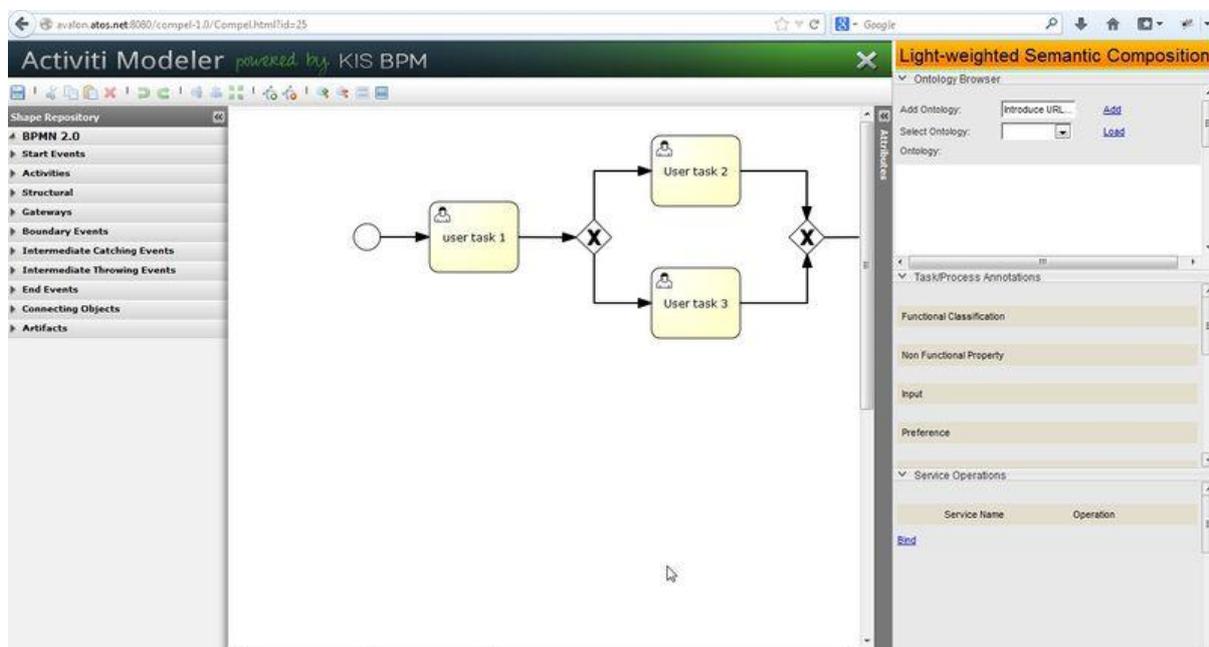
After login, the following dashboard appears:



Click on the Process-->Model Work Space



When you create a “New model” or “Edit an existing one”, YourBPM Tool, including Activiti Modeler (based in Oryx) appears in the following screen.



For further information about how to manage Activiti engine, you can consult the documentation of the Activiti (www.activiti.org)

1. ↑ Firstly create this directory in the ROOT directory

2. ↑ <Server name or IP>:<port> where you have Tomcat running with YourBPM modules

7.3.4 Development

7.3.4.1 *Environment*

- Eclipse IDE for Java Developers (tested on version Indigo 3.7.2)
- Required plugins:
 - Eclipse Web Developer Tools
 - JST Server Adapters
 - JST Server Adapters Extensions
 - Subversive SVN Connectors
 - Subversive Java HL 1.6 Implementation
 - SVNKit 1.3.5 Implementation
 - WST Server Adapters
 - M2e - Maven Integration for Eclipse.
 - Google Plugin for Eclipse (tested with version 3.7)
 - Google Web Toolkit SDK (tested with version 2.5.0)

7.3.4.2 *Activiti Modeler (based in Oryx Editor)*

Code changes in activiti-webapp-explorer2 module :

Current changes in activiti/activiti-webapp-explorer2/editor/scripts/Plugins

Modified plugins:

- shapemenu.js

NOTE: All changes tagged with //COMPEL

Update changes in Activiti Editor:

`$YOUR_BPM/activiti/activiti-webapp-explorer2/editor/>ant js.compress`

Regenerate the associated war:

`$YOUR_BPM/activiti/activiti-webapp-explorer2/mvn clean install`

Deploy the new ActivitiCompelExplorer2.war in Tomcat webapps directory.

