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

D.3.3.3: FI-WARE Installation and Administration Guide

Project acronym: FI-WARE
Project full title: Future Internet Core Platform
Contract No.: 285248
Strategic Objective: FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform
Project Document Number: ICT-2011-FI-285248-WP3-D.3.3.3
Project Document Date: 2014-04-30
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 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.
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. 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 number 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.
- **Business Modeler and Business Calculator** - handle the monetization of services or applications as well as their compositions/aggregations. Pricing schemes are modelled in business model definitions. The information in these business model definitions is stored and handled in the rating/charging/billing systems.
- **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 interoperate 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.org/

The following resources were used to generate this document:

D.3.3.3_Installation_and_Administration_Guide_front_page
Application Mashup - Wirecloud - Installation and Administration Guide
Light Semantic Composition - Installation and Administration Guide
Marketplace - 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 our wikis. The project is currently working on the migration of as many deliverables as possible towards the new system.

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

1.6.1 Links within this document

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

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

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

1.6.2 Figures

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

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

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

1.6.3 Sample software code

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

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

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

1.8 Keyword list


1.9 Changes History

<table>
<thead>
<tr>
<th>Release</th>
<th>Major changes description</th>
<th>Date</th>
<th>Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>First Version</td>
<td>2014-06-18</td>
<td>E-IIS</td>
</tr>
<tr>
<td>v2</td>
<td>Second Version</td>
<td>2014-06-19</td>
<td>E-IIS</td>
</tr>
<tr>
<td>v3</td>
<td>Third Version</td>
<td>2014-06-27</td>
<td>E-IIS</td>
</tr>
</tbody>
</table>

1.10 Table of 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 Application Mashup - Wirecloud - Installation and Administration Guide .................................. 13
  2.1 Introduction .............................................................................................................................. 13
  2.2 Installation ............................................................................................................................... 13
### 2.2.1 Requirements

- 2.2.2 Installing basic dependencies .................................................. 14
- 2.2.3 Installing WireCloud using pip .................................................. 15
- 2.2.4 Installing WireCloud from sources ........................................... 15
- 2.2.5 Creating a new instance of WireCloud ......................................... 15
- 2.2.6 Database installation and configuration ....................................... 16
- 2.2.7 Extra options .................................................................................. 18
- 2.2.8 Django configuration ........................................................................ 20
- 2.2.9 Advanced configurations .............................................................. 20
- 2.2.10 Running WireCloud ............................................................... 22
- 2.2.11 FAQ ......................................................................................... 24

### 2.3 Sanity check procedures

- 2.3.1 End to End testing ........................................................................ 26
- 2.3.2 List of Running Processes ............................................................. 30
- 2.3.3 Network interfaces Up & Open .................................................. 31
- 2.3.4 Databases ..................................................................................... 31

### 2.4 Diagnosis Procedures

- 2.4.1 Resource availability ....................................................................... 32
- 2.4.2 Remote Service Access ............................................................... 32
- 2.4.3 Resource consumption ................................................................... 32
- 2.4.4 I/O flows ....................................................................................... 32

### 3 Light Semantic Composition - Installation and Administration Guide

#### 3.1 Light Semantic Composition Installation and Administration Guide

#### 3.2 System Requirements

- 3.2.1 Hardware Recommendations ..................................................... 33
- 3.2.2 Software Requirements ............................................................... 33

#### 3.3 Software Installation and Configuration

- 3.3.1 Download release ........................................................................ 33
- 3.3.2 Checkout code ............................................................................... 34
- 3.3.3 Database creation .......................................................................... 34
- 3.3.4 Configuration ................................................................................ 34
- 3.3.5 Compilation and Deployment when you've check out the source code .................................................................................. 36
- 3.3.6 Deployment when you've downloaded the release ......................... 37
- 3.3.7 Execution ...................................................................................... 38

#### 3.4 Development

- 3.4.1 Environment ................................................................................ 40
- 3.4.2 Activiti Modeler (based in Oryx Editor) .......................................... 41
- 3.4.3 Eclipse ......................................................................................... 41
5.2.1 Hardware Requirements ................................................. 55
5.2.2 Operating System Support ............................................ 55
5.2.3 Software Requirements ............................................... 55
5.3 Software Installation and Configuration .............................. 56
5.3.1 Application Server Configuration .................................. 56
5.4 Sanity check procedures ................................................... 56
5.4.1 End to End testing ...................................................... 56
5.4.2 List of Running Processes ............................................. 57
5.4.3 Network interfaces Up & Open .................................... 57
5.4.4 Databases ............................................................... 58
5.5 Diagnosis Procedures ...................................................... 58
5.5.1 Resource availability .................................................. 58
5.5.2 Remote Service Access .............................................. 58
5.5.3 Resource consumption ............................................... 58
5.5.4 I/O flows ............................................................... 58

6 Registry - Installation and Administration Guide ..................... 59
6.1 Registry Installation and Administration Guide ..................... 59
6.2 System Requirements ..................................................... 59
6.2.1 Hardware Requirements ............................................. 59
6.2.2 Operating System Support ......................................... 59
6.2.3 Software Requirements ............................................. 59
6.3 Software Installation and Configuration .............................. 60
6.3.1 MongoDB Configuration ............................................ 60
6.3.2 Registry Configuration ............................................... 60
6.4 Sanity check procedures ................................................... 60
6.4.1 End to End testing ...................................................... 61
6.4.2 List of Running Processes ............................................. 62
6.4.3 Network interfaces Up & Open .................................... 62
6.4.4 Databases ............................................................... 62
6.5 Diagnosis Procedures ...................................................... 63
6.5.1 Resource availability .................................................. 63
6.5.2 Remote Service Access .............................................. 63
6.5.3 Resource consumption ............................................... 63
6.5.4 I/O flows ............................................................... 63

7 Repository - Installation and Administration Guide .................. 64
7.1 Repository Installation and Administration Guide .................. 64
7.2 System Requirements ..................................................... 64
7.2.1 Hardware Requirements ............................................. 64

D.3.3.3: FI-WARE Installation and Administration Guide 9
Future Internet Core Platform

7.2.2 Operating System Support ................................................................. 64
7.2.3 Software Requirements ................................................................. 64
7.2.4 Performance .................................................................................. 65
7.3 Software Installation and Configuration .................................................. 68
  7.3.1 Install all required software packages .............................................. 68
  7.3.2 MongoDB Configuration ............................................................. 68
  7.3.3 Application Server Configuration .................................................. 69
  7.3.4 Repository Configuration ............................................................ 69
7.4 Sanity check procedures ..................................................................... 69
  7.4.1 End to End testing ........................................................................ 69
  7.4.2 List of Running Processes ............................................................. 70
  7.4.3 Network interfaces Up & Open ..................................................... 70
  7.4.4 Databases .................................................................................... 71
7.5 Diagnosis Procedures ......................................................................... 71
  7.5.1 Resource availability ..................................................................... 71
  7.5.2 Remote Service Access ............................................................... 72
  7.5.3 Resource consumption .................................................................. 72
  7.5.4 I/O flows .................................................................................... 72
8 RSS - Installation and Administration Guide ............................................. 73
  8.1 Introduction ...................................................................................... 73
  8.2 System Requirements ....................................................................... 73
    8.2.1 Hardware Requirements ............................................................ 73
    8.2.2 Operating System Support ......................................................... 73
    8.2.3 Software Requirements ............................................................. 74
  8.3 Software Installation ......................................................................... 77
    8.3.1 RSS API .................................................................................... 77
    8.3.2 Limit Management and Balance-accumulated API ....................... 77
    8.3.3 Data Base .................................................................................. 78
    8.3.4 Pentaho Data Integration Infrastructure ..................................... 78
    8.3.5 Pentaho BI Server Infrastructure .............................................. 78
  8.4 Configuration .................................................................................... 78
    8.4.1 Pentaho BI Server Configuration .............................................. 78
    8.4.2 rss.properties Configuration file ............................................... 81
    8.4.3 database.properties Configuration file ...................................... 82
    8.4.4 oauth.properties Configuration file ........................................... 82
    8.4.5 jdbc.properties Configuration file ............................................ 83
    8.4.6 Scripts Configuration ............................................................... 83
  8.5 Sanity check Procedures ................................................................. 84

D.3.3.3: FI-WARE Installation and Administration Guide 10
10 Business Modeler - Installation and Administration Guide
10.1 Business Modeler Installation and Administration Guide
10.2 System Requirements
10.2.1 Software requirements
10.3 Software Installation and Configuration
10.3.1 Port configuration
10.3.2 Starting the NodeJS server
10.3.3 Stopping the Node.js server
10.3.4 Viewing the web application .......................................................... 113
10.4 Sanity check procedures ....................................................................... 113
  10.4.1 End to End testing ........................................................................... 113
  10.4.2 List of Running Processes .............................................................. 116
  10.4.3 Network interfaces Up & Open ....................................................... 116
  10.4.4 Databases ..................................................................................... 117
10.5 Diagnosis Procedures ........................................................................... 117
  10.5.1 Resource availability ...................................................................... 117
  10.5.2 Remote Service Access .................................................................. 118
  10.5.3 Resource consumption .................................................................. 119
  10.5.4 I/O flows ....................................................................................... 119
11 Business Calculator - Installation and Administration Guide .................... 120
  11.1 Business Calculator Installation and Administration Guide ................. 120
  11.2 System Requirements ....................................................................... 120
    11.2.1 Software requirements ................................................................. 120
  11.3 Software Installation and Configuration ............................................. 121
  11.4 Sanity check procedures .................................................................... 121
    11.4.1 End to End testing ....................................................................... 121
    11.4.2 List of Running Processes ............................................................ 123
    11.4.3 Network interfaces Up & Open .................................................... 123
    11.4.4 Databases ................................................................................... 123
  11.5 Diagnosis Procedures ....................................................................... 123
    11.5.1 Resource availability ................................................................... 123
    11.5.2 Remote Service Access ............................................................... 124
    11.5.3 Resource consumption ................................................................. 125
    11.5.4 I/O flows ..................................................................................... 125
2 Application Mashup - Wirecloud - Installation and Administration Guide

2.1 Introduction

This Installation and Administration Guide covers WireCloud versions since 0.6.0 (corresponding to FI-WARE release 3.3). Any feedback on this document is highly welcomed, including bugs, typos or things you think should be included but aren't. Please send it to the "Contact Person" email that appears in the Catalogue page for this GE.

2.2 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 the github's issue tracker of the project.

2.2.1 Requirements

This section describes all the requirements of a basic WireCloud installation. However, these dependencies are not meant to be installed manually in this step, as they will be installed throughout the documentation:

- A Database Manager (MySQL, PostgreSQL, SQLite3...)
- Python 2.6 or 2.7. Python 3 and other versions are currently not supported. Also the following python packages must be installed:
  - Django 1.5+
  - South 0.7.3+
  - lxml
  - requests 2.0.0+
  - selenium
  - django-compressor 1.2+
  - rdflib 3.2.0+
  - pytz
  - django_relatives
  - user-agents
  - regex

All these dependencies are available for Linux, Mac OS and Windows, so WireCloud should work on any of these operating systems. However, it is better to use Debian Wheezy+, CentOS 6.3+, Ubuntu 11.10+ or Mac OS X as these operating systems are actively tested.

**NOTE:** WireCloud can make use of the Marketplace, Store and Repository GEs. If you want to exploit this support, you can choose between installing these GEs or using any of the instances publicly available, for example, on FI-LAB (see the "Instances" tab of the corresponding entries at http://catalogue.fi-ware.org).
2.2.2 Installing basic dependencies

Before installing WireCloud, you will need to have some basic dependencies installed: python and pip.

**NOTE:** Although virtualenv is not required, you should install it before installing WireCloud if you intend to use it. It is highly recommended to use virtualenv (see the [using virtualenv section](#) for more info) when installing WireCloud in CentOS/RedHat as those systems usually raise problems when installing python packages using their official repositories and pip (a common case, as some packages should be updated for being compatible with WireCloud). Anyway, it is possible to install WireCloud in those systems without using virtual environments.

2.2.2.1 **Debian/Ubuntu**

```bash
$ apt-get install python python-pip
```

2.2.2.2 **CentOS/RedHat**

Python itself can be found in the official CentOS/RedHat repositories:

```bash
$ yum install python
```

Whereas pip and other packages should be installed from 3rd party repositories. The most common one is the EPEL repository (see [http://fedoraproject.org/wiki/EPEL](http://fedoraproject.org/wiki/EPEL) for instructions about how to add it). If you have such a repository, you will be able to install pip using the following command:

```bash
$ yum install python-pip
```

2.2.2.3 **Mac OS**

Python comes installed by default in Mac OS, so you don't need to install it. Pip can be installed using the following command:

```bash
$ sudo easy_install pip
```

However, we recommend you to upgrade your python installation using the [Homebrew](https://github.com/Homebrew/homebrew) tools for Mac:

```bash
$ brew install python
```

This command will install, as bonus, the pip command tool.

2.2.2.4 **Using virtualenv**

`virtualenv` is a tool to create isolated Python environments. Those Virtual Environments, are an isolated working copy of Python which allows you to work on a specific project without worry of affecting other projects.

`virtualenv` can be installed using pip:

```bash
$ pip install virtualenv
```

Once installed virtualenv, you will be able to create virtual environments using the following command:

```bash
$ virtualenv ENV
```
This will create an ENV folder for storing all the resources related to the virtual environment. To begin using the virtual environment, it needs to be activated:

```
$ source venv/bin/activate
```

You can then begin installing any new modules without affecting the system default Python or other virtual environments. If you are done working in the virtual environment for the moment, you can deactivate it:

```
$ deactivate
```

This puts you back to the system’s default Python interpreter with all its installed libraries. To delete a virtual environment, just delete its folder.

