Fusion Engine V0.3

INSTALLATION GUIDE

TABLE OF CONTENTS

1 - Introduction

1.1 - Enablers. FICONTENT and FIWARE

In order to strengthen a powerful European industry market for the upcoming years in the context of the Future Internet (FI), an FI-PPP called FIWARE was created (http://www.fiware.org/). The main outputs of this project are:

- To provide a reference platform for developing smart applications in an easy and efficient way.
- To provide a set of core (horizontal) components that build the core of this platform, upong which new (vertical) components can be built.

In order to further spread the usage of FIWARE in the European industry, new actions and use case projects were created, one of which was called FICONTENT (http://mediafi.org/) , focussed on topics such as social connected TV, smart city guides and pervasive games.

The components developed in FIWARE are called Generic Enablers (GEs), as they are envisioned for a general (horizontal) market. On the other side, components developed in FIWARE use case projects such as FICONTENT are called Specific Enablers (SEs), as they are primarily intended for special purposes (e.g. multimedia, TVs, databases, etc.).

All components (enablers) are just middleware that should facilitate big and small companies develop next generation Internet applications in order to be more competitive and/or productive. Some GEs and SEs are open source, and other can be used under specific terms of use defined by each project partner, but all of them have been built with the intention of providing helpful tools for SMEs.

1.2 - Fusion Engine Definition and Contextualization

The Fusion Engine (FE) is an open source specific enabler (SEs) developed in the framework of the FICONTENT2 project. The specific area of interest of this enabler is Smart Cities, as it provides a tool to fusion data from different data providers to be used for further smart city applications.

The Fusion Engine is about data fusion for smart cities. In this context, the most useful piece of data for general purposes is called a POI (Point of Interest). Though POIs are mostly used for navigational GPS and touristic applications, it can be easily extended to any smart city environment you can imagine. Basically, a POI is something located at a certain position with certain properties that might be of interest while developing your application.

It is important to clarify what fusion means in the context of FE and why it might be helpful. With the appearance of the smart city concept, many cities (town council) are showing strong interest in providing local open data for their citizens. Other companies are also providing more and more open data repositories (e.g. for touristic and environmental purposes). There are also other global data repositories (e.g. OSM and DBPedia) that can act as additional data sources. So many data sources might be apparently helpful, but it typically results in POI replication and multiple connections to different data sources. Why not making this process offline before? One may wish to:

- Select only those data sources that are of interest
- Select only those categories from the sources that are of interest
- Get only one single merged POI from multiple ones
- Get only one single access interface to retrieve the resulting POIs

The result of the fusion generated by the FE is a georeferenced database of POIs. We called this Open City Database (OCD) as the input data (POIs) might typically come from open data. Any way you can also use proprietary data if you comply with the corresponding license. This is much more a legal issue than a technological issue. In fact, the FE supports in its data model the multiple usage of licenses. It is up to the user to correctly (legally) exploit the data.

1.3 - Fusion Challenges

There are 3 technological challenges targeted by data fusion in general and FE in particular, as depicted in the following Figure 1 (see C in red).

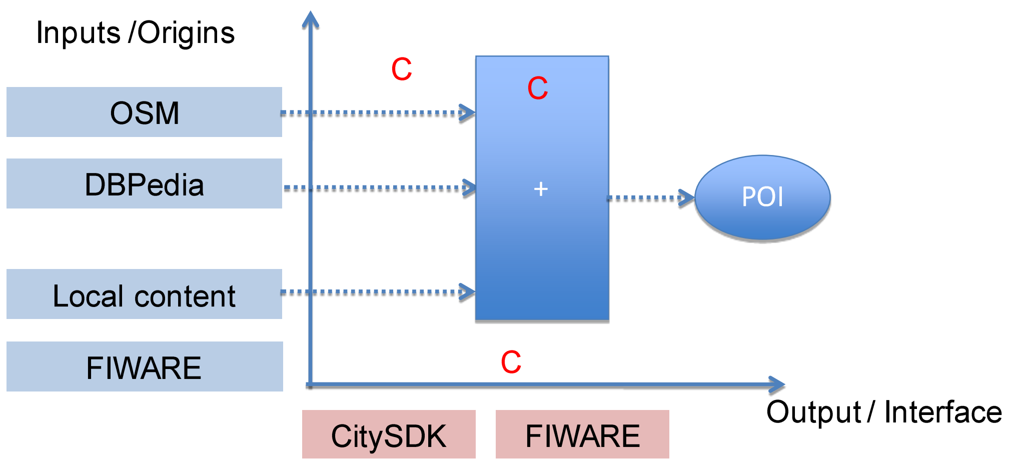


Figure 1. Fusion challenges in FiContent.

The first challenge relates to the input data. There are many data sources to be used:

- Global data sources, such as OSM and DBPedia
- Local data sources, such as the data portals of many cities (e.g. Valencia datos abiertos, opendatabcn, opendatacanarias, etc.)
- Particular data sources, such as FIWARE GEs

Note here that each data source has its own way (format and API) for providing the data. Besides, data categories are not necessarily preserved between data sources and are typically different.

The second challenge is how we are going to perform the fusion, the merging of POIs:

- How do we define that one POI from data source A and another POI from data source B refer to the same POI?
- When two or more POIs match, how do we create one single POI (e.g. which name and location do we preserve?

The third challenge is about the output of the fusion. Even if it is a georeferenced POI repository, there should be a single API access. One would think in a standard way of defining POIs. Unfortunately, there is no such. Several years ago there was a first POI draft specification by the W3C (http://www.w3.org/2010/POI/), which later moved to OCG (http://www.opengeospatial.org/projects/groups/poiswg). However, the activity within this last group is frozen for the last two years. Thus, there were basically two alternatives:

- Define a new API format and API interface. This has been the approach within FIWARE
- Use a current POI specification that most resembles the POI draft, called CitySDK (http://www.citysdk.eu/).

2 - ARCHITECTURE OF THE FUSION ENGINE

The FE has two main components: (i) the frontend and (i) the service, as depicted in Figure 2.

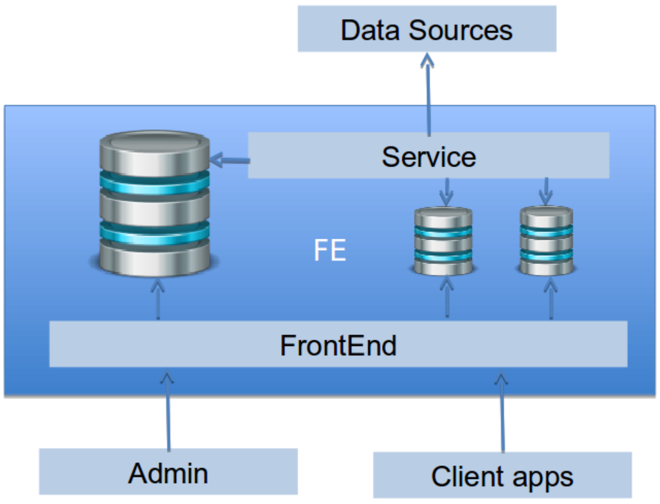


Figure 2. FE general architecture.

2.1 - FE Frontend

The FE frontend acts as interface with the outside world:

- Administrators access the frontend to set up the fusion engine and generate OCD instances
- Users and client applications access the frontend to request for POI data of a particular OCD instance

Note in Figure 2 that the FE has many different databases.

- There is a main database, called OCD_BASE, which indexes and configures all OCD instances. The data stored in this database is entered by the administrator.
- There are multiple databases, corresponding to each of the OCDs that are being built. This allows separating the information about POIs and securing data access. It also facilitates exporting operations of an OCD. Client apps access one of these OCD (small) databases through the frontend.

For client applications, there is a tiny difference with CitySDK interface. As the FE is able to generate as many OCDs as set up by the administrator, you also have to provide in the access API and 'ocdName' parameter, in order to select the city (OCD) you are interested in. The management of different databases allows introducing different access models and therefore business models. For example, an admin (SME) can offer free access to a basic OCD but pay-per-access to an enriched OCD.

2.2 - FE service

The FE service is a standalone process that reads the set up information in the OCD_BASE database and builds the different OCDs taking information from the corresponding (selected) data sources. This is an offline process that can take several minutes depending on the configuration (number of sources, number of categories, city size, etc.)

3 - INSTALLATION

Installing the Fusion Engine enabler is quite easy. Just go to Github:
https://github.com/satrd/poi_fusion_engine3
You can select to download the whole project (Download ZIP in the right-middle area of the page) or just go to the release directory and download the packaged version, both for frontend and service components.

3.1 - Requirements and previous steps

This installation guide is described for Ubuntu 14.10. For other versions or operating systems, please take care of the corresponding differences.
Required software components for a successful installation and operation:

- JDK v.1.7.x or higher (the FE is developed in Java. You can use either OpenJDK or the official JDK provided by Oracle
  ```
      $ sudo apt-get install openjdk-8-jdk
  ```
- POSTGIS 2.1.2 or higher with POSTGRESQL 9.1 or higher. You also need the PSQL client
  ```
  $ sudo apt-get install postgis postgresql-9.3-postgis-scripts postgresql postgresql-client
  ```
  If required, remember to grant user access at /etc/postgresql/9.X/main/pg_hba.conf
- Tomcat v7.0 or higher (tested with Tomcat8)
  ```
  $ sudo apt-get install tomcat8 tomcat8-admin
  ```
  Remember to assign a manager user in /var/lib/tomcatX/conf/tomcat-users.xml
- Apache Ant 1.9.3 or higher (to execute the FE). Make sure ANT is in your PATH if you install it on Windows
  ```
  $ sudo apt-get install ant
  ```

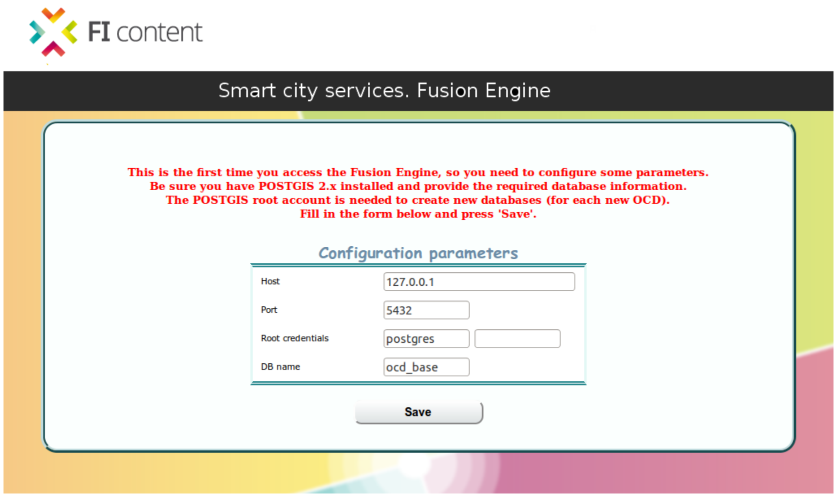
Optional components (for developers)

- pgAdminIII (this will help you accessing the POSTGIS database)
  ```
  $ sudo apt-get install pgadmin3
  ```
- Eclipse Luna (with this Java IDE developers may change the code to adapt it you their application needs) Download the latest version (Eclipse Luna EE) at www.eclipse.org
- Git (if you want to download easily the code)
  ```
  $ sudo apt-get install git
  $ git clone https://github.com/satrd/poi_fusion_engine3
  ```

Developers can also download the code with Eclipse (see http://wiki.eclipse.org/EGit/User_Guide#Starting_from_existing_Git_Repositories for more info) or import it later from Eclipse.

3.2 - FE Frontend

We take as starting point Github:
https://github.com/satrd/poi_fusion_engine3
Go to the folder Release→ Frontend and download the WAR file (fic2_fe_v3_frontend.war)
Deploy this WAR file on Tomcat (be sure that you have installed the prerequisites explained in the previous section).
For the first time, the application will detect that it has to be configured, showing some default parameters to be used, as depicted in Figure 3. These parameters are taken from the config.xml file located under the 'config' folder where the application has been deployed.


Smart city services. Fusion Engine

This is the first time you access the Fusion Engine, so you need to configure some parameters.
Be sure you have POSTGIS 2.x installed and provide the required database information.
The POSTGIS root account is needed to create new databases (for each new OCD).
Fill in the form below and press 'Save'.

Configuration parameters

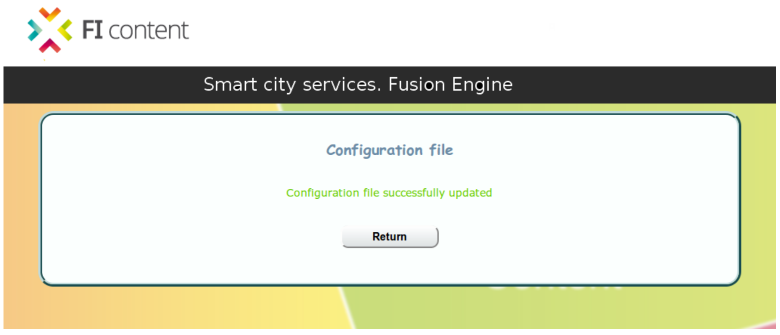
Host	127.0.0.1	
Port	5432	
Root credentials	postgres	
DB name	ocd_base	

Figure 3. FE frontend initial configuration

Just press 'Save' and everything should be done automatically, which is:

- Create the main database (ocd_base)
    - Create two demo databases as examples : valencia_demo and tenerife_demo , Note here that is information is stored under the ../config/dbScripts folder. Here there are two SQL files that contains the dump of a previous fusion. It contains (hardcoded) the values of an Ubtuntu's tomcat8 installation. If you are using tomcat7, then change it, e.g:

```
sed -i 's/tomcat8/tomcat7/' ocd_valencia_demo.sql
```


Smart city services. Fusion Engine

Configuration file

Configuration file successfully updated

Figure 4. Initial configuration successful message

If you don't get this successful message, then there should have been some error during the initialization. You may want to check the log file located in:
<tomcat_dir>/logs/fic2_fe_v3_frontend.log

You can also check (e.g. via pgAdmin3) that three databases have been created in postgis: *ocd_base*, *ocd_valencia_demo* and *ocd_tenerife_demo*
If everything went right, you should see the main page after pressing the 'Return' button in Figure 4.

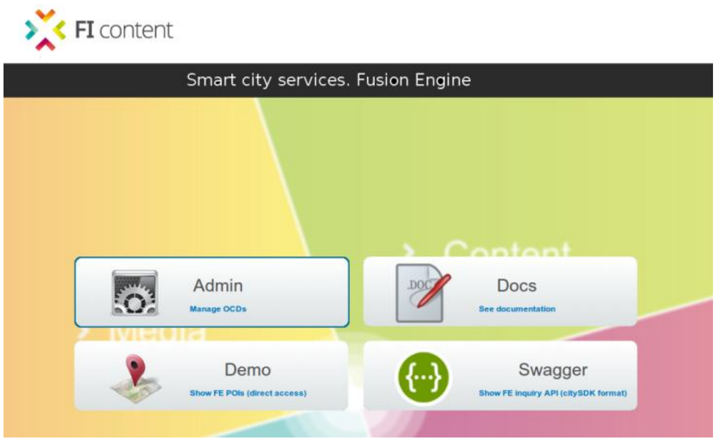
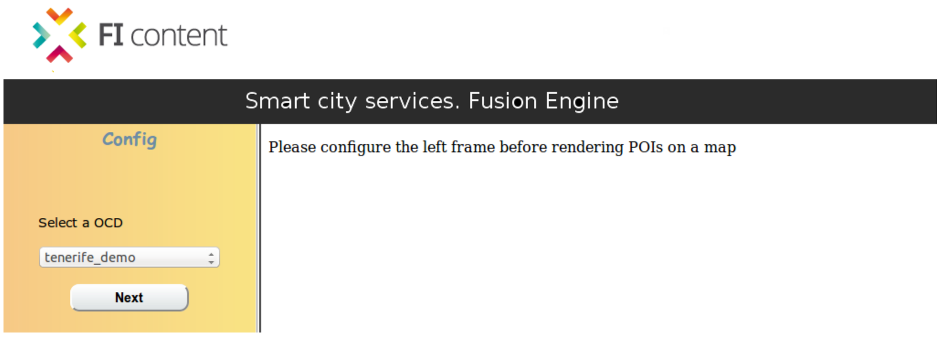

FI content
Smart city services. Fusion Engine
Admin
Manage OCDs
Docs
See documentation
Demo
Show FE POIs (direct access)
Swagger
Show FE inquiry API (citySDK format)

Figure 5. FE frontend main page

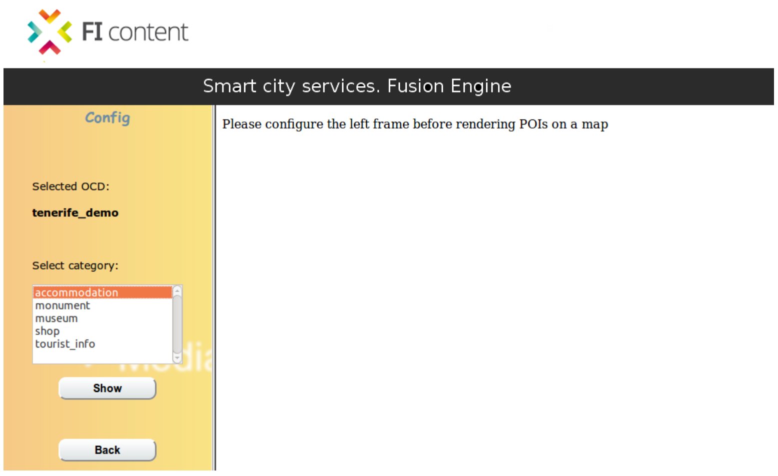
For a quick check that everything went smoothly, you might click on 'Demo' and Select one city (e.g. tenerife_demo).



Please configure the left frame before rendering POIs on a map

Figure 6. Demo (Step 1)

Click 'Next' and select one category (e.g. accommodation) and press 'Show'



Please configure the left frame before rendering POIs on a map

Figure 7. Demo (Step 2)

You will see the POIs in a google maps view on the right frame. You can repeat for other categories and or OCDs (e.g. valencia_demo).

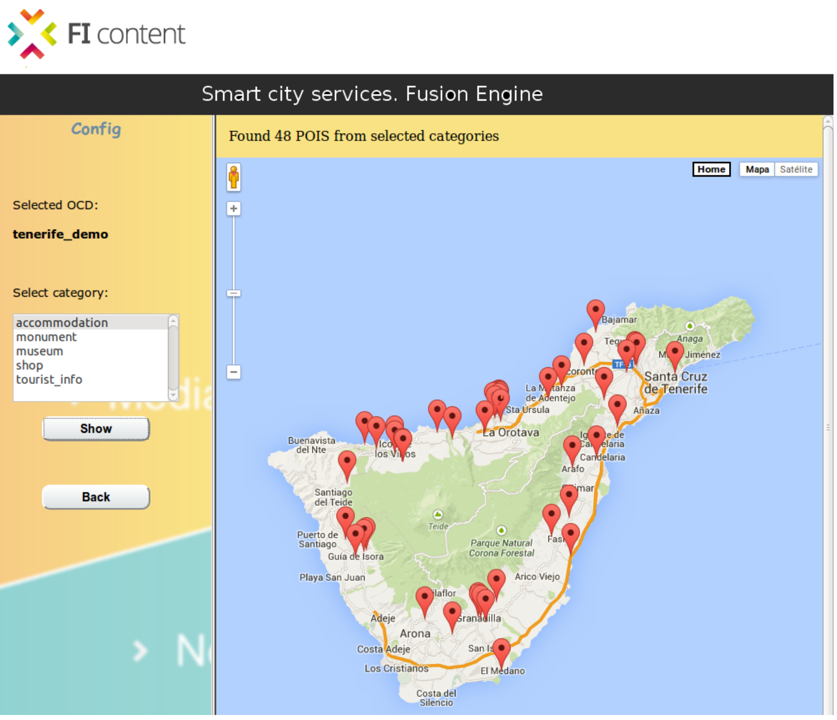
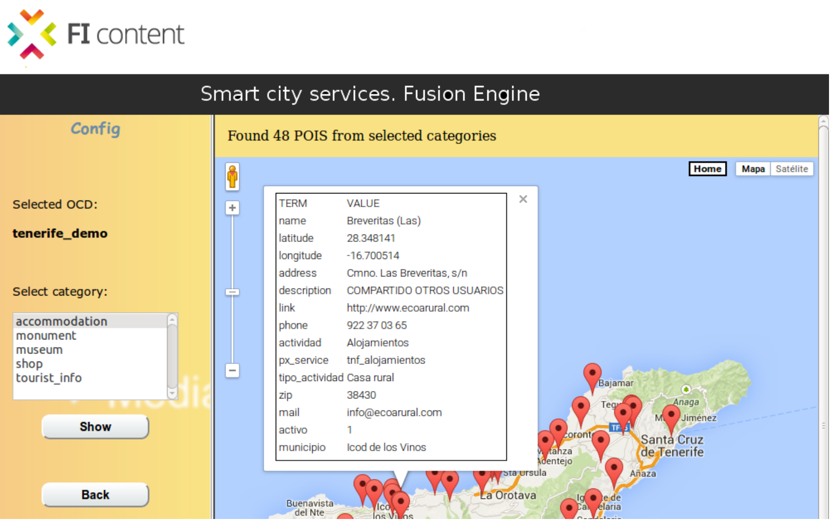


Figure 8. Demo (step 3). Visualization of POIs

If you click on a particular POI, you can even show more detailed information about it.



TERM	VALUE
name	Breveritas (Las)
latitude	28.348141
longitude	-16.700514
address	Cmno. Las Breveritas, s/n
description	COMPARTIDO OTROS USUARIOS
link	http://www.ecoarural.com
phone	922 37 03 65
actividad	Alojamientos
px_service	tnf_alojamientos
tipo_actividad	Casa rural
zip	38430
mail	info@ecoarural.com
activo	1
municipio	Icod de los Vinos

Figure 9. Demo (step 4)

For more information about how to further use the FE please see the *User-Admin Guide* in the *doc* folder.

3.3 - FE Service

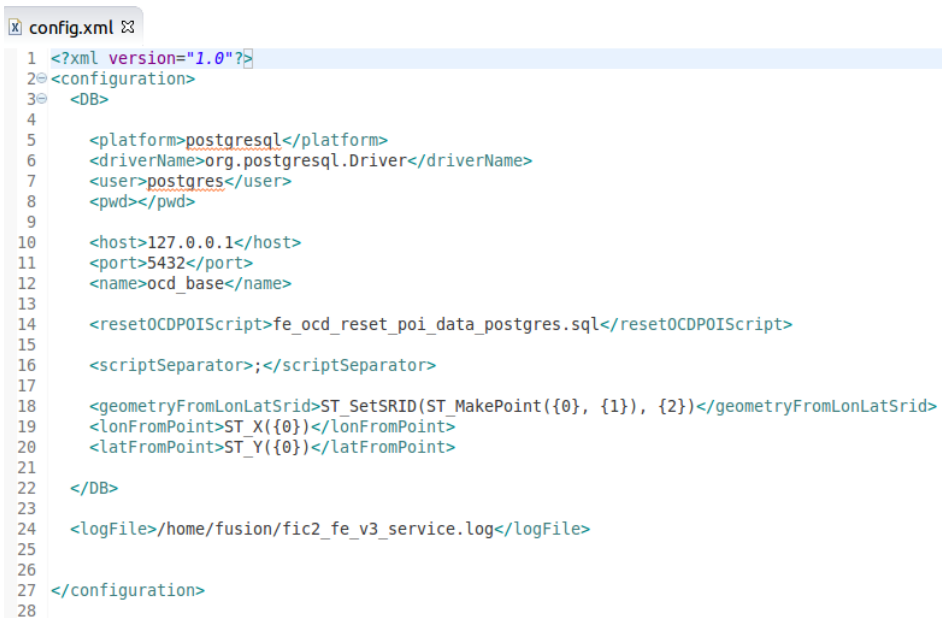
Just go to Github:
https://github.com/satrd/poi_fusion_engine3

Go to the folder Release→ Service and download the four files in the same folder
You will run the service with a console:

```
$ java –jar fic2_fe_v3.jar
```

But PREVIOUSLY you need the other files in the same directory:

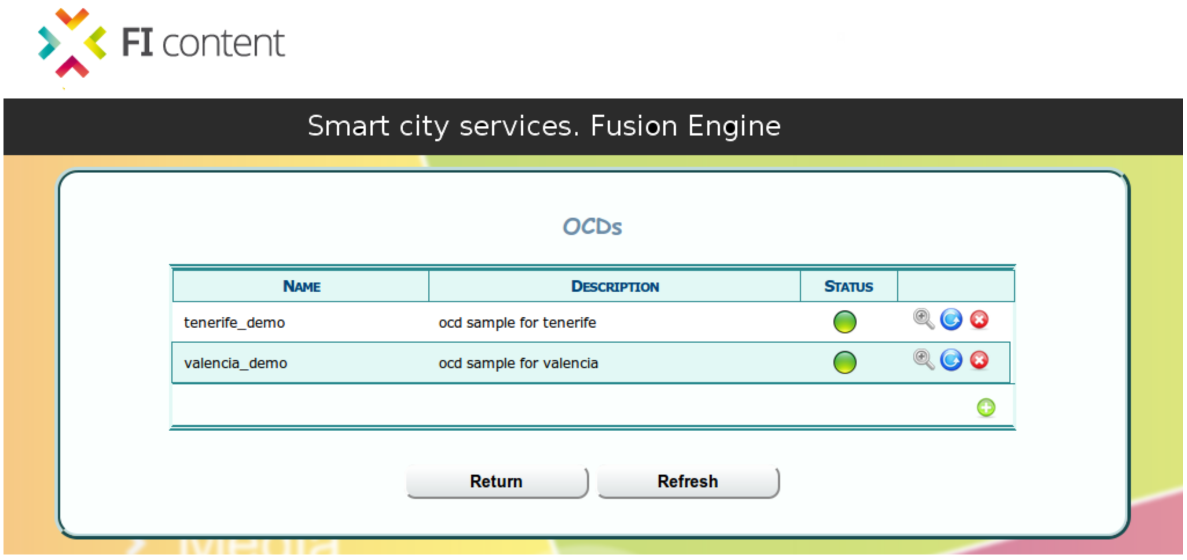
- Config.xml : this has a similar structure as for the frontend service, but here you will have to edit the values manually. You only need to configure some values of this configuration file (see Figure 10) :
  - The connection to the *ocd_base* database (user,pwd,host,port and name). This should be the same as for the frontend service. You have to use here also the root user, as each OCD has its own database and the service needs to create one per each OCD.
  - Logfile : this is the directory where to log data. Be sure that you have write permissions on this directory. This directory is important because :
    - The main thread will log data in the file specified by the 'logFile' parameter
    - The other threads (one per OCD) will log data in the same directory with the name <id_ocd>.log
- fe_ocd_reset_poi_data_postgres.sql : this is just a simple script that allows to reset an already created OCD to perform the fusion again. You may notice that the name of this file is given in the config.xml file.



```
<?xml version="1.0"?>
<configuration>
  <DB>

    <platform>postgresql</platform>
    <driverName>org.postgresql.Driver</driverName>
    <user>postgres</user>
    <pwd></pwd>

    <host>127.0.0.1</host>
    <port>5432</port>
    <name>ocd_base</name>

    <resetOCDPOIScript>fe_ocd_reset_poi_data_postgres.sql</resetOCDPOIScript>

    <scriptSeparator>;</scriptSeparator>

    <geometryFromLonLatSrid>ST_SetSRID(ST_MakePoint({0}, {1}), {2})</geometryFromLonLatSrid>
    <lonFromPoint>ST_X({0})</lonFromPoint>
    <latFromPoint>ST_Y({0})</latFromPoint>

  </DB>

  <logFile>/home/fusion/fic2_fe_v3_service.log</logFile>


</configuration>
```

Figure 10. Config.xml example

Check the config.xml before running the service. Once executed, the FE will look up in the ocd_base database for OCDs to be fusioned. As long as there is no new OCD, you may only see in the main log file (fic2_fe_v3_service.log) this loop.
If you want to force the service without setting up a new OCD, you can go to the FrontEnd service's main menu and press 'Admin' → OCDs.
You should see (Figure 11) two demo OCDs. At the right side, there is a blue button that allows you to restart fusion.



Name	Description	Status	
tenerife_demo	ocd sample for tenerife	●	
valencia_demo	ocd sample for valencia	●	

Figure 11. OCD menu

If you click on one of them, you should trigger a new Fusion Engine process, and a new log file should appear in the configured FE service log directory. Besides, the status of the clicked OCD in the Frontend Service turns to 'Running' until the Fusion process ends (this may take several minutes).

For more information about the FE service, you may also look the Frontend Service, as both are complimentary services. You may therefore have a look at the *User-Admin guide* or even the Developer *gui*de.
Both files are available in G*ithub* under the '*doc*' folder.