D3.17

D3.17 - Data Quality Assessment Tools Implementation (Release 3.1)

Work package:	**WP3**
Version number:	**Version 3.0**
Dissemination level:	**PU**
Date:	**3/11/2016**

7th RTD Framework Programme
Directorate General for Communications Networks, Content & Technology
Cooperative Systems for energy efficient and sustainable mobility (FP7-ICT-2011-6.7)
Contract Type: Collaborative project
Grant agreement no.: 318485

Version Control

Version history			
Version	**Date**	**Main author**	**Summary of changes**
0.1	16/12/2013	Francesco Alesiani	TOC
0.6	31/01/2014	Francesco Alesiani	Reviewed before partner submission
0.8	02/05/2014	Francesco Alesiani	Updated with implementation of R1
0.9	15/12/2014	Francesco Alesiani	New FVD service specification (draft)
1.0	05/05/2015	Francesco Alesiani	Review of the deliverable for release II
1.1	28/02/2016	Francesco Alesiani	Review of the deliverable for release III
1.2	17/03/2016	Jihed Khiari	Data Quality for FCD sensors
1.3	27/06/2016	Francesco Alesiani	New authorization API
2.0	30/09/2016	Francesco Alesiani	D3.17 formatting
2.3	17/10/2016	Francesco Alesiani	Version Submitted for peer-review
2.4	24/10/2016	Francesco Alesiani	Inclusion of comments from peer-review
3.0	03/11/2016	Jihed Khiari	Final version
	Name		**Date**
Prepared	Francesco Alesiani, Jihed Khiari		17/10/2016
Reviewed	Alessandro Tadei (INFOBLU), Silvia Alén González (CTAG), Yanying Li (ERTICO)		02/11/2016
Authorised	Rasmus Lindholm		03/11/2016
Circulation			
Recipient		**Date of submission**	
European Commission		03/11/2016	
Project partners		03/11/2016	

Authors

Francesco Alesiani, NEC Europe Ltd

Jihed Khiari, NEC Europe Ltd

Table of contents

Table of figures

Abbreviations and definitions

Abbreviation	Definition
ASCII	American Standard Code for Information Interchange
CSV	Comma Separated Value
DQA	Data Quality Assessment
ESRI	Environmental Systems Research Institute
IDM	Identity Management (part of MOBiNET Platform)
FCD	Floating Car Data
FVD	Floating Vehicle Data
FTS	Fix Traffic Sensor
GIS	Geographical Information System
NA	Not Applicable
REST	Representational state transfer

Executive Summary

This document presents the specification for the Traffic Data Quality Assessment Tool for two distinct data sources: data from fixed sensors (e.g. loop counters) and data from probe vehicles i.e. Floating Vehicle Data (FVD), as deployed in Release III of the MOBiNET Platform.

The specification follows the general guidelines of the MOBiNET platform since the service has similar characteristics to Platform components.

Traffic data quality assessment component is meant to support Service Level Agreement. The service results are based on the statistical analysis of the sensor measure or localization traces of vehicles and provide information on possible anomalies in the data or value in the data.

1. Introduction

1.1. Scope of the Deliverable

The deliverable describes the Data Quality Assessment (DQA) Service for two distinct data sources: data from Fixed sensors (e.g. loop counters) and data from probe vehicle i.e. Floating Vehicle Data (FVD). This service is provided in order to support Service Level Agreement. The service analyses the quality of traffic sensor data and tries to identify the reasons in case of low quality. As for FVD, the service measures the value of the data for delivering Traffic Service in general by calculating quality indicators for the provided data.

1.2. Service Introduction

The service is supposed to be consumed by two actors: 1) the data provider and 2) the data consumer. Figure 1 shows the relationship of the actors and the service. Differentiation of the actors is not relevant to the service and is traced by the MOBiNET Platform via the IDM.

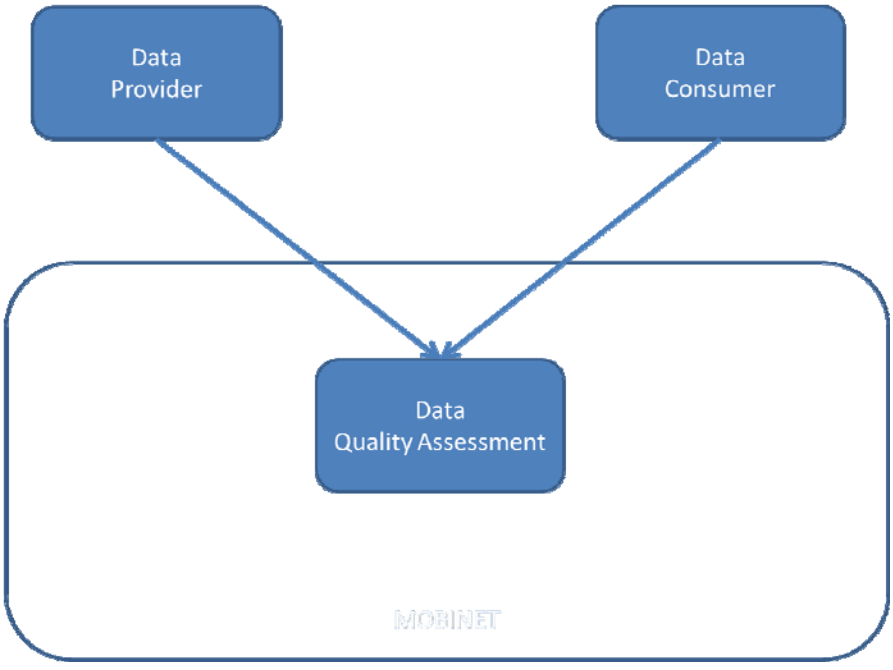

Figure 1 - Service Overview

1.3. Deliverable Structure

The deliverable is divided in the following chapters:

1) The introductory chapter presenting the scope of the deliverable
2) The description of the DQA service for Fix sensor

3) The description of the DQA service for FVD data
4) Installation instruction for the DQA service
5) Step by Step use of services
6) DQA Python API use

1.4. Deliverable Changes (Release 3.1)

This deliverable was first published in relation to “Task 2.4 as D24.111 - 2.11a Service data quality assessment utilities implementation (release 3)”.

The DQA service has been deployed in the MOBiNET Platform in Release 3. The service had similar API but was changed following the feedback received during the Hackathon in Bordeaux.

With Respect to Release 3.0, the service now includes the DQA for FVD and the OAuth2.0. Previously the service has used its internal authentication mechanism, then it was integrated with the REST API from IDM and currently in Release 3.1 it used the OAuth2.0 authentication mechanism.

A dedicated Widget has been developed for loading the FVD data and integrated in the MOBiNET Platform.

2. Data Quality Assessment – Fix Traffic Sensor (FTS)

2.1. Concept

2.1.1. Basic Service

This document describes the service that is deployed with the released 3.1 and reference to difference with previous releases.

Let consider the situation described in Figure 2. One or two measured site collects information on vehicles. The problem is to somehow assess the quality of the provided data.

The quality is defined as a real value, an indicator between 0 and 1 and represented internally as a float32 bit. 0 is indicating a low quality, while 1 corresponds to and high quality. Quality indicator can be at level of the whole day or at the single sample period. Data are supposed to be provided in a regular interval. When an interval is not present the data is considered missed.

There are multiple indicators:

1) Type 1: Zero occupancy
2) Type 2: Non-zero occupancy with zero flow
3) Type 3: High occupancy
4) Type 4: Constant measure

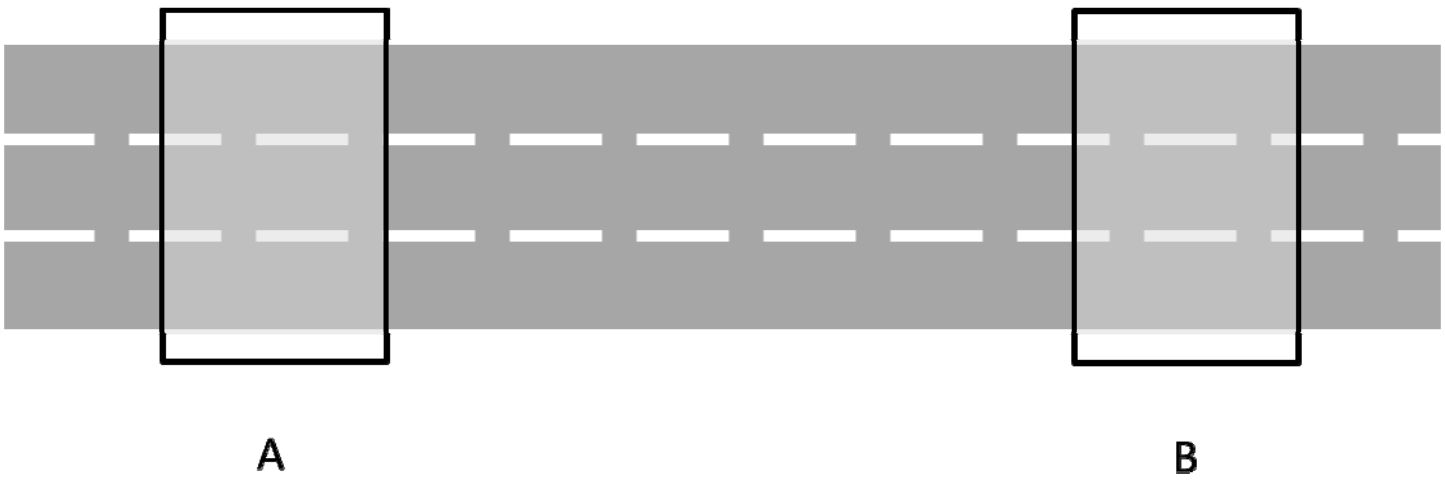

Figure 2 - Concept

a) Type 1

The Zero occupancy indicates if the measure has many zero occupancy values, it this is the case, the sensor may not properly working or is simply placed in an area where no vehicle is passing.

b) Type 2

Non-zero occupancy with no zero flow measures consistency between the flow and occupancy measures. If the flow is non zero, then the occupancy shall not be low or zero. That condition may indicate that the sensor is not properly functioning.

c) Type 3

High occupancy measures if the sensors are constantly occupied. This is also a condition that may indicate an anomaly.

d) Type 4

Constant measure tries to identify is the sensor is measuring correctly but measuring the change in the sensor values. If the value is constant for a long period then probably the sensor is not measuring correctly.

2.1.2. Technology

The provided service is implemented using the HTTP API, according to the concept of RESTfull API [1].

Server side the implementation is based on python and SQLite.