7.3.4.3 *Eclipse*

Create Eclipse project configurations from Maven

`$COMPEL> maven eclipse:eclipse`

This command will create Eclipse project configuration for each COMPEL module.

Import individual COMPEL modules in Eclipse using File/Import/Existing Projects into Workspace. Browse \$COMPEL directory for modules. Mark option “Copy projects into workspace” if \$COMPEL directory is out of your workspace.

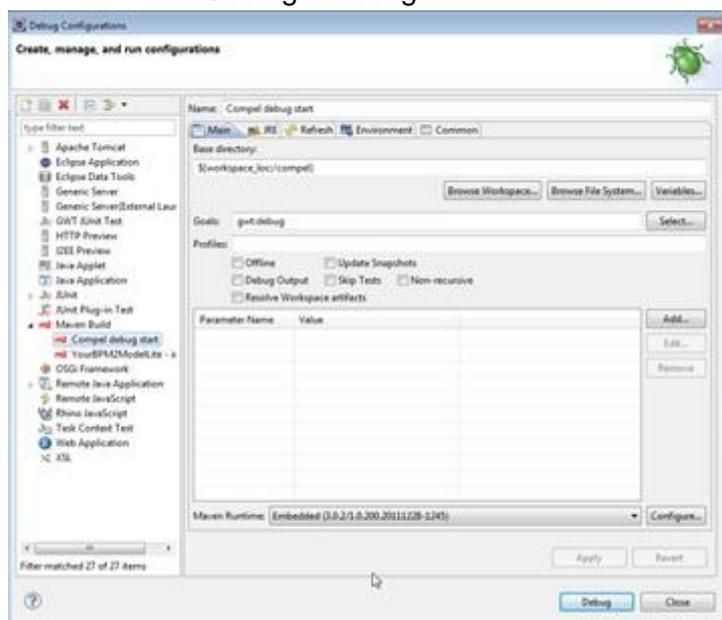
The following modules are recommended for COMPEL development and debugging:

- Activiti-explorer: `$YOUR_BPM/BPMModel/activiti/activiti-explorer`

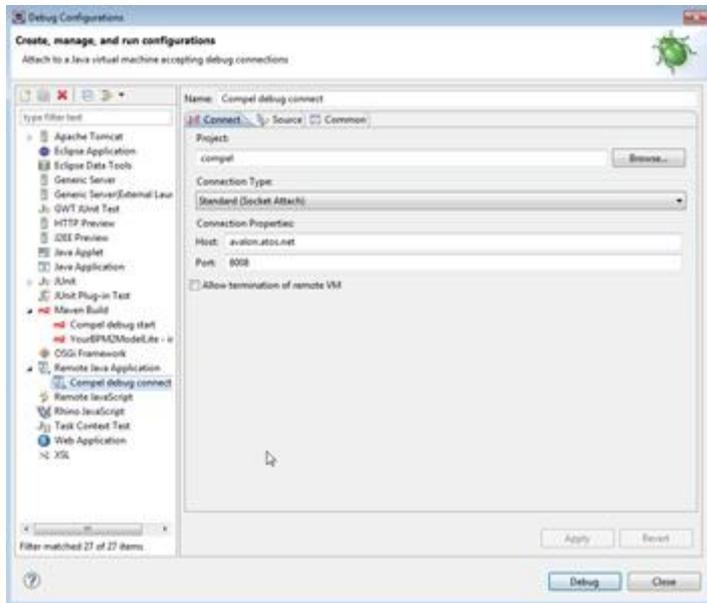
- Activiti-webapp-explorer2: \$YOUR_BPM/BPMModel/activiti/activiti-webapp-explorer2
- BPMModel: \$YOUR_BPM/BPMModel
- BPMN20Manager: \$YOUR_BPM/BPMN20/BPMN20Manager
- BPMN20ModelLite: \$YOUR_BPM/BPMN20/BPMN20ModelLite
- BPMN20Module: \$YOUR_BPM/BPMN20/BPMN20Module
- BPMN20Service: \$YOUR_BPM/BPMN20/BPMN20Service
- Compel: \$YOUR_BPM/Compel
- Design Time Composer: \$YOUR_BPM/DTC/DesignTimeComposer
- DTCCClient: \$YOUR_BPM/DTCCClient
- DTCCConfiguration: \$YOUR_BPM/DTC/DTCCConfiguarion
- KBClient: \$YOUR_BPM/KBClient
- WSDLManager: \$YOUR_BPM/WSDLManager

Eclipse debugging environment

- Create a Tomcat 6.X server configuration, following Eclipse manuals: Preferences/Server/Runtime Environments
- Create a Tomcat execution server in Servers View
 - Open server configuration (double click on tomcat server in servers view)
 - In server locations section, select Use Tomcat installation
 - In deploy path browse to \$TOMCAT_HOME/webapps
 - Open launch configuration. In Arguments tab, VM Arguments add the following JAVA options
 -Xms256m -Xmx1024m -XX:PermSize=256m -XX:MaxPermSize=512m -XX:+CMSClassUnloadingEnabled -XX:+UseConcMarkSweepGC
- We assume yourBPM required modules are already deployed in \$TOMCAT_HOME/webapps, otherwise deploy them
- Create an Eclipse debug configurations:
 - Type --> Maven build
 - Name: Compel debug start
 - Goal: gwt:debug



- Type --> Remote Java Application
- Name: Compel debug connect
- Type: Remote Java Application
- Port: 8008



- In servers view select create Tomcat execution server and launch it as debug using the icon in top right button bar
- Launch Compel debug start debug configuration
- Launch Compel debug connect debug configuration
- Access YourBPM using the URL provided by GWT Dev Mode Window
- Set breakpoints in Compel client and server code and other modules server code, including Activiti Editor.

7.4 Troubleshooting

7.4.1 Activiti database creation

[Linux] If you experience problems to authenticate postgres or activiti users into the postgresql server, Follow instructions in: <http://www.cyberciti.biz/faq/psql-fatal-ident-authentication-failed-for-user/>

```

/var/lib/postgresql/data/pg_hba.conf looks locally as:
# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all trust
# IPv4 local connections:
host all all 127.0.0.1/32 trust
# IPv6 local connections:
host all all ::1/128 trust
    
```

[Linux] If the commands to create activiti user, database and population fail with message "createuser: could not connect to database postgres: FATAL: Peer

authentication failed for user "postgres" Log as postgres user into the computer (sudo su - postgres) and issue these command and the followings without -U postgres, for instance
 createuser -e -P -E activiti

7.4.2 Tomcat

[All OS] If you get permgen memory exception when launching Tomcat.

Set JAVA_OPTS environment variable to:

```
"-Xms256m -Xmx1024m -XX:PermSize=256m -XX:MaxPermSize=512m -
XX:+CMSSClassUnloadingEnabled -XX:+UseConcMarkSweepGC"
```

7.4.3 Development Eclipse

It is required to debug Compel in Firefox. If you need a GXT DevMode plugin for Firefox, download it from <http://notepad2.blogspot.com.au/2012/10/gwt-developers-plugin-for-firefox-16.html>

Compel pom.xml is configured to debug running in own server or launched by gwt:debug, by changing value `<noServer>true</noServer>`. If set to false, Compel is launched in internal jetty.

If set to true, you should launched in an external server (i.e. in the Tomcat configuration described above)

IMPORTANT NOTE: Compel and Activiti Editor must be located in the same server (<http://<host>:<port>>) otherwise, the connection between both will not work. This means that during debugging, compel and oryx must be hosted by the sameserver, whereby `<noServer>>false</noServer>` will not work.

Launch Tomcat with increased pergem memory, for instance setting the following JAVA_OPTS:

```
JAVA_OPTS="-Xms256m -Xmx1024m -XX:PermSize=256m -
XX:MaxPermSize=512m -XX:+CMSSClassUnloadingEnabled -
XX:+UseConcMarkSweepGC"
```

If you experience RPC serialization problems such as those described in:

<http://stackoverflow.com/questions/2122798/gwt-occasional-com-google-gwt-user-client-rpc-serializationexception>

```
delete $TOMCAT_HOME/webapps/compel-1.0/ directory and deploy a new compel
war
```

7.5 Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator has to take in order 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.