### 2.2.3 Installing WireCloud using pip

WireCloud can be easily installed using `pip`. To install WireCloud from a FI-WARE release, download the desired version from the FI-WARE PPP Public Files area.

Once downloaded, you can install it using the following command (assuming you downloaded APPS-Application-Mashup-Wirecloud-3.3.1.tar.gz):

```
$ sudo pip install APPS-Application-Mashup-Wirecloud-3.3.1.tar.gz
```

You can always install the latest version of WireCloud from PyPI using the following command:

```
$ sudo pip install wirecloud
```

### 2.2.4 Installing WireCloud from sources

The WireCloud source code is available from the [GitHub WireCloud repository](https://github.com/Wirecloud/wirecloud.git).

To get the latest development version of the code, you can choose between two options:

- 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](https://github.com/Wirecloud/wirecloud.git). Unzip it.
- Or use a [GIT](https://git-scm.com) client to get the latest development version via Git:

```
$ git clone git://github.com/Wirecloud/wirecloud.git
```

Once downloaded the source code, you can install wirecloud using the setup.py script (this step requires root privileges):

```
$ cd <path/to/source/code>/src
$ sudo python setup.py sdist
$ sudo pip install dist/wirecloud-<version>.tar.gz
```

Where `<version>` is the version of WireCloud to install.

### 2.2.5 Creating a new instance of WireCloud

Once installed WireCloud, you will have access to the wirecloud-admin script. This script is, among other things, used for deploy new instances of WireCloud. Before creating the instance, we recommend you to create a special user for managing and running WireCloud. For example, in Debian/Ubuntu:

```
$ adduser --system --group --shell /bin/bash wirecloud
```
Then, create a new instance directory using the \texttt{startproject} command. This will create a new directory containing the configuration files, also, you can add new python modules into this directory to customise your instance.

\begin{verbatim}
$ cd /opt
$ wirecloud-admin startproject wirecloud_instance
\end{verbatim}

After creating the new instance, you have to configure it choosing a database, populating it and performing final django configurations. These steps can be skiped using the \texttt{--quick-start} option. This will configure the instance to use SQLite3 with a default \textit{admin} user (password: \textit{admin}). This method is very useful for creating a WireCloud instance for testing:

\begin{verbatim}
$ cd /opt
$ wirecloud-admin startproject wirecloud_instance --quick-start
\end{verbatim}

If you make use of the \texttt{--quick-start} option, you should be able to go directly to the \textit{Running WireCloud} section.

2.2.6 Database installation and configuration

To set up the database engine, it is necessary to modify the \texttt{DATABASE} configuration setting in the instance \texttt{settings.py} file (e.g. \texttt{/opt/wirecloud_instance/wirecloud_instance/settings.py}). You can use any of the database engines supported by Django.

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

2.2.6.1 \textit{SQLite}

Setting up a SQLite database can be just accomplished within seconds by using the following parameters into the \texttt{settings.py} file:

\begin{verbatim}
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': '<dbfile>',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
    }
}
\end{verbatim}

where \texttt{<dbfile>} is the path to the database file.

Python directly comes with support for SQLite, but we recommend you to install the \texttt{pysqlite2} module as it provides a more updated driver:

\begin{verbatim}
$ sudo pip install pysqlite
\end{verbatim}

Finally, please take into account that SQLite database is \textbf{not recommended for production purposes}. It is only useful for evaluation purposes.
2.2.6.2 PostgreSQL

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

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

where `<dbname>` represents the name of the database, `<dbuser>` is the name of the user with privileges on the database and `<dbpassword>` is the password to use for authenticating the user. `<dbhost>` and `<dbport>` are the host and the port of the database server to use (leave these settings empty if the server is running on the same machine as WireCloud).

The only thing that remains is installing the python bindings for PostgreSQL:

```
$ sudo pip install psycopg2
```

Or alternatively, for Debian/Ubuntu:

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

2.2.6.2.1 Installing PostgreSQL on Debian/Ubuntu

First install the object-relational database system.

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

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

```bash
$ sudo service postgresql reload
```

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

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

### 2.2.6.2.2 Installing PostgreSQL on other platforms

Please, follow the [official PostgreSQL installation guide](https://www.postgresql.org/download/).

---

### 2.2.6.3 Database population

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

```bash
# 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:

```plaintext
...
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):

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

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

---

### 2.2.7 Extra options

Here’s a list of general settings available in WireCloud and their default values. These settings are configured in the `settings.py` file. Also, take into account that most of these settings are based on settings provided by Django (see [Django documentation](https://docs.djangoproject.com) for more info).
2.2.7.1 **ADMINS**

(Tuple, default: () [Empty tuple])

A tuple that lists people who get code error notifications. When DEBUG=False and a view raises an exception, WireCloud will email these people with the full exception information. Each member of the tuple should be a tuple of (Full name, email address). Example:

```
(('John', 'john@example.com'), ('Mary', 'mary@example.com'))
```

Note that Django will email all of these people whenever an error happens.

2.2.7.2 **ALLOW_ANONYMOUSE_USER**

(Boolean; default: True)

A boolean that turns on/off anonymous user access. Take into account that disabling anonymous access will reduce the usefulness of embedded and public workspaces as they will require users to be logged in.

2.2.7.3 **DEBUG**

(Boolean; default: False)

A boolean that turns on/off debug mode.

Never deploy a site into production with DEBUG turned on.

One of the main features of debug mode is the display of detailed error pages. If your app raises an exception when DEBUG is True, Django will display a detailed traceback, including a lot of metadata about your environment, such as all the currently defined Django settings (from settings.py).

2.2.7.4 **DEFAULT_LANGUAGE**

(String; default: "browser")

Language code to use by default (e.g. "en"). This setting also support other values: "browser", meaning "use the language detected from browser" and "default" for using the value of the LANGUAGE_CODE setting.

2.2.7.5 **LANGUAGE_CODE**

(String; default: "en-us")

A string representing the language code to use as fallback when no translation exist for a given literal to the user’s preferred language. For example, U.S. English is "en-us".

2.2.7.6 **SERVER_EMAIL**

(String; default: 'root@localhost')

The email address that error messages come from, such as those sent to ADMINS.

2.2.7.7 **THEME_ACTIVE**

(String, default: "wirecloud.defaulttheme")
A string representing the module that will be use for theming WireCloud. Current themes shipped with WireCloud are "wirecloud.defaulttheme", "wirecloud.fiwaretheme" and "wirecloud.oiltheme".

2.2.7.8  **URL_MIDDLEWARE_CLASSES**

(Dictionary; default: A middleware configuration dictionary)

A data structure containing the middleware configuration per URL group where the URL group name are the keys of the dictionary and the value should be a tuple of middleware classes to use for that group.

You should use this setting as replacement of the Django's MIDDLEWARE_CLASSES setting (See [Django's middleware documentation](https://docs.djangoproject.com/en/2.2/topics/http/middleware/))

Currently available groups are "default", "api" and "proxy". For example, if you want to add a middleware class to the "api" group, you can use the following code:

```python
URL_MIDDLEWARE_CLASSES['api'] += ('my.middleware.module.MyMiddleware',)
```

2.2.8  **Django configuration**

The `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:

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

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:

```bash
$ 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 or the default theme is changed.

2.2.9  **Advanced configurations**

2.2.9.1  **Installing the WireCloud Pub Sub add-on**

The development of the Pub Sub add-on is carried out at [github.com](https://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:

```bash
$ pip install wirecloud-pubsub
```

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

```python
INSTALLED_APPS = {
    ...,
    'wirecloud_pubsub',
    ...
}
```
As last step, add a DEFAULT_SILBOPS_BROKER setting with the URL of the broker to use:

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

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

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

### 2.2.9.2 NGSI proxy

WireCloud comes with a javascript library that allows widgets and operators to connect to NGSI-9/10 servers. This support works out of the box when installing WireCloud except for receiving notification directly to widgets and operators. To enable it WireCloud requires what is called NGSI proxy, this proxy is a facade that receives NGSI notifications and passes them to Widgets or Operators.

This NGSI proxy doesn't need to be installed in the same machine as WireCloud and can be shared with other WireCloud instances. WireCloud will use the NGSI proxy passed to the ngsi_proxy_url option of the NGSI.Connection object. This URL can be obtained from Widget/Operator preference defined in its config.xml.

You can install a NGSI proxy following those steps:

```bash
$ 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:

```bash
$ npm run start
```

### 2.2.9.3 Integration with the IdM GE

1. Create a new Application using the IdM server to use (for example: [https://account.lab.fiware.org](https://account.lab.fiware.org)).

   - Redirect URI must be: http(s)://<wirecloud_server>/complete/fiware/

2. Install the social-auth django module (e.g. pip install django-social-auth)

3. Edit settings.py:

   - Remove wirecloud.oauth2provider from INSTALLED_APPS
   - Add social_auth to INSTALLED_APPS
   - Add 'wirecloud.fiware.social_auth_backend.FiwareBackend' to AUTHENTICATION_BACKENDS. example:

```python

AUTHENTICATION_BACKENDS = (
    'wirecloud.fiware.social_auth_backend.FiwareBackend',
    'django.contrib.auth.backends.ModelBackend',
)
```

D.3.3.3: FI-WARE Installation and Administration Guide 21
Future Internet Core Platform

- Add a FIWARE_IDM_SERVER setting pointing to the IdM server to use (e.g. FIWARE_IDM_SERVER = "https://account.lab.fiware.org")
- Add FIWARE_APP_ID and FIWARE_APP_SECRET settings using the id and secret values provided by the IdM. You should end having something like this:

```
  o FIWARE_APP_ID = "43"
  o FIWARE_APP_SECRET = "a6ded8771f7438ce430dd93067a328fd282c6df8c6c793fc8225e2cf940f746e6b229158b5e3828e2716b915d2c4762a34219e1792b85e4d3cdf66d70d72840b"
```

4. Edit urls.py:
   - Replace the login endpoint:
     - Remove: url(r'^login/?$', 'django.contrib.auth.views.login', name="login"),
     - Add: url(r'^login/?$', 'wirecloud.fiware.views.login', name="login"),
   - Add social-auth url endpoints at the end of the pattern list: url(r'', include('social_auth.urls')),

5. Run "python manage syncdb --migrate; python manage.py collectstatic --noinput; python manage.py compress --force"

2.2.10 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.

2.2.10.1 Running WireCloud using the Django internal web server

Please note:
- Be aware that this way of running WireCloud should be used for evaluation/testing 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/](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.

2.2.10.2 Integrating WireCloud with Apache

If you choose to deploy WireCloud in Apache, the 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
```

Once you have installed Apache and mod_wsgi, add a VirtualHost to the Apache's configuration files. For example, in Debian and Ubuntu, you can edit the /etc/apache2/sites-available/default configuration file:

D.3.3.3: FI-WARE Installation and Administration Guide 22
<VirtualHost *:80>
  ...
  ### WireCloud ###
  WSGIPassAuthorization On

  WSGIDaemonProcess wirecloud user=<wirecloud_user> group=<wirecloud_group>
  python-path=<path_to_wirecloud>[:<path_to_virtualenv>/lib/python2.7/site-packages]

  WSGIScriptAlias / <path_to_wirecloud_wsgi.py>
  <Location />
    WSGIProcessGroup wirecloud
  </Location>

  Alias /static <path_to_wirecloud>/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>

The main configuration directive used in this file is the WSGIDaemonProcess, please visit its documentation for more info. Assuming that your WireCloud instance is available at /opt/wirecloud_instance and you don't make use of virtualenvs, you should have something similar to:
WSGIPassAuthorization On

WSGIDaemonProcess wirecloud user=wirecloud group=wirecloud python-path=/opt/wirecloud_instance
WSGIScriptAlias /opt/wirecloud_instance/wirecloud_instance/wsgi.py
<Location />
    WSGIProcessGroup wirecloud
</Location>

Alias /static /opt/wirecloud_instance/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>

Once you have the site enabled, restart Apache

# apache2ctl graceful

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

2.2.11 FAQ

2.2.11.1 *pip has problems installing lxml. What I have to do?*

See http://lxml.de/installation.html#installation for more detailed info.

For instance, in Debian and Ubuntu you probably have to install the python-dev, libxml2-dev and libxslt1-dev packages:
$ sudo apt-get install python-dev libxml2-dev libxslt1-dev

In Mac OS, remember to install XCode and its Command Line Tools. If this doesn't work and you use're using the Homebrew tools for Mac, you can try the following commands:

$ brew install libxml2
$ pip install lxml

2.2.11.2  I'm getting strange errors. Is there any way to get better info about the problem?
You can set the DEBUG setting to True

2.2.11.3  I don't remember the admin credentials. How can I recover it?
You have two options:
  - change the password of your admin user: see python manage.py changepassword
  - create a new admin user: see python manage.py createsuperuser

2.2.11.4  I get errors while running the manage.py script or when running the startproject command
If the error is similar to the following one:

```
Traceback (most recent call last):
  File "./manage.py", line 8 in <module>
      from django.core.management import execute_from_command_line
ImportError: No module named django.core.management
```

check that you python installation is correctly configured (using the python interpreter used for running WireCloud):

```
$ python
Python 2.7.6 (default, Nov 13 2013, 20:19:29)
 [GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.VERSION
(1, 5, 5, 'final', 0)
```

2.2.11.5  WireCloud server is giving 503 error responses
If you see messages in the apache log file like:

```
(13)Permission denied: mod_wsgi (pid=26962): Unable to connect to WSGI \ daemon process '<process-name>' on '/etc/httpd/logs/wsgi.26957.0.1.sock' \ after multiple attempts.
```

edit /etc/httpd/conf.d/wsgi.conf and add the following line:

```
WSGISocketPrefix /var/run/wsgi
```
See the following link for more information about this problem.

2.2.11.6 **I get the following error**

2.2.11.6.1 *Error processing proxy request: 'HTTPResponse' object has no attribute 'stream'*

Check your python requests module version.

2.2.11.6.2 *AttributeError: This StreamingHttpResponse instance has no `content` attribute. Use `streaming_content` instead.*

Remove MIDDLEWARE configuration from your settings.py file.

2.3 **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.

2.3.1 **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_1.0.3.wgt](http://conwet.fi.upm.es/docs/download/attachments/1278018/CoNWeT_weather-example_1.0.3.wgt)
  - [http://conwet.fi.upm.es/docs/download/attachments/1278018/CoNWeT_wms-viewer-geowidget_0.5.2.2.wgt](http://conwet.fi.upm.es/docs/download/attachments/1278018/CoNWeT_wms-viewer-geowidget_0.5.2.2.wgt)
  - [http://conwet.fi.upm.es/docs/download/attachments/1278018/CoNWeT_weather-mashup-example_2.0.wgt](http://conwet.fi.upm.es/docs/download/attachments/1278018/CoNWeT_weather-mashup-example_2.0.wgt)

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

1. Open a browser and type [http://computer_name_or_IP_address/login](http://computer_name_or_IP_address/login) in the address bar.

2. The following user login form should appear:
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_1.0.3.wgt` file using the "Adding widgets from packages" form.

7. The new widget should be now available on the local catalogue.
8. Repeat steps 6 and 7 using the CoNWeT_wms-viewer-geowidget_0.5.2.2.wgt and CoNWeT_weather-mashup-example_2.0.wgt files.

9. All the widgets and mashups should be now available on the local catalogue.

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

11. Click on the "New workspace" button.
12. The view should automatically change to the "Weather Example Mashup" view and widgets should appear in it.

13. Select the pin tool in the "Web Map Service" widget clicking the appropriated button as shown in the image.

14. 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.

### 2.3.2 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:

```
postgres  1631     1  0 25212  9452   0 Jul03 ?        00:00:19
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
```
2.3.3 Network interfaces Up & Open

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

```bash
$ 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
```

2.3.4 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:

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

It should show a message text similar to the following:

```
psql (9.1.4)
Type "help" for help.
wirecloud=>
```

2.4 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.

2.4.1 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.

2.4.2 Remote Service Access

N/A

2.4.3 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.

2.4.4 I/O flows

The only expected I/O flow is of type HTTP or HTTPS, on port defined in Apache Web Server configuration files.
3 Light Semantic Composition - Installation and Administration Guide

3.1 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.

3.2 System Requirements

3.2.1 Hardware Recommendations

- RAM: 4GB
- Disk: 20GB

3.2.2 Software Requirements

- Java JDK 6 (tested with version 1.6.0_32) – Note: Current version does not work with OpenJDK. (can be downloaded from [http://www.oracle.com/technetwork/java/javaee/downloads/java-ee-sdk-6u3-jdk-6u29-downloads-523388.html](http://www.oracle.com/technetwork/java/javaee/downloads/java-ee-sdk-6u3-jdk-6u29-downloads-523388.html))
- Tomcat 6.X or above (tested with version 6.0.35, current version does not work in Tomcat 7.X). (can be downloaded from [https://tomcat.apache.org/download-60.cgi](https://tomcat.apache.org/download-60.cgi))
- Postgresql 8.3.X (can be downloaded from [http://www.postgresql.org/download/](http://www.postgresql.org/download/))

3.3 Software Installation and Configuration

There are two options to make the installation. On one side you can download a binary release with war files and configuration files; in such a case it will only work for the port 80 in the application server. On the other side it is possible to retrieve the source code from svn, configure any port and build it.

3.3.1 Download release

It is possible to download the 3.3.3 release from:

3.3.2 Checkout code

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

3.3.3 Database creation

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

  $COMPEL/resources/myconfiguration.properties

3.3.4 Configuration

This option will only make sense if you’ve downloaded the source code. Configure the following file

  $COMPEL/resources/myconfiguration.properties

The parameters that can be configured with this file are the following:
- **tomcat.dir**: webapps folder from tomcat. All war files will be deployed automatically to this folder when build with maven.
- **tomcat.host**: tomcat host address. By default it’s set to localhost. **tomcat.port**: tomcat port. We change the default port to 80
- **db.username**: Database user name. Default value is activiti.
- **db.password**: Default password. Default value is activiti.
- **singlelogon.idmauthentication**: true or false. True means that the system will be configured to use the OAuth2.0 servicers from IDM to logon; admin and user roles will be defined in IDM application. False means that the system will use their own login page; admin and user roles will be defined by activiti application.

In case you have the ontologies hosted in another machine or it is not hosted in the same tomcat, please change Tomcat <host>:<port> in following configuration file:

```
$YOUR_BPM/Compel/src/main/resources/compel.properties
```

If you set the singlelogon.idmauthentication to true, you might need to change the file to configure the system to use the proper IDM:

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

There you have 4 variables set to:

- **idmauthenthication.clientid**: a number given by FiWare
- **idmauthenthication.authorizationAddress**: address of the machine where the IDM is installed
- **idmauthenthication.callbackURL**: callback that must match with FiWare. This callback has to build following the pattern of `http://<TOMCAT_HOST>:<TOMCAT_PORT>/ActivitiCompelExplorer2/CallbackAuthorizationServlet`
- **idmauthenthication.applicationKey**: a string give by FiWare.

For a better understanding of how this values have to be configured. We put here a example ho we have configured an application in the `account.testbed.fi-ware.org` machine.

The following figure shows how it is configured in FiWare.
Here we have how this values would be matched in the $YOUR_BPM\activiti\activiti-webapp-explorer2\src\main\resources\ui.properties configuration file:

```ini
idmauthenthication.clientid = 22
idmauthenthication.authorizationAddress = account.testbed.fi-ware.org
idmauthenthication.callbackURL = http://compositioned.testbed.fi-ware.eu/ActivitiCompelExplorer2/CallbackAuthorizationServlet
idmauthenthication.applicationKey = f2adcf80d6426c04b0d676066b4fb51793591da538ad337bd0ff8c589d9bb37cccb12de973c19b3cd538b55db255104c42be979838009e5a044bf6b037ea8ac0
```

The `idmauthenthication.clientid`, `idmauthenthication.applicationKey` are values given by the FiWare site. The `idmauthenthication.callbackURL` must match with the Callback URL from the site.

### 3.3.5 Compilation and Deployment when you've check out the source code

To compile the modules:

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

War files created for some modules in their target directories. To check that it has worked properly, make sure that once it has finished you have the following files in $TOMCAT_HOME/webapps directory:

- ActivitiCompelExplorer2.war
- compel-1.0.war
- DesignTimeComposer.war
- BPMN20Module.war
- openrdf-sesame.war
• openrdf-workbench.war
• AirCongo.war
• brusselAirlinesService.war
• fisrtHotelService.war
• HostelWorldService.war
• HotelFindService.war
• /ROOT/ontologies/pizza.owl
• /ROOT/ontologies/wine.owl
• /ROOT/ontologies/wp7-businessregistration-fc.wsml.rdf.xml
• /ROOT/ontologies/wp7-io.wsml.rdf.xml
• /ROOT/ontologies/Turismo.owl.xml

If any of the files doesn’t exist, refer to the Troubleshooting section, the Deployment of war and ontologies files section.

Change JAVA_OPTS environment variable to in the $TOMCAT_HOME/bin/startup.sh file adding

```bash
export JAVA_OPTS=-Xms256m -Xmx1024m -XX:PermSize=256m -XX:MaxPermSize=512m -XX:+CMSClassUnloadingEnabled -XX:+UseConcMarkSweepGC
```

Launch Tomcat:

```
$TOMCAT_HOME/bin>startup.sh
```

Wait until all war files have been deployed.

To deploy COMPEL ontologies do:

```
$COMPEL> java -jar console/target/RepositoryCreatorForSesame.jar console/target/repositoryBean.properties
```

Deploying Semantic Web Service (SWS) descriptions

```
$COMPEL> mvn install -f KBClient/pom.xml
```

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

3.3.6 Deployment when you’ve downloaded the release

War files can be found in the 3 zip files you’ve downloaded. Copy the files to the $TOMCAT_HOME/webapps directory:

• ActivitiCompelExplorer2.war
• compel-1.0.war
• DesignTimeComposer.war
• BPMN20Module.war
• openrdf-sesame.war
• openrdf-workbench.war
• AirCongo.war
• brusselAirlinesService.war
• fisrtHotelService.war
Future Internet Core Platform

- HostelWorldService.war
- HotelFindService.war
- /ROOT/ontologies/pizza.owl
- /ROOT/ontologies/wine.owl
- /ROOT/ontologies/wp7-businessregistration-fc.wsml.rdf.xml
- /ROOT/ontologies/wp7-io.wsml.rdf.xml
- /ROOT/ontologies/Turismo.owl.xml

Change JAVA_OPTS environment variable to in the $TOMCAT_HOME/bin/startup.sh file adding:

```
export JAVA_OPTS=-Xms256m -Xmx1024m -XX:PermSize=256m -XX:MaxPermSize=512m -XX:+CMSClassUnloadingEnabled -XX:+UseConcMarkSweepGC
```

Launch Tomcat:

```
$TOMCAT_HOME/bin>startup.sh
```

Wait until all war files have been deployed. Remember that in this case tomcat must be configured with port 80.

3.3.7 Execution

Access Activiti/YourBPM using Firefox browser (not Chrome) at:

```
http://<COMPEL_SERVER>:<COMPEL_PORT>/ActivitiCompelExplorer2
```

<COMPEL_SERVER> and <COMPEL_PORT> are the configured tomcat server and port.

Depending of the **singlelogon.idmauthentication** variable in $COMPEL/resources/myconfiguration.properties you will be redirected to the FiWare page where you have to login with you FiWare user.

or you are requested to login in the following page (this is the option you will have if you've installed the release)
using one of the three users in the Activiti engine that are also able to access to the environment integrated with Compel Editor.

- kermit/kermit --> Role: admin
- gonzo/gonzo --> Role: manager
- fozzie/fozzie --> Role: user

After login, the following dashboard appears:

Click on the Processes --> Model workspace
When you create a “New model” or “Edit an existing one”, YourBPM Tool, including Activiti Modeler (based in Oryx) appears in as next screen.

Click in Marketplace link and it appears a pop-up with the Marketplace search. Introduce The “logistic” keyword and click the search button. The system call the remote Marketplace GE and it shows the list of services.

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

3.4 Development

3.4.1 Environment

- Eclipse IDE for Java Developers (tested on version Indigo 3.7.2)
- Required plugins:
Future Internet Core Platform

- 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)

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:
$COMPEL/activiti/activiti-webapp-explorer2/editor/ant js.compress
Regenerate the associated war:
$COMPEL/activiti/activiti-webapp-explorer2/mvn clean install
Deploy the new ActivitiCompelExplorer2.war in Tomcat webapps directory.

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: $COMPEL/BPMModel/activiti/activiti-explorer
  - Activiti-webapp-explorer2: $COMPEL/BPMModel/activiti/activiti-webapp-explorer2
  - BPMModel: $COMPEL/BPMModel
  - BPMN20Manager: $COMPEL/BPMN20/BPMN20Manager
  - BPMN20ModelLite: $COMPEL/BPMN20/BPMN20ModelLite
  - BPMN20Module: $COMPEL/BPMN20/BPMN20Module

D.3.3.3: FIWARE Installation and Administration Guide 41
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.

### 3.5 Troubleshooting

#### 3.5.1 Activiti database creation


/var/lib/pgsql/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 ::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

3.5.2 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:+CMSClassUnloadingEnabled -XX:+UseConcMarkSweepGC"

3.5.3 Deployment of war and ontologie files
In case the files are no automatically copied to the $TOMCAT_HOME/webapps directory, you can do it manually. Here we list where they can be found.

- $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/firstHotelService.war
- $COMPEL/MockServices/HostelWorld/target/HostelWorldService.war
- $COMPEL/MockServices/HotelFind/target/HotelFindService.war

Deploy optional domain specific ontologies in $TOMCAT_HOME/webapps/ROOT/ontologies/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
3.5.4 Development Eclipse


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>](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:+CMSClassUnloadingEnabled -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](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

### 3.6 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.

#### 3.6.1 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_ADRESS/ActivitiCompelExplorer2](http://IP_ADRESS/ActivitiCompelExplorer2)
- Depending Depeding of the `singlelogon.idmauthentication` variable in `$COMPEL/resources/myconfiguration.properties` you will be redirected to the FIWare login page where you login using the FIWare user, or you are redirected to the Activi login page where you have to 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.
- Click in the Marketplace link and it appears a pop-up to search in the Marketplace GE.
- Introduce the "logistic" keyword and and click the search button.
### 3.6.2 List of Running Processes

Processes that should be working are

- java instance for the tomcat (type the command `ps -aux | grep tomcat`)

```
```

- a postgres instance for the database (type the command `ps -aux | grep postgres`)

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

### 3.6.3 Network interfaces Up & Open

- 5432 --> For the postgres connection (type the command `netstat -an | grep 5432 | grep LISTEN`, no output means that is not properly installed)

- 80 --> For the web application (type the command `netstat -an | grep 80 | grep LISTEN`, no output means that is not properly installed)

### 3.6.4 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=>
```

### 3.7 Diagnosis Procedures

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

#### 3.7.1 Resource availability

Minimal System Requirements:

- RAM:2GB
Storage: 5GB

3.7.2 Remote Service Access
N/A

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

3.7.4 I/O flows
- Requests to port 8080 (port defined in Apache Tomcat configuration files).
4 Marketplace - Installation and Administration Guide

4.1 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 relies on a SQL database. If you want to install the additional Pricing Support Component as well, follow this guide and deploy the Pricing Simulator WAR package by simply copying it into "webapp" folder of Apache Tomcat. Alternatively, use a Tomcat administration tools like its web GUI to perform the deployment.

4.1.1 Background and Detail

This Installation and Administration Guide relates to the Marketplace GE which is part of the Architecture of Applications and Services Ecosystem and Delivery Framework Applications and Services chapter. Please find more information about this Generic Enabler in the following FIWARE.OpenSpecification.Apps.Marketplace Open Specification.

4.2 System Requirements

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

4.2.1 Hardware Requirements

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

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

4.2.2 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 installation process on a Linux based System.

4.2.3 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
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
```

### 4.3.1 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 ...
...
...
...
```

### 4.3.2 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.
### 4.3.3 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

```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

```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.

### 4.4 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.

#### 4.4.1 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**

PUT the following body information:
<user username="demoCompany2"><company>demoCompany2</company><email>demo12132@sap.com</email><password>demoCompany2</password></user>

to this URL:

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

using the demo user account demo1234 (Password: demo1234).

Curl Example:

```bash
curl -v -H "Content-Type: application/xml" -X PUT --data-"@messageBody.xml" -u "demo1234:demo1234" http://[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.

### 4.4.2 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
         --pid-file=/opt/bitnami/mysql/data/ip-10-234-150-94.pid --lower-case-table-names=1
1055 ?   Sl 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
         --socket=/opt/bitnami/mysql/tmp/mysql.sock --port=3306
1100 ?   S 0:07 /opt/bitnami/java/bin/java -Djava.util.logging.config.file=/opt/bitnami/apache-tomcat/conf/logging.properties
```
4.4.3 Network interfaces Up & Open

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

```bash
$ sudo netstat -ltp
```

The expected results must be somewhat 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/mysqld.bin
tcp6       0      0 [::]:8009               [::]:* LISTEN      1100/java
```

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:

```bash
$ 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        |
```
4.5 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.

4.5.1 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

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 Tomcat application server should be between 48MB and 1024MB. These numbers can vary significantly 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 Apache Tomcat configuration files, inbound and outbound. Requests interactivity should be low.
5 Mediator - Installation and Administration Guide

5.1 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)

Reference Forums

- Stackoverflow
- Apache-camel User Forum

5.2 System Requirements

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

5.2.1 Hardware Requirements

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

<table>
<thead>
<tr>
<th>Resource</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1-2 cores with at least 2.4 GHZ</td>
</tr>
<tr>
<td>Physical RAM</td>
<td>2GB</td>
</tr>
<tr>
<td>Disk Space</td>
<td>30GB</td>
</tr>
</tbody>
</table>

5.2.2 Operating System Support

The Mediator has been tested against the following Operating Systems:

- Ubuntu 10.04, 12.04 LTS

5.2.3 Software Requirements

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

- Java 1.6.x or greater - mandatory
5.3 Software Installation and Configuration

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

1. Download the Mediator binary distribution from the repository:
   - https://forge.fi-ware.org/frs/download.php/1055/Mediator_3.2.2_patch01.tar.gz
2. Extract the archive where you want the Mediator GE installed (e.g. into /opt)
3. Create wso2 group: groupadd wso2
4. Create wso2 user in wso2 group: useradd -g wso2 wso2
5. Set wso2 as group and user for the Mediator folder and its subfolders: chown -R wso2:wso2 <Mediator base folder>
6. 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
7. Edit the <Mediator base folder>/repository/conf/wrapper.conf file and set in the property wrapper.java.command the absolute path to your Java Home /bin/java (e.g. wrapper.java.command=/usr/lib/jvm/java-6-sun/bin/java )
8. Execute the Mediator daemon script from the bin directory. e.g. ./daemon.sh start
9. 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)
10. Login as "admin" using the default password "admin"

5.3.1 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

5.4 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.

5.4.1 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://<host>: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.

5.4.2 List of Running Processes

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

Execute:

```
pgrep -lf Mediator.*wrapper
```

It should show something similar to the following:

```
root 10287 1 0 11:36 ? 00:00:00 /opt/Mediator_3.2.2_patch01./bin/native/wrapper-linux-x86-32
/opt/Mediator_3.2.2_patch01./repository/conf/wrapper.conf wrapper.syslog.ident=WSO2Carbon
wrapper.pidfile=/opt/Mediator_3.2.2_patch01./WSO2Carbon.pid wrapper.daemonize=TRUE
```

Then execute:

```
pgrep -lf wso2.*RUN
```

It should show something similar to the following:

```
```

5.4.3 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
```
5.4.4 Databases
N/A

5.5 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.

5.5.1 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

5.5.2 Remote Service Access

N/A

5.5.3 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 continous rate of request, even a CPU load above 50% is normal.

5.5.4 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
6 Registry - Installation and Administration Guide

6.1 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.

6.1.1.1 Background and Detail

This Installation and Administration Guide relates to the Registry GE which is part of the Applications and Services chapter. Please find more information about this Generic Enabler in the following Open Specification.

6.2 System Requirements

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

6.2.1 Hardware Requirements

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

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

6.2.2 Operating System Support

The Registry has been tested against the following Operating Systems:

- Ubuntu 13.10.11, 12.04 LTS

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

6.2.3 Software Requirements

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

- MongoDB 2.x - mandatory
- Nodejs 0.10.26 or higher - mandatory
- Registry Software - mandatory
- NPM 1.4.3 - 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)
6.3 Software Installation and Configuration

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

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

Depending on the operating system version there might only be older versions in the standard repository. The most recent version of node can be obtained from the maintainers repository (see https://github.com/joyent/node/wiki/Installing-Node.js-via-package-manager).

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

```
$ npm install https://github.com/service-business-framework/Registry-RI/tarball/master
```

6.3.1 MongoDB Configuration

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

```
$ mongo
MongoDB shell version: 2.4.6 connecting to: test
> 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`

6.3.2 Registry Configuration

By default it is not necessary to change anything. In case of non standard installations, the Registry is using a number of environment variables:

- `$VMC_APP_HOST` - the host name of the server running the Registry
- `$VMC_APP_PORT` - the port number of the service (default: 5000)
- `http_proxy` - system proxy settings
- `$IDServerHostname` - the host name of the ID Management Service
- `$IDServerPort` - the port number of the ID Management Service (default: 443)

6.4 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
6.4.1 End to End testing

From the directory where the registry is installed change to node_modules/ longitude and start the Registry server:

```
node app.js
```

To quickly check if the application is running open

```
http://localhost:5000/
```

in your web browser. If you see a directory page with the message "Unauthorized", 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. The Registry - User and Programmer Guide gives some examples of valid requests. For testing purpose use the bearer token "123456789".

E.g.:

```
```

with "test.json" containing:

```json
{
    "type": "Person",
    "firstName": "Joe",
    "lastName": "Random"
}
```

You can check whether a resource is in the registry also with mongo:

```
> mongo
MongoDB shell version: 2.4.6
connecting to: test

> use registry
switched to db registry
> show collections
system.indexes
/registry/de/acme
> db["/registry/de/acme"].find()
```
6.4.2 List of Running Processes

You can execute the following command to check that the Registry and MongoDB database are running:

```bash
$ ps ax | grep -i 'node \| mongo'
```

It should show something similar to the following:

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

6.4.3 Network interfaces Up & Open

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

```bash
$ 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:28017 *:* LISTEN 646/mongod
udp 0 0 *:* 0:* LISTEN 5670/node
```

6.4.4 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
```

6.5 Diagnosis Procedures

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

6.5.1 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

6.5.2 Remote Service Access

N/A

6.5.3 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.

6.5.4 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.
7 Repository - Installation and Administration Guide

7.1 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.

7.1.1 Background and Detail

This Installation and Administration Guide relates to the Repository GE which is part of the Applications and Services chapter. Please find more information about this Generic Enabler in the following Open Specification.

7.2 System Requirements

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

7.2.1 Hardware Requirements

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

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

7.2.2 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.

7.2.3 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)
7.2.4 Performance

As part of the testing, we evaluated how the Repository GE performs under load. The tests were performed using SoPeCo, a framework for systematic performance evaluations of software systems, developed within the Tools Chapter WP9.

During the tests, the following hardware was used for running the Repository component:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Core 2 (6600), 2.40 GHz</td>
</tr>
<tr>
<td>Physical RAM</td>
<td>4 GB</td>
</tr>
<tr>
<td>Disk Space</td>
<td>200 GB</td>
</tr>
</tbody>
</table>

**Operating System**: Ubuntu, Linux kernel version 3.5.0-23

7.2.4.1.1 Process & Setup

The tests have been conducted with a focus on request response times. Here we are interested in understanding and optimizing the request processing. In addition we are also interested in the performance over time in order to understand how stable response times are during the run of service.

Conducting the tests, we started with a dedicated Repository GE instance on the machine outlined above. Next we created a SoPeCo Measurement Controller (MEController). The MEController has multiple roles and can be seen as the interface between the SoPeCo framework and the system under test. SoPeCo uses the MEController to coordinate the test by (i) acting as a workload driver and (ii) collecting the measurement results.
7.2.4.1.2  Test 1: Retrieving meta information of a single resource

For a detailed description on how to install SoPeCo and how to write a MEController, please refer to the documentation.

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

Further analyzing the response times we looked at the performance over time for the HTML and the XML output format. We chose these two, as they represent two extremes with regards to implementation complexity. The HTML output format is an optimized custom implementation that simply assembles the result string. The XML output on the other hand is using a generic external library which comes with overhead and more room for external factors like Garbage Collection etc. Below you see a plot for the response times for each of the 1000 calls. The first chart is for the HTML output format, while the second one is for XML.
Repository GE request response times

Repository GE request response times
7.2.4.1.3 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).

7.3 Software Installation and Configuration

7.3.1 Install all required software packages

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

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

7.3.2 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.

```bash
$ 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. mongod.conf
   1. Where to store the data.

   dbpath=/var/lib/mongodb

7.3.3 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.

7.3.4 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.

7.4 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.

7.4.1 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)

### 7.4.2 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 ? S 0:12 /opt/bitnami/java/bin/java -Djava.util.logging.config.file=/opt/bitnami/apache-tomcat/conf/logging.properties

-Djavax.util.logging.manager=org.apache.juli.ClassLoaderLogManager

-XX:MaxPermSize=512m -Xms256m -Xmx512m -Djavax.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
```

### 7.4.3 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:
### D.3.3: FIWARE Installation and Administration Guide

<table>
<thead>
<tr>
<th>Active Internet connections (only servers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proto</td>
</tr>
<tr>
<td>tcp</td>
</tr>
<tr>
<td>tcp</td>
</tr>
<tr>
<td>tcp6</td>
</tr>
<tr>
<td>PID/Program name</td>
</tr>
<tr>
<td>tcp</td>
</tr>
<tr>
<td>tcp</td>
</tr>
<tr>
<td>tcp6</td>
</tr>
</tbody>
</table>

#### 7.4.4 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:

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

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

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

show dbs

local   (empty)
test    4.3625GB
```

#### 7.5 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.

##### 7.5.1 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
7.5.2 Remote Service Access
N/A

7.5.3 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.

7.5.4 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.
8 RSS - Installation and Administration Guide

8.1 Introduction
This section explains the process to install the RSS GE (Revenue Settlement and Sharing Generic Enabler). The following services are offered:

- REST service to receive CDRs (Charging Data Records)
- REST service to control the expenditure limits for users
- A Web Interface service for RSS administrators
  - Management of settlement process, files and transactions
  - Management of RS models
  - Management of stores and application providers
  - Access to graphical reports
- A Web Interface service for Store administrators
  - Management of their settlement process, files and transactions
  - Management of their RS models
  - Management of their application providers
  - Access to their graphical reports

This is its Installation and Administration Guide.

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

8.2.1 Hardware Requirements
The following table contains the minimum resource requirements for running the Revenue Settlement and Sharing system:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2 cores with at least 2.0 GHZ</td>
</tr>
<tr>
<td>Physical RAM</td>
<td>2GB</td>
</tr>
<tr>
<td>Disk Space</td>
<td>4GB The actual size will depend on the amount of logging and configuration of the Application.</td>
</tr>
</tbody>
</table>

8.2.2 Operating System Support
The Application has been tested on Ubuntu 12.04.2 LTS (GNU/Linux 3.2.0-29-generic x86_64).
This Installation Guide describes the installation process as based on mentioned operating system.
8.2.3 Software Requirements

- Oracle Java JDK 6
- Apache Tomcat 6
- Pentaho Data Integration 4.40
- Pentaho Business Intelligence (BI) Server 4.8.0
- MySQL JDBC Connector
- MySQL server 5.5

8.2.3.1 Java JDK 6 installation

To do it, use the application manager installer `apt-get` from linux, using the following commands:

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:eugenesan/java
sudo apt-get update
sudo apt-get install openjdk-6-jdk
```

To check if it's installed:

```
java -version
```

If everything ok, the version of Java software just installed will be shown.

8.2.3.2 Apache Tomcat 6 installation

Download last version of Apache Tomcat 6 (available from https://tomcat.apache.org/download-60.cgi):

```
```

Extract it:

```
tar xvzf apache-tomcat-6.0.39.tar.gz
```

Move tomcat folder to a permanent location:

```
sudo mv apache-tomcat-6.0.39 /opt/
```

Edit your bashrc file:

```
vi ~/.bashrc
```

Add the following line:

```
export JAVA_HOME=/usr/lib/jvm/java-6-oracle
```

To make tomcat a service:

```
sudo vi /etc/init.d/tomcat
```

Paste the following:

```
# Tomcat auto-start
#
# description: Auto-starts tomcat
```
# processname: tomcat
# pidfile: /var/run/tomcat.pid

export JAVA_HOME=/usr/lib/jvm/java-6-oracle

case $1 in
  start)
    sh /opt/apache-tomcat-6.0.39/bin/startup.sh
    ;;
  stop)
    sh /opt/apache-tomcat-6.0.39/bin/shutdown.sh
    ;;
  restart)
    sh /opt/apache-tomcat-6.0.39/bin/shutdown.sh
    sh /opt/apache-tomcat-6.0.39/bin/startup.sh
    ;;
esac
exit 0

You'll need to make the script executable by running the chmod command:

```
sudo chmod 755 /etc/init.d/tomcat
```

To config admin user:

```
vi /opt/apache-tomcat-6.0.39/conf/tomcat-users.xml
```

Add the following lines:

```
<role rolename="manager-gui"/>

<user username="<admin_user>" password="<admin_password>" roles="manager-gui"/>
```

where `<admin_user>` is the user name for the tomcat administrator and the `<admin_password>` the administration password.

Restart tomcat:

```
service tomcat restart
```

8.2.3.3 **Pentaho Data Integration (PDI) and MySQL Connector installation**

Download PDI, rename the file downloaded if needed, extract it and move to a permanent location:

```
mv download pdi-ce-4.4.0-stable.tar.gz
tar xvzf pdi-ce-4.4.0-stable.tar.gz
mkdir /opt/pentaho
mv data-integration /opt/pentaho/
```
Download MySQL Connector, rename the file if needed, extract it and move the jar file to libext directory inside PDI:

```bash
mv index.html mysql-connector-java-5.1.30.tar.gz
tar xvzf mysql-connector-java-5.1.30.tar.gz
cp mysql-connector-java-5.1.30/mysql-connector-java-5.1.30-bin.jar
/opt/pentaho/data-integration/libext/
```

### 8.2.3.4 Pentaho BI Server and MySQL Connector installation

Download Pentaho BI Server, rename the file downloaded if needed, extract it and move to a permanent location:

```bash
mv download biserver-ce-4.8.0-stable.tar.gz
tar xvzf biserver-ce-4.8.0-stable.tar.gz
mv biserver-ce /opt/pentaho/
mv administration-console /opt/pentaho/
```

Download MySQL Connector (if done in the step before, use the file downloaded), rename the file if needed, extract it and move the jar file to the required directories inside Pentaho BI Server.

```bash
cp mysql-connector-java-5.1.30/mysql-connector-java-5.1.30-bin.jar
/opt/pentaho/administration-console/jdbc
```

Start the server with the same user as the home directory where the configuration will be placed (fiware user for this manual):

```bash
cd /opt/pentaho/biserver-ce
./start-pentaho.sh
```

### 8.2.3.5 Apache Tomcat and Pentaho BI Server Connector Port

Due to Pentaho BI Server starts up another Apache Tomcat, it is important configuration issue, if the two components are started up in the same machine. Apache Tomcat port by default (in Pentaho BI Sever too) is port 8080. So, if this port is not changed, it will cause a conflict between the two components. In order to avoid the problem, one port must be change. To do it, it should be done, following the tips from:

- **Pentaho BI Server**: Section 9 (Rename your Server (and Port if necessary)) from [http://interestingittips.wordpress.com/tag/bi-production-server-modifications/](http://interestingittips.wordpress.com/tag/bi-production-server-modifications/)

In particular for the current installation, it have been followed the tips from the last link.
8.2.3.6 **MySQL installation**

Install MySQL server:

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

Edit `/etc/mysql/my.cnf` file and change the property `bind-address` from `localhost` to `0.0.0.0` so MySQL listens to all interfaces:

```
bind-address = 0.0.0.0
```

Restart MySQL server:

```
sudo service mysql restart
```

Grant all privileges to user root from any IP so you can create a database later. To do it, log in into MySQL database:

```
mysql -h localhost -u root -p
```

Mysql prompt will appear. Then execute the following command:

```
grant all privileges on *.* to 'root'@'%' identified by 'root';
```

After this command, execute quit command to return to Linux shell again.

---

8.3 **Software Installation**

The software is delivered in 5 files:

- `fiware-rss_<release_number>.war` : containing the RSS API and back end interface
- `expenditureLimit_<release_number>.war` : containing the Limit management API
- `settlement_dump_<release_number>.sql` : containing the database basic information
- `pentaho_files_<release_number>.zip` : containing the Pentaho Data Integration infrastructure
- `fiware_reports_files_<release_number>.zip` : containing the Pentaho BI Infrastructure for reports

Where `<release_number>` is compound of 3 numbers “x.y.z” (x: major release, y: minor release, z: sprint number).

It must be taken into account that, for a proper installation, before deploying the .war files, their names must be changed to the ones without version number (in example: `fiware-rss.war` or `expenditureLimit.war`).

For these steps, a functioning machine with the software requirements is assumed, as well as the same user permission as the required software uses.

It's recommended to use the same paths as in this guide. If you want to use different paths, you'll have to change all configuration files and scripts.

8.3.1 **RSS API**

Go to Tomcat Web Application Manager with admin user and deploy the `fiware-rss.war`.

8.3.2 **Limit Management and Balance-accumulated API**

Go to Tomcat Web Application Manager with admin user and deploy the `expenditureLimit.war`. 
8.3.3 Data Base

Enter mysql with root user.

```
mysql -u root
```

Create Database with the desired name <db_name>

```
cREATE DATABASE <db_name>;
```

Exit mysqladmin and restore the database dump file. settlement_dump.sql

```
mysql --user=<admin_user> --password=<admin_password> <db_name> < /Path/To/settlement_dump.sql
```

8.3.4 Pentaho Data Integration Infrastructure

Unzip the file in the user home directory (/home/fiware/ in this example). This zip file contains the scripts used to calculate the revenue sharing.

```
unzip pentaho_files.zip
```

It is needed to give execution permission to these sh files:

```
chmod +x *.sh
```

It is also needed to configure user and password of local mysql inside run_report_all.sh script.

```
mysql -u<database_user> -p<database_password>
```

8.3.5 Pentaho BI Server Infrastructure

Unzip the file in /opt/pentaho/biserver-ce/pentaho-solutions/.

```
unzip fiware_reports_files.zip
```

8.4 Configuration

8.4.1 Pentaho BI Server Configuration

To configure some aspects of Pentaho BI Server, it is needed to use the Administration Pentaho Console.

To start it, it is needed to execute the following script:

```
cd /opt/pentaho/administration-console
./start-pac.sh
```

In a browser put the URL corresponding to the Pentaho installation: http://host:8099/. The user will be asked for a login and password (by default user:admin password:password). After filling them in, it will be shown the main administration console page:
Choose "Administration" option on the left panel and "Database Connections" on top options. The following page will be shown:

Then, click plus button and the following window will be shown:

Add the following data to create the two needed datasources:

1. **Name:**
2. **Driver Class:** `org.hsqldb.jdbcDriver`
3. **User Name:**
4. **Password:**
5. **URL:**
6. **Test:**

---

D.3.3.3: FI-WARE Installation and Administration Guide 79
Name: settlement
Driver class: com.mysql.jdbc.Driver
User: <database_user>
Password: <database_password>
URL: jdbc:mysql://localhost:3306/<database_name>
Name: dbe
Driver class: com.mysql.jdbc.Driver
User: <database_user>
Password: <database_password>
URL: jdbc:mysql://localhost:3306/<database_name>

where the values of the parameter in example are:
- <database_name>: FIWARE_SETTLEMENT
- <database_user>: root
- <database_password>: root

Pentaho BI Server administrator can also add new users able to access the Pentaho Reports, as new Stores for instance. To do it:

Choose "Administration" option on the left panel and "Users and Roles" on top options. The following page will be shown:

Then click plus button and the following window will be shown:
Fill in the proper information and click OK button to create the user. After that, it is necessary to assign roles to the user just created. This is important, because the information a user is able to manage depends on the roles the user has got. To do it, you had to click on plus button of the Assign Roles part of user information. The following windows will be shown:

Here, it is possible to assign different roles to the user. For example, it is important to add Authenticated role that allows the user to see and generate reports.

After saving the configuration, stop the administration console (Ctrl + C)

8.4.2 rss.properties Configuration file

This file is located at: {Apache Tomcat installation}/webapps/fiware-rss/WEB-INF/classes/rss.properties

This file contains the properties that specify where the sh files are located.

Example:

```plaintext
cdrfilepath=/home/fiware/bills/cdr_fixed/
reportsPath=/home/fiware/pentaho/reports/
cdrToDBScript=/home/fiware/pentaho/insert_cdrs.sh
settlementScript=/home/fiware/pentaho/run_report.sh
pentahoReportsUrl=http://host:port/pentaho
```

The description of each property is:
- `cdrfilepath` is the path where CDR files will be saved.
• reporsPath is the path where settlement files will be stored after generation.
• cdrToDBScript is the path of the script that launches the insertion of the CDRs into the database.
• settlementScript is the path of the script that generate de settlement files for each of the providers.
• pentahoReportsUrl is the path URL where Pentaho BI Server is installed.

8.4.3 database.properties Configuration file
This file is located at: {Apache Tomcat installation}/webapps/expenditureLimit /WEB-INF/classes/database.properties and in {Apache Tomcat installation}/webapps/fiware-rss /WEB-INF/classes/database.properties
This file contains the properties that give access to the database.

Example:

```
database.url=jdbc:mysql://localhost:3306/FIWARE_SETTLEMENT
database.username=<database_user>
database.password=<database_password>
database.driverClassName=com.mysql.jdbc.Driver
```

The description of each property is:
- database.url is the location of the database.
- database.username is the username to access to the database.
- database.password is the password to access to the database.
- database.driverClassName is the driver user in the connection to the database.

8.4.4 oauth.properties Configuration file
This file is located at: {Apache Tomcat installation}/webapps/expenditureLimit /WEB-INF/classes/oauth.properties and in {Apache Tomcat installation}/webapps/fiware-rss /WEB-INF/classes/oauth.properties
As the RSS generic enabler delegates the authentication to the Identity Manager (IDM) enabler, you need to setup the properties needed to access the accounting system. This file contains them.

Example:

```
############## IDM configuration ################
config.externalLogin=Y
config.useOauth=Y
config.baseUrl=https://account.lab.fi-ware.org
config.client_id=client_id
config.client_secret=client_secret
config.callbackURL=http://host:port/fiware-rss/settlement/settlement.html
config.authorizeUrl=/oauth2/authorize
```

D.3.3.3: FI-WARE Installation and Administration Guide 82
config.accessTokenUrl =/token
config.userInfoUrl =/user?access_token=
config.grantedRole=Provider
config.getApplications=/applications.json?actor_id=

The description of each property is:

- config.externalLogin says if IDM external login will be used.
- config.useOauth says if security authentication header will be needed in API request.
- config.baseUrl is the base URL where IDM is located.
- config.client_id is the client id that is configured inside IDM account.
- config.client_secret is the client secret that is configured inside IDM account.
- config.callbackURL is the back URL, where IDM will redirect the request after the login (this URL must match the one configure inside IDM account).
- config.authorizeUrl is the path of where IDM expect authorize request.
- config.accessTokenUrl is the path of where IDM expect access token request.
- config.userInfoUrl is the path where IDM gives information regarding an user id.
- config.grandtedRole is the type of role granted to administrate RSS private interface.
- config.getApplicattions is the path where IDM gives information of the applications granted to an user.

8.4.5 jdbc.properties Configuration file

This file contains the properties of the datasources that Pentaho BI Server will use. It is located at: /home/fiware/pentaho/simple-jndi/jdbc.properties

Add the following lines to the file:

```java
settlement/type=javax.sql.DataSource
settlement/driver=com.mysql.jdbc.Driver
settlement/url=jdbc:mysql://localhost:3306/FIWARE_SETTLEMENT
settlement/user=<database_user>
settlement/password=<database_password>
```

If you've created the database with other name you must change FIWARE_SETTLEMENT with the name of the database.

8.4.6 Scripts Configuration

There are different scripts used by RSS System that launch the Pentaho Data Integration transformations that must be configured before being used. These are

1) insert_cdrs.sh

In this script configure the correct path were scripts are stored:

cd /home/fiware/pentaho/

Write the path where the script is. Usually /home/fiware/pentaho.

In the same script, check the following line:
Put the path where Pentaho Data Integration is installed.

2) run_report.sh

Pentaho Data Integration installation path must be corrected in this script if it's different to the default path: /opt/pentaho/data-integration/

3) run_report_all.sh

This script is used to launch manually the generation of the corresponding reports. Here, it is necessary to configure the following:

- The place where the scripts are stored as it was done in the first script.
- Pentaho Data Integration installation path must be corrected in this script if it's different to the default path: /opt/pentaho/data-integration/
- It retrieves information from local database. So, it is necessary to configure login and password to access it.

After this configuration the Sanity check procedures and the Diagnosis procedures can be followed for initial tests.

8.5 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. It is a preliminary set of tests to ensure that obvious or basic malfunctioning are fixed before proceeding to unit tests, integration tests and user validation.

The following steps have to be performed to ensure the Application is working correctly. It is assumed that all installation steps have been followed correctly. If there is something not working correctly, it is recommended to consult the diagnosis procedures.

8.5.1 End to End testing

In a browser, you can go to: http://host:port and the Tomcat start page should appear.

Go to http://host:port/fiware-rss/rss. The following text should appear:

Available RESTful services:
Endpoint address: http://host:port/fiware-rss/rss/cdrs
WADL: http://host:port/fiware-rss/rss/cdrs?_wadl

Go to http://host:port/expenditureLimit/. The following text should appear:

Available RESTful services:
Endpoint address: http://host:port/expenditureLimit/limitManagement
Endpoint address: http://host:port/expenditureLimit/balanceAccumulated
WADL: http://host:port/expenditureLimit/balanceAccumulated? wadl

Go to http://host:port/pentaho/Login. A login web page will appear. Use the login data according your credentials. By default:
Use the login data:
*login: joe
*password: password

In the main page, in the left side, a folder called "FI-WARE reports" bellow browse title will appear. Click over the folder. Below the title “files” the available reports will appear. Click again over one of the available reports; this will appear in the main frame.

8.5.2 List of Running Processes

Make sure that the Tomcat server and BI server Tomcat are up and running. This can be verified by issuing the command:

```
ps -fe | grep tomcat
```

At least, the following list of processes, with your installation corresponding parameters, must appear:

```
```

```
```

8.5.3 Network interfaces Up & Open

Check the ports used by the applications. By default the ports used by Tomcat, Pentaho BI Server and Mysql are: 8080, 8081 and 3306 respectively.

- Make sure that Tomcat is listening in the correct port:

```
etstat -na | grep <tomcat_port>
```

tcp6 0 0 ::::<tomcat_port> ::::* LISTEN

- Make sure that Pentaho BI server Tomcat is listening in the correct port:

```
netstat -na | grep <pentaho_port>
```

tcp6 0 0 ::::<pentaho_port> ::::* LISTEN

- Make sure the MySQL server is listening in the correct port and interface:

```
netstat -na | grep <mysql_port>
```

tcp 0 0 0.0.0.0:<mysql_port> 0.0.0.0:* LISTEN
8.5.4 Databases

- Make sure mysql database is running with the command:

  ```sh
service mysql status
  ``

  The result should be something like: mysql start/running, process 25680

8.6 Diagnosis Procedures

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

Apart from the tests specified in the standard sections that follow, the logs can provide relevant diagnosis information:

- The logs of the RSS API are stored in {Apache Tomcat Installation}/logs/fiware-rss/main.log
  - After testing the REST service, with the command of the previous section, you can see if there is an error in the log.
  - The developer team must be consulted in that case.

- The logs of the Balance Accumulate and Limit Management API are stored in {Apache Tomcat Installation}/logs/expendLimit/expendLimit.log
  - After testing the REST service, with the command of the previous section, you can see if there is an error in the log.
  - The developer team must be consulted in that case.

- The logs of the Pentaho transformations are stored in /home/fiware/pentaho/logs if the paths used were the same as in this guide.
  - These logs can be checked for errors too.

8.6.1 Resource availability

The application will run correctly if the system adheres to the minimal requirements.

8.6.2 Remote Service Access

The REST RSS API is an HTTP service located on the port 8080 by default.

The following example command can be issued to test if it is working correctly.

```sh
curl -X POST -H 'Content-Type: application/xml' -d '<cdrs></cdrs>' http://host:port/fiware-rss/rss/cdrs
```

The following output should be returned

CDR saved

The following example command can be issued to test if it the balanceAccumulated API is working correctly (the command must be executed in one line without CR).

```sh
curl -X POST -H 'Content-Type: application/json'
-d '{"service": "fiware","appProvider": "conwet","currency": "EUR","chargeType": "C","amount": 1}'
```

D.3.3.3: FI-WARE Installation and Administration Guide 86
The following output should be returned

```
{"service":"fiware","appProvider":"conwet","accums":[{"type":"daily","currency":"EUR","nextPeriodStartDateTime":"2013-11-13T23:00:00.000Z","expensedAmount":0.0000},{"type":"monthly","currency":"EUR","nextPeriodStartDateTime":"2013-11-30T23:00:00.000Z","expensedAmount":0.0000}]
```

8.6.3 Resource consumption

There are three main processes consuming resources:

- Mysql Server
- 2 Apache Tomcat servers (for APIs and Pentaho)

Resource consumption strongly depends on the load, especially on the number of concurrent transactions and in the number of concurrent requests by administrators. So, the expected resource consumption for these processes is quite low.

8.6.4 I/O flows

N/A
9 Store - W-Store - Installation and Administration Guide

9.1 Introduction
This Installation and Administration Guide covers WStore versions since 0.3 (corresponding to FI-WARE release 3.3.3). Any feedback on this document is highly welcomed, including bugs, typos or things you think should be included but aren't. Please send it to the “Contact Person” email that appears in the Catalogue page for this GE.

9.2 Installation
This page contains the Installation and Administration guide of WStore, a reference implementation of the Store GE.

9.2.1 Requirements
This section describes all the requirements of a basic WStore installation. However, these dependencies are not meant to be installed manually in this step, as they will be installed throughout the documentation:

- A Web Server (i.e Apache)
- MongoDB
- Python 2.5, 2.6 or 2.7. Python 3 and other versions are not supported.
- Django nonrel 1.3 or 1.4
- djangotoolbox
- django_mongodb_engine
- lxml
- rdflib 3.2.0+
- rdflib-jsonld
- Pymongo
- Whoosh
- paypalpy
- django-crontab
- django-social-auth
- wkhtmltopdf

All this dependences are available for Linux and Mac OS, so WStore can work on this operative systems. However, it is better to use Debian Wheezy or Ubuntu 11.10+ as these operating systems are actively tested.

NOTE: WStore can make use of the Marketplace and RSS GEs, and requires a Repository GE. If you want to exploit this support, you can choose between installing these GEs or using any of the instances publicly available, for example, on FI-LAB (see the "Instances" tab of the corresponding entries at http://catalogue.fi-ware.org).
9.2.2 Getting the source code

The source code can be downloaded from WStore GitHub repository. To get the latest development version you have two options:

- Go to the WStore GitHub repository and download the zip file.
- Use a Git client to get the code via git

```
$ git clone git://github.com/conwetlab/wstore.git
```

9.2.3 Installing basic dependencies

Before installing WStore, you will need to have some basic dependencies installed: MongoDB, wkhtmltopdf, python and pip.

9.2.3.1 Debian/Ubuntu

To install Python and pip

```
$ apt-get install python python-pip
```

To install MongoDB

```
$ apt-get install mongodb
```

To install wkhtmltopdf:

```
$ apt-get install wkhtmltopdf
```

9.2.3.2 CentOS/RedHat

Python itself can be found in the official CentOS/RedHat repositories:

```
$ yum install python
```

Whereas pip and other packages should be installed from 3rd party repositories. The most common one is the EPEL repository (see http://fedoraproject.org/wiki/EPEL for adding it). If you have such a repository, you will be able to install pip using the following command:

```
$ yum install python-pip
```

MongoDB is included in the official MongoDB downloads repositories. Once the related repositories has been included (see http://docs.mongodb.org/manual/tutorial/install-mongodb-on-red-hat-centos-or-fedora-linux/) install MongoDB with the command:

```
$ yum install mongo-10gen mongo-10gen-server
```

To install wkhtmltopdf get the related binary for your system from http://code.google.com/p/wkhtmltopdf/downloads/list:

```
$ tar -xvf wkhtmltopdf-0.11.0_rcl-static-amd64.tar.bz2 -C /opt
$ cd /opt
$ mv wkhtmltopdf-amd64 wkhtmltopdf
$ mv wkhtmltopdf /usr/local/bin
```

**Note:** The command will change depending on the selected package
9.2.3.3 **Mac OS**

Python comes installed by default in Mac OS, so you don't need to install it. pip can be installed using the following command:

```
$ sudo easy_install pip
```

However, we recommend you to upgrade your python installation using the Homebrew tools for Mac:

```
$ brew install python
```

This command will install, as bonus, the pip command tool.

To install MongoDB:

```
$ brew install mongodb
```

To install wkhtmltopdf:

```
$ brew install wkhtmltopdf
```

9.2.4 **Installing WStore using Scripts**

Once basic dependencies have been resolved, it is possible to install python and Django dependencies using the provided scripts. You can execute the script setup.sh to perform the complete installation.

```
$ ./setup.sh
```

This script will create a virtual environment for the project with the corresponding packages, resolve all needed python and django dependencies, and execute a complete test in order to ensure that WStore is correctly installed. To use this script you need virtualenv2.7 and python 2.7.

```
$ pip install virtualenv
```

**NOTE:** WStore uses wkhtmltopdf for creating invoices. This software requires an X Server to work. If you do not have one, WStore will try to run Xvfb on the display :98. To install Xvfb use the following command.

```
$ apt-get install xvfb
```

**NOTE:** It is possible that the setup.sh script fails while installing lxml. See [http://lxml.de/installation.html#installation](http://lxml.de/installation.html#installation) if in trouble installing lxml. In Debian and Ubuntu you probably have to install the libpython-dev, libxml2-dev and libxslt1-dev packages ( `sudo apt-get install python-dev libxml2-dev libxslt1-dev` ).

The setup.sh script will also offers you a wizard to ease the configuration process. This wizard will generate the settings.py file for you, so if you follow the wizard, you can avoid following the Configuration section (unless you want to introduce some specific configuration). However, it is highly recommended to read the Configuration section for a better understanding of the parameters. To use this wizard, just type 'y' when asked:

```
Do you want to create an initial configuration? [y/n]:
```

```
$ y
```

First, you will be required a database name. You can introduce the name that you want:
Include a database name:

wstore_db

Then, you should include a site name. This value is up to you:

Include a site name:

store

After that, the script will ask you the domain where the Store is to going run. You must introduce a valid domain because otherwise the Store won't run.

Include a site domain:

http://host:port

Later, you will be required to introduce the name of your store instance. You are free to introduce any name that you want. This will be the name used to register your WStore instance in external components such as the Marketplace:

Include a name for your instance:

FI-WARE

Then, the script will ask you for a basic e-mail configuration. If you don't want to provide a mail configuration, just type 'n' when asked.

Do you want to include email configuration? [y/n]:

<y/n>

If you choose to include the mail configuration, you will be asked for a SMTP server, a mail address, a mail user, and the password associated to that user. This mail configuration will be used as the source address for notifications sent by email. You will be also asked for a requests mail that will be used as the destination mail for user requests asking for the provider role:

Include email smtp server endpoint:

<YOUR_SMPT_SERVER>

Include WStore email:

<YOUR_EMAIL_ADDRESS>

Include WStore email user:

<USER_NAME>

Include WStore email password:

<PASSWORD>

Include WStore provider requests email:

<REQUEST_MAIL>
Finally, you must choose the authentication method. You have two possible options: use (1) an identity manager or (2) the Django Authentication System.

Select authentication method:
1) Identity manager
2) WStore

If you choose the identity manager option, you will be asked for the identity manager endpoint, and the basic OAuth2 configuration (Client ID and Client Secret). You can avoid to introduce the basic OAuth2 configuration if you don't have the credentials at that moment. However, in order to start the Store, you need to introduce this information in the settings.py file as explained in the Configuration section. Note that for using this authentication method you must have registered your WStore instance in the identity Manager using the Callback URL explained in the configuration section of this document.

Include Identity manager endpoint:
<IDM_END_POINT>

Do you want to include OAuth2 configuration? [y/n]:
y
Include Client id:
<CLIENT_ID>

Include client secret:
<CLIENT_SECRET>

In you choose the Django Authentication System and you don't have a superuser in the selected database, you will be asked to create a new superuser in order to be able to manage the Store.

Would you like to create one now? (yes/no): yes
Username (leave blank to use 'basic'): <USERNAME>
E-mail address: <MAIL_ADDR>
Password: <PASS>
Password (again): <PASS>

If you don't the wizard to start when the script is executed, you must run the script as follow:

$ ./setup.sh --noinput

9.2.5 Manually resolving dependencies
Python and django dependencies can be easily installed using pip

$ pip install lxml "rdflib>=3.2.0" pymongo

NOTE: See http://lxml.de/installation.html#installation if in trouble installing lxml in Debian and Ubuntu you probably have to install the libpython-dev, libxml2-dev and libxslt1-dev packages ( $ sudo apt-get install python-dev libxml2-dev libxslt1-dev ).
WStore requires the Django nonrel framework ready to work with MongoDB. To install this framework in its version 1.4 as well as djangotoolbox and django_mongodb_engine for this version use the following commands:

```
$ pip install https://github.com/django-nonrel/django/archive/nonrel-1.4.zip

$ pip install https://github.com/django-nonrel/djangotoolbox/archive/toolbox-1.4.zip

$ pip install https://github.com/django-nonrel/mongodb-engine/archive/mongodb-engine-1.4-beta.zip
```

To install the rdflib plugin for json-ld format use the following command:

```
$ pip install https://github.com/RDFLib/rdflib-jsonld/archive/master.zip
```

To install the PayPal module paypalpy use the following command:

```
$ pip install https://github.com/conwetlab/paypalpy/archive/master.zip
```

WStore uses some plugins for django, to install them use the following commands:

```
$ pip install nose django-nose

$ pip install django-social-auth

$ pip install django-crontab

$ pip install whoosh
```

9.2.6 Configuration

9.2.6.1 Basic Configuration

The preliminary configuration of the database connection is included in settings.py and is ready to work using MongoDB in the default host and port, with a database called wstore_db, and without security. To modify the database connection configuration edit the DATABASES setting:

```
DATABASES = {
    'default': {
        'ENGINE': 'django_mongodb_engine',
    }
```
Using this setting is possible to change the database name and the test database name, include an user and password, and specify the host and port of MongoDB.

**Note:** The engine field cannot be changed, since WStore only works with MongoDB.

The name of the instance is included in the `STORE_NAME` setting:

```
STORE_NAME = 'WStore'
```

### 9.2.6.2 Creating the default site

WStore (and any software using `django_mongodb_engine` and `django sites framework`) requires the creation of a default `Site` model. To create the default site execute the following command including a site name and the site domain where your instance is going to run:

```
$ python manage.py createsite site_name http://host:port
```

Get the default site id:

```
$ python manage.py tellsiteid
```

Include the site id in `settings.py` updating the `SITE_ID` setting:

```
SITE_ID = u'515ab0738e05ac20b622888b'
```

### 9.2.6.3 PayPal Credentials Configuration

WStore uses PayPal to perform chargings (both charging with credit card or charging with PayPal account). In order to receive the payments, it is necessary to include the credentials of a Business PayPal account in the `settings.py` file. In this file is also possible to configure the endpoints used by PayPal, this settings contain by default the testing sandbox endpoints.

```
# Paypal credentials
PAYPAL_USER = '<PayPal_user_name>'
PAYPAL_PASSWD = '<PayPal_password>'
PAYPAL_SIGNATURE = '<PayPal_signature>'
PAYPAL_URL = 'https://api-3t.sandbox.paypal.com/nvp'
```
9.2.6.4 **Pay-per-use Cron Configuration**

WStore uses a Cron task to perform the aggregation and charging of Pay-per-use information. The periodicity of this task can be configured using the `CRONJOBS` setting of `settings.py` using the standard Cron format.

```python
CRONJOBS = [ 
    ('0 5 * * *', 'django.core.management.call_command', ['resolve_use_charging']), 
]
```

Once the Cron task has been configured, it is necessary to include it in the Cron tasks using the command:

```bash
$ python manage.py crontab add
```

It is also possible to show current jobs or remove jobs using the commands:

```bash
$ python manage.py crontab show
$ python manage.py crontab remove
```

9.2.6.5 **Email Configuration**

WStore uses some email configuration for sending notifications. To configure the source email used by WStore for sending notifications include the following settings:

```python
WSTOREMAILUSER = 'email_user'
WSTOREMAIL = 'wstore_email'
WSTOREMAILPASS = 'wstore_email_passwd'
SMTPSERVER = 'email_smtp_server'
```

It is also possible to configure a provider notification email. This email will be used by WStore as the destination email when an user requests the provider role. To set this email, include it in the `WSTOREPROVIDERREQUEST` setting:

```python
WSTOREPROVIDERREQUEST = 'provider_requ_email'
```

9.2.6.6 **Authentication Method Configuration**

WStore allows two different methods for the authentication of users. The method for users management should be selected in the initial configuration of the WStore instance. Note that WStore does not store exactly the same info for the two methods, so, changing between...
authentication methods when the system has started to be used may cause unexpected
behaviours.

9.2.6.6.1  FI-WARE Identity management

It is possible to delegate the authentication of users to the FI-WARE Identity Management system
on a FI-WARE instance. View FI-LAB info in:

- http://lab.fi-ware.eu

To do that, the first step is setting up the OILAUTH setting to True (Note that this is the default
value).

```
OILAUTH=True
```

Then configure the authentication endpoint in the settings.py file:

```
FIWARE_AUTHORIZATION_URL = 'https://fiware_idm_url/
```

Next, register WStore as an application in the identity management portal, to do that WStore uses
the following URL as as callback URL for OAuth2 authentication:

```
<host_wstore>/complete/fiware/
```

Once you have registered your WStore instance, get OAuth2 credentials needed for the
authentication of your application. You will need to create some roles in your application, one for
offering provider, other for offering customer, and a final role for developers. This roles

9.2.6.6.2  WStore Identity Management

WStore has its own authentication mechanism based on django auth. To enable WStore
authentication, set up the OILAUTH setting to False:

```
OILAUTH=False
```

For API accesses, WStore has an OAuth2 server that can be enabled by including the
oauth2provider in the INSTALLED_APPS setting.

```
INSTALLED_APPS = (  
    ...  
    wstore.oauth2provider  
    ...  
)
```

Applications can be registered in WStore using the django admin view.

9.2.6.7  Database Population

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

```
$ python manage.py syncdb
```

This command creates indexes for the different models of the database and ask if you want to
create a Django superuser. In case you are using WStore authentication, this superuser is required
in order to perform administrative tasks. If you are using FI-WARE authentication, users are taken
from the identity management system, so this superuser should not be created. In this case, users with corresponding role (Provider) will be able to perform the administrative tasks. An example of the output of this command follows:

```
...

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 'francisco'): admin
E-mail address: admin@email.com
Password: ***** (admin)
Password (again): ***** (admin)
Superuser created successfully.
...
```

### 9.2.7 Final Steps

Make sure that the directories `<wstore_path>/src/media`, `<wstore_path>/src/media/resources`, `<wstore_path>/src/media/bills`, `<wstore_path>/src/wstore/search/indexes`, and `<wstore_path>/src/wstore/social/indexes` exist, and that the server has sufficient permissions to write on them. To give permission to the server use the following commands:

```
# chgrp -R www-data <wstore_path>/src/media
# chgrp -R www-data <wstore_path>/src/wstore/search/indexes
# chgrp -R www-data <wstore_path>/src/wstore/social/indexes
# chmod g+wrX <wstore_path>/src/media
# chmod g+wrX <wstore_path>/src/wstore/search/indexes
# chmod g+wrX <wstore_path>/src/wstore/social/indexes
```

It is possible to collect all static files in WStore in a single directory using the following command and answering yes when asked.

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

Note that this is already done if you have used the script for installation.

In addition, you should serve the static files with a fast performance http server like Nginx or Apache. Django has documentation for this topic.

### 9.2.8 Running WStore

#### 9.2.8.1 Running WStore using the Django internal web server

Please note:

- Be aware that this way of running WStore should be used for evaluation purposes. Do not use it in a production environment.
NOTE: Since the installation scripts create a virtualenv to install the dependencies, you must activate virtualenv before running the runserver command if you have installed and configured the Store using these scripts. To do so, you must run the following command (in the src folder):

$ source virtenv/bin/activate

To start WStore, type the following command (in the src folder):

$ python manage.py runserver 0.0.0.0:8000

Then, go to http://computer_name_or_IP_address:8000/ where computer_name_or_IP_address is the name or IP address of the computer on which WStore is installed, and use the username and password you provided when populating the database to sign in on WStore.

9.2.8.2 Integrating WStore with Apache

If you choose to deploy WStore 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 wsgi.py file anywhere, but we recommend to have it inside your WStore installation (i.e. <path_to_wstore>/src/wsgi.py):

```python
import os
import sys
path = '<path_to_wstore/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()
```

If you are running WStore using a virtualenv environment your wsgi.py file should have the following structure:

```python
import os
import sys
import site

site.addsitedir('<virtualenv_path>/local/lib/python2.7/site-packages')

path = '<path_to_wstore/src>'
if path not in sys.path:
    sys.path.insert(0, path)
```
os.environ['DJANGO_SETTINGS_MODULE'] = 'settings'

# Activate your virtual env
activate_env=os.path.expanduser("<virtualenv_path>/bin/activate_this.py")
execfile(activate_env, dict(__file__=activate_env))

import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()

Please, pay attention that you set the right path to the wtore/src directory.
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>
  ...
  ### WStore ###
  WSGIScriptAlias / <path_to_django_wsgi>
  WSGIPassAuthorization On
  Alias /static <path_to_wstore>/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 <path_to_wstore>/src/static directory.

Once you have the site enabled, restart Apache
and go to http://computer_name_or_IP_address/ to get into WStore.

**Note:** The user www-data used by apache must be able to access WStore code and must have read and write rights in the different directories as specified in the final steps section.

### 9.3 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.

#### 9.3.1 End to End Testing

Please note that the following information is required before performing this process.

- The computer name or the IP address where WStore is running.
- Valid credentials for WStore (i.e credentials created during the syncdb command or an Identity Manager user).
- Access to a Repository GE (a new deployed instance or an existing one)
- A logo image.
- Some images to be used as screenshots

To check if WStore is running follow these steps:

1. Open a browser and enter WStore.
2. The login window should appear (WStore or idM depending on the configuration)
3. Introduce your credentials and click *login*
4. Go to the *Administration* view

5. Click add Repository
6. Fill Repository info and press *Register*

7. Go to the *Catalogue view*
8. Select the *Options* Tab and choose *Create new offering*

9. Fill the name, and the version

10. Include the logo and the screenshots

11. Select the option for not providing a notification URL

12. Press *Next*
13. Fill USDL Info and press Next
14. Press *Next* in the Application selection form (This form only appears if using idM for authentication).

15. Press *Accept*
16. The created offering should appear in the *Provided* Tab

![Catalogue](image)

### 9.3.2 List of Running Processes

We need to check that the Apache web server and the MongoDB database are running. WStore 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\|mongodb' | grep -v grep
```

It should show something similar to the following:

```
$ ps -ewF | grep 'apache2\|mongodb' | grep -v grep
root    1154   1  0  22744  3584  1 11:07 ?    00:00:00 /usr/sbin/apache2 -k start
www-data 1157  1154  0  22677  2620  2 11:07 ?    00:00:00 /usr/sbin/apache2 -k start
www-data 1178  1154  0  111374 6672  0 11:07 ?    00:00:00 /usr/sbin/apache2 -k start
www-data 1179  1154  0  111374 6672  2 11:07 ?    00:00:00 /usr/sbin/apache2 -k start
```
9.3.3  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:27017         *:* LISTEN    4879/mongod
tcp    0      0 localhost:28017         *:* LISTEN    4879/mongod
tcp6   0      0 [::]:http              [::]:* LISTEN    1154/apache2

9.3.4  Databases

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

```
$ mongo wstore_db -u wstore -p wstore
```

It should show a message text similar to the following:

```
MongoDB shell version: 2.0.4
connecting to: wstore_db
>
```

9.4  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.

9.4.1  Resource availability

Memory use depends on the number of concurrent users as well as the free memory available and
the hard disk. WStore requires a minimum of 512 MB of available RAM memory, but 1024 MB of
free memory are recomended. Moreover, WStore requires at least 10 GB of hard disk space.
9.4.2 Remote Service Access
N/A

9.4.3 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.
- MongoDB main memory consumption should be between 30 MB and 500 MB.

9.4.4 I/O flows
The only expected I/O flow is of type HTTP, on port defined in Apache Web Server configuration files.
10 Business Modeler - Installation and Administration Guide

10.1 Business Modeler Installation and Administration Guide

This document explains how to install and host the Business Modeler on the Node.js run-time. The Business Modeler is a single page web application, using HTML5, CSS3 and JavaScript and needs to be viewed with a common, up-to-date web browser.

10.2 System Requirements

This section covers the requirements needed to install and host the Business Modeler web application.

10.2.1 Software requirements

The following software is required to install and host the Business Modeler web application:

- NodeJS
- GruntJS

10.2.1.1 Node.js

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

First we need to install NodeJS (v0.10.18 or greater required). Since installation procedures are very different for each platform (and out of the scope of this manual) we would like to refer to the official Node.js installation guides:

- Installing Node.js via package managers
- Download Node.js binaries

10.2.1.2 GruntJS

Next we use the NPM (Node Package Manager, included in Node.js) to install a global instance of the Grunt command line interface:

```bash
$ sudo npm install -g grunt-cli
```

10.3 Software Installation and Configuration

The Business Modeler web application itself requires no installation, other than placing the application directory on a Node.js accessible location.
10.3.1 Port configuration

By default the Business Modeler web application will be served on port 8080. Should you wish to change this, you can change the port in the configuration file `<Business Modeler web application directory>/env/live.json`.

Contents of this file should be similar to:

```json
{
    "NODE_ENV" : "live",
    "DEST"    : ".",
    "HOST"    : "localhost",
    "PORT"    : "8080"
}
```

10.3.2 Starting the NodeJS server

Grunt is used to simplify the process of running the Business Modeler web application in Node.js:

```bash
$ cd <Business Modeler web application directory>
$ grunt server:start
```

Output should be similar to:

```
Running "server:start" (server) task

Running "env:live" (env) task

Running "forever:start" (forever) task
Attempting to start ./index.js as daemon.
Logs can be found at <Business Modeler web application directory>/logs.

Done, without errors.
```

In case a more detailed view of the bootstrap sequence is required, use the `--verbose` flag:

```bash
$ grunt server:start --verbose
```

This will outline the separate bootstrapping steps:

```
Initializing
Command-line options: --verbose
```
Reading "Gruntfile.js" Gruntfile...OK

Registering Gruntfile tasks.

Registering "grunt-contrib-connect" local Npm module tasks.
Reading <Business Modeler web application directory>/node_modules/grunt-contrib-connect/package.json...OK
Parsing <Business Modeler web application directory>/node_modules/grunt-contrib-connect/package.json...OK
Loading "connect.js" tasks...OK
  + connect

Registering "grunt-open" local Npm module tasks.
Reading <Business Modeler web application directory>/node_modules/grunt-open/package.json...OK
Parsing <Business Modeler web application directory>/node_modules/grunt-open/package.json...OK
Loading "open.js" tasks...OK
  + open

Registering "grunt-casperjs" local Npm module tasks.
Reading <Business Modeler web application directory>/node_modules/grunt-casperjs/package.json...OK
Parsing <Business Modeler web application directory>/node_modules/grunt-casperjs/package.json...OK
Loading "casperjs.js" tasks...OK
  + casperjs

Registering "grunt-env" local Npm module tasks.
Reading <Business Modeler web application directory>/node_modules/grunt-env/package.json...OK
Parsing <Business Modeler web application directory>/node_modules/grunt-env/package.json...OK
Loading "env.js" tasks...OK
  + env

Registering "grunt-forever" local Npm module tasks.
Reading <Business Modeler web application directory>/node_modules/grunt-forever/package.json...OK
Parsing <Business Modeler web application directory>/node_modules/grunt-forever/package.json...OK
Loading "forever-task.js" tasks...OK
+ forever
Initializing config...OK
Loading "Gruntfile.js" tasks...OK
+ default, server, tests

Running tasks: server:start

Running "server:start" (server) task

Running "env:live" (env) task
Verifying property env.live exists in config...OK
Files: env/live.json
Reading env/live.json...OK

Running "forever:start" (forever) task
Attempting to start ./index.js as daemon.
Logs can be found at <Business Modeler web application directory>/logs.

Done, without errors.

When the server is up and running the final line in the standard output stream is:

Done, without errors.

In case anything went wrong a warning will be thrown to the standard error stream:

Warning: <warning message>

Aborted due to warnings.

Follow Diagnosis Procedures to diagnose the problem.

10.3.3 Stopping the Node.js server

The Node.js server can be stopped using Grunt:

$ grunt server:stop

Output should be similar to:

Running "server:stop" (server) task
Running "env:live" (env) task

Running "forever:stop" (forever) task

Attempting to stop ./index.js...

<table>
<thead>
<tr>
<th>uid</th>
<th>command</th>
<th>script</th>
<th>forever pid</th>
<th>pid</th>
<th>logfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>yv_v</td>
<td>/usr/local/bin/node ./index.js</td>
<td>58246</td>
<td>58247</td>
<td>/Users/creynder/.forever/yv_v.log 0:0:11:44.502</td>
</tr>
</tbody>
</table>

Done, without errors.

10.3.4 Viewing the web application

The application is hosted on port 8080. You can open a web browser of your choice and point it to http://localhost:8080 or run the following grunt command:

$ grunt server:view

Your default browser will automatically open, showing the Business Modeler web application.

10.4 Sanity check procedures

The System Administrator needs to perform the following verifications, to ensure the application is fully functional and correctly served by the NodeJS server.

10.4.1 End to End testing

To check if the application is running correctly, please verify the following:

- It shows an empty business modeler canvas, menu bar with "File", "Edit", "View" and "Snap" buttons at the top and a "Toolbox" and "Business Model" panel at the right.
Business Modeler web app

- Clicking any of the "Elements" in the sidebar creates a corresponding business element on the canvas:

Creating an element

- Clicking the "Edit Timerange" button in the "Business Model" panel opens a dialog window:
Clicking the "File" > "Save" menu item will, depending on the browser you use, either open a business model XML file in a new tab or present you with an OS-native filesystem browser dialog:

It accepts a GET REST service call on [http://localhost:8080/api/bmd/models](http://localhost:8080/api/bmd/models):

```
$ curl -G http://localhost:8080/api/bmd/models --verbose
```

Output should be similar to:
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /api/bmd/models HTTP/1.1
> User-Agent: curl/7.19.7 (universal-apple-darwin10.0) libcurl/7.19.7 OpenSSL/0.9.8r zlib/1.2.3
> Host: localhost:8080
> Accept: */*
>
< HTTP/1.1 200 OK
< X-Powered-By: Express
< Vary: Accept
< Content-Type: application/vnd.iminds.bme+json
< Content-Length: 96
< Date: Thu, 03 Oct 2013 10:50:29 GMT
< Connection: keep-alive
<
> Connection #0 to host localhost left intact
> Closing connection #0
{
  "BMD": {
    "document": {
      "uuid": "SingleActor",
      "title": "SingleActor",
      "link": {
        "href": "/api/bmd/models/SingleActor",
        "rel": "/api/schemas/iminds.be/bmd/model"
      }
    }
  }
}

If all of the above steps were possible the application was deployed correctly.

10.4.2 List of Running Processes

Execute the following commands to check whether the relevant processes are up and running

```
$ ps ax | grep 'node'
```

Output should be similar to:

```
  58434 ?? 0:00.30 /usr/local/bin/node <Business Modeler web application directory>/node_modules/grunt-forever/node_modules/forever/bin/monitor ./index.js
  58435 ?? 0:00.44 /usr/local/bin/node ./index.js
  58484 ttys000 0:00.00 grep node
```

10.4.3 Network interfaces Up & Open

The Node.js HTTP Server listens on port 8080:
D.3.3.3: FI-WARE Installation and Administration Guide

$ netstat -anf inet | grep '8080'

Output should be similar to:

```
tcp4 0 0 *.8080 *.* LISTEN
```

10.4.4 Databases
N/A

10.5 Diagnosis Procedures

If any of the above sanity checks produced errors or unexpected results the System Administrator should use the following guidelines as a first measure diagnosis. Extensive testing and running the Unit Tests may be necessary to pinpoint the exact location of the error(s).

The System Administrator can view the error and access log files to verify no errors occurred when serving the files and/or whether the service was correctly started. The default location of the log files is `<Business Modeler web application directory>/logs/err.log` and `<Business Modeler web application directory>/logs/out.log`.

Grunt can log its output to the stdout stream. You need to add the `--debug` flag to any failing process:

```
$ grunt server:start --debug
```

10.5.1 Resource availability

The application has been tested on a medium end client machine (not a dedicated high end server) with four cores CPU, 4G ram and 200+G hard drive. Running the application on fewer resources would be possible but has not been tested.

The web application consists of multiple files, which are GET requested from the Node.js HTTP Server. Using browser dev tools the System Administrator can verify they are correctly requested and served:
The above example shows a full resource delivery, meaning the application is correctly served.
Another useful diagnostic is to verify no errors were thrown to the JavaScript console:

The console should be free of error messages (as in the above example).

10.5.2 Remote Service Access
The Node.js HTTP server will provide the necessary remote service capabilities for the services to be run.
10.5.3 Resource consumption
Node should appear twice in the running processes list and both processes should consume maximally 30Mb of main memory.

10.5.4 I/O flows
The only expected I/O flow is of type HTTP, on the port as defined in the configuration files, "8080" by default), both inbound and outbound.
11 Business Calculator - Installation and Administration Guide

11.1 Business Calculator Installation and Administration Guide

This document explains how to install and host the Business Calculator on an application server. The Business Calculator in its current implementation is actually the ECMN Calculator and consists of three parts.

1. The first part is the front-end which is a web application, using HTML5, CSS3 and Javascript which needs to be viewed with a common, up-to-date web browser. It needs to be served from within an HTTP server (or an application server as these typically pack this functionality as well).

2. The second part is the back-end of the editor. Currently, the implementation requires a back-end for storage, although it will work without storage as a proof of concept in a stand-alone fashion as well. In following versions the front-end will be able to work as a stand-alone application as well as in combination with a back-end. The back-end is implemented in java using a REST interface and requires as such an application server to be started for this back-end. At start-up the front-end will ask to link to this back-end.

3. The third part, which is actually the real generic enabler, is the ECMN Calculator service. This is also a java based implementation which exposes one REST interface as described in the programmer manual. It needs to be hosted on an application server as well. Finally the editor front-end links to this calculator for inspection purposes and will ask, at start-up, for the location of this service.

Before being able to use the editor in the intended manner, both services (back-end and calculator) need to be started and their location linked at start-up of the editor. When this is not the case the editor can be used, but will make no link to neither the back-end nor the calculator.

11.2 System Requirements

This section covers the requirements needed to install and host the Business Calculator elements. As mentioned before, the application consists of three separate (but linked) components. None of the three components needs dedicated installation steps, they only need to be dropped into the right server and hosted from there. The best approach here is to work with an application server which can host both the java-web-application as well as the front-end for download. The following sections detail how to install and configure an application server to host the three components.

11.2.1 Software requirements

This manual guides you through the process of setting up the JBoss application server as the server of choice as this is both open and can be used by everybody and as this server has been used in the development process as well. Of course other application servers should be capable of hosting the Business Calculator components as well. During development we took the necessary measures to guarantee that the application should run on other application servers as well, but all internal testing occurred on the mentioned JBoss instances.

To install and run the open source JBoss Application Server or the new version renamed into Wildfly Application Server take a look at their very short and self explanatory installation guides:

- WildFly
  - getting started and installation
Future Internet Core Platform

- JBoss 7.1.1
  - on a linux/unix machine
  - on a windows machine
  - from within your IDE

The deployment of the software has been tested on a clean JBoss 7.1.1 as well as on a clean wildfly 8.0 installation. It should be noticed that it is required to have the latest Java runtime (and/or development kit) installed (at least version 1.7) on an up-to-date operating system. The installation has been tested on clean a windows 7 as well as an up-to-date linux (Ubuntu) distribution.

11.3 Software Installation and Configuration

The Business Calculator components itself typically require no installation.

Note: Due to a minor issue/bug in the JBoss (7.1.1) a library gets incorrectly linked and might cause some war-files to fail. In case this issue pops up, this can be solved by adding the jar-file as a module in the JBoss server, which can be accomplished by following the steps:

1. go to the Jboss installation directory (we refer to this later as <JBossAS7_HOME>)
2. get the commons-httpclient-3.1 jar from its location
3. add new directories (if not existing) to get the path <JBossAS7_HOME>/modules/org/apache/commons/httpclient/main
4. copy the file commons-httpclient-3.1.jar in this directory
5. add the file module.xml at this location

It is preferable to work from within the administration panel of the JBoss running instance. In this administration panel, new instances can simply be added and linked to a specific location by dropping the .war-file into a new location and starting this service. It should be noted that for the time being - due to shared storage locations - both the back-end as the calculator should be hosted from the same application server instance.

11.4 Sanity check procedures

Once installed, the System Administrator needs to perform the following verifications, to ensure the application is fully functional and correctly served by the HTTP server.

11.4.1 End to End testing

The three components - editor front-end, editor back-end and calculator should be tested by starting the editor at [1]. This requires the Jboss to be running and hosting the three services, which each should also be up and running. the editor will automatically open up a window for linking to the other services with a correct path to each of those services predefined and a connection-test button. Clicking on this button and receiving and OK is all it takes to verify the three components.

To check if the editor (and linked back-ends) is running correctly, please verify the following:

- start-up the application (browse to the location to which you linked the war file) and you should get the following screen. It contains a popup configuration of the linked services (back-end and calculator):
Business Calculator Editor

- clicking on close - not selecting a connection should lead you to the offline version (no storage existing yet) and you should see the following screen:

Business Calculator Offline Editor

- Once opened you should click on model - load. There should be one model selectable (hardcoded test). Opening this should lead you to the following screen:
From here on you should be able to work with the editor in the offline (as well as the online version) as specified in the user manual.

11.4.2 List of Running Processes
The JBoss Server should be running as a standalone process. In the Jboss server administration panel all three services should be up and running.

11.4.3 Network interfaces Up & Open
the JBoss server should have the 8080 (tcp) port opened for connections. Additionally to use the administration panel, the port 9990 should be opened at the same machine. Of course all administration can be run from the localhost or locally in the JBoss configuration files as well.

11.4.4 Databases
N/A

11.5 Diagnosis Procedures
If there are errors or unexpected results the System Administrator should use the following guidelines as a first measure diagnosis. Extensive testing and running the Unit Tests may be necessary to pinpoint the exact location of the error(s).

**JBoss Application Server**
The System Administrator can view the error and access log files to verify no errors occurred when serving the files and/or whether the service was correctly started. The default location of the log files is `jboss/server/(serverinstance - generally default)/log/server.log`.

11.5.1 Resource availability
All business calculator services together with a JBoss runtime can be hosted on a single low end machine. This has been tested on two test-machines: (1) A four core CPU, 4G ram and 250G hard
drive. No bad performance has been experienced and no tests below these resources have been performed. (2) A single core CPU 2.4Ghz, 1G ram and 128G hard drive. No bad performance has been experienced and resource consumption indicated less than 2G hard disk space was required and typically less than 200MB memory was consumed. CPU ratings show no hard constraint here either. Of course multiple users might increase requirements.

The web application consists of multiple files, which are GET requested from the HTTP Server. Using browser dev tools the System Administrator can verify they are correctly requested and served:

![Chrome dev tool: resource tab](image)

The above example shows a full resource delivery, meaning the application is correctly served. Another useful diagnostic is to verify no errors were thrown to the JavaScript console:

![Chrome dev tool: console tab](image)

The console should be free of error messages (as in the above example).

### 11.5.2 Remote Service Access

The Jboss server will provide the necessary remote service capabilities for the three services to be run. In order to provide the functionality to external parties, the JBoss server should be binded to all network interfaces and these interfaces should have public access at port 8080.
11.5.3 Resource consumption
As mentioned before, a low end regular PC will suffice to run these services for a limited amount of users (<10).

11.5.4 I/O flows
The firewalls should allow regular http traffic (and https for remote JBoss administration)