2.1.3. References

[1] RESTfull definition, http://en.wikipedia.org/wiki/Representational_state_transfer

[2] Restful Objects, http://en.wikipedia.org/wiki/Restful_Objects

2.2. Data Quality Service

Following the graphical representation for Restful Objects [2], the following picture represents the elements and the possible operation on each element composing the service.

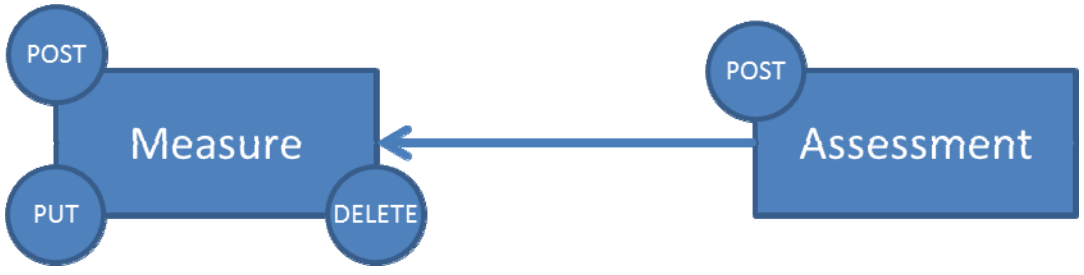

Figure 3 - Service Elements

Element	Description	Comment
Measure	The group of measurement referred to an	Required
Assessment	The assessment of the quality of the data	Required

Element	HTTP Op	Effect
Measure	PUT	Insert or update an object element
Measure	POST	Return the value of the selected element
Measure	DELETE	Remove the element
Assessment	GET	Perform the assessment of the data

2.3. Assessment

2.3.1. /dataquality/assessment

2.3.1.1. Description

The interface allows retrieving the quality assessment value.

2.3.1.2. URL Structure

http://<server>:<port>/dataquality/assessment

2.3.1.3. POST

2.3.1.3.1. Input

Name	Type	Description
netid	long	Id of the network element
objid	long	Id of the object element
from	long	Date from when execute the assessment; if means from today
to	long	Date to when execute the assessment; if not provided means till now

2.3.1.3.2. Response (json)

It contains the full description of the element as defined in the Data Model.

2.3.1.3.3. Return Codes

Code	Meaning
200	No error
400	Bad Request

2.3.1.3.4. Example

GET

http://<server>:<port>/dataquality/assessment

{"header": {"access_token": "YTcxNzkzODktNzlmNS00NzFjLTk5NjctMGRhMDFjYmUzMjQ1",
"client_id": "TI.CSE.client.dqaFixTestUser1.4605715283281456328", "oid": "4742", "nid": "M25"",
"where": "sid between 0 and 1432"}}

Reply

[{"gdate": -1, "oid": "4742", "nid": "M25", "type4": 0.7881188627771927, "type1": 0.9873726525163128,
"type3": 0.9794202147444259, "type2": 1.0}]

2.4. Measure

2.4.1. /dataquality/measure

2.4.1.1. Description

The interface allows manipulate the measure element.

2.4.1.2. URL Structure

http://<server>:<port>/dataquality/measure

2.4.1.3. POST

2.4.1.3.1. Input (json)

The following fields are used to create a where clause to get the data. The condition is concatenated as AND. The “where” field is used to consider different additional and more sophisticated conditions.

2.4.1.3.1.1. Header

Name	Type	Description
nid	long	Id of the element; when not id is provided, all elements are returned
oid	long	Id of the element; when not id is provided, all elements are returned
gdate	long	A specific generation date
sid	long	Sample number
where	string	String containing directly the condition to check

2.4.1.3.2. Response (json)

It contains the full description of the element as defined in the Data Model, Table 1 - Measure Table.

2.4.1.3.3. Return Codes

Code	Meaning
200	No error
400	Bad Request

2.4.1.3.4. Example

GET

http://<server>:<port>/dataquality/measure

Input (json)

{"header": {"oid": "4742", "nid": "M25", "access_token": "YTcxNzkzODktNzlmNS00NzFjLTk5NjctMGRhMDFjYmUzMjQ1", "client_id": "TI.CSE.client.dqaFixTestUser1.4605715283281456328", "where": "sid between 0 and 1432", "dbname": "lucas"}}

Output (json)

[{"Flow4": -2, "gdate": -1, "Flow1": -2, "ostatus": 1, "oid": "4742", "flow": 23, "hw": 92, "nid": "M25", "Flow2": -2, "nlanes": 4, "sid": 1430, "hspeed": -2, "Flow3": -2, "speed": 107, "occ": 3, "slimit": 112, "hflow": -2}, {"Flow4": -2, "gdate": -1, "Flow1": -2, "ostatus": 1, "oid": "4742", "flow": 21, "hw": 76, "nid": "M25", "Flow2": -2, "nlanes": 4, "sid": 1431, "hspeed": -2, "Flow3": -2, "speed": 103, "occ": 5, "slimit": 112, "hflow": -2}, {"Flow4": -2, "gdate": -1, "Flow1": -2, "ostatus": 1, "oid": "4742", "flow": 18, "hw": 108, "nid": "M25", "Flow2": -2, "nlanes": 4, "sid": 1432, "hspeed": -2, "Flow3": -2, "speed": 111, "occ": 2, "slimit": 112, "hflow": -2}]

2.4.1.4. PUT

2.4.1.4.1. Input

Json file containing the information of table Table 1 - Measure Table in the body part, plus a header with the authentication (see section 2.7).

2.4.1.4.2. Response

NA

2.4.1.4.3. Return Codes

Code	Meaning
200	No error
400	Bad Request

2.4.1.4.4. Example

PUT

http://<server>:<port>/dataquality/measure

{"header": {"access_token": "YTcxNzkzODktNzlmNS00NzFjLTk5NjctMGRhMDFjYmUzMjQ1",
"client_id": "TI.CSE.client.dqaFixTestUser1.4605715283281456328",
"body": [{"gdate": "-1", "Flow4": "-2", "ostatus": "1", "oid": "4742", "Flow1": "-2", "flow": "15",
"nid": "M25", "hw": "111", "nlanes": "4", "sid": "1433", "hspeed": "-2", "Flow3": "-2", "Flow2": "-
2", "speed": "105", "occ": "2", "slimit": "112", "hflow": "-2"}, {"gdate": "-1", "Flow4": "-2",
"ostatus": "1", "oid": "4742", "Flow1": "-2", "flow": "24", "nid": "M25", "hw": "104", "nlanes": "4",
"sid": "1434", "hspeed": "-2", "Flow3": "-2", "Flow2": "-2", "speed": "111", "occ": "2", "slimit":
"112", "hflow": "-2"}, {"gdate": "-1", "Flow4": "-2", "ostatus": "1", "oid": "4742", "Flow1": "-2",
"flow": "16", "nid": "M25", "hw": "110", "nlanes": "4", "sid": "1435", "hspeed": "-2", "Flow3": "-2",
"Flow2": "-2", "speed": "113", "occ": "2", "slimit": "112", "hflow": "-2"}, {"gdate": "-1", "Flow4": "-
2", "ostatus": "1", "oid": "4742", "Flow1": "-2", "flow": "15", "nid": "M25", "hw": "133", "nlanes":
"4", "sid": "1436", "hspeed": "-2", "Flow3": "-2", "Flow2": "-2", "speed": "117", "occ": "2", "slimit":
"112", "hflow": "-2"}, {"gdate": "-1", "Flow4": "-2", "ostatus": "1", "oid": "4742", "Flow1": "-2",
"flow": "8", "nid": "M25", "hw": "85", "nlanes": "4", "sid": "1437", "hspeed": "-2", "Flow3": "-2",
"Flow2": "-2", "speed": "115", "occ": "1", "slimit": "112", "hflow": "-2"}, {"gdate": "-1", "Flow4": "-
2", "ostatus": "1", "oid": "4742", "Flow1": "-2", "flow": "21", "nid": "M25", "hw": "66", "nlanes":
"4", "sid": "1438", "hspeed": "-2", "Flow3": "-2", "Flow2": "-2", "speed": "116", "occ": "2", "slimit":
"112", "hflow": "-2"}, {"gdate": "-1", "Flow4": "-2", "ostatus": "1", "oid": "4742", "Flow1": "-2",
"flow": "23", "nid": "M25", "hw": "84", "nlanes": "4", "sid": "1439", "hspeed": "-2", "Flow3": "-2",
"Flow2": "-2", "speed": "104", "occ": "3", "slimit": "112", "hflow": "-2"}]}

2.4.1.5. DELETE

2.4.1.5.1. Input (Header)

Name	Type	Description
nid	long	Id of the element; when not id is provided, all elements are returned
oid	long	Id of the element; when not id is provided, all elements are returned
gdate	long	A specific generation date
sid	long	Sample number
where	string	String containing directly the condition to check

2.4.1.5.2. Response

NA

2.4.1.5.3. Return Codes

Code	Meaning
200	No error
400	Bad Request

2.4.1.5.4. Example

DELETE

http://<server>:<port>/dataquality/measure

{"header": {"oid": "4742", "nid": "M25", "access_token": "YTcxNzkzODktNzlmNS00NzFjLTk5NjctMGRhMDFjYmUzMjQ1", "client_id": "TI.CSE.client.dqaFixTestUser1.4605715283281456328", "where": "sid between 1430 and 1432", "dbname": "lucas"}

2.5. Summary

2.5.1. /dataquality/summary

2.5.1.1. Description

The interface allows retrieving a summary of the measure data.

2.5.1.2. URL Structure

http://<server>:<port>/dataquality/summary

2.5.1.3. POST

2.5.1.3.1. Input

Name	Type	Description
netid	long	Id of the network element;
objid	long	Id of the object element;

2.5.1.3.2. Response (json)

It contains a summary of the data present in the database.

2.5.1.3.3. Return Codes

Code	Meaning
200	No error
400	Bad Request

2.5.1.3.4. Example

GET

http://<server>:<port>/dataquality/assessment

{"header": {"nid": "M25", "access_token":
"YTcxNzkzODktNzlmNS00NzFjLTk5NjctMGRhMDFjYmUzMjQ1", "client_id":
"TI.CSE.client.dqaFixTestUser1.4605715283281456328"}}

Reply

[{"s0": 986, "gdate": 16, "s3": 6, "s2": 0, "s1": 22, "oid": "4742A", "nid": "M25"}, {"s0": 1433, "gdate": 17, "s3": 76, "s2": 0, "s1": 18, "oid": "4742A", "nid": "M25"}, {"s0": 1438, "gdate": 18, "s3": 35, "s2": 0, "s1": 28, "oid": "4742A", "nid": "M25"}, {"s0": 1394, "gdate": 19, "s3": 21, "s2": 0, "s1": 26, "oid": "4742A", "nid": "M25"}, {"s0": 986, "gdate": 20, "s3": 6, "s2": 0, "s1": 22, "oid": "4742A", "nid": "M25"}, {"s0": 1440, "gdate": 16, "s3": 17, "s2": 0, "s1": 21, "oid": "4747A", "nid": "M25"}, {"s0": 1436, "gdate": 17, "s3": 86, "s2": 0, "s1": 20, "oid": "4747A", "nid": "M25"}, {"s0": 1440, "gdate": 18, "s3": 36, "s2": 0, "s1": 28, "oid": "4747A", "nid": "M25"}]

2.6. Header and Body in the request

The request to the server uses json for formatting the input. The input is made of:

1) A **header**, with the keyword “header” and
2) A **body**, using the keyword “body”; the body part is only present if needed

An example of the PUT request of the measure is as follow, where the two elements are in bold.

{"**header**": {"access_token": "YTcxNzkzODktNzImNS00NzFjLTk5NjctMGRhMDFjYmUzMjQ1", "client_id": "TI.CSE.client.dqaFixTestUser1.4605715283281456328"},

"**body**": {{"gdate": "-1", "Flow4": "-2", "ostatus": "1", "oid": "4742", "Flow1": "-2", "flow": "21", "nid": "M25", "hw": "66", "nlanes": "4", "sid": "1438", "hspeed": "-2", "Flow3": "-2", "Flow2": "-2", "speed": "116", "occ": "2", "slimit": "112", "hflow": "-2"}, {"gdate": "-1", "Flow4": "-2", "ostatus": "1", "oid": "4742", "Flow1": "-2", "flow": "23", "nid": "M25", "hw": "84", "nlanes": "4", "sid": "1439", "hspeed": "-2", "Flow3": "-2", "Flow2": "-2", "speed": "104", "occ": "3", "slimit": "112", "hflow": "-2"}]}

2.7. Authentication (Release I only, obsolete)

The authentication is done using a secret password set in the system and a hash function. The user generates a string and encrypts with the secret key. The content is added to the json elements.

Name	Type	Description
user	string	Name of the user
c1	string	String for the authentication
c2	string	Return string of the hash of c1 + private password, according to sha256 function (hexadecimal version)
dbname	string	Name of the database

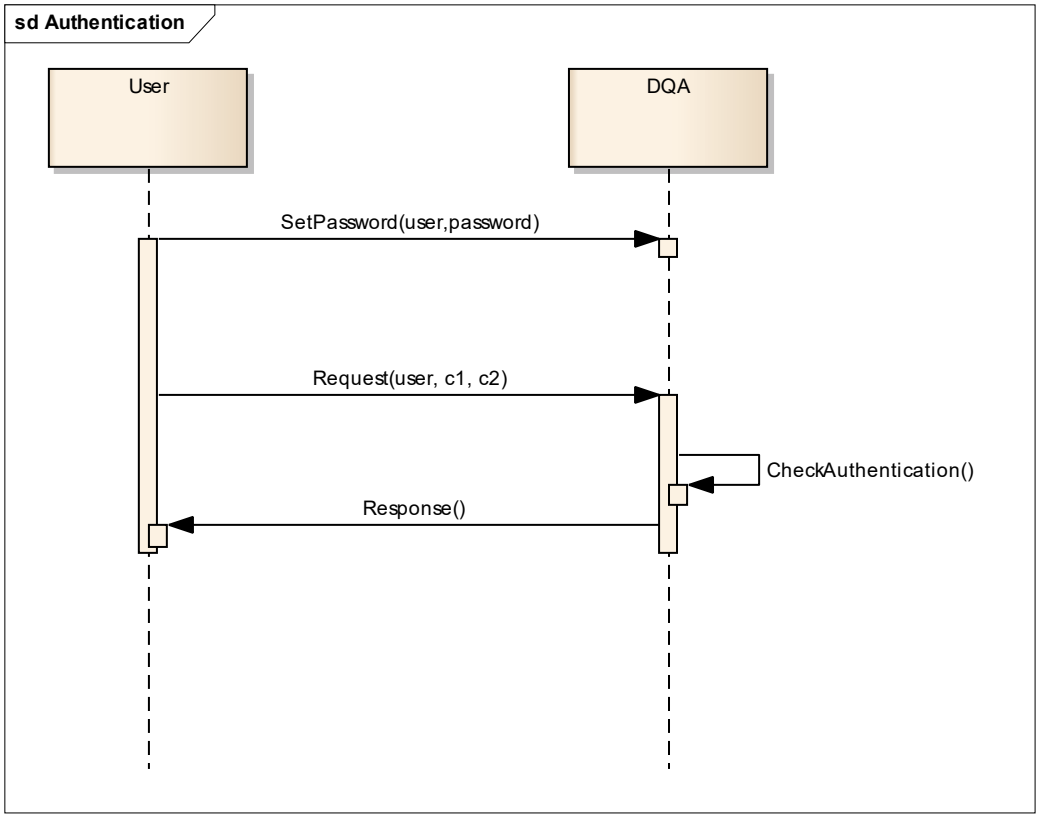

Figure 4 - Authentication Release I

Using python this can be generated by the following line

```
c2 = hashlib.sha256(c1+password).hexdigest()
```

2.8. Authentication using MOBiNET Identity Management (Release II & III, obsolete)

MOBiNET Identity Management Provider is implemented by Telecom Italia at the following url:

http://identitymgr.test1.mobinet.eu/IdentityManager/authenticate

The user is able by using the credential from the Identity Management to access to the service and not using the internal authentication service.

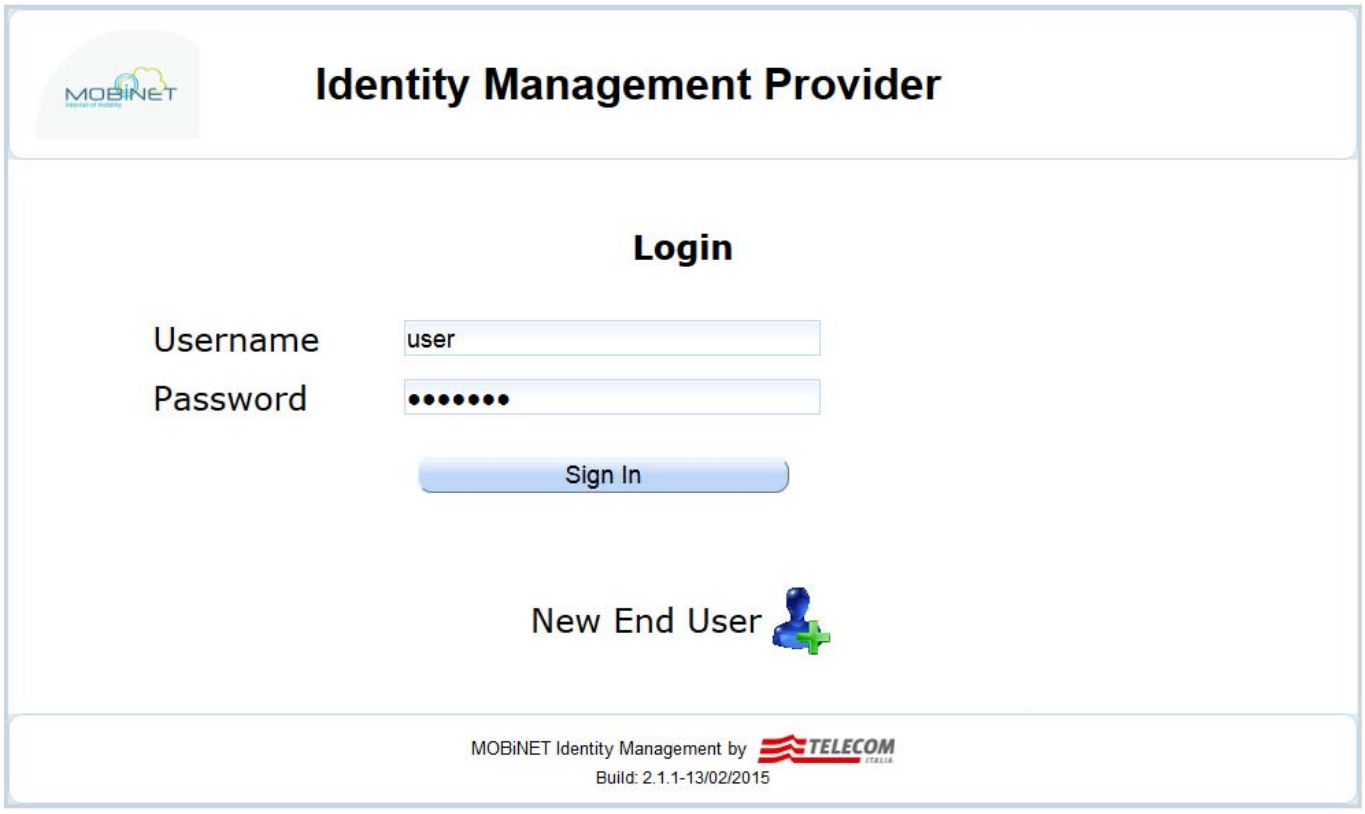

Figure 5 - Authentication using MOBiNET Identity Management

2.9. Authentication based on OAuth2.0 (Release III)

The authentication is done using a client id and an access_token provided by the IdM. The content is located in the header of the json request. These are the only information required.

Name	Type	Description
client_id	string	ID of the user as defined in from the IdM
access_token	string	The Authentication string as defined by the IdM

An example of the PUT request of the measure is as follow, where the two elements are in bold.

{"**header**": {"client_id": " TI.CSE.client.dqaTestUser.4605242329131027131", "access_token": "M2JjNzJmZTAtMzJhYS00MmU0LWE2YTItNjBkODNiZTljNzVh"},

"**body**": {{"gdate": "-1", "Flow4": "-2", "ostatus": "1", "oid": "4742", "Flow1": "-2", "flow": "21", "nid": "M25", "hw": "66", "nlanes": "4", "sid": "1438", "hspeed": "-2", "Flow3": "-2", "Flow2": "-2", "speed": "116", "occ": "2", "slimit": "112", "hflow": "-2"}, {"gdate": "-1", "Flow4": "-2", "ostatus": "1", "oid": "4742", "Flow1": "-2", "flow": "23", "nid": "M25", "hw": "84", "nlanes": "4", "sid": "1439", "hspeed": "-2", "Flow3": "-2", "Flow2": "-2", "speed": "104", "occ": "3", "slimit": "112", "hflow": "-2"}]}

2.10. Service Deployment

The Data Quality Service is deployed in the MOBiNET center.

2.11. Data Model

The reference data model is represented in the following figure.

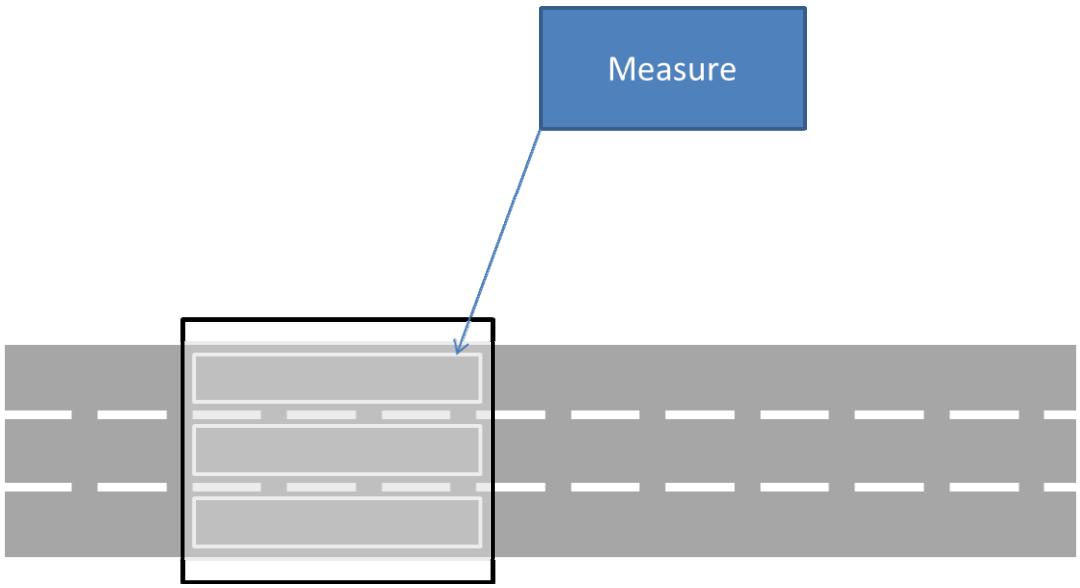

Figure 6 - Data Model

2.11.1. Measure Specification – Measure

Table 1 - Measure Table

Position	Field Content	Name	Unit	Type	Comment	Required
1	Network Id	nid	Identifier	Long	Id of the sender	Y
2	Object Id	oid	Identifier	Long	Unique Object Id	Y
3	Generation Date	gdate	Seconds	Long	Date-time	Y
4	Sample number	sid	Sample progressive number	int	Is the id of the sample within the day; 0 is the first	Y
5	Object status	ostatus	Boolean	int	0: ok, 1: down (default 0)	
6	Speed Limit	slimit	Km per hour	Int		Y
7	Number of Lanes	nlanes	Number	Int	If is a sensor typically it is 1	Y

8	Flow (All)	flow	Vehicles per minute	Int		Y
9	Flow (Heavy)	hflow	Vehicles per minute	Int		
10	Speed (All)	speed	Km per hour	Int		Y
11	Speed (Heavy)	hspeed	Km per hour	Int		
12	Occupancy (All)	occ	% of time	int		Y
13	Headway (All)	hw	meters	int		13
14	Flow on category 1	Flow1	Vehicles per minute 1	Int	Not used	14
15	Flow on category 2	Flow2	Vehicles per minute 2	int	Not used	15
16	Flow on category 3	Flow3	Vehicles per minute 1	Int	Not used	16
17	Flow on category 4	Flow4	Vehicles per minute 2	int	Not used	17

2.11.2. Assessment specification

Table 2 - Assessment Table – sample/day level

Position	Field Content	Name	Unit	Type	Comment	Required
1	Network Id	nid	Identifier	Long	Id of the sender	Y
2	Object Id	oid	Identifier	Long	Unique Object Id	Y
3	Day	gdate	Seconds	Long	Date-time	Y
4	Time	sid	Sample number	int	Is the id of the sample within the day; 0 is the first It is present only on specific indicators	*
5	Quality type	qtype		int	Type of the analysis	Y
6	Indicator	qlevel	0-1	float	Number between 0 and 1 indicating the quality of the measure	Y

2.12. Design choices

A database stores the user information (name, password, location of the local db).

Each user has a separate database.

There is no rights restriction, once is authenticated the user can do all the possible actions (POST/PUT/DELETE).

POST must been selected instead of GET since the GET interface in some implementation does not allow payload parameters.

3. Data Quality Assessment – FCD/FVD Sensor

3.1. Overview

We present in this section a brief overview of DQA solution for FVD. We aim namely to assess the data quality of floating car datasets by calculating four distinct indicators: the granularity, missing data, reliability and accuracy. The indicators are normalized to have values between 0 and 1. All the indicators are real number values range between 0 and 1 represented internally as float32 bit. A brief explanation of their logic is provided below, along with some practical functional examples.

3.1.1. The Data Quality Indicators

e) Granularity

This indicator gives insight about the GPS sampling rate within a dataset, meaning the overall frequency of GPS transmissions within a dataset. The granularity thus measures the regularity of the information in comparison to the expectation we have based on a specific context.

For example, an urban setting characterized by a high road network density requires a low sampling rate, e.g. 15 seconds, which translates into a high granularity i.e. close to 1. In the same setting, if the sampling rate is of 60 seconds, the granularity would be low, i.e. close to 0. However, a sampling rate of 60 seconds on a highway is considered an acceptable sampling rate given the low road network density. Depending on the scenario, the parameters used for calculating the granularity can be tuned by an expert user.

f) Missing Data

GPS devices generally transmit data at a fixed rate. However, due to malfunctions or human error, datasets may present gaps, i.e. blackholes between some GPS transmissions. We define a blackhole by a period of time higher to 2*the sampling rate where no information about a trip is supplied. We consider, for example, a dataset from a taxi fleet in an urban setting, characterized by short trips of typically 8-10 minutes and a sampling rate of 15 seconds. The minimum length of a blackhole would then be 30 seconds. If the trips within this dataset typically have blackholes of 60 seconds, the missing data indicator would be low i.e. close to 0, considering the high relative loss of information. However, if we consider a floating vehicle dataset from cars on a highway where the trips are typically of 30-60 minutes and the sampling rate is of 60 seconds, the minimum length of a blackhole is 2 minutes. Consequently, if the trips within this dataset have blackholes of a minute, the relative loss if information is not high and the missing data indicator approaches 1. To sum up, this indicator takes into account the number of blackholes and their length for all the trips as well as the sampling rate to estimate the severity of the information loss depending on a given setting.

g) Reliability

This indicator assesses how logical the data is in describing mobility traces, i.e. if the provided data is reliable. Examples of ‘unreliable’ data include stopped cars that feed data for a long time, cars that travel an illogically long distance within a short span of time (e.g. more than 20 meters in a second) or cars that repeatedly do circular movements or clear patters within a short span of time. As a result, if the data

presents many samples of ‘unreliable’ data such as the above mentioned examples, the reliability indicator approaches 0. Otherwise, it approaches 1.

h) Accuracy

Given the urban context, the possibly high sampling rate and the possibly low precision of the GPS device, the actual position of the vehicle may be different than the navigation position indicated in the dataset at a given time. Using the map-matching technique, this indicator measures the overall error between these values within a dataset.

The figure below illustrates this discrepancy and the location error.

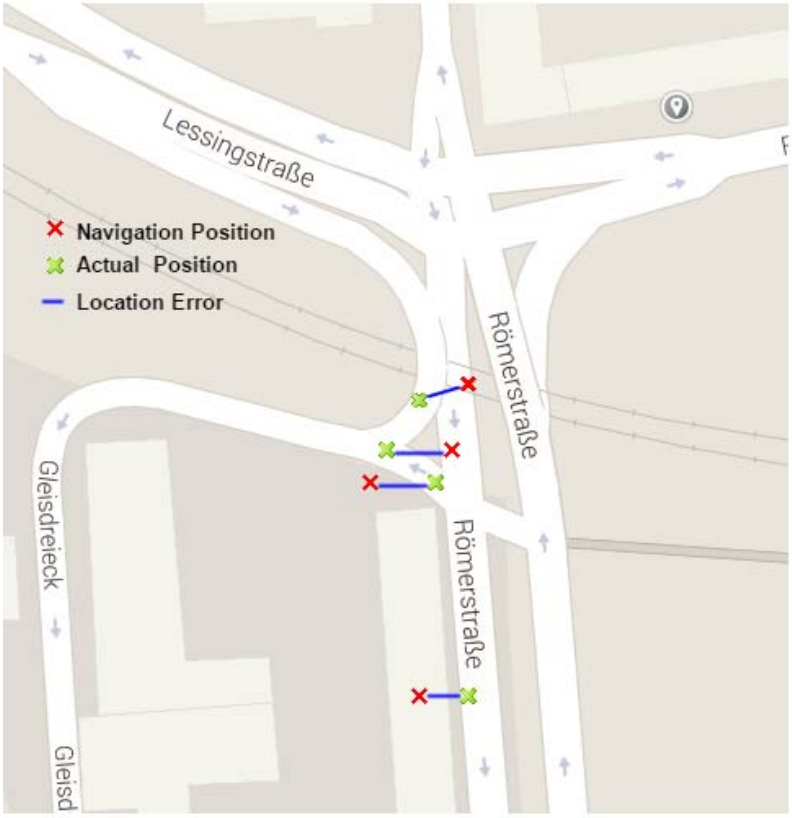

Figure 7 - Examples of possible location errors

3.1.2. Input

The proposed file format for the input file is the CSV (Comma Separated Values) where ‘,’ is used as token separator.

Position	Field Content	Unit	Description	Comment
1	tripid	string	Identifier of the trip. Each vehicle will have more a unique id for each trip. Vehicle id is not required	required

2	gpstime	string	Date of the gps coordinate Yyyy-MM-dd hh:mm:ss 2015-09-09 03:58:50	required
3	gpslatitude	decimal degree	The latitude of the vehicle	required
4	gpslongitude	decimal degree	the longitude of the vehicle	required

A possible example of a row within an accepted input file format:

```
tripid, gpstime, gpslatitude, gpslongitude

da970980689816ccdf112d15b453d283,2015-09-09 03:58:50,43.00347,12.65704
```

3.1.3. Detail field description

The mandatory input fields are described as followed:

tripid

An alphanumeric string uniquely identifying a trip. A tripid is shared over many entries thus describing a trip.

gpstime

Timestamp indicating the exact time in which GPS was acquired.