7.5.1 End to End testing

To check that everything is up and running, you can follow the next steps:

- Access to the login Web page at http://IP_ADDRESS:8080/ActivitiCompelExplorer2
- Introduce the user fozzie and password fozzie, and enter in the application.
- Select the link Process-->Model Work Space
- create a "New model" and enter the name.
- Appear the Editor, integrating the Activiti Modeler and all the widgets.

7.5.2 List of Running Processes

Processes that should be working are

- java instance for the tomcat

```
bpmn@debianVM:~$ bpmn 1963 1.6 18.1 1952428 342368 pts/0 Sl 13:40 0:33 /usr/bin/java
-Djava.util.logging.config.file=/home/bpmn/apache-tomcat-6.0.35/conf/logging.properties -
Djava.awt.headless=true -Dfile.encoding=UTF-8 -server -Xms1536m -Xmx1536m -
XX:NewSize=256m -XX:MaxNewSize=256m -XX:PermSize=256m -XX:MaxPermSize=256m
-XX:+DisableExplicitGC -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -
Djava.endorsed.dirs=/home/bpmn/apache-tomcat-6.0.35/endorsed -classpath
/home/bpmn/apache-tomcat-6.0.35/bin/bootstrap.jar -Dcatalina.base=/home/bpmn/apache-
tomcat-6.0.35 -Dcatalina.home=/home/bpmn/apache-tomcat-6.0.35 -
Djava.io.tmpdir=/home/bpmn/apache-tomcat-6.0.35/temp
org.apache.catalina.startup.Bootstrap start
```

- a postgres instance for the database

```
postgres 2063 0.0 0.4 51540 9240 ? Ss 13:41 0:00 postgres: poem poem 127.0.0.1(41572)
idle
```

7.5.3 Network interfaces Up & Open

- 5432 --> For the postgres connection
- 8080 --> For the web application

7.5.4 Databases

The last step in the sanity check, once that we have identified the processes and ports, is to check the database. If we execute the following command:

```
$ psql -U activiti
password: activiti
```

It should show a message text similar to the following:

```
activiti=>
```

7.6 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.

The following sections have to be filled in with the information or an “N/A” (“Not Applicable”) where needed. Do not delete section titles in any case.

7.6.1 Resource availability

Minimal System Requirements:

RAM:2GB

Storage: 5GB

7.6.2 Remote Service Access

N/A

7.6.3 Resource consumption

- Normal parameters: 2GB memory consumption, 7GB total file system storage

7.6.4 I/O flows

- Requests to port 8080 (port defined in Apache Tomcat configuration files).

8 Service Composition - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

Disclaimer:

The partner providing this component as software asset left the related work package and can't provide further support for the software and software related deliverables.

8.1 General

The Ericsson Composition Engine implements the "Service Composition" GE and is provided as a combination of SaaS and downloadable software. A new user can register at the following address: <http://compositionex.lab.fi-ware.eu>.

- The Composition Editor is provided as a downloadable user-end installer for the Windows environment. Once a new user is registered, a link for downloading the Composition Editor installer is provided.
- The Execution Engine is provided as a service (SaaS) and can be found at the following address: <http://compositionex.lab.fi-ware.eu:8080>. No installation and administration is needed.

8.2 Composition Editor Installation and Administration

For installation, run the downloaded 'setup-ace.exe' and apply your personal preferences. Once the Composition Editor is opened a connection to the Execution Engine is preconfigured. In case the Composition Editor is used within an environment that uses a www proxy the relevant information can be configured in Menu->Edit->Preferences->General->Network Connections. For further information on using the Composition Editor please consult the [Service Composition - User and Programmer Guide](#). No installation and administration is needed within the FIWARE testbed.

8.3 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.

8.3.1 End to End testing

Once the Composition Editor is started, the 'ACE Explorer' widow should show the 'ECE4Fiware' node. Expanded, this should show 'Services' and 'Skeletons'. Both 'Services' and 'Skeletons' should be expandable and show the actual loaded services and skeletons.

8.3.2 List of Running Processes

Once the Composition Editor is started, the Windows Task Manager should show an entry in the tab 'Applications' called 'Demo Service Composition Environment - Advanced Composition Editor'. The 'Processes' tab should show the item 'ace.exe'.

8.3.3 Network interfaces Up & Open

N/A

8.3.4 Databases

N/A

8.4 Diagnosis Procedures

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

8.4.1 Resource availability

The Composition Editor requires for installation 80 MB hard disk. During run-time it requires 70 MB RAM.

8.4.2 Remote Service Access

The Composition Editor talks to the Execution Engine and the Repository. This should only be after specific user actions (load, save, import, etc).

8.4.3 Resource consumption

For RAM, any strong deviation from the figure above is deviant. For CPU only a small percentage CPU load is normal.

8.4.4 I/O flows

The Composition Editor issues only outgoing requests.

9 Mediator - Installation and Administration Guide

You can find the content of this chapter as well in the [wiki](#) of fi-ware.

9.1 Mediator Installation and Administration Guide

The purpose of this document is to describe how to install and administrate the necessary software on a server to run the Mediator.

The Mediator itself is a Java Enterprise application based on OSGi Equinox framework and Tomcat appserver, packaged in a tar.gz that includes the application servers.

Reference Administration Guides

The Mediator is based on the open source packages WSO2 ESB and Apache Camel. The user and programmer guide of these two open source technologies can be found respectively at:

[wso2-esb 4.0.0. Administration Guide](http://wso2.org/project/esb/java/4.0.0/docs/admin_guide.html)
 (http://wso2.org/project/esb/java/4.0.0/docs/admin_guide.html)
[Apache-camel 2.7.0 Manual](http://camel.apache.org/manual/camel-manual-2.7.0.pdf) (<http://camel.apache.org/manual/camel-manual-2.7.0.pdf>)

Reference Forums

[Stackoverflow](#)
[Apache-camel User Forum](#)

9.2 System Requirements

This section covers the requirements needed to install and use the Mediator.

9.2.1 Hardware Requirements

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

Resource	Requirement
CPU	1-2 cores with at least 2.4 GHZ
Physical RAM	2GB
Disk Space	30GB .

9.2.2 Operating System Support

The Mediator has been tested against the following Operating Systems:

- Ubuntu 10.04, 12.04 LTS

9.2.3 Software Requirements

In order to have the Mediator running, the following software is needed:

- Java 1.6.x - mandatory

9.3 Software Installation and Configuration

All the mandatory dependencies can be easily installed on a debian based Linux distribution using `apt-get`:

```
$ sudo apt-get install sun-java6-jre
```

The following steps will take you through the binary distribution installation on Unix/Linux systems.

1. Download the Mediator binary distribution from the repository. You can download it from one of these sources:
 - SVN repository: https://forge.fi-ware.eu/scmrepos/svn/fiware/trunk/FI-WARE/Apps/Mediator/Mediator_TI/2.2.1-SNAPSHOT
 - Fusion Forge: https://forge.fi-ware.eu/frs/?group_id=12
 1. download the four Mediator files: Mediator_TI_2.2.1-SNAPSHOT.tar.gz.1aa Mediator_TI_2.2.1-SNAPSHOT.tar.gz.1ab Mediator_TI_2.2.1-SNAPSHOT.tar.gz.1ac Mediator_TI_2.2.1-SNAPSHOT.tar.gz.1ad. These files have been created using the linux split command.
 2. join the four files, on linux you can use this comamnd: `cat Mediator_TI_2.2.1-SNAPSHOT.tar.gz.1aa Mediator_TI_2.2.1-SNAPSHOT.tar.gz.1ab Mediator_TI_2.2.1-SNAPSHOT.tar.gz.1ac Mediator_TI_2.2.1-SNAPSHOT.tar.gz.1ad > Mediator_TI_2.2.1-SNAPSHOT.tar.gz`
2. Extract the archive where you want the Mediator GE installed (e.g. into /opt)
3. Set the JAVA_HOME environment variable to your Java home using the export command or by editing /etc/profile, and add the Java /bin directory to your PATH
4. Execute the Mediator daemon script from the bin directory. e.g. `./daemon.sh start`
5. Check your Mediator instance using the URL <https://<host>:9443/carbon> which will take you to the Mediator Management Console. (Note that server start up may take time)
6. Login as "admin" using the default password "admin"

9.3.1 Application Server Configuration

It is possible to leave the Embedded Apache Tomcat Application server as is, without any further configuration. The default configuration allows:

- HTTPS connections at 9443 port for the console
- HTTP connection at 9763 port for the console (redirected to the previous one)
- HTTPS connection at 8243 for the mediation services
- HTTP connection at 8280 for the mediation services

9.4 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.

9.4.1 End to End testing

To quickly check if the Mediator is running and deployed correctly, open in your web browser the Mediator interface URL:

```
https://SERVER\_URL:9443/carbon
```

A user login page is shown.

Login with the following credentials:

```
Username: admin
```

```
Password: admin
```

The home page of the Mediator will be shown.

Click on "List" under the "Web Services" section in the left menu (in the "Main" tab, the one selected by default).

A list of Services is shown.

Check that the "fiwareSmsProxy" is shown in the Service list, among the others.

9.4.2 List of Running Processes

You can execute the following two commands to check that the Mediator is running.

Execute:

```
ps -ef | grep 'Mediator_TI'
```

It should show something similar to the following:

```
root      10287      1  0 11:36 ?          00:00:00
/opt/Mediator_TI_2.2.1-SNAPSHOT/./bin/native/wrapper-linux-x86-32
/opt/Mediator_TI_2.2.1-SNAPSHOT/./repository/conf/wrapper.conf
wrapper.syslog.ident=WSO2Carbon
wrapper.pidfile=/opt/Mediator_TI_2.2.1-SNAPSHOT/./WSO2Carbon.pid
wrapper.daemonize=TRUE
```

Then execute:

```
ps -ef | grep 'wso2'
```

It should show something similar to the following:

```
root      10289 10287  2 11:36 ?          00:00:42 /usr/lib/jvm/java-6-
sun/bin/java -Xms256m -Xmx512m -XX:MaxPermSize=256m -Dcarbon.home=.
-Djava.endorsed.dirs=lib/endorsed -Dcom.sun.management.jmxremote -
Dwso2.server.standalone=true -Djava.io.tmpdir=tmp -
Dwso2.transports.xml=repository/conf/mgt-transports.xml -
Dcarbon.registry.root=/ -Xms256m -Xmx512m -
Djava.library.path=bin/native -classpath lib/wrapper-
3.2.3.jar:bin/log4j-1.2.13.jar:bin/org.wso2.carbon.bootstrap-
3.2.0.jar:repository/conf -Dwrapper.key=UJlvP2gyHHKc122i -
```

```
Dwrapper.port=32000 -Dwrapper.jvm.port.min=31000 -
Dwrapper.jvm.port.max=31999 -Dwrapper.pid=10287 -
Dwrapper.version=3.2.3 -Dwrapper.native_library=wrapper -
Dwrapper.service=TRUE -Dwrapper.cpu.timeout=10 -Dwrapper.jvmid=1
org.tanukisoftware.wrapper.WrapperSimpleApp
org.wso2.carbon.bootstrap.Bootstrap RUN
```

9.4.3 Network interfaces Up & Open

To check the ports in use and listening, execute the command:

```
$ sudo netstat -ltp | grep '9443\|9763'
```

The expected results must be something similar to the following:

```
Active Internet connections (only servers)

tcp6      0      0 [::]:9763          [::]:*
LISTEN    5021/java

tcp6      0      0 [::]:9443          [::]:*
LISTEN    5021/java
```

9.4.4 Databases

N/A

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

The resource load of the Mediator strongly depends on the number and the rate of concurrent requests received as well as free main memory:

- minimum available main memory: 512 MB
- minimum available hard disk space: 1 GB

9.5.2 Remote Service Access

N/A

9.5.3 Resource consumption

Resource consumption depends on the load, especially on the number of concurrent requests. The main memory consumption of the WSO2 ESB application server (i.e. Tomcat) should be between 512MB and 1024MB.

For the CPU, in a normal usage only a small percentage of CPU load is normal. In case of a significant and continuous rate of request, even a CPU load above 50% is normal.

9.5.4 I/O flows

The expected I/O flow is:

- HTTP on port configured (default are: 9763 and 8280) inbound
- HTTPS, on port configured (default are 9443 and 8243), inbound