fiware-rss Documentation

Release develop

March 10, 2016

RSS-RI is the reference implementation of the Revenue Settlement and Sharing System GE. RSS-RI is in charge of distributing the revenues originated by the usage of a given service among the involved stakeholders. In particular, it focuses on distributing part of the revenue generated by a service between the Store Provider and the Service Provider(s) responsible for the service. With the term "service" we refer to both final applications and backend application services (typically exposed through an API). Note that, in the case of composite services, more than one service provider may have to receive a share of the revenues.

This project is part of FIWARE.

CHAPTER 1

Index

Installation and Administration Guide The guide for RSS maintainers that explains how to install and configure it.

User and Programmer Guide The guide for user and programmers where it is explained how to use RSS-RI and how to integrate the exposed API with third party applications.

1.1 Installation and Administration Guide

1.1.1 Introduction

This Installation and Administration Guide covers RSS-RI version 4.4.3. 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 GEi.

1.1.2 System Requirements

Hardware Requirements

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

- CPU: 1-2 cores with at least 2.0 GHZ
- Physical RAM: 2GB
- Disk Space: 10GB The actual disk space depends on the amount of transactions strored in the database.

Operating System Support

The RSS has been tested in the following Operating Systems:

- Ubuntu 12.04, 14.04
- CentOS 7.0

Software Requirements

In order to have the RSS running, the following software is needed. However, these dependencies are not meant to be installed manually in this step, as they will be installed throughout the documentation:

- Java JDK 7
- Apache Tomcat 7
- MySQL >= 5.5

1.1.3 Software Installation

Getting the RSS Software

The packaged version of the RSS software can be downloaded from:

- The FIWARE catalgue.

This package contains the war files of the RSS as well as the installation scripts used in this document.

Alternatively, it is possible to install the RSS from the sources published in GitHub. To clone the repository, the git package is needed:

```
# Ubuntu/Debian
$ apt-get install git

# CentOS
$ yum -y install git
```

To download the source code usig git, execute the following command:

```
$ git clone https://github.com/conwetlab/fiware-rss.git
```

Installing the RSS Using Scripts

In order to facilitate the installation of the RSS, the script *install.sh* has been provided. This script installs all needed dependencies, configures the RSS and deploys it.

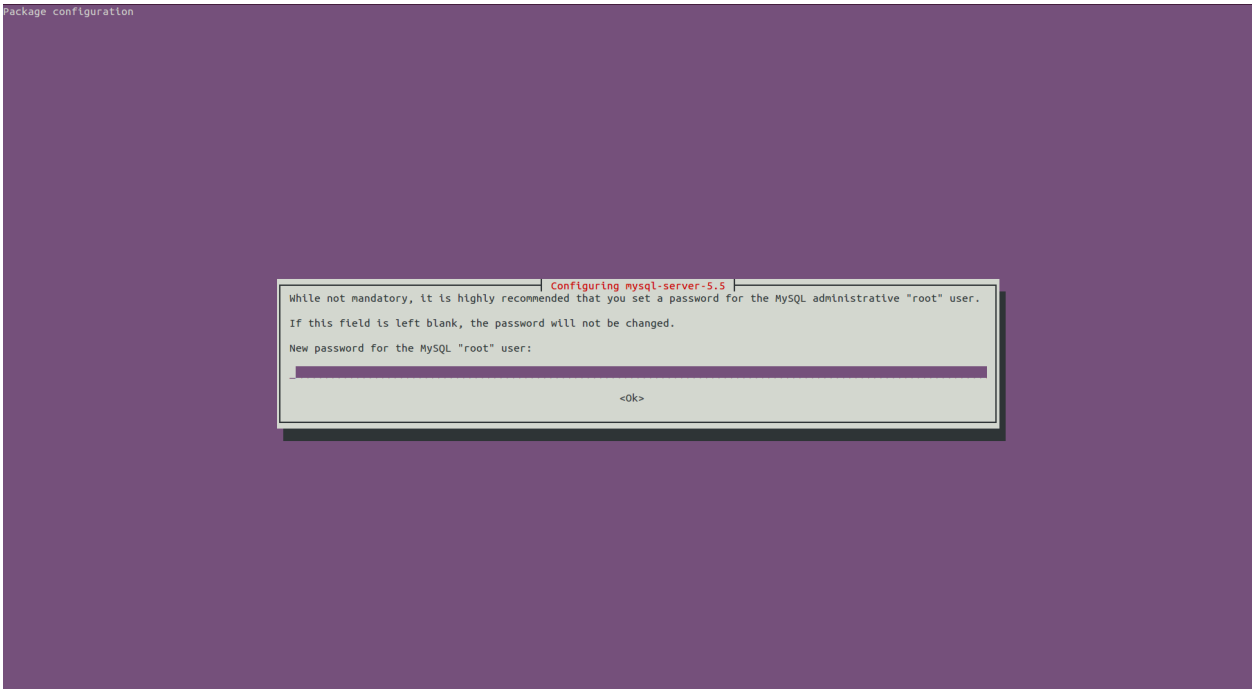
Note: The script *install.sh* installs java and tomcat. If you have those systems already installed, you may want to install the RSS manually as explained in the next section.

To use the installation script execute the following command:

```
$ ./install.sh
```

The installation script, installs MySQL and creates the root user. During this process you will be asked to provide a password for this user:

- Ubuntu



- CentOS

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the current
password for the root user.  If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):

OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MySQL
root user without the proper authorisation.

Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
    ... Success!

By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n]

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.
```

```
    Disallow root login remotely? [Y/n]

    ... skipping.

    By default, MySQL comes with a database named 'test' that anyone can
    access.  This is also intended only for testing, and should be removed
    before moving into a production environment.

    Remove test database and access to it? [Y/n]

    Reloading the privilege tables will ensure that all changes made so far
    will take effect immediately.

    Reload privilege tables now? [Y/n]
```

Then, the installation script creates the *RSS* database. In order to be able to do that, you are asked to provide root MySQL credentials.

```
'RSS' database is going to be created, Please introduce your mysql user and password with administrat
> user:
root
> Password:
```

The RSS uses the FIWARE Identity Manager for authenticating users. In this regard, the installation script asks you to provide valid OAuth2 credentials for your application. Additionally, it is also required to include the URL where the service is going to run (only host and port). You can find more details on how register your RSS instance in the IdM in section *OAuth2 Configuration*

```
----------------------------------------------------------------------
The RSS requires a FIWARE IdM to authenticate users. Please provide valid FIWARE credentials for this
> FIWARE Client ID:
{FIWARE CLIENT ID}
> FIWARE Client Secret:
{FIWARE CLIENT SECRET}
> Include the URL (including port) where the RSS is going to run:
http://[HOST]:[PORT]
```

During this installation process, the properties files are created in */etc/default/rss* using the provided information.

Manually Installing the RSS

Installing Basic Dependencies

The basic dependencies of the RSS can be easily installed using *apt-get* or *yum*, depending on the system.

- Ubuntu

```
    # apt-get install -y openjdk-7-jdk tomcat7 mysql-client mysql-server
```

- CentOS

```
    # yum install -y java-1.7.0-openjdk-devel tomcat
    # rpm -Uvh http://dev.mysql.com/get/mysql-community-release-el7-5.noarch.rpm
    # yum -y install mysql-community-server
    # /usr/bin/systemctl enable mysqld
```

Compiling Source Code

If you have downloaded the source code of the RSS from its GIT repository, you will need to compile the sources. To do that it is needed to have *maven* installed.

- Ubuntu

```
# apt-get install maven
```

- CentOS

```
# yum install maven
```

Once maven is installed, you can compile the source code executing the following command:

```
# mvn install
```

Note: In this case war files will be available at *fiware-rss/target/fiware-rss.war* and *rss-expendLimit/el-server/target/expenditureLimit.war*

Deploying the Software The RSS reads its properties from *database.properties* and *oauth.properties* files, located at */etc/default/rss*, so the first step for deploying the RSS is creating this directory.

```
# mkdir /etc/default/rss
```

Once this directory has been created, the next step is copying the properties files (located in the properties folder) to this location.

```
# cp properties/database.properties /etc/default/rss/database.properties
# cp properties/oauth.properties /etc/default/rss/oauth.properties
```

The concrete values contained in the properties files are described in *Configuration* section.

Finally, the last step is deploying the war files in Tomcat.

- Ubuntu

```
# cp fiware-rss.war /var/lib/tomcat7/fiware-rss.war
# cp expenditureLimit.war /var/lib/tomcat7/expenditureLimit.war
```

- CentOS

```
# cp fiware-rss.war /var/lib/tomcat/fiware-rss.war
# cp expenditureLimit.war /var/lib/tomcat/expenditureLimit.war
```

1.1.4 Configuration

This section explains how to configure the RSS. If you have used the provided script, you can skip this step as your properties files are already created. However, it is highly recomended to read this section in order to understand the existing preferences.

Database Configuration

Database connection in configured in */etc/default/rss/database.properties*, which has the following structure:

```
## Filter usage
database.url=jdbc:mysql://localhost:3306/RSS
database.username=root
database.password=root
database.driverClassName=com.mysql.jdbc.Driver
```

This file contains the following properties:

- **database.url**: URL where the MySQL database is located. it includes the host, the port, and the database name.
- **database.username**: User name used to access the database.
- **database.password**: Password of the user used to access the database.
- **database.dirverClassName**: Name of the driver class used to connect to the database

OAuth2 Configuration

TThe RSS uses the FIWARE Identity Manager. In this regard, it is needed to register the application in this system in order to retrieve valid credentials. For registering the application is required to provide the following information:

- A name.
- A description.
- The URL of the RSS. Must be something like *http://[HOST]:[PORT]/fiware-rss/*
- The callback URL of the RSS. Must be something like *http://[HOST]:[PORT]/fiware-rss/callback?client_name=FIWAREClient*

OAuth2 information is configured in */etc/default/rss/oauth.properties*, which has the following structure:

```
############## IDM configuration ################
config.baseUrl=https://account.lab.fiware.org
config.logoutPath=/auth/logout
config.client_id=
config.client_secret=
config.callbackURL=http://localhost:8080/fiware-rss/callback
config.callbackPath=/callback
config.authorizeUrl=/oauth2/authorize
config.accessTokenUrl=/oauth2/token
config.userInfoUrl=/user?access_token=
config.grantedRole=Provider
```

This file contains the following properties:

- **config.baseUrl**: URL of the FIWARE Identity Manager used to authenticate users.
- **config.logoutPath**: URL path used for logging out users from the RSS.
- **config.client_id**: ID of the application in the identity manager.
- **config.client_secret**: Secret of the application in the identity manager.
- **config.callbackURL**: URL of the RSS used to receive authorization callbacks.
- **config.callbackPath**: URL path of the RSS used to receive authorization callbacks.
- **config.authorizeUrl**: URL path of the identity manager used for making authorization requests.
- **config.accessTokenUrl**: URL path of the identity manager used for making access token requests.
- **config.userInfoUrl**: URL path of the identity manager used for retrieving user information.

- **config.grantedRole**: Role defined in the application in the identity manager for identifying admins of the RSS.

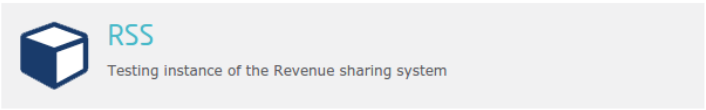
1.1.5 Sanity check procedures

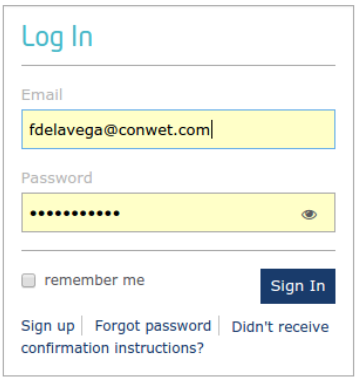
The Sanity Check Procedures are those activities that a System Administrator has to perform to verify that an installation is ready to be tested. Therefore there is a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and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

End to End testing

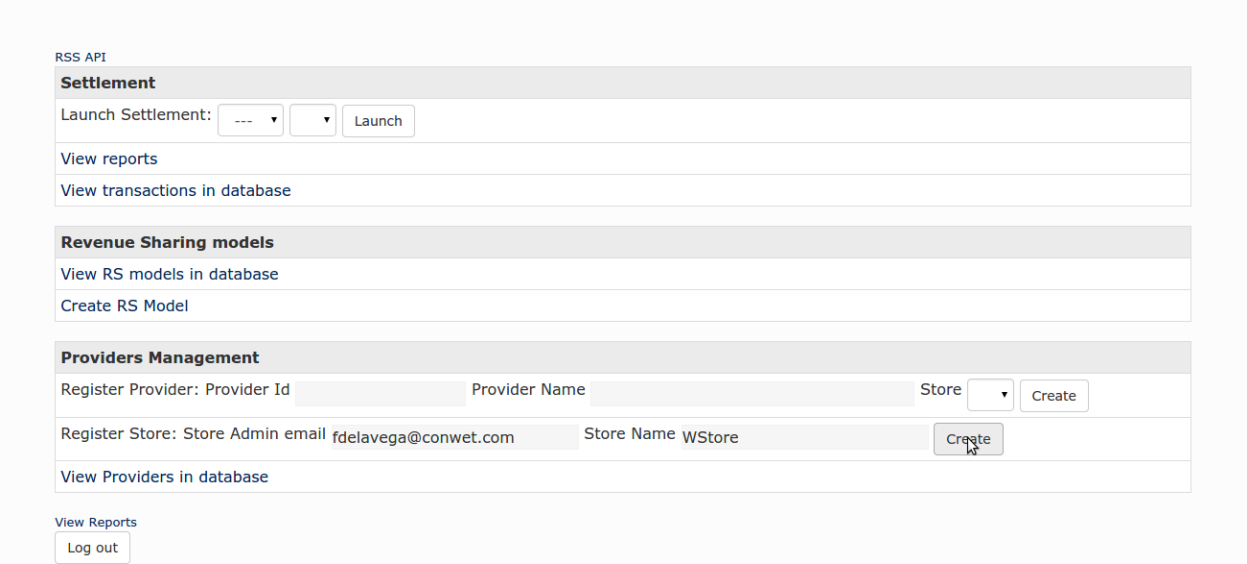
Although one End to End testing must be associated to the Integration Test, we can show here a quick testing to check that everything is up and running. The following process can be performed by a system administration in order to verify the installation.

1. Access the URL of the RSS (http://HOST:PORT/fiware-rss). You should be redirected to the IdM in order to login.

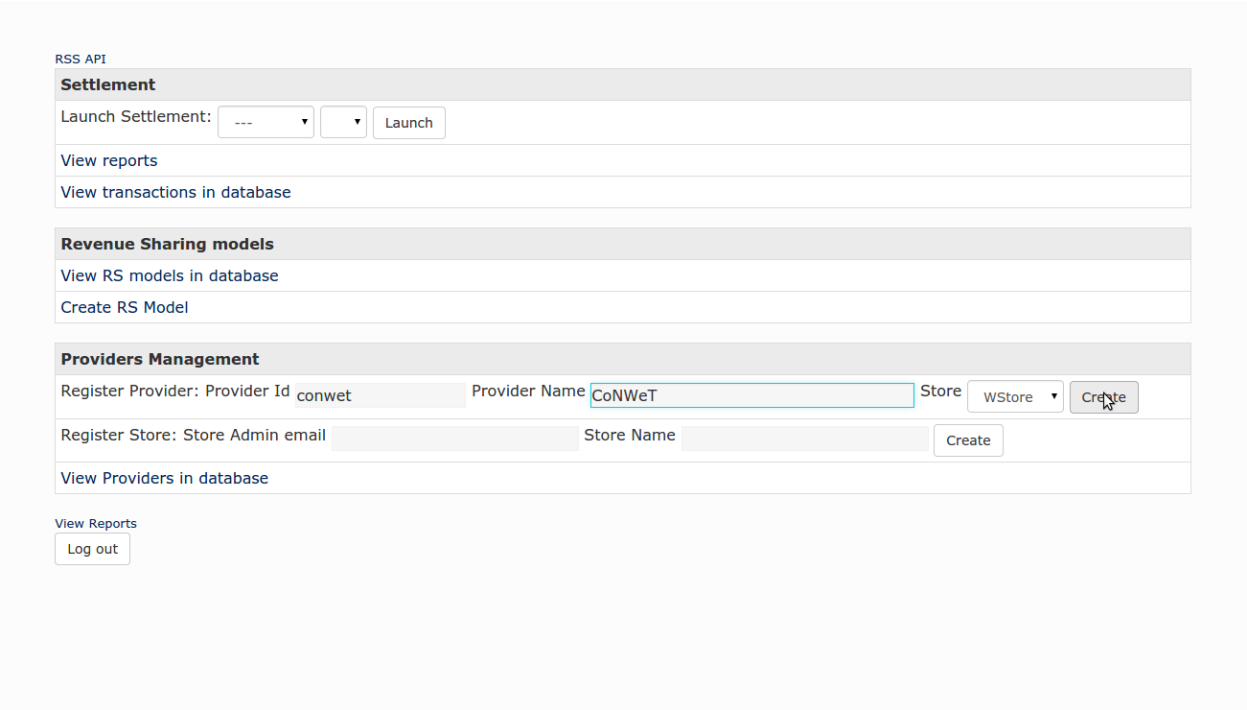




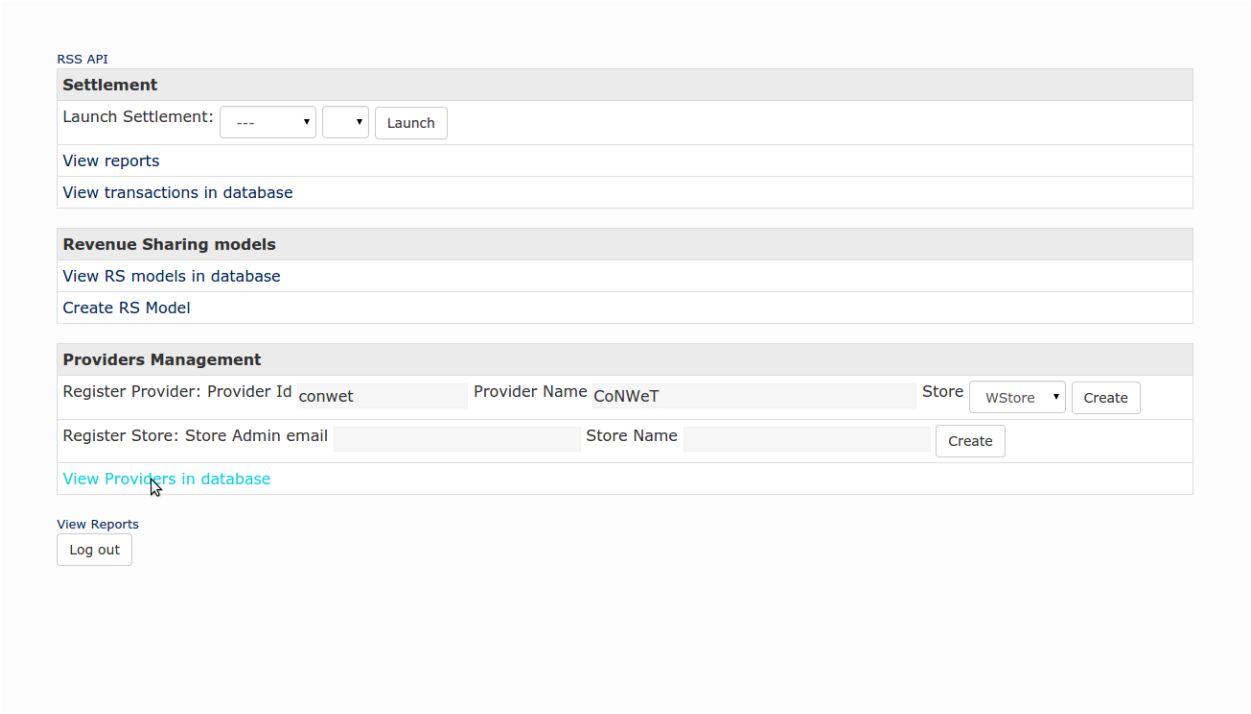
2. Register a new Store, providing the admin email and a display name.

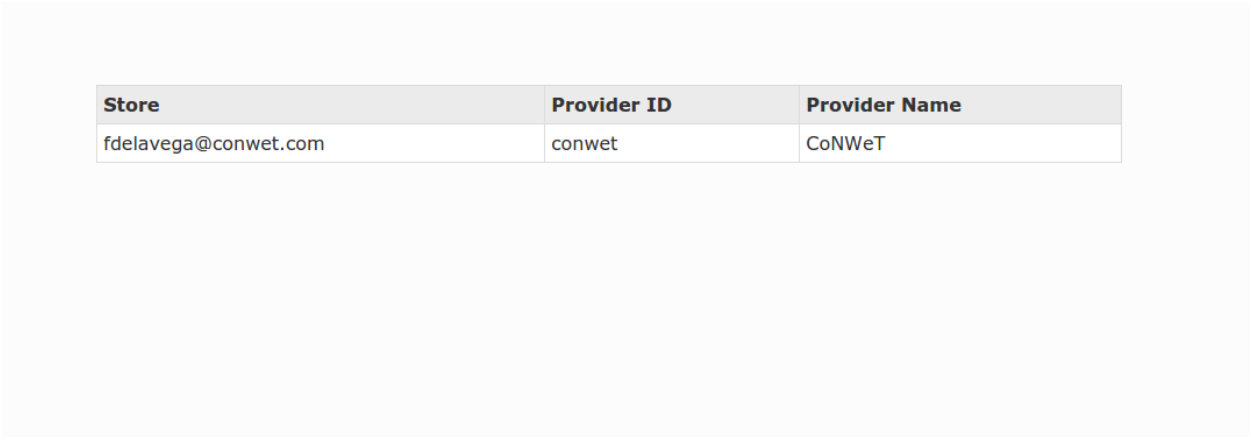


3. Register a new provider, including an id and a display name, and selecting the previously registered store.



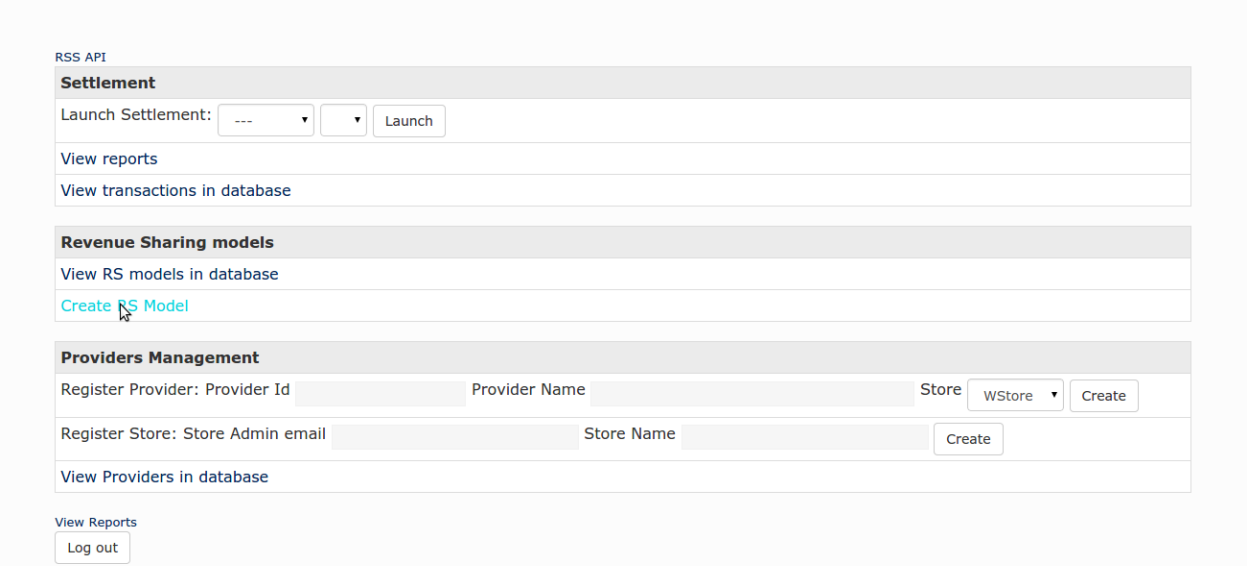
4. Verify that the provider has been created by clicking on *View Providers in database*.



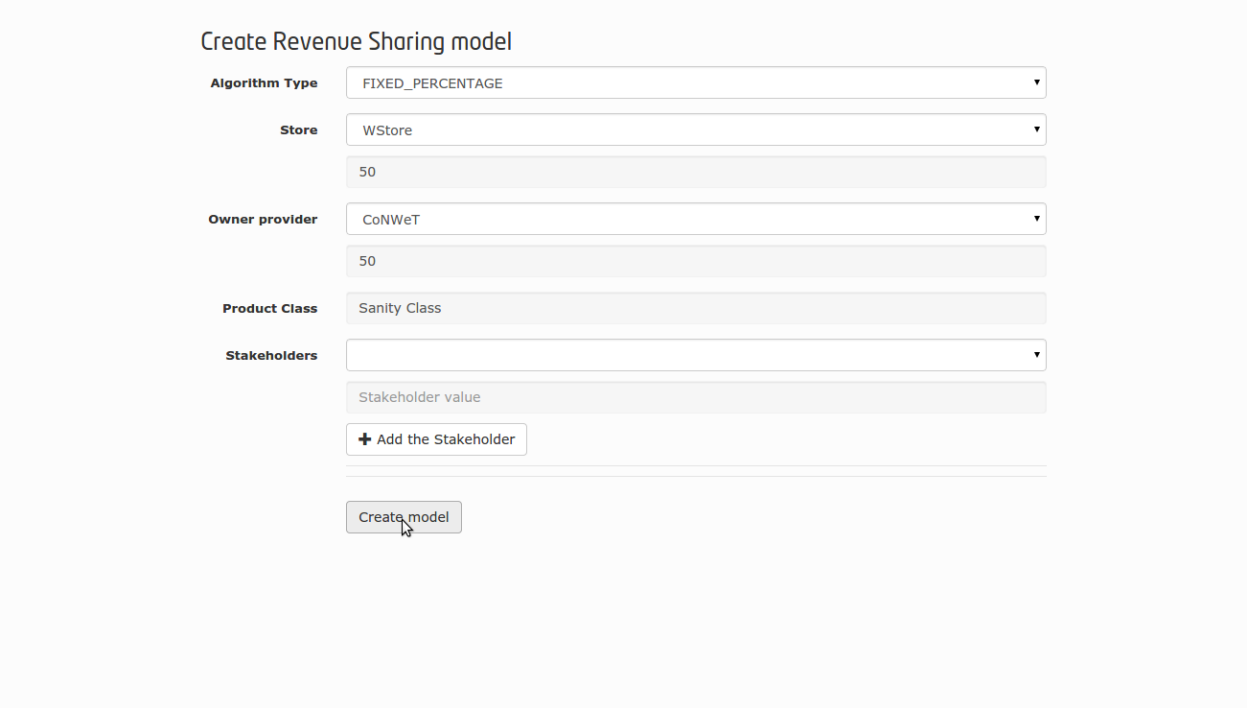


Store	Provider ID	Provider Name
fdelavega@conwet.com	conwet	CoNWeT

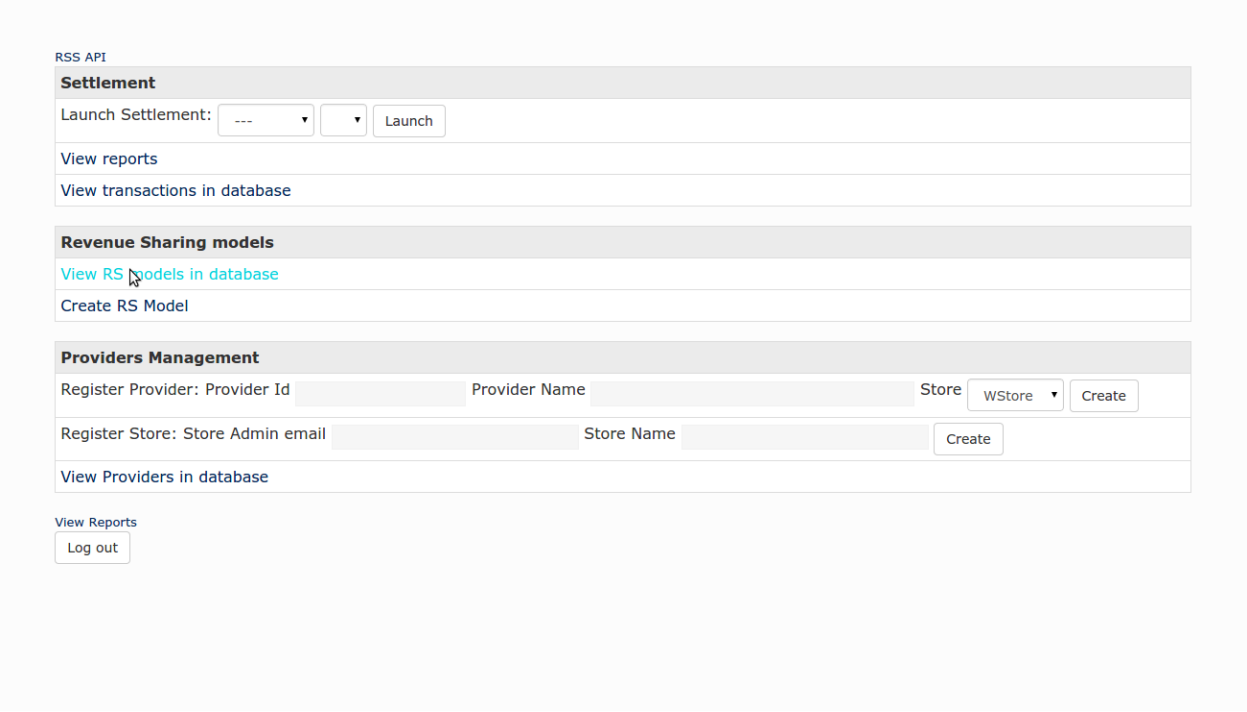
5. Go back to the home page and click *Create RS model*.



6. Include a percentage value for the store and for the provider (The total must be equal to 100). Provide a product class for identifying the model and click on *Create*.



7. Go back to the home page, and verify that the model has been created clicking on *View RS models in database*.



Algorithm	Product Class	Store	Store Value	Provider	Provider Value
FIXED_PERCENTAGE	Sanity Class	fdelavega@conwet.com	50	conwet	50

List of Running Processes

You can execute the command `ps -ax | grep 'tomcat\|mysql'` to check that the Tomcat web server and the MySQL database. It should show a message text similar to the following:

```
23397 ?        Ssl    0:00 /usr/sbin/mysqld
24459 ?        Sl     1:15 /usr/lib/jvm/java-7-openjdk-amd64/bin/java -Djava.util.logging.config.file
24921 pts/0    S+     0:00 grep --color=auto tomcat\|mysql
```

Network interfaces Up & Open

To check whether the ports in use are listening, execute the command `netstat -ntpl`. The expected results must be somehow similar to the following:

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN      -
tcp6       0      0 :::8080                 :::*                    LISTEN      -
```

Databases

In order to check that MySQL is running and that the *RSS* database has been set up, MySQL client can be used.

- Open MySQL Client and enter *RSS* database:

```
$ mysql -u root -proot RSS
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 143
Server version: 5.5.41-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

- Check that tables has been created:

```
mysql> show tables;
+-----------------------------+
| Tables_in_RSS               |
+-----------------------------+
| bm_country                  |
| bm_currency                 |
| bm_customer_type            |
| bm_language                 |
| bm_methods_of_payment       |
| bm_ob                       |
| bm_ob_country               |
| bm_ob_mop                   |
| bm_paymentbroker            |
| bm_pb_mop                   |
| bm_price_point              |
| bm_product                  |
| bm_product_vs_ob            |
| bm_servdeploy_mop           |
| bm_service                  |
| bm_service_deployment       |
| bm_service_product_type     |
| dbe_aggregator              |
| dbe_aggregator_appprovider  |
| dbe_appprovider             |
| dbe_appprovider_application |
| dbe_expend_control          |
| dbe_expend_limit            |
| dbe_system_properties       |
| dbe_transaction             |
| set_revenue_share_conf      |
| share_conf_provider         |
+-----------------------------+
27 rows in set (0.00 sec)
```

Note: This Test is asumming that you are using the user *root* with password *root* and a database called *RSS*.

1.1.6 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.

The first step that can be follow in order to locate a problem is running the tests of the software:

```
$ mvn test -fae
```

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

- The logs of the RSS and RSModels API are stored in {Apache Tomcat Installation}/logs/fiware-rss/main.logs
- The logs of the Balance Accumulate and Limit Management API are stored in {Apache Tomcat Installation}/logs/expendLimit/expendLimit.log

Resource availability

The resource load of the RSS strongly depends on the number of concurrent requests received as well as on the free main memory and disk space. In this regard, the application will run correctly if the system adheres to the minimal requirements.

Resource consumption

There are two main processes consuming resources:

- MySQL Server
- 2 Apache Tomcat Server

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.

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.

1.2 User and Programmer Guide

1.2.1 Introduction

This page contains the User and Programmer guide of the reference implementation of the Revenue Settlement and Sharing Sytem GE.

1.2.2 User Guide

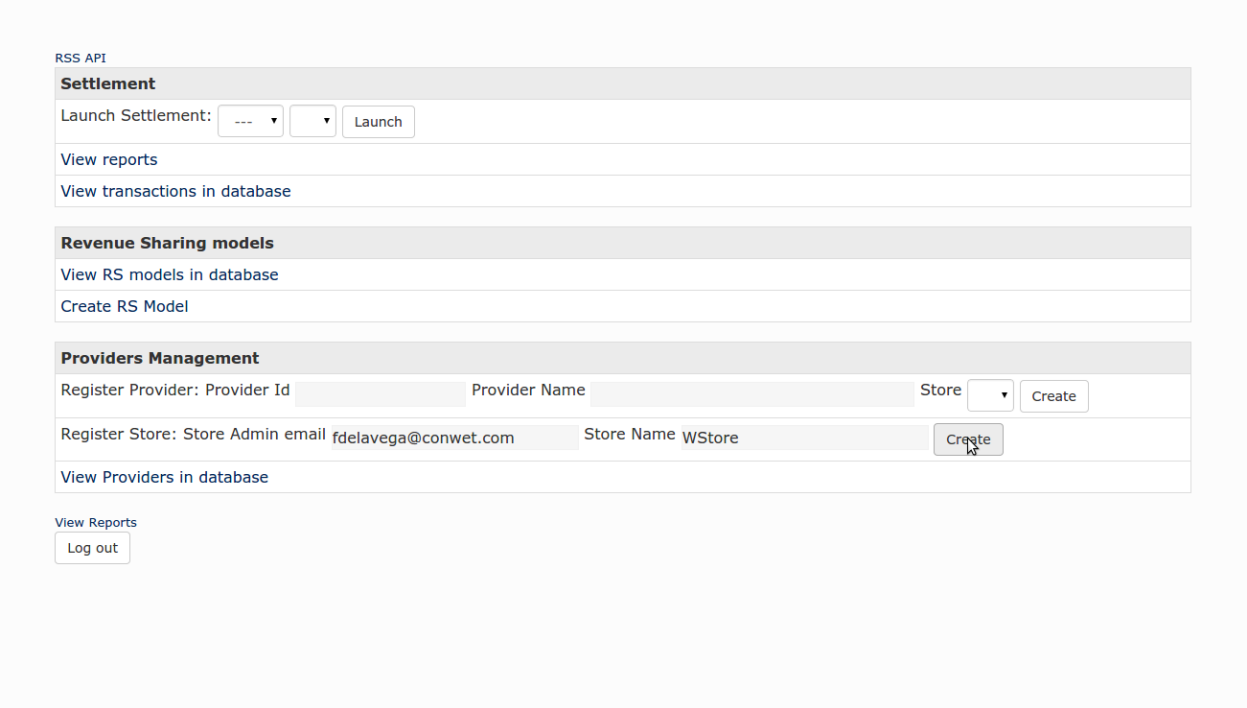
The RSS is a software which provides pure backend functionality to other applications (e.g. Generic Enablers or end user facing applications). However, a web interface is offered in order to allow RSS and store administrators interact with the back-end functionality. This section covers the functionality of the administration interface.

The administration interface only can be accesed by those users that have been regsitered as administrators of the RSS in the FIWARE Identity Manager (Using roles). Aditionally, administrators of the stores already registered can also access this interface. In this case, they will be only allowed to view and use it limited to the scope of the assets that belongs to their store.

Register Store

The administration interface allows to register new stores, which are charging information sources. To do that, it is required to include the email of the user of the FIWARE identity manager who admins the concrete store instance, and a display name for the store.

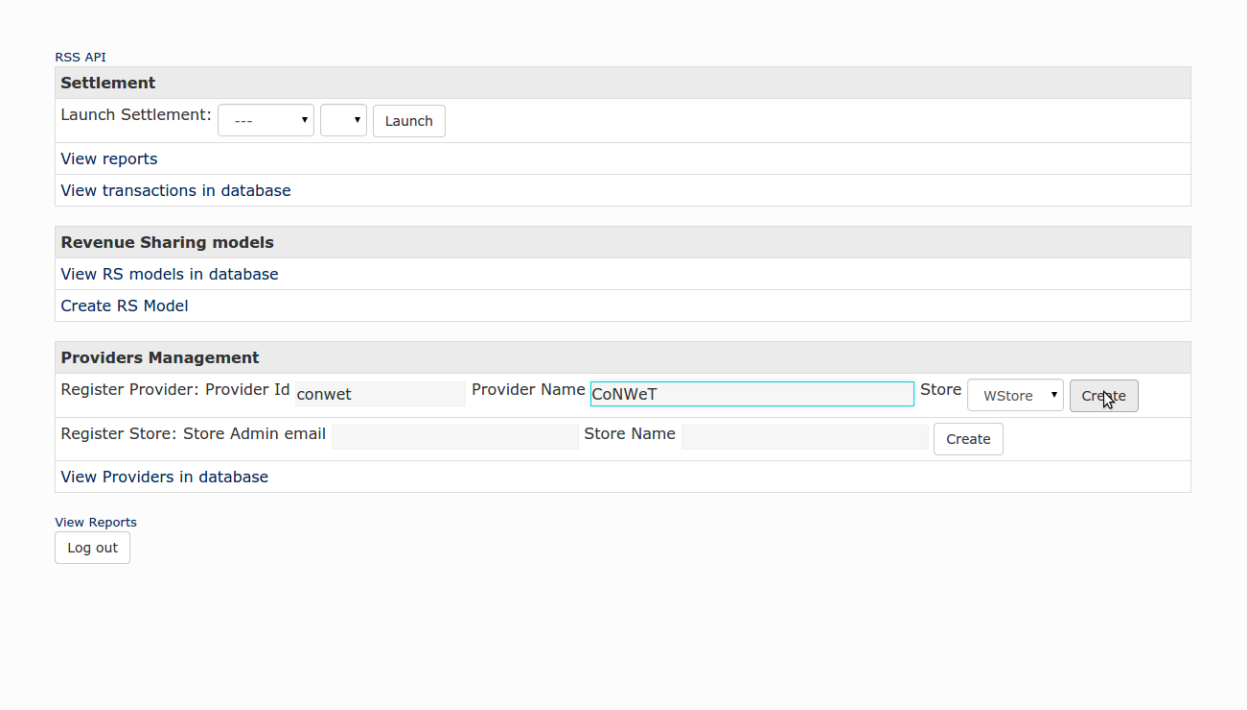
Once a Store is registered, it is possible to send charging information to the RSS in order to perform the revenue sharing of its providers.



Note: Store administrators cannot perform this action

Register Provider

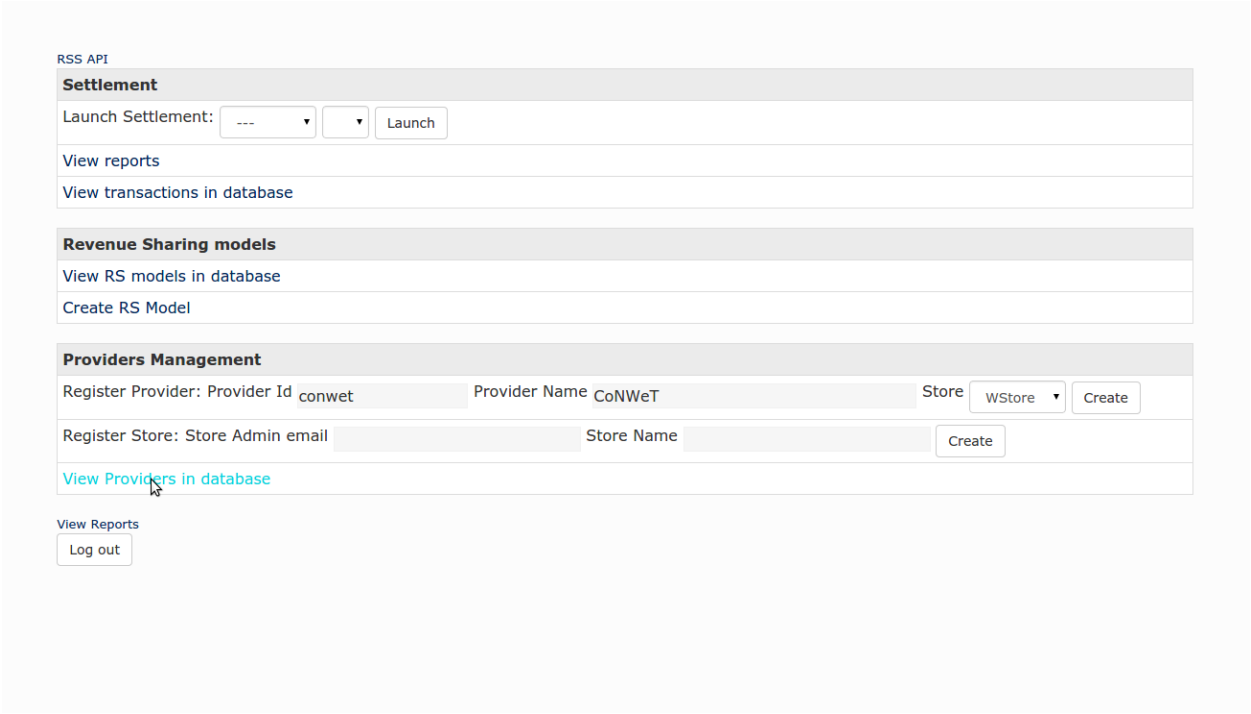
Additionally, it is possible to use the administration interface to register providers of a given store. To do that, it is necessary to provide an id, a display name, and select a store.

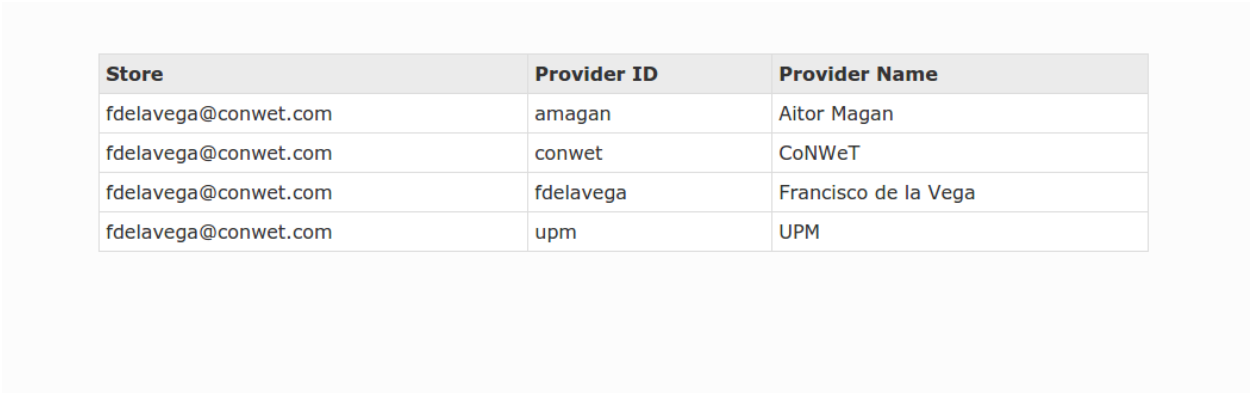


Note: Store administrators only can register providers of the store they are owning

View Providers

It is possible to view the providers already registered in the RSS using the administration interface. To view existing providers click on *View Providers in database*.



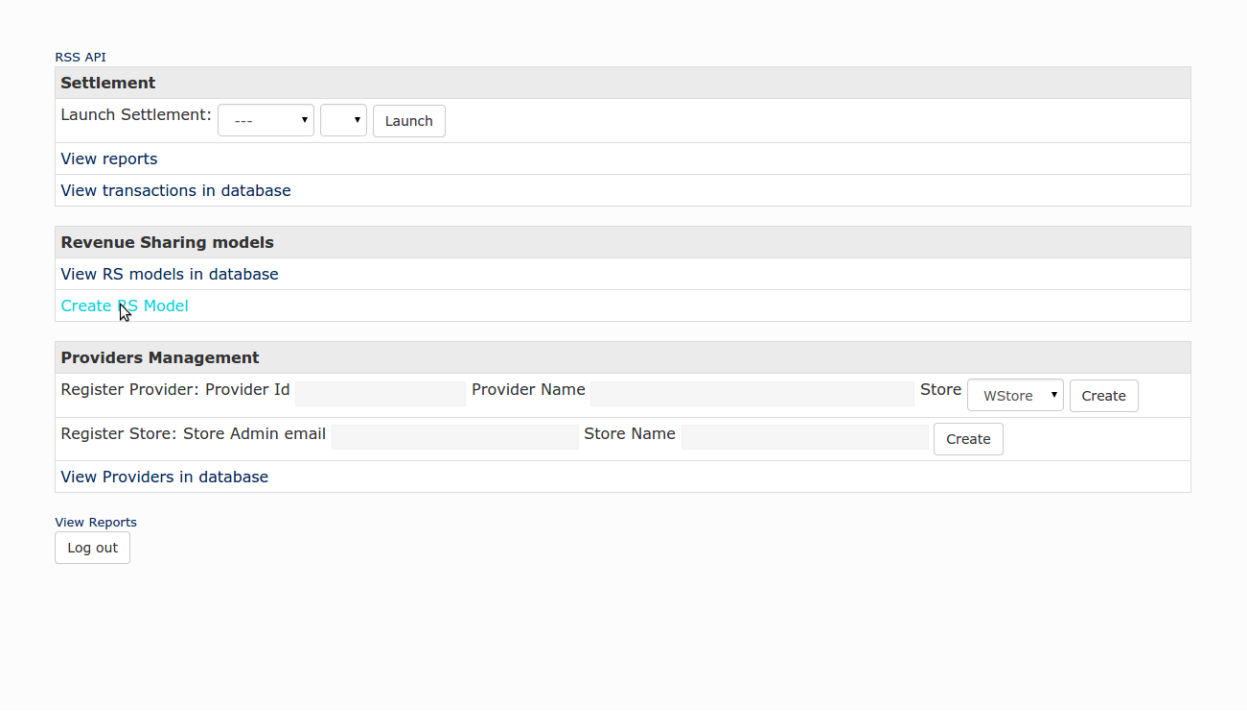


Store	Provider ID	Provider Name
fdelavega@conwet.com	amagan	Aitor Magan
fdelavega@conwet.com	conwet	CoNWeT
fdelavega@conwet.com	fdelavega	Francisco de la Vega
fdelavega@conwet.com	upm	UPM

Note: Store administrators only can view providers of the store they are owning

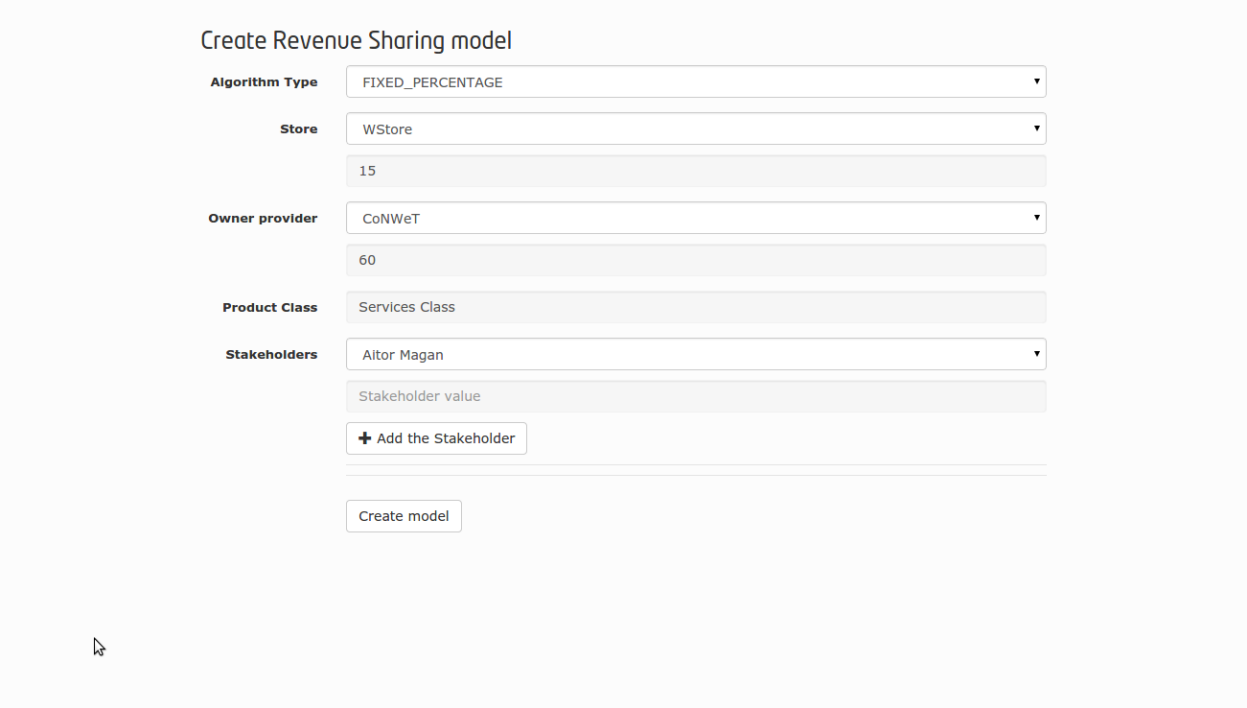
Create Revenue Sharing Model

Revenue Sharing Models in the RSS specify how the revenues generated by a set of offerings must be distributed. In this regard, the administration interface allows to create revenue sharing models by clicking on *Create RS Model*.



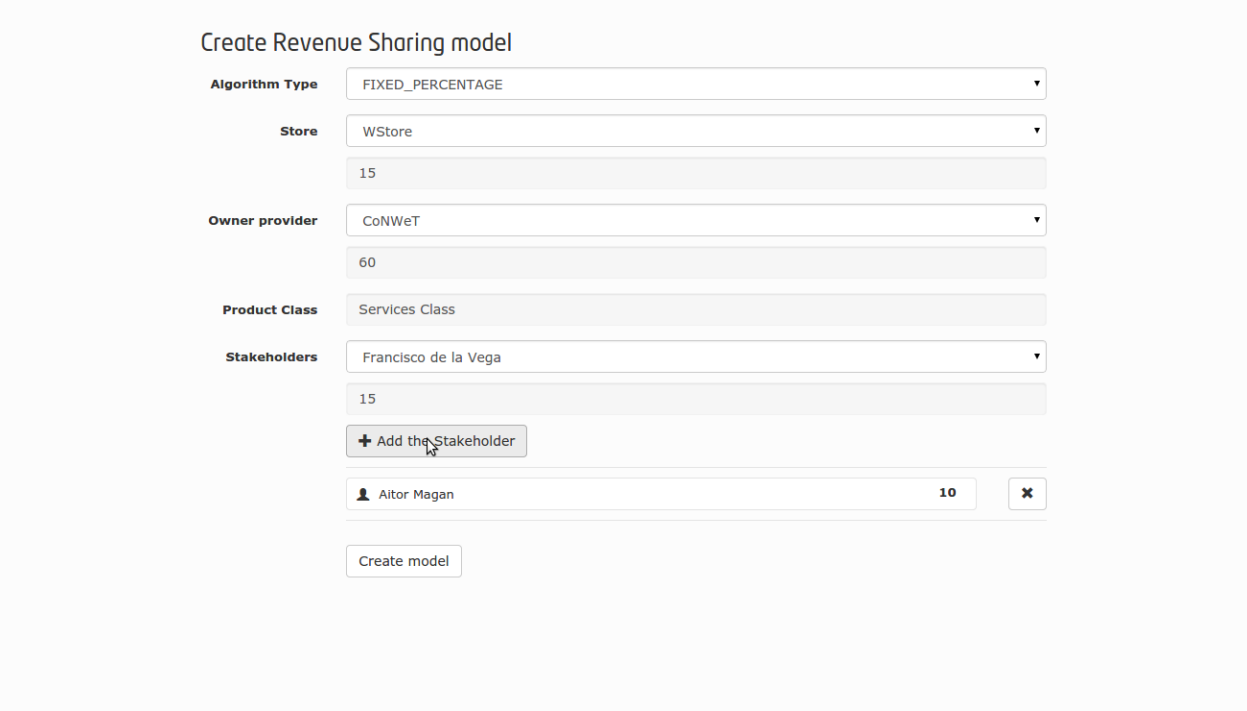
The first step for creating a RS model is selecting the Store where the charging information is being generated, and specify the percentage of the revenue that belongs to their owners (*Store Value*). Then, it is necessary to specify the provider, who owns the offering or group of offerings whose revenues are going to be distrubuted using the current model, and provide the percentage of the revenue that belongs to her (*Provider Value*).

Next, it is required to fill the *Product Class*. This value is used as the identifier of the RS model and identifies an offering or group of offerings.

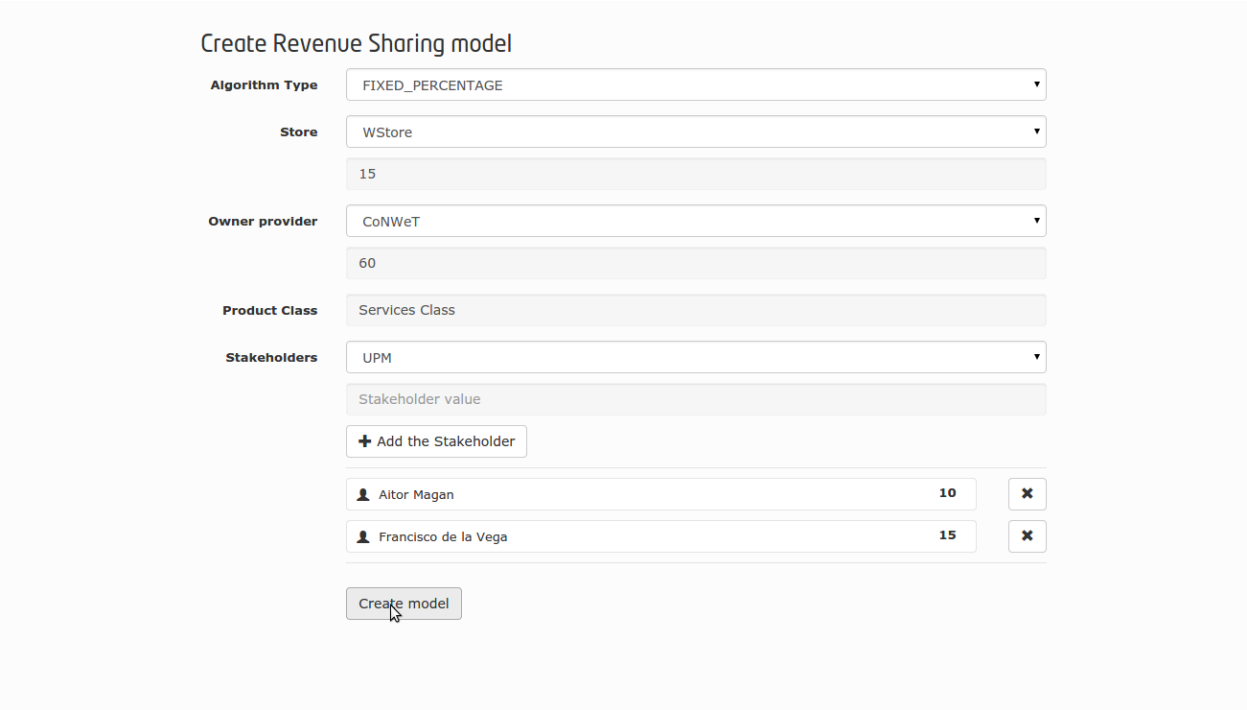


There are some offerings, that are composed of services belonging to different providers. To deal with that, it is

possible to specify additional stakeholders to the revenue sharing model including their percentage of the revenues.



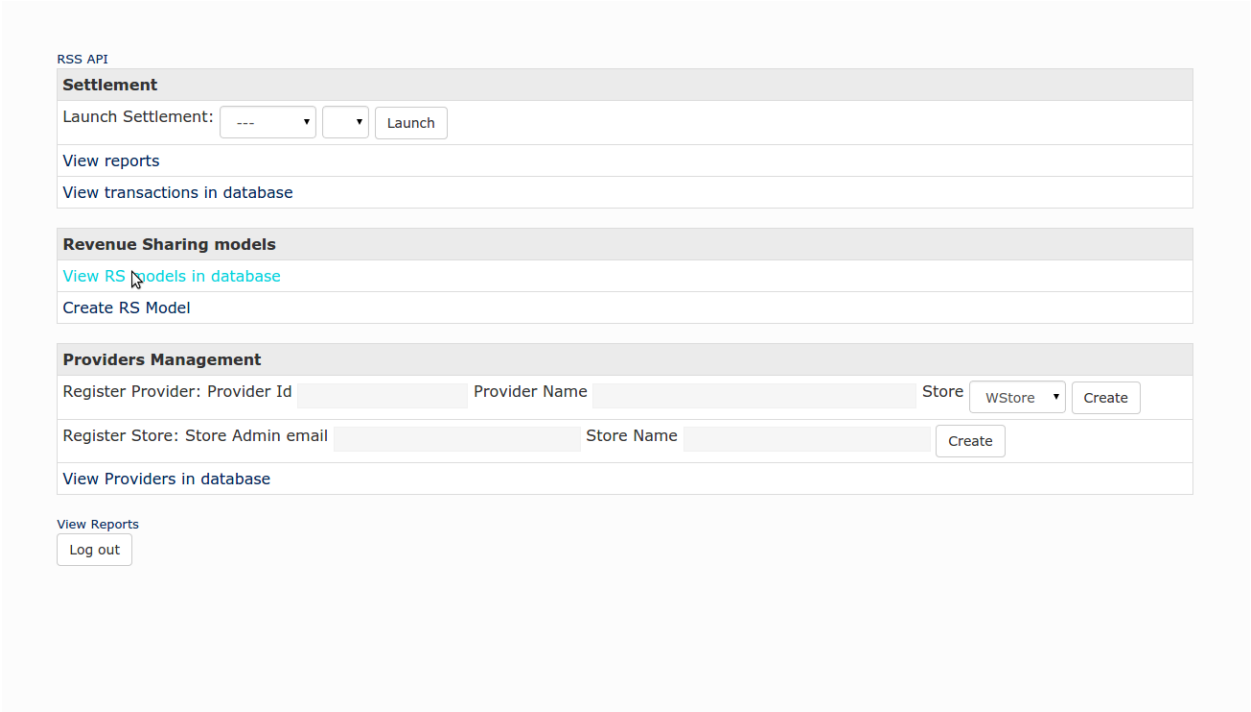
Finally, the model is created by clicking on *Create Model*



Note: Store administrators only can create RS models for the providers of the store they are owning

View Revenue Sharing Models

The administration interface allows to view existing RS models by clicking on *View RS models in database*.



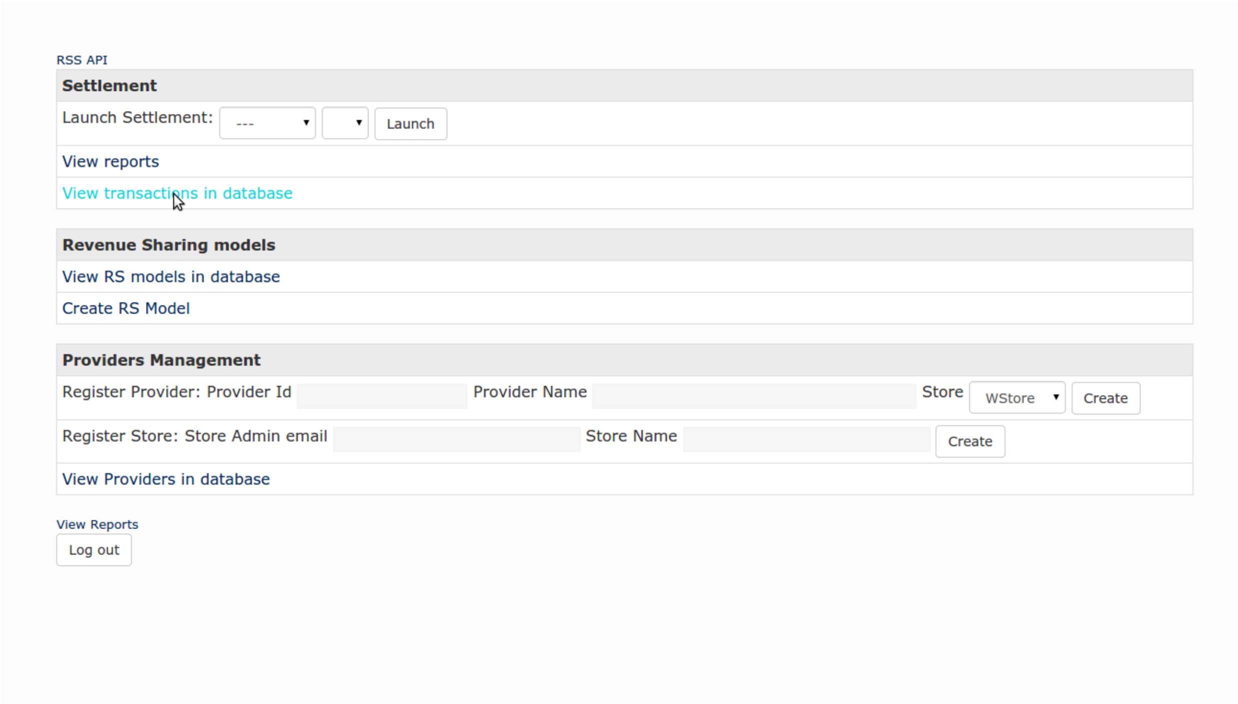
Algorithm	Product Class	Store	Store Value	Provider	Provider Value
FIXED_PERCENTAGE	Sanity Class	fdelavega@conwet.com	50	conwet	50

Algorithm	Product Class	Store	Store Value	Provider	Provider Value
FIXED_PERCENTAGE	Services Class	fdelavega@conwet.com	15	conwet	60
Stakeholder			**Stakeholder Value**		
fdelavega			15		
amagan			10		

Note: Store administrators only can view RS models of the store they are owning

View Transactions

As stated, the RSS provides pure backend functionality. In this respect, the different transactions with charging information generated in the stores are fed to the RSS via API. Nevertheless, the administration interface allows to view the existing (not aggregated) transactions. To do that, click on *View transactions in database*.



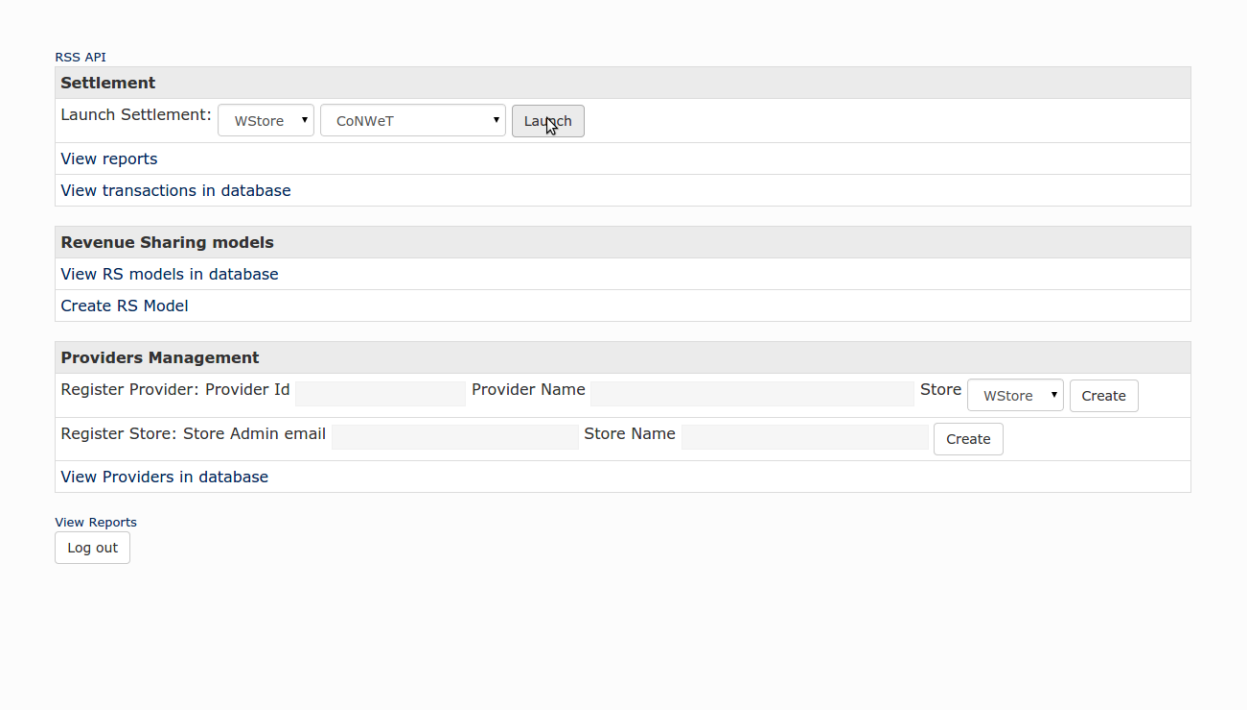
Product Class	Provider ID	User ID	Tx Type	App Id	Request Time	Ref Code	Amount	Tax Amount	Currency	Description
Services Class	conwet	aarranz	C	Service app	2015-10-09T19:00:01.000Z	2015/10/01001	100	10	EUR	Usage charge
Services Class	conwet	aarranz	C	Service app	2015-10-09T19:00:01.000Z	2015/10/01001	100	10	EUR	Usage charge
Services Class	conwet	aarranz	C	Service app	2015-10-09T19:00:01.000Z	2015/10/01001	100	10	EUR	Usage charge
Services Class	conwet	aarranz	C	Service app	2015-10-09T19:00:01.000Z	2015/10/01001	100	10	EUR	Usage charge
Services Class	conwet	aarranz	C	Service app	2015-10-09T19:00:01.000Z	2015/10/01001	100	10	EUR	Usage charge
Services Class	conwet	aarranz	C	Service app	2015-10-09T19:00:01.000Z	2015/10/01001	60	10	EUR	Usage charge
Services Class	conwet	aarranz	C	Service app	2015-10-09T19:00:01.000Z	2015/10/01001	80	10	EUR	Usage charge
Services Class	conwet	aarranz	R	Service app	2015-10-09T19:00:01.000Z	2015/10/01001	40	10	EUR	Usage charge

Launch Settlement

Transactions stored in the RSS contain charging information that is used to calculate the revenue sharing using the corresponding RS model. In this regard, the administration interface allows to launch the process that aggregates charging info and calculates the revenue sharing (The Settlement process).

The interface allows to launch the settlement for all the pending transactions, for the all the pending transactions generated in a store, or for all the pending transactions belonging to a concrete provider.

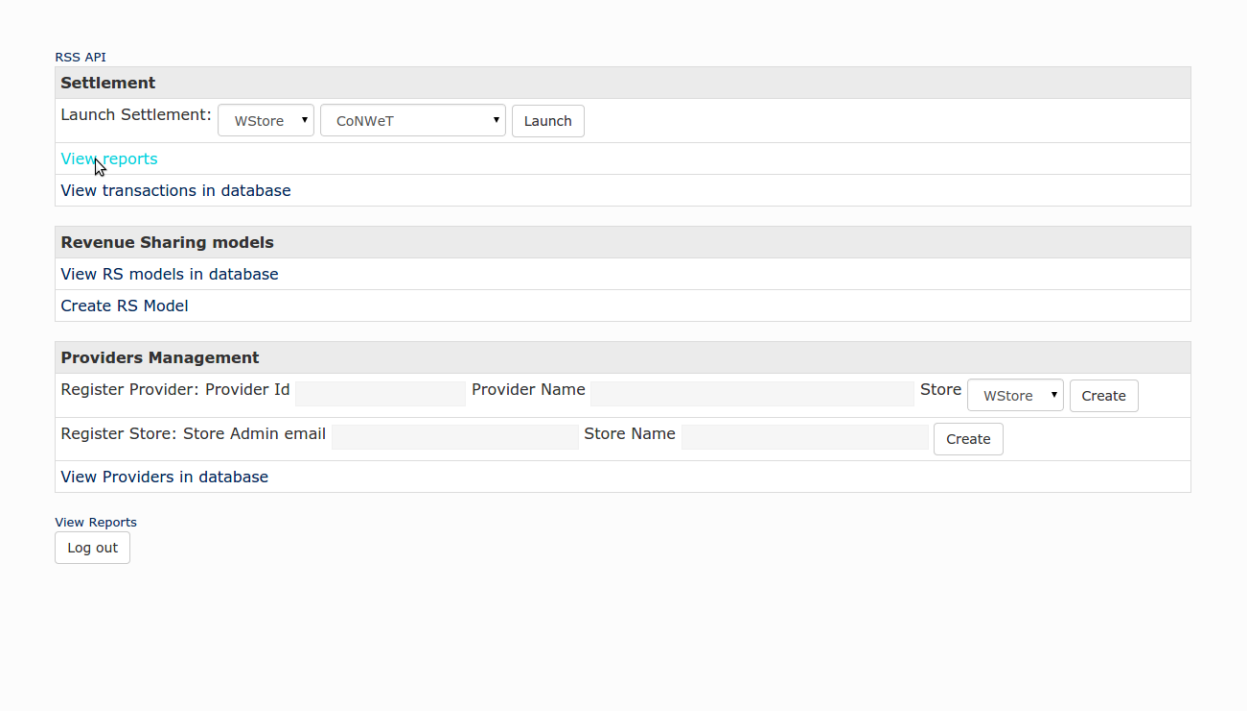
This can be done in the settlement section by selecting the store, the provider, and clicking *Launch*

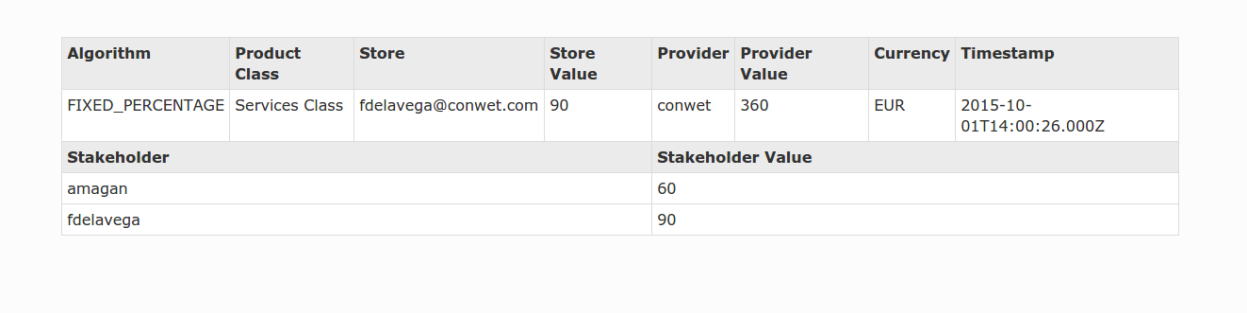


Note: Store administrators only can launch the process for their store or its providers.

View Revenue Sharing Reports

The settlement process generates a set of reports that specify how revenues must be distributed. This reports can be viewed in the admin interface by clicking on *View reports*.





Algorithm	Product Class	Store	Store Value	Provider	Provider Value	Currency	Timestamp
FIXED_PERCENTAGE	Services Class	fdelavega@conwet.com	90	conwet	360	EUR	2015-10-01T14:00:26.000Z

Stakeholder	Stakeholder Value
amagan	60
fdelavega	90

1.2.3 Programmer Guide

The RSS offers its functionality as a REST API that can be used by developers to integrate revenue sharing functionality with their own solutions. This section covers the main aspects of the RSS API and the actions that can be performed with it.

This section is not a detailed reference of the RSS API. You can find this documentation in:

- Apiary
- GitHub pages

It is important to remark that the RSS is integrated with the FIWARE Identity Manager for authenticating and authorizing users. In this regard, all the requests made to the RSS API must include an *Authorization* header containing a valid OAuth2 token.

Stores and Providers Management

The first step for a developer to integrate the RSS APIs is integrating the aggregator (Store admin) and provider APIs. These APIs are available at:

- Aggregator: `/fiware-rss/rss/aggregators`
- Provider: `/fiware-rss/rss/providers`

Both resources support `POST` and `GET` operations for creating an retrieving entities using JSON.

Following you can find the JSON serialization of an aggregator.

```
{
    "aggregatorName": "WStore",
    "aggregatorId": "fdelavega@conwet.com"
}
```

The aggregator model contains the following fields:

- **aggregatorName** - Display name of the given aggregator
- **aggragtorId** - Email used to identify the user that is authorized to send changing information (typically an admin of a Store instance)

Following you can find the JSON serialization of a provider.

```
{
    "aggregatorId": "fdelavega@conwet.com",
    "providerId": "conwet",
    "providerName": "CoNWeT"
}
```

The provider model contain the following fields:

- **aggregatorId** - Aggregator email that identifies the charging information source (Store instance)
- **providerId** - Id of the given provider. Note that this id only needs to be unique in the context of an aggregator, so the same providerId can be used for different providers if the aggregator is different
- **providerName** - Display name of the given provider

Revenue Sharing Models Management

To be able to calculate the revenue sharing, it is required to have revenue sharing models. RS models are managed in the RSS API using the resource:

- `/fiware-rss/rss/models`

These models can be created using a `POST` request, and retrieved with a `GET` request with JSON content type. Following you can find the JSON serialization of a revenue sharing model.

```
{
    "ownerProviderId": "fdelavega",
    "ownerValue": 60,
    "productClass": "orionServices",
    "algorithmType": "FIXED_PERCENTAGE",
    "aggregatorId": "fdelavega@conwet.com",
    "aggregatorValue": 20,
    "stakeholders": [
        {
            "stakeholderId": "aarranz",
            "modelValue": 20
        }
    ]
}
```

These models manage the following fields:

- **ownerProviderId** - Provider Id of the owner of the model. This provider is the owner of the application and services whose revenues will be distributed using the Revenue Sharing Model
- **ownerValue** - Value of the owner provider in the Revenue Sharing Model. The semantics of this field depends on the algorithm specified, for example if the algorithm is a fixed precentage, this field will contain the percetage of the revenue that belongs to the owner provider.
- **productClass** - Id of the Revenue Sharing Model. This field represents a group of services or applications whose revenues are distributed in the same way
- **algorithmType** - ID of the algorithm that is used in this model
- **aggregatorId** - Id of the aggregator that represents the Store instance where the applications and services are offered, and thus, must receive part of the revenues
- **aggregatorValue** - Value of the aggregator in the Revenue Sharing Model
- **stakeholders - List of providers that are stakeholders of the applications and services included in a given product class, an**
  - **stakeholderId** - provider Id of the Stakeholder
  - **modelValue** - Value of the stakeholder in the Revenue Sharing Model

Transactions Management

Once the RSS has RS models, it is needed to receive charging information. The different transactions that contain the charging information are managed in the RSS using CDR (Charging Detailed Records) documents. CDRs are managed in the RSS using the resource:

- `/fiware-rss/rss/cdrs`

These CDRs are created using a `POST` request and retrieved using a `GET` request with JSON content type. Following, you can find the JSON serialization of a CDR.

```
{
    "cdrSource": "fdelavega@conwet.com",
    "productClass": "orionServices",
    "correlationNumber": 112,
    "timestamp": "2015-07-15T19:00:01.000Z",
    "application": "OrionStarterKit",
    "transactionType": "C",
    "event": "use",
    "referenceCode": "555b079d8e05ac213ff15827",
    "description": "Usage of OrionStarterKit Offering",
    "chargedAmount": 10,
    "chargedTaxAmount": 3,
    "currency": "EUR",
    "customerId": "amagan",
    "appProvider": "fdelavega"
}
```

CDRs contain the following fields:

- **cdrSource** - Id of the aggregator that represent the Store instance that is generating the charging information
- **productClass** - Product Class used to identify the revenue sharing model that will be used to distribute the revenues generated in the current transaction
- **correlationNumber** - Correlation number of the transaction

- **timestamp** - Timestamp of the transaction
- **application** - Textual field with the id of the application or service that generates the transaction
- **transactionType** - Type of transaction. This field can contain "C" for charges and "R" for refunds
- **event** - Textual field that describes the event that generated the transaction (e.g pay-per-use)
- **referenceCode** - Reference code that identifies the purchase in the Store instance that generates the transaction
- **description** - Textual description of the transaction
- **chargedAmount** - Part of the total charged amount to be distributed. The total amount charged to the customer includes also the field chargedTaxAmount
- **chargedTaxAmount** - Part of the total charged amount that are taxes. The total amount charged to the customer includes also the field chargedAmount
- **currency** - Currency of the transaction
- **customerId** - Id of the customer that acquires the given service or application
- **appProvider** - provider Id of the owner of the charged applications or services

Settlement Management

If some transactions have been received and there are RS models able to manage them, then, it is possible to launch the settlement process. The settlement process is launched using the resource:

- `/fiware-rss/rss/settlement`

To launch the process is needed to make a `GET` request using query strings to filter the scope:

- None: if no query string is provided the settlement process is launched for all the pending transactions.
- aggregatorId: Id of a given aggregator. If this query string is provided the settlement process is launched only for those pending transactions generated in the given store.
- providerId: Id of a given provider. If this query string is provided the settlement process is launched only for those pending transactions generated in the given store, and belonging to the given provider.
- productClass: Product class of the RS models. If this query string is provided the settlement process is launched only for those pending transactions generated in the given store, belonging to the given provider, and with the given product class.

The result of the settlement process are a couple of reports that specify the concrete amount that has to paid to the concrete stakeholders involved. The reports can be accesed using the resource:

- `/fiware-rss/rss/settlement/reports`

RS Reports can be retrieved using a `GET` request. Following you can find a report serialized in JSON format.

```
{
    "ownerProviderId": "fdelavega",
    "ownerValue": 4578,
    "productClass": "orionServices",
    "algorithmType": "FIXED_PERCENTAGE",
    "aggregatorId": "fdelavega@conwet.com",
    "aggregatorValue": 3000,
    "currency": EUR,
    "timestamp": "2015-07-15T19:00:01",
    "stakeholders": [
        {
            "stakeholderId": "aarranz",
```

```
            "modelValue": 2500
        }
    ]
}
```

These reports contain the following fields:

- **ownerProviderId** - Provider Id of the owner of the model. This provider is the owner of the application and services whose revenues has been aggregated.
- **ownerValue** - Amount that has to be paid to the provider.
- **productClass** - Id of the Revenue Sharing Model that have been applied. This field represents a group of services or applications whose revenues are distributed in the same way
- **algorithmType** - ID of the algorithm that have been used.
- **aggregatorId** - Id of the aggregator that represents the Store instance where the applications and services are offered, and thus, must receive part of the revenues
- **aggregatorValue** - Amount that has to be paid to the store owners.
- **currency**: Currency of the different amounts.
- **timestamp**: Timestamp of the reports.
- **stakeholders - List of providers that are stakeholders of the applications and services included in a given product class, an**
  - **stakeholderId** - provider Id of the Stakeholder
  - **modelValue** - Amount that has to be paid to the concrete stakeholder