The time shall be in UTC time and its format shall be "yyyy-mm-dd hh:mm:ss".

The Time Zone (if any) shall then be according to ISO 86011 “The offset from UTC is given in the format ±[hh]:[mm], ±[hh][mm], or ±[hh]. So if the time being described is one hour ahead of UTC (such as the time in Berlin during the winter), the zone designator would be "+01:00", "+0100", or simply "+01". This is appended to the time in the same way that 'Z' was above. The offset from UTC changes with daylight saving time, e.g. a time offset in Chicago, would be "−06:00" for the winter (Central Standard Time) and "−05:00" for the summer (Central Daylight Time).

The following times all refer to the same moment: "18:30Z", "22:30+04", "1130−0700", and "15:00−03:30". Nautical time zone letters are not used with the exception of Z. To calculate UTC time one has to subtract the offset from the local time, e.g. for "15:00−03:30" do 15:00 − (−03:30) to get 18:30 UTC.”

gpslatitude

WGS84 coordinates.

Format in decimal degrees with '.' as decimal separator.

At least 5 decimal digits to be present.

gpslongitude

WGS84 coordinates.

Format in decimal degrees with '.' as decimal separator.

At least 5 decimal digits to be present.

3.1.4. Output

This output values indicates the state of the execution (0 if error/ 1 if success) and the values of the four abovementioned indicators.

An example of the output values is shown below:

Status, 1
Accuracy, 0.85
Reliability, 0.75
Missing Data, 0.65
Granularity, 0.75

3.2. Implementation and Integration into the MOBiNET platform

Given a Floating Car dataset, a Python script returns the abovementioned Data Quality. Since our service is to be integrated into the MOBiNET platform we have devised the architecture shown in the figure below.

The widget is developed in Javascript, HTML and CSS. We also used the jQuery library to facilitate Cross-domain AJAX requests. Connection between the widget and the Apache web server is also ensured by enabling CORS (Cross Origin Resource Sharing) on the server side.

As shown in the above figure, the user first uploads a dataset to the web server. To execute NLE's service, we use a PHP script which opens a pipe to the Python script. Both scripts exchange parameters using JSON. This notation also allows to return the output indicators' values to the widget.

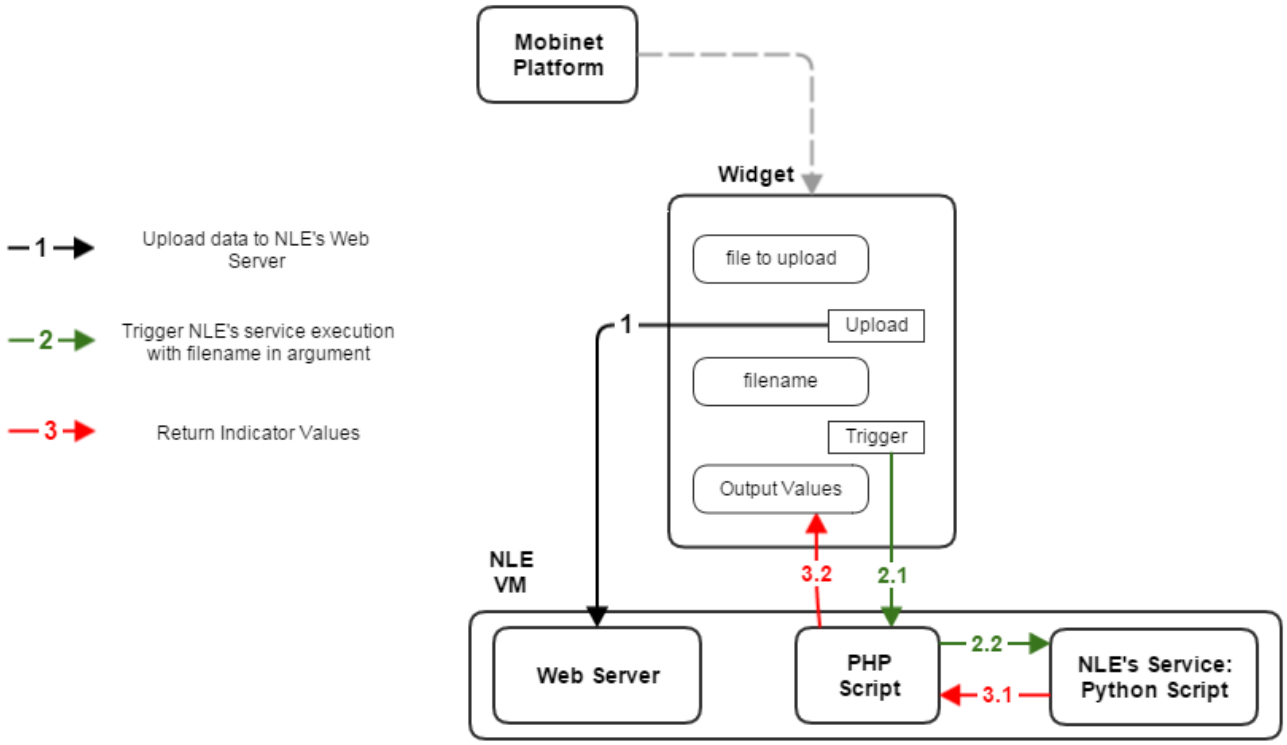

Figure 8 - Information Flow of the deployment of the FVD DQA service

Integrating the widget into the MOBiNET platform requires a prior service registration in the Dashboard using our registered user. A further requirement is to implement the OAuth 2.0 authorization mechanism using the IdentityManager component as an authorization server. The figure below illustrates the different necessary steps to fill this requirement.

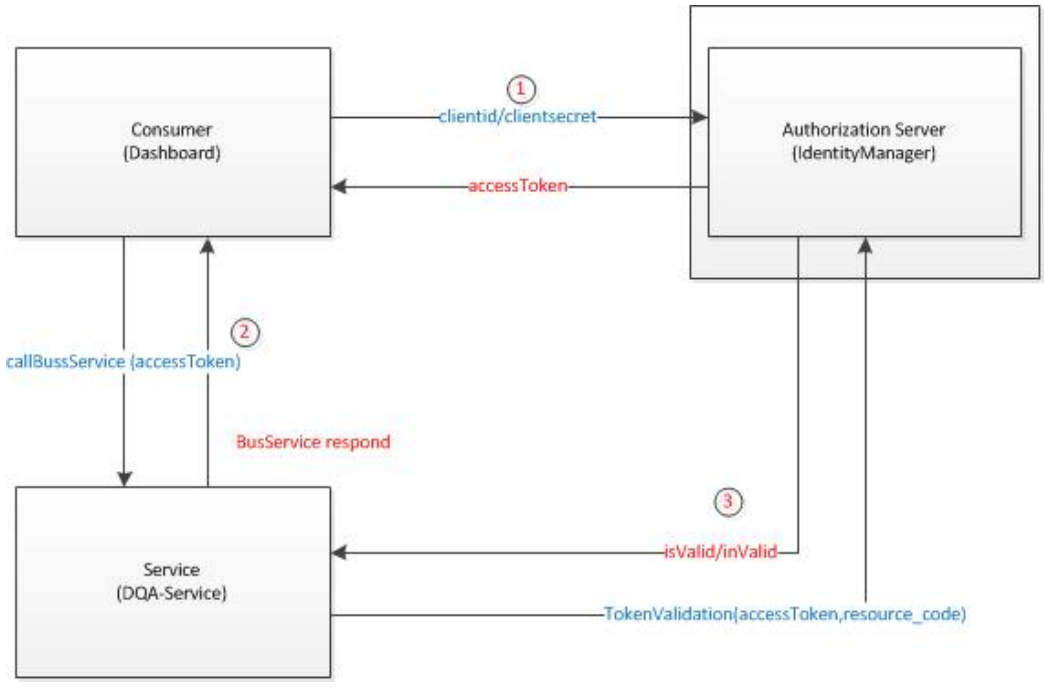

Figure 9 - Authorization information flow between IDM, DQA and Dashboard

4. DQA - MOBiNET center installation instructions (Fix Traffic Sensor)

4.1. Architecture

The service is composed of a frontend that is responsible of handling REST request. The DQA library computes the Quality indicators and manages the interface with the internal structures.

The DQA service is composed of some entities. The main entity is the measure. The Assessment is the function that acts on the measure. Summary entity provides a summary of the information on the data quality indicators. Measures are stored in separate databases. The access to the databases is defined by the role entity. Each user can be associated to one or more database via the role entity. The following figure shows the REST API entities and actions of the DQA service.

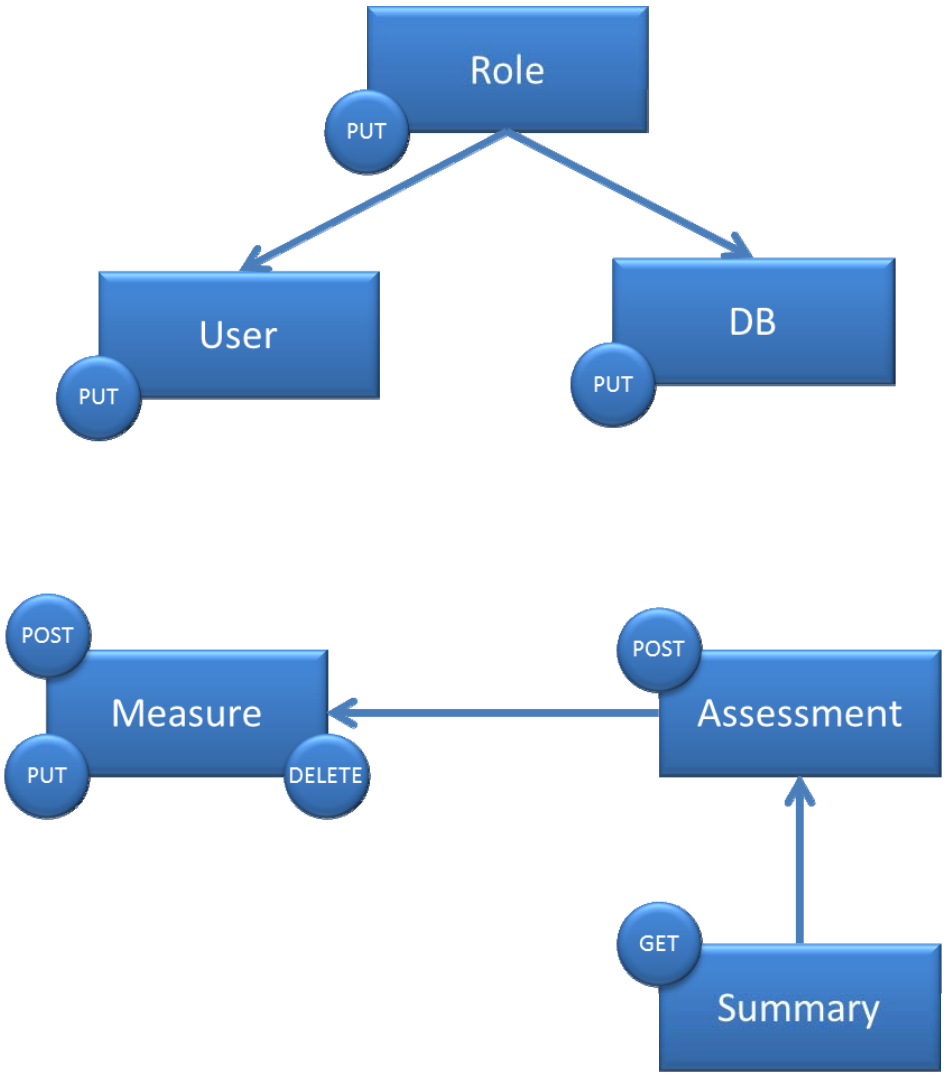

Figure 10 - Object Diagram of DQA release 3.0(upper and lower part)-3.1(only lower part)

4.2. Database Description

The following database table will be used by the store in the configuration database (Release 3.0):

Table name	Description
users	Store the user information, namely the user name and the authentication parameters
databases	Stores the databases connected to the service
roles	Store the connection between users and databases
measure	Store the actual measure. This table is actually distributed in each database

4.3. Interface Description

The DQA offers an REST-API as an external interface for other components or services.

Interface Name	Port	Type	Description
REST-API	8082 (via https://mobicentre2.mobinet.eu)	TCP/IP	access to the REST API

4.4. Installation Plan

i)Pre-requisites

- Python 2.7
- python libraries: random, logging, sys, os,datetime,time, sqlite3,json,hashlib,string, math, csv, signal, BaseHTTPServer, ssl, cgi, urlparse, ConfigParser, httplib

j)Installation

```
yum install mobinet-dqa
```

k) Uninstall / Service Removal

```
yum remove mobinet-dqa
```

l)Start/ Stop service

```
/etc/init.d/mobinet-dqa start|stop
```

4.5. Service configuration

To Start and stop the service it is possible to use the command.

```
/etc/init.d/mobinet-dqa start|stop
```

The service can be configured by editing the configuration file dqa.cnf.

The database is located in the “db” folder. Each database has a separate file.

There is a common configuration file with the user information.

m) Files

- Install path: /opt/mobinet-dqa/
- Log is placed in: /opt/mobinet-dqa/log/
- Databases are placed in: /opt/mobinet-dqa/db/
- Configuration files is located in: /opt/mobinet-dqa/dqa.cnf

n) Basic operational check

- Insert/Get/Remove Measure
- Get the assessment result
- Add User, Database and Role

4.6. DQA API

o) API urls

http://[ip address]:8082//dataquality/assessment

http://[ip address]:8082//dataquality/measure

http://[ip address]:8082//dataquality/summary

http://[ip address]:8082//dataquality/user

http://[ip address]:8082//dataquality/db

http://[ip address]:8082//dataquality/role

Commissioning environment host:

http://???.com1.mobinet.eu:8082//dataquality

Test Environment host:

http://???.test1.mobinet.eu:8082//dataquality

4.7. Installing Python2.7

Install pre-requiste packages

```
yum groupinstall "Development tools"
yum install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel xz-libs gcc
cd /opt
wget --no-check-certificate https://www.python.org/ftp/python/2.7.10/Python-2.7.10.tar.xz
tar xf Python-2.7.10.tar.xz
cd Python-2.7.10
./configure --prefix=/usr/local
make && make altinstall
wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py
/usr/local/bin/python2.7 ez_setup.py
/usr/local/bin/easy_install-2.7 pip
```

Verify the different version of Python

```
ls -ltr /usr/bin/python*
ls -ltr /usr/local/bin/python*
echo $PATH
```

If local is not present, it is possible to add as follow

```
export PATH="/usr/local/bin:$PATH"
```

To install the setuptools

```
# downloading the installation file using wget:
wget --no-check-certificate https://pypi.python.org/packages/source/s/setuptools/setuptools-18.0.tar.gz

# Extract the files from the archive:
tar -xvf setuptools-18.0.tar.gz

# Enter the new directory:
cd setuptools-18.0

# Install setuptools
python2.7 setup.py install

# to install additional packages
# pip2.7 install [package name]
pip2.7 install requests
```

5. Annex 1 – MOBiNET Dashboard

5.1. FVD Widget

From the MOBiNET Website (https://mobicentre2.mobinet.eu/) is possible to upload the FVD data set.

Figure 11 – MOBiNET Website (https://mobicentre2.mobinet.eu/)

The Dashboard presents the DQA FVD service under Analytics.

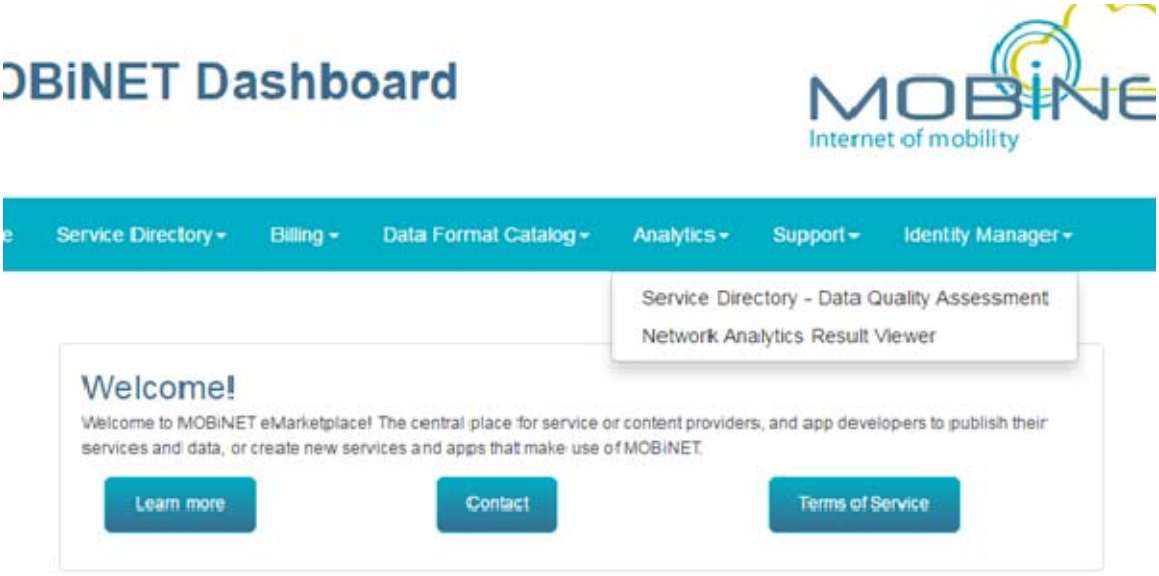

Figure 12 - DQA item

After selecting the service it is possible to upload the data set.

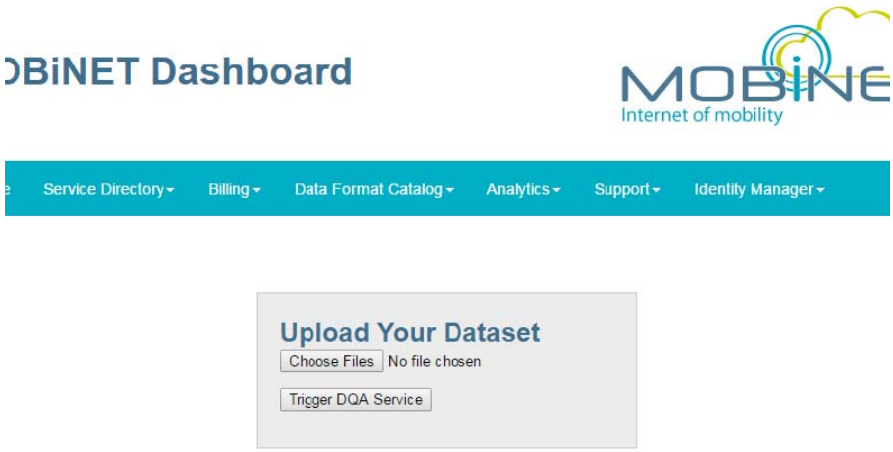

Figure 13 - DQA (FVD) Dataset upload page

After the file has been uploaded the service either return an error message if any problem occurred or return the indicators.

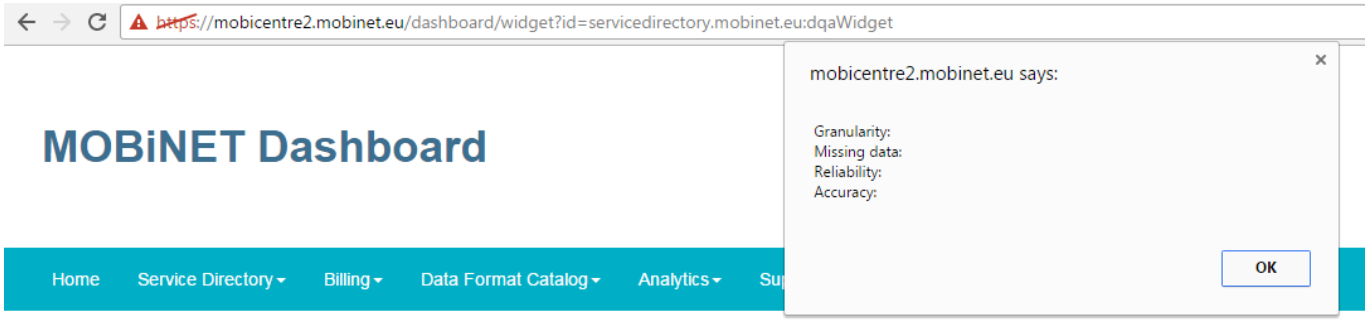

Figure 14 - DQA (FVD) return message

5.2. Consumer Registration for DQA Authentication

In order to use the DQA service, user shall register the DQA resource using the Authentication portal provided in the MOBiNET Dashboard.

After accessing the MOBiNET website (https://mobicentre2.mobinet.eu/), select Authorization item from the Identity Manager.

Figure 15 - MOBiNET Website (https://mobicentre2.mobinet.eu/)

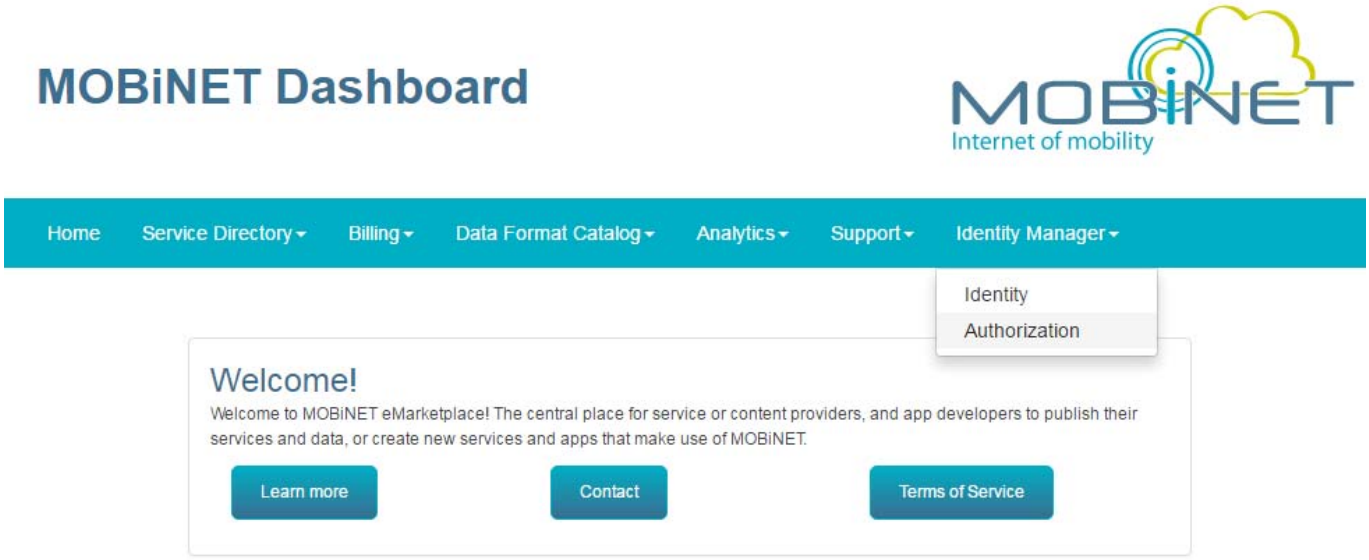

Figure 16 - Authorization Item

From them, select New Consumer from the Registration page.

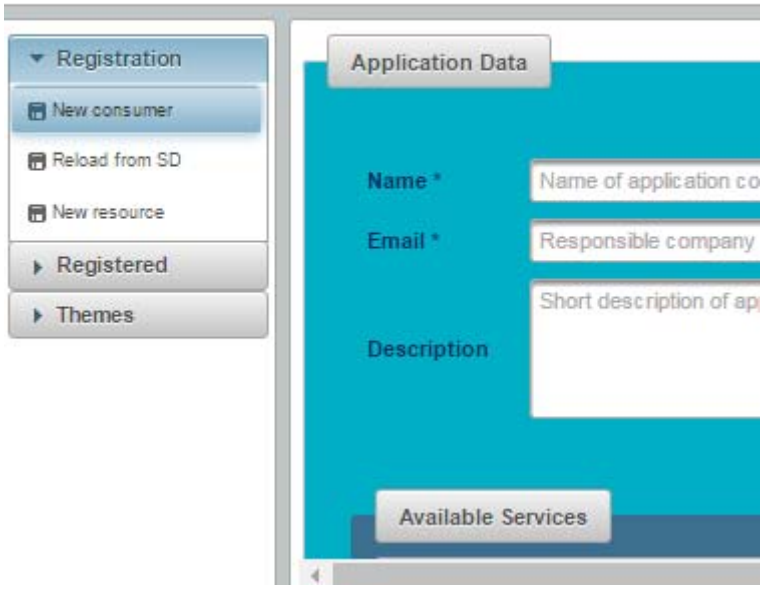

Figure 17 - Generation of new Consumer

Associate the Consumer with the DQA service (currently named “necDQAFix5”).

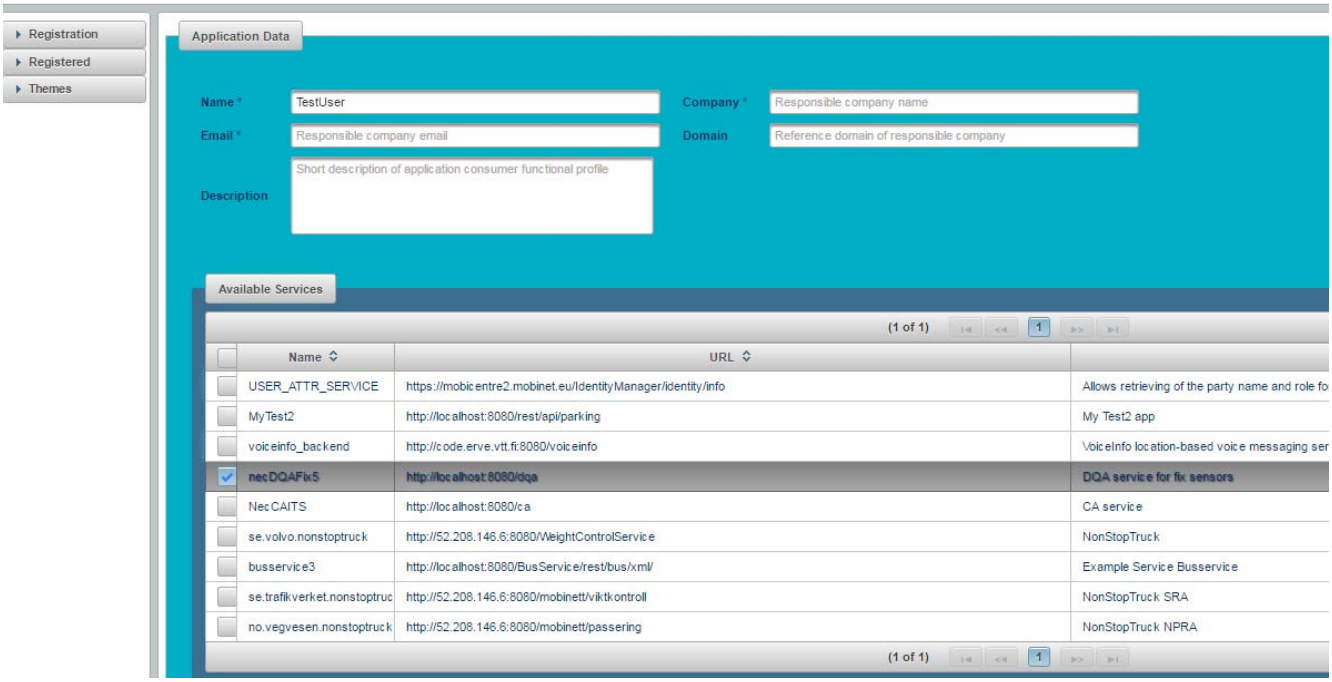

Figure 18 - Selection of the Service

Use the provided information to access the service. An example of Client Registration information to be used for authentication is reported below.

Parameter	Value
client_id :	TI.CSE.client.Francesco_Alesiani.4601286295766036528
client_secret :	3fe5d8ea422f892b
grant_type :	client_credentials
token_url :	https://mobicentre2.mobinet.eu/AuthorizationServer/resources/oauth/token

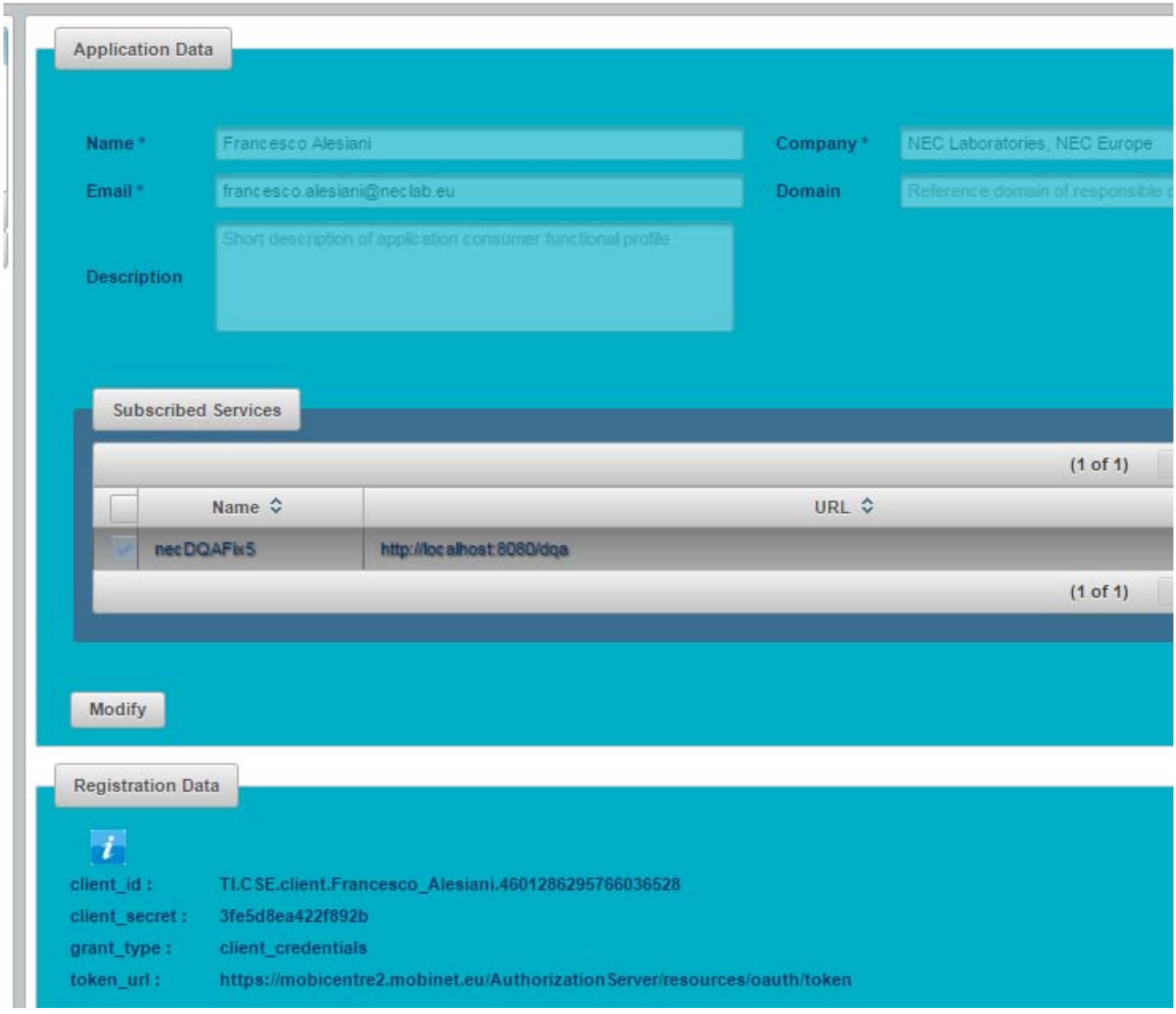

Figure 19 - Customer Registration

6. Annex 2 – DQA Python API (SDK)

In the following section, the use of the DQA API for fix sensor is presented.

6.1. DQA API use: send Measurements and get result

```
##/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##*        MOBINET
##*
##*   File: dqa_client_r3.py
##*
##*         NEC Europe Ltd. PROPRIETARY INFORMATION
##*
##* This software is supplied under the terms of a license agreement
##* or nondisclosure agreement with NEC Europe Ltd. and may not be
##* copied or disclosed except in accordance with the terms of that
##* agreement. The software and its source code contain valuable
##* trade secrets and confidential information which have to be
##* maintained in confidence.
##* Any unauthorized publication, transfer to third parties or
##* duplication of the object or source code - either totally or in
##* part - is prohibited.
##*
##*
##*      Copyright (c) 2016 NEC Europe Ltd. All Rights Reserved.
##*
##* Authors: Francesco Alesiani, francesco.alesiani@neclab.eu
##* Date: 06.10.2016
##*
##* NEC Europe Ltd. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
##* INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY
##* AND FITNESS FOR A PARTICULAR PURPOSE AND THE WARRANTY AGAINST LATENT
##* DEFECTS, WITH RESPECT TO THE PROGRAM AND THE ACCOMPANYING
##* DOCUMENTATION.
##*
##* No Liability For Consequential Damages IN NO EVENT SHALL NEC Europe
##* Ltd., NEC Corporation OR ANY OF ITS SUBSIDIARIES BE LIABLE FOR ANY
##* DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS
##* OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF INFORMATION, OR
##* OTHER PECUNIARY LOSS AND INDIRECT, CONSEQUENTIAL, INCIDENTAL,
##* ECONOMIC OR PUNITIVE DAMAGES) ARISING OUT OF THE USE OF OR INABILITY
##* TO USE THIS PROGRAM, EVEN IF NEC Europe Ltd. HAS BEEN ADVISED OF THE
##* POSSIBILITY OF SUCH DAMAGES.
##*
##*     THE ABOVE HEADER MAY NOT BE EXTRACTED OR MODIFIED IN ANY WAY.
```

```
##*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##*/
#!/usr/bin/env python

#upload data
data,status = dqa_client_lib.measure_put_mydata(filename,
header=header,address=mobinet_address,port=mobinet_port,prefix=mobinet_prefix)

#get data
data,status = dqa_client_lib.something_what_how_body(what="measure", how="POST",
BODY=json.dumps({'header':header}),address=mobinet_address,port=mobinet_port,prefix=mobinet_pre
fix)

#get assessment
data,status = dqa_client_lib.something_what_how_body(what="assessment", how="POST",
BODY=json.dumps({'header':
header}),address=mobinet_address,port=mobinet_port,prefix=mobinet_prefix)

#get summary
data,status = dqa_client_lib.something_what_how_body( what="summary", how="POST",
BODY=json.dumps({'header':header}),address=mobinet_address,port=mobinet_port,prefix=mobinet_pre
fix)

#delete data
data,status = dqa_client_lib.something_what_how_body(what="measure", how="DELETE",
BODY=json.dumps({'header': header}),address=mobinet_address,port= mobinet_port,prefix=
mobinet_prefix)
```

6.2. Request of IDM Access Token

DQA uses authentication mechanism of OAuth2.0 provided by IDM. The user shall create a proper header for the request to the DQA API. Below is an example of generation of the header by using the DQA Client API.

```
##/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##*         MOBINET
##*
##*   File: dqa_client_r3.py
##*
##*         NEC Europe Ltd. PROPRIETARY INFORMATION
##*
##* This software is supplied under the terms of a license agreement
##* or nondisclosure agreement with NEC Europe Ltd. and may not be
##* copied or disclosed except in accordance with the terms of that
##* agreement. The software and its source code contain valuable
##* trade secrets and confidential information which have to be
```

```
##* maintained in confidence.
##* Any unauthorized publication, transfer to third parties or
##* duplication of the object or source code - either totally or in
##* part - is prohibited.
##*
##*
##*      Copyright (c) 2016 NEC Europe Ltd. All Rights Reserved.
##*
##* Authors: Francesco Alesiani, francesco.alesiani@neclab.eu
##* Date: 06.10.2016
##*
##* NEC Europe Ltd. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
##* INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY
##* AND FITNESS FOR A PARTICULAR PURPOSE AND THE WARRANTY AGAINST LATENT
##* DEFECTS, WITH RESPECT TO THE PROGRAM AND THE ACCOMPANYING
##* DOCUMENTATION.
##*
##* No Liability For Consequential Damages IN NO EVENT SHALL NEC Europe
##* Ltd., NEC Corporation OR ANY OF ITS SUBSIDIARIES BE LIABLE FOR ANY
##* DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS
##* OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF INFORMATION, OR
##* OTHER PECUNIARY LOSS AND INDIRECT, CONSEQUENTIAL, INCIDENTAL,
##* ECONOMIC OR PUNITIVE DAMAGES) ARISING OUT OF THE USE OF OR INABILITY
##* TO USE THIS PROGRAM, EVEN IF NEC Europe Ltd. HAS BEEN ADVISED OF THE
##* POSSIBILITY OF SUCH DAMAGES.
##*
##*     THE ABOVE HEADER MAY NOT BE EXTRACTED OR MODIFIED IN ANY WAY.
##*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##*/
#!/usr/bin/env python

access_token = dqa_client_lib.getToken(client_id, client_secret, client_auth_server)
token = access_token['access_token']
header={'client_id': client_id, 'access_token': token}
```

6.3. DQA Client Library

DQA Client Library is used to facilitate exchange of information to the MOBiNET DQA service.

```
##/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##*        MOBINET
##*
##*   File: dqa_client_lib_r3.py
##*
```

```
##*          NEC Europe Ltd. PROPRIETARY INFORMATION
##*
##* This software is supplied under the terms of a license agreement
##* or nondisclosure agreement with NEC Europe Ltd. and may not be
##* copied or disclosed except in accordance with the terms of that
##* agreement. The software and its source code contain valuable
##* trade secrets and confidential information which have to be
##* maintained in confidence.
##* Any unauthorized publication, transfer to third parties or
##* duplication of the object or source code - either totally or in
##* part - is prohibited.
##*
##*
##*      Copyright (c) 2016 NEC Europe Ltd. All Rights Reserved.
##*
##* Authors: Francesco Alesiani, francesco.alesiani@neclab.eu
##* Date: 10.10.2016
##*
##* NEC Europe Ltd. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
##* INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY
##* AND FITNESS FOR A PARTICULAR PURPOSE AND THE WARRANTY AGAINST LATENT
##* DEFECTS, WITH RESPECT TO THE PROGRAM AND THE ACCOMPANYING
##* DOCUMENTATION.
##*
##* No Liability For Consequential Damages IN NO EVENT SHALL NEC Europe
##* Ltd., NEC Corporation OR ANY OF ITS SUBSIDIARIES BE LIABLE FOR ANY
##* DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS
##* OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF INFORMATION, OR
##* OTHER PECUNIARY LOSS AND INDIRECT, CONSEQUENTIAL, INCIDENTAL,
##* ECONOMIC OR PUNITIVE DAMAGES) ARISING OUT OF THE USE OF OR INABILITY
##* TO USE THIS PROGRAM, EVEN IF NEC Europe Ltd. HAS BEEN ADVISED OF THE
##* POSSIBILITY OF SUCH DAMAGES.
##*
##*     THE ABOVE HEADER MAY NOT BE EXTRACTED OR MODIFIED IN ANY WAY.
##*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##*/
#!/usr/bin/env python

import httplib
useHttps = False

global_address ="mobicentre2.mobinet.eu"
global_prefix="/dqa"
global_portnum = 80

import json
def read_file(measure_file):
```

```
    with open (measure_file, "r") as myfile:
        data=myfile.read().replace('\n', '')
        return data
    return None

def measure_put_mydata(measure_file, header, verbose_flag=False,
address=global_address,port=global_portnum,prefix=global_prefix):
    if useHttps:
        conn = httplib.HTTPSConnection(address,port)
    else:
        conn = httplib.HTTPConnection(address,port)

    BODY = read_file(measure_file)
    data = json.loads(BODY)
    BODY = json.dumps({"body":data,"header":header})
    if verbose_flag:  print BODY
    conn.request("PUT", prefix+"/measure", BODY)
    r1 = conn.getresponse()
    if r1.status!=200 and verbose_flag:
        print(r1.status, r1.reason)
    data1 = r1.read()
    if data1 is not None and verbose_flag:
        print data1
    conn.close()
    return data1, r1.status==200

def something_what_how_body(what="user", how="PUT", BODY=json.dumps({'header':{'nid':'Z1',
"access_token": "YTcxNzkzODktNzlmNS00NzFjLTk5NjctMGRhMDFjYmUzMjQ1", "client_id":
"TI.CSE.client.dqaFixTestUser1.4605715283281456328"}}), verbose_flag=False,
address=global_address,port=global_portnum,prefix=global_prefix):
    if useHttps:
        conn = httplib.HTTPSConnection(address,port)
    else:
        conn = httplib.HTTPConnection(address,port)

    if verbose_flag: print 'BODY:',BODY
    conn.request(how, prefix+"/"+what, BODY)
    r1 = conn.getresponse()
    if r1.status!=200 and verbose_flag:
        print(r1.status, r1.reason)
    data1 = r1.read()
    if data1 is not None  and verbose_flag:
        print data1
    conn.close()
    return data1, r1.status==200

import requests, urllib, json
```

```
def getToken(client_id,client_secret,client_auth_server):
    resp = requests.post(client_auth_server, headers={"Content-type": "application/x-www-form-
urlencoded"}, data=urllib.urlencode({'grant_type': 'client_credentials', 'client_id': client_id, 'client_secret':
client_secret}), verify=False)
    access_token = None
    if resp.status_code==200:
        access_token =  json.loads(resp.text)
    return access_token
```