



D2.1 Technical Documentation of R1: Operation and Management Module

WP2 Development of Computed Assisted Integral Waste Management Systems

FDEUSTO

Report

Public

Reviewers: UPATRAS, ENBIO, ENG

Version	Date	Description of main changes	Author
1.0	27/11/2017	Frist draft	FDEUSTO
1.1	04/12/2017	Review from UPATRAS	UPATRAS
1.2	11/12/2017	Review from ENBIO	ENBIO
1.3	11/12/2017	Review from ENG	ENG
2.0	15/12/2017	Second draft	FDEUSTO
2.1	18/12/2017	Third draft	FDEUSTO
3.0	25/05/2018	Added Backend Security, Open Data Platform, Pilot Data Ingestion sections, User Stories as requested in the review letter	FDEUSTO
4.0	08/09/2018	Updated User Stories	FDEUSTO
4.1	27/06/2019	Updated deliverable with data model for the Food App according to reviewer comments.	FDEUSTO

Table of Contents

Table of Contents	2
Table of Figures.....	3
1. Introduction.....	6
2. Functional view of the system	10
3. FIWARE-LAB infrastructure	17
4. Backend	25
5. Design of the frontend.....	41
6. Sensor Layer technical documentation	64
7. Data model	75
8. Namespaces.....	103
9. References	111
Annex A: Deployment methodology	113
A1. Git Lab Deployment.....	113
A2. W4T Core Deployment	114
Annex B: NGSI API.....	117
Annex C: Cascais Lock System	121
Annex D: Halandri Weight Sensor.....	122
Annex E: Community Composting Plant Sensors	123
Annex F: Zamudio Citizen Card	124
Annex G: Truck Data App (confidential).....	125
Annex H: Kitchen Data App (confidential)	126
Annex I: Data Ingestion Template (confidential).....	127
Comments from external reviewers.....	128

Table of Figures

Figure 1. Generic IoT communication architecture.....	9
Figure 2. Relation of f R1 with the overall architecture of Waste4Think.....	11
Figure 3. FIWARE Lab Waste4Think Organization.	18
Figure 4. Waste4Think instances.....	18
Figure 5. Physical architecture.....	20
Figure 6. Security Group Rules.....	21
Figure 7. Waste4Think GitLab based development environment.	22
Figure 8. Waste4Think GitLab Projects.....	23
Figure 9. FIWARE Cloud Portal.	24
Figure 10. FIWARE Cloud - Management Storage Volumes.....	24
Figure 11. Conceptual back-end architecture.....	26
Figure 12. Logical architecture of the Publish/Subscribe Context Broker GE.	28
Figure 13. Waste Management data ingestion.....	32
Figure 14. Waste Management Data storage.....	33
Figure 15. Real-time processing.	34
Figure 16. Waste4Think backend scenario.	34
Figure 17. FIWARE Security GEs.	35
Figure 18. W4T backend security layer.	35
Figure 19. Security level 1 - Authentication.....	36
Figure 20. Security level 2 - Basic authorization.....	36
Figure 21. Security level 3 - Advanced authorization.....	36
Figure 22. NGSi data model.	37
Figure 23. Example of a WasteContainer entity.	37
Figure 24. Example of a Custom Attribute Metadata.....	39
Figure 25. Conceptual diagram of the MVC pattern.	41
Figure 26. W4T-Core.....	42
Figure 27. Examples of configurable dashboards.	43
Figure 28. Tab menu to change between dashboards.	44
Figure 29. Dashboard button to access the Dashboard Manager.....	44
Figure 30. Dashboard Manager.	45
Figure 31. Properties button within the Dashboard tool.....	45
Figure 32. Dashboard properties menu.....	46

Figure 33. Widgets button.....	47
Figure 34. Widgets toolbar.....	48
Figure 35. Behaviour properties of the widget.....	48
Figure 36. Data source properties of the widget.....	49
Figure 37. Data attribute properties of the widget.....	49
Figure 38. Live dashboard.....	50
Figure 39. Historical dashboard.....	51
Figure 40. Scenario dashboard.....	51
Figure 41. Event dashboard.....	51
Figure 42. Rule Manager view.....	52
Figure 43. Example of a rule.....	53
Figure 44. View to add operation to a rule.....	54
Figure 45. View to add a constant to a rule.....	54
Figure 46. View to link operations and operands.....	54
Figure 47. Add variable view.....	55
Figure 48. Main view of the Public Observatory.....	56
Figure 49. Baseline comparison of the Public Observatory.....	56
Figure 50. Main view of the Citizen Behaviour Tracking.....	57
Figure 51. List of available Citizen Behaviour Tracking reports.....	58
Figure 52. Creation of a new Citizen Behaviour Tracking report.....	59
Figure 53. Types of Open Data.....	60
Figure 54. Connection with the Open Data Platform.....	61
Figure 55. User interface of the Open Data Platform.....	61
Figure 56. Conceptual communication architecture of the CASCAIS sensors.....	66
Figure 57. Conceptual communication architecture of the HALANDRI sensors.....	67
Figure 58. Conceptual communication architecture of the SEVESO sensors.....	68
Figure 59. Conceptual communication architecture of the ENBIO treatment plant.....	70
Figure 60. Conceptual communication architecture of the GreenTech treatment plant.....	72
Figure 61. Linear and graph representation of an SLP to evaluate the function $2y(x + 1)^2 - 2y$	100
Figure 62. Namespace structure for the Context Broker.....	104
Figure 63. Pilots data document - Categories.....	105
Figure 64. Pilots data document - Entities.....	106
Figure 65. Pilots data document - Attributes.....	106
Figure 66. Pilots data document - Descriptions.....	106

Figure 67. Pilots data document – Editable part.....	107
Figure 68. Uploading systems.....	107
Figure 69. NgsiConnectorAPI	108
Figure 70. NgsiConnectorWEB - Login	108
Figure 71. NgsiConnectorWEB - Home page.....	109
Figure 72. NgsiConnectorWEB – Create entities page.....	109
Figure 73. NGSi Create Entity	110
Figure 74. NGSi Update Entity attribute	110
Figure 75. GitLab Volume	113

1. Introduction

From a generic point of view, an Internet of Thing (IoT) architecture is composed of the following four layers:

1. **Sensor / perception layer.** This layer is composed of the series of sensors that collect data from either an object under measurement in the environment or from the environment itself and turn it into useful data.
2. **Transport layer.** The transport layer transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, LAN, Bluetooth, RFID, and NFC. Ideally, each observed object would have an intelligent sensor attached, capable of retrieving the necessary data and then connecting over the Internet to send it to the middleware.
3. **Middleware / processing layer.** The processing layer is also known as the middleware layer. It stores, analyses, and processes huge amounts of data that come from both the transport layer and the business/services layer.
4. **Business/services layer.** This layer is responsible for delivering application specific services to the user. It defines various applications in which the IoT can be deployed.

Figure 1 shows the translation of this generic architecture to the components identified in Waste4Think, where:

1. **Sensor Layer (green).** This layer concentrates all the sensors/devices that act as input to define the waste management system. In particular:
 - **Sensors deployed at the bins and clean points:** Waste4Think will deploy identification tags, electronic locks and filling levels sensors in different containers, but there are other sensors available in the market, such as those to measure temperature, humidity, and position of the bins. Additionally, the Waste4Think clean points will have the same sensors as the bins, but in other cases, weighing systems may also be deployed.
 - **Sensors deployed at the truck:** several sensors are traditionally installed in the trucks, such as the bin identification system, the weighing sensor and the GPS system. Moreover, Waste4Think will also deploy a system to retrieve the information from the electronic locks and from a high-speed camera to estimate the amount of improper waste in the bins.
 - **Sensors deployed at the treatment plants:** Waste4Think will monitor different treatment plants. On the one hand, composting plants will have temperature and humidity sensors while, on the other hand, the Biowaste Treatment Unit and the Waste Treatment Unit will monitor the amount of the input material, the amount and quality of the different outputs and different control parameters of the treatment plant.
 - **Other sources of information:** These sources comprise a series of different input mechanisms such as surveys, focus groups, APPs, the serious games and learning materials (Learning Analytics) and other monitoring actions that feed information to the system.

Additional information about these sensors is covered in Section 6 of the current deliverable.

2. **Transport Layer** (red). In nearly all the cases, the sensors will use a wireless transport layer, in some cases the connection with upper layers will be direct, and in other cases there will be a device acting as a gateway. Nevertheless, in the case of the surveys and other monitoring actions, the transport layer will be wired based.
3. **Middleware** (blue). The selected middleware revolves around the use of the components of the FIWARE ecosystem. These layers are thoroughly detailed in Sections 2-4 of the current deliverable.
4. **Business/services** layer (purple). The middleware layer will feed the data to several applications (verticals) built upon it. These applications can be grouped into several categories:
 - *Persistence*. This category will group all the services responsible for storing the information for long term archiving. We can identify two components of this module:
 - Databases that are responsible for storing the relevant information fed from the sensor layer for internal reuse.
 - Open Data module that filters and consolidates the information to publish to the relevant repositories for external reuse.
 - *Operation and management services*. This category gathers all services in charge of the management of the waste collection system under six main groups.
 - Dashboard that graphically represents the information from the sensor layer as well as what is typically called Business Intelligence (alerts, forecasting, knowledge extraction, etc.).
 - Collection Management that will help in the management of the daily collection routes. This service will be covered in detail in Deliverable D2.5.
 - Long Term Planning that will perform long term simulations to help plan investments, the waste collection system and eco-events. This service will be covered in detail in Deliverables D2.7 and D2.9.
 - Zero Waste ecosystems / events that will allow the planification and monitorization of the events and ecosystems. This service will be covered in detail in Deliverable D1.3 and Deliverable 2.7.
 - Invoice Module that analyses the information stored in the persistence layer and produces invoices accordingly to the PAYT rules. This service will be covered in detail in Deliverable D5.1.
 - Public Observatory that will allow the citizens and entrepreneurs to access and exploit open data information. This service will be covered in detail in Section 5.4 of the current deliverable.
 - Citizen Behaviour Tracking that will help managers and teachers perceive the problems of the society concerning the understanding of the waste management system. This particular service will be covered in several deliverables as it is spread among several components and services.
 - Green Procurement that will help managers introducing green procurement into their workflows. This service will be covered in detail in Deliverable D1.3 and D2.7.

- Circular Economy that will help assessing the impact of implementing different best practices in an area. This service will be covered in detail in Deliverable D2.9.
- *Serious Games and Learning Materials*. This category groups the services that will be used to teach lessons about the waste collection system in different contexts. These services will be developed in Deliverables D4.3 to D4.5.
- *APPs layer*. This category groups three apps.
 - Citizen app that will help citizens retrieving useful information about the waste collection system.
 - Food waste app that will give food producers the opportunity to avoid generating food waste by looking for potentials re-users of the food.
 - Local trade app that will help citizens to retrieve useful information about the actions taking by local business to reduce their waste generation

In this deliverable, we provide a technical description of R1: Operation and Management Module. This module will feature several of the core components of the solution developed in Waste4Think. Section 2 will provide a broad description of them and their relationship with the rest of modules of the project. Section 3 focuses on the technical description of the environment (FIWARE-LAB) where the solution will be deployed. Section 4 contains a description of the middleware (FIWARE) and the Generic Enablers (GE) used as building blocks of R1. Section 5 details the mock-ups of the user interface for the components of R1: Dashboard, Rules Creator, Public Observatory, What-if scenarios creator, and the Citizen Behaviour Tracking. Section 6 contains the technical description of the new sensors that will be used in the project (both, the ones being developed and the commercial ones already available). Finally, Section 7 describes the complete data model that will be developed in the project and on which the whole intelligent waste management system approach will be based.

Please note that the status of the development for the R1: Operation and Management Module is provided in **the Monitoring Action Reports (Deliverables D1.4 to D1.7)**. Also, specifics related to the **deployment plan** can be found in **Table 11, Section 4.2 of Deliverable D1.4**.

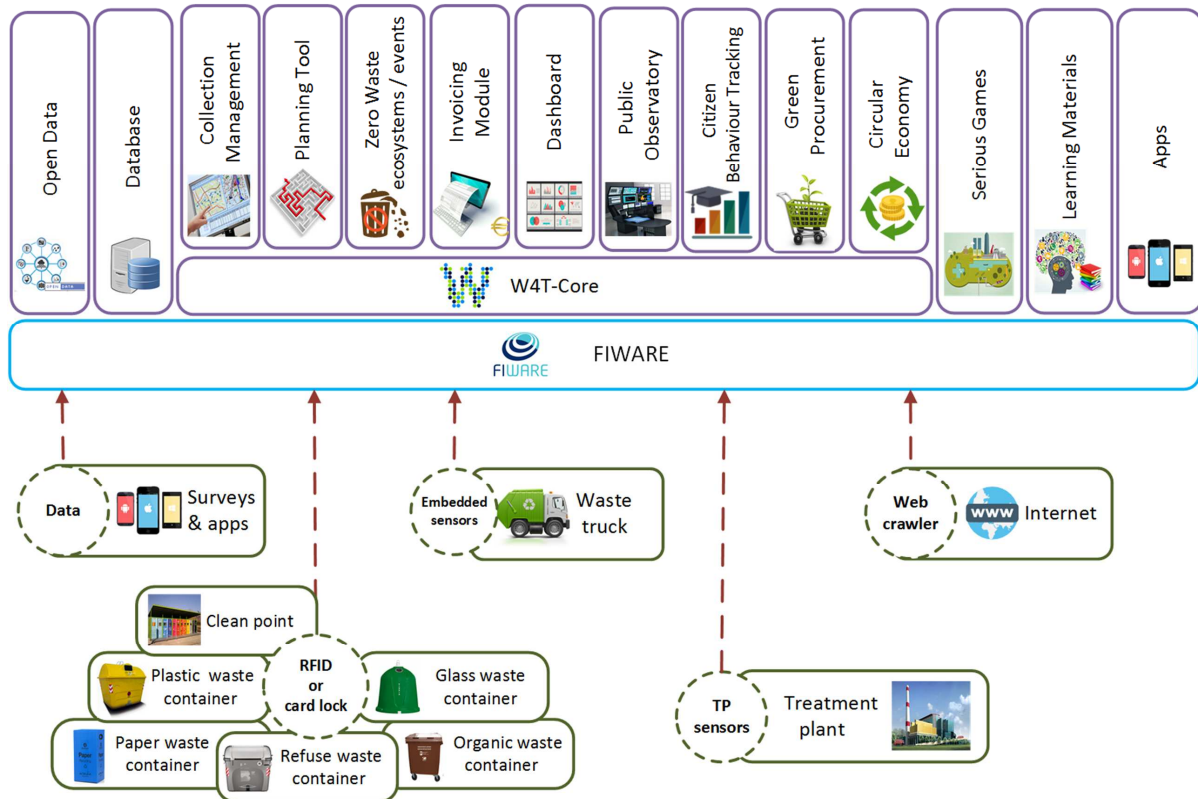


Figure 1. Generic IoT communication architecture

2. Functional view of the system

The overall functional view is provided in Section 6.3 of Deliverable D1.1. The current document exclusively details the functional view of the Operation and Management Module (R1). R1 is one of the biggest modules of the solutions developed in Waste4Think. It is composed of several components that span over all the layers of the solution:

- **Sensor/perception layer:** Several physical sensors are part of R1: the identification system deployed on the bins and clean points, weighing sensors deployed on trucks, the characterization sensors deployed on several places, the web crawlers, and, finally, the sensors of the community composting plan. Section 6 will cover these components in detail, and Deliverable D2.2 will feature the demonstration documentation.
- **Transport layer:** Different sensors will use different transport layers (GRPS but also custom RF-protocols) that could be considered as part of this eco-solution. Section 6 will describe the specific details of this layer, and Deliverable D2.2 will contain the demonstration documentation.
- **Middleware/processing layer:** This module is basically the middleware layer of Waste4Think. It will be based on the FIWARE middleware and will be responsible for receiving the information from the sensors and dispatching it to components that are in the Business / Service layer. Several components (called Generic Enablers) compose this layer. Section 4 contains a detailed description of these components, and Deliverable D2.3 will contain the demonstration documentation.
- **Business/Service layer:** The contribution of R1 to the Business and Service layer is centralized in the W4T-core module. This core component allows to access the rest of the modules in general, and the ones developed as part of R1, in particular: the Dashboard, the Public Observatory, and the Citizen Behaviour Tracking. Section 5 features a detailed description of each of the modules along with several mock-ups that will help define the final user interface. Please note that Deliverable D2.4 will contain the demonstration documentation.

Figure 2 shows the relations of the components of R1 (in blue) with the rest of eco-solutions of Waste4Think. As seen, R1 is directly or indirectly connected to almost all the other components of the project. This approach provides the following advantages:

- Eases the collection of information from the sensors in R19 and R20. Likewise, it will centralize the collection of information from the Apps (R5-R7), the Learning Analytics (R8-R13), and indirectly, from the Citizen Science Actions (R13-R15) as well as the different Social Actions (R17).
- Provides an abstraction layer to the real-time, historic or pre-processed information to the rest of eco-solutions.

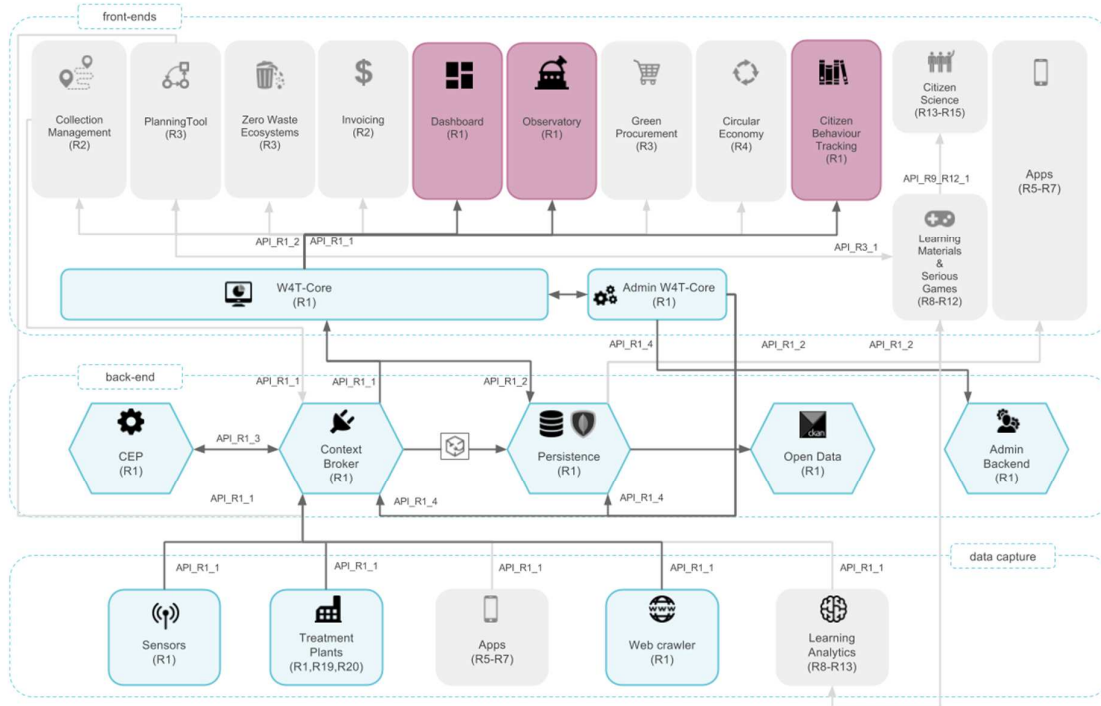


Figure 2. Relation of f R1 with the overall architecture of Waste4Think.

Several use cases are related to the different components of R1, in particular: Z1-Z9, S1-S5, S10, S12, A2, A6, C1-C10, C13 (see Deliverable D1.1). The lists of functional requirements fulfilled by the components of R1 and the related components of R2 are detailed in Table 1 and Table 2. Functional requirements of R3.

Table 1. Functional requirements of R1.

ID	FR	Sensors	Web crawler	Citizen app	Context Broker	Persistence	CEP	Open Data	Dashboard
R1_FR1	To monitor the waste generation	X		X	X				
R1_FR1.1	Allow to identify the users to the collection system	X		X	X				
R1_FR1.1.1	Allow to identify the users by an ID card (NFC)	X		X	X				
R1_FR1.1.2	Allow to identify the user by an iBag (RFID)	X		X	X				
R1_FR1.1.3	Allow to identify the bin by a RFID tag (D2D system)	X		X	X				
R1_FR1.2	Monitor the delivery of waste (control access)								
R1_FR1.2.1	Control the access to the containers with a black list mechanism	X							



R1_FR1.2.2	Control the access to the containers with a white list mechanism								
R1_FR1.2.3	Locks must be reprogrammable								
R1_FR1.3	Allow to characterize the waste deposited	X		X	X				X
R1_FR1.3.1	Identify the type of waste deposited (information of manual characterizations)	X		X	X				X
R1_FR1.3.2	Provide help in the identification of the improper waste when emptying the bin/bag	X		X	X				X
R1_FR1.3.3	Get the amount of waste deposited	X		X	X				X
R1_FR1.3.4	Get the time when the waste is generated (deposited or surplus)	X		X	X				X
R1_FR1.3.5	Get the localization where the waste is deposited	X		X	X				X
R1_FR1.3.6	Get the number of iBags delivered to users	X		X	X				X
R1_FR1.3.7	Monitoring the waste deposited in Eco-events	X		X	X				X
R1_FR1.4	Get information about the state of the deposit point and composting plants	X		X	X				X
R1_FR1.4.1	Allows the monitorization of the filling level (Bins)	X		X	X				X
R1_FR1.4.2	Allows the monitorization of the temperature (Composting)	X		X	X				X
R1_FR1.4.3	Allows the monitorization of the moisture (Composting)	X		X	X				X
R1_FR1.5	Be a fallback solution to publish information in the context broker			X	X				X
R1_NFR1.1	The system must be auditable			X	X	X			X
R1_FR2	Get information of a collection route			X	X	X			X
R1_FR2.1	Get the total fuel consumption of a collection route			X	X	X			X
R1_FR2.2	Estimate the total environmental impact of a collection route			X	X	X			X
R1_FR2.3	Get the GPS Trace of a collection route			X	X				X
R1_FR2.4	Get the FMS information from the CAN BUS			X	X				X
R1_FR2.5	Get the total weight of waste collected			X	X	X			X
R1_FR2.6	Get the places where the bin has been collected			X	X				X
R1_FR2.7	Get the time when the bin has been collected			X	X				X
R1_FR2.8	Allow to emit a certification of the work done during the shift				X				X
R1_FR3	Allow to report issues			X	X	X	X		X
R1_FR3.1	Allow to report the overfill of a container	X		X	X		X		X
R1_FR3.2	Allow to report vandalizing of a container	X		X	X		X		X
R1_FR3.3	Allow to report incorrect use of a container or bag			X					
R1_FR3.3.1	Allows to report the presence of improper waste	X		X	X	X			
R1_FR3.3.2	Allows to detect if a container is not inventoried	X							



R1_FR3.3.3	Allow to report a missing container		X					
R1_FR3.4	Allow to take evidences (photos or videos) of the incidences		X					
R1_FR3.5	Allow to request of specific collections (bulky waste)		X					
R1_FR4	To display the state of the waste collection system			X	X	X		X
R1_FR4.1	Allow to show the state of the waste collection system			X				X
R1_FR4.1.1	Allow to show the {past, current, planned} state of the system			X	X			X
R1_FR4.2	Detect anomalies in the waste generation patterns			X		X		X
R1_FR4.3	Allows to personalize the monitoring system			X	X			X
R1_FR4.3.1	Allows to filter the information provided by time, type and geographical distribution					X		X
R1_FR4.3.2	Allows to select and configure the KPIs to show							X
R1_FR4.3.3	Allows to define alerts					X		X
R1_FR4.4	Allows to create automatic reports in PDF, Excel and CSV			X	X			X
R1_FR4.5	The system must be able to crawl the web to get incidences	X						
R1_FR4.5.1	Crawl social networks like Twitter and Facebook	X						
R1_FR4.5.2	Crawl blogs and webpages	X						
R1_FR5	Provide a system to persist the information			X	X		X	
R1_FR5.1	Provide a communication channel to the persistence solution			X	X			
R1_FR5.2	Provide a communication channel to CKAN storage systems						X	
R1_NFR6.1	The communication channel should be secure (un-authorized person should not be able to read a message)			X	X			
R1_NFR6.2	The communication channel should be reliable (no information can be lost)			X	X			
R1_NFR6.3	Provide bidirectional communication system			X	X			
R1_NFR6.3.1	Provide bidirectional communication system between the {back-end, MAWIS and Wintarif and the sensors							
R1_NFR6.3.2	Provide bidirectional communication system between the {back-end, MAWIS and Wintarif and the truck							
R1_NFR6.3.3	The communication system should use the FIWARE platform			X				
R1_FR5.3	Data Transmission to the Server (for MOBA components)			X				
R1_FR5.3.1	Secured data transmission for PAYT			X				
R1_FR5.3.2	Data transmission for real time reporting							

Table 2. Functional requirements of R3.

ID	FR	Mid-Term	Long-Term	Green Procurement	Zero-Waste Ecosystems/Events	Invoicing
R3_FR1	To support the long-term planning	X	X			
R3_FR1.1	Provide visual tools to create predictive and explicative models		X			
R3_FR1.2	Provide visuals tools to help in the creation of baselines and what-if scenarios		X			
R3_FR1.2.1	Allow to modify in what-if scenarios the projected population and their spatial distribution		X			
R3_FR1.2.2	Allow to modify in what-if scenarios the waste type and quantity		X			
R3_FR1.2.3	Allow to modify in what-if scenarios the introduction of new treatment plants		X			
R3_FR1.2.4	Allow to modify in what-if scenarios the introduction of new collection systems	X	X			
R3_FR1.2.5	Allow to modify in what-if scenarios the mean for waste collection (electric truck, etc.)		X			
R3_FR1.2.6	Allow to modify in what-if scenarios the localization of the collection spots	X	X			
R3_FR1.2.7	Allow to modify in what-if scenarios the prevention campaigns		X		X	
R3_FR1.2.8	Allow to introduce and / or modify the economic instruments (tariff, PAYT, etc.)	X	X			X
R3_FR1.3	Perform long term simulations of the waste management system	X	X			
R3_FR1.3.1	Simulate the combination of collection system plus treatment plans		X			
R3_FR1.3.2	Simulate the localization and size of the collection points and treatment plans	X	X			
R3_FR1.3.3	Simulate combinations of prevention campaigns				X	
R3_FR1.3.4	Take into consideration multiple criteria (economical, environmental and social impacts)		X			
R3_FR1.3.5	Simulate different transport means for waste collection		X			
R3_FR1.3.6	Simulate the effect of different combination of economic instruments					X
R3_FR1.4	Allow the optimization of the waste collection for special events				X	
R3_FR1.5	Help in the creation of green procurements			X		
R3_FR1.5.1	To identify criteria for the tendering			X		
R3_FR1.5.2	To evaluate the applicants of the tender			X		
R3_NFR1	Open Source solution					



2.1. User Stories

Emilio Rivas lives in Bilbao, Spain, and he is the Head of the Waste Management Service of the city council. His responsibilities extend to the provision of all aspects of waste management: refuse collection, recycling, street cleanliness, transportation to landfill sites, and environmental waste contracts with the private sector.

Emilio arrives to his office. As usual, the first thing he does is open his Dashboards to see a broad overview of the status of the waste management system. He has previously personalized his Dashboards to see the following information:

- Alerts feed (table) of types:
 - container moved, container broken, overflow/overweight, improper waste, impossible access to the container,
 - route incidence, truck incidence,
 - special service requested
- Map with the predicted filling level of the containers
- Planned collection routes to be made today overlaid in the previous map
- Graph with the total amount of waste collected: last day, last week and last invoicing period

Emilio sees something strange in yesterday's measurements for one bin. The map widget allows the selection of one of the elements displayed and see its historical values. He does so and sees that this bin has been reporting erroneous values from some time. He opens the incidence system and creates a ticket to make someone check the sensor in this bin. Finally, he opens the CRUD tool. Then he searches for all the values of this sensor and deletes the erroneous values.

Emilio realises that he has been exclusively focusing on the technical aspects of the collection system, and now he also wants to include the monitoring of the economic, social and environmental aspects related to it. For this end, Emilio must:

1. Select the corresponding KPIs to monitor
2. Select the right widget to show the information

Emilio opens the configuration interface of the Dashboard module, analyses the list of KPIs and takes a note of the KPIs he wants to track. Now, he does the same with the screen that contains the list of widgets. He writes down the following table in a paper:

	KPI	Widget
Economic	Total amount of revenues (New)	Chart
Environmental	Total amount of GHG	Heat map
Social	Total negative quotes about waste collection system (New)	Historical Graph



Now, Emilio opens the configuration interface of his own personal dashboard. There he finds an “add new widget” button. A new screen appears where he has to select the widget to visualize the information. He will start with the environmental information that he wants to track as he does not need to create any new KPI. He selects a Heat Map and configures its visualization parameters (colour gradient, scale of the values, etc.) and the source of information. For this last step, Emilio opens the KPI search box and looks for the right KPI. In this case, this is not just one KPI since he wants to track the variation of a specific attribute for a list of entities. To this end, he has to write a little query where he selects the attribute “emission” of all entities “routes”. Next, he just has to drag and drop to the place where he wants to so see this map.

Then, Emilio wants to add the economic information to the Dashboard. He performs the same actions as before, but in this case, he selects a chart widget and when prompted to select a source of information, he hits the “create a new KPI” button. Now, Emilio has the visual programming tool where he can define a new KPI. In this case, the KPI is just the aggregation of all the revenues of the project, so he only needs make a couple of simple queries to select the sources of information and select the aggregation component. Then, he completes the form with the rest of information about the KPI, and he finally can continue creating the Dashboard as in the environmental information.

Next, Emilio wants to add the social information. He performs the same actions as in the previous cases, and when prompted to select a source of information he hits the “create a new KPI” button. As before, Emilio uses the visual programming tool to define this KPI. In this case, the KPI is a search of negative quotes about the waste collection, so he selects the “social behaviour tracking component” in the tool and configures it to perform an analysis of the texts published. Then, as before, he completes the form with the rest of information about the KPI and he finally can continue creating the Dashboard as in the environmental information.

Finally, he thinks that this information is quite interesting to the citizen. To allow this public visualization, he now goes to the Public Observatory Dashboard. This dashboard is a regular dashboard, but it is shown in the webpage and in the Citizen App. To this end, Emilio just repeats the previous actions in this Dashboard. Note that now, Emilio goes quite fast as all the KPIs have already been created, so he only has to choose the widgets, configure its shapes, link the widget to the sources of information and finally, and place it in the Public Observatory.



3. FIWARE-LAB infrastructure

FIWARE Lab is a non-commercial sandbox environment where innovation and experimentation based on FIWARE technologies takes place. Entrepreneurs and individuals can test the technology as well as their applications on FIWARE Lab, exploiting Open Data published by cities and other organizations. FIWARE Lab is deployed over a geographically distributed network of federated nodes leveraging on a wide range of experimental infrastructures [1].

The Waste4Think project, as defined in the “Description of Action” document [2], will exploit FIWARE Lab as its own IT infrastructure.

Three different systems will be deployed on FIWARE Lab:

- a Development Platform that is used for code versioning and sharing, collaborative documentation creation and sharing through wiki, issue tracking, continuous integration and delivery (see Section 2.4)
- a Back-End management system which is a middleware platform based on FIWARE technologies (see section 3)
- the Front-End systems (monitoring dashboard, the what-if scenario tool and the green procurement tool) [see Section 5 of the current Deliverable and Deliverable 2.7]

These services will be provisioned and managed using the FIWARE Cloud capabilities of the FIWARE LAB node located in SPAIN.

To access the FIWARE Lab resources, on January 2017, a FIWARE Community Account [6] has been released and the following Cloud resources have been requested:

- FIWARE Lab Node: Spain
- Number of VMs: 8
- Public IPs: 2
- Total vCPUs: 16
- Total RAM: 32
- Total hard disk: 320

The FIWARE Community Account has a default duration of 9 months and an extension will be asked after this time period. The procedure to extend the FIWARE account will be made available in due time by the FIWARE support team as specified in the FIWARE User policy [6].

Moreover, a Waste4Think Organization (Figure 3) has been created on the FIWARE Lab to let more than one user access the shared cloud resources.

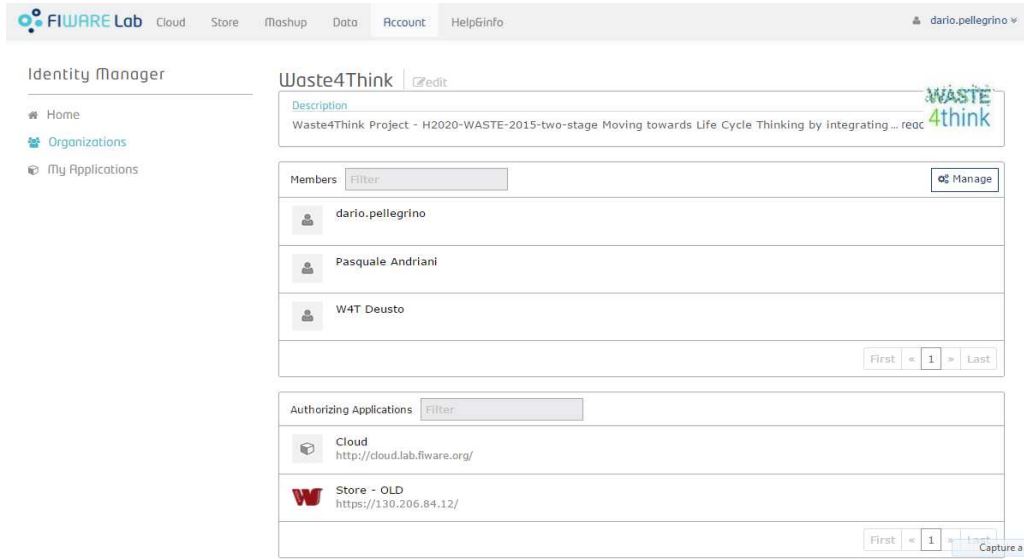


Figure 3. FIWARE Lab Waste4Think Organization.

At the time of writing, the deployment of the Waste4Think project on the FIWARE Lab (Spain Node) includes the following instances, also shown in Figure 4:

- Development Environment VM (GITLab) [3]
- NGNIX Reverse Proxy [19]
- Publish/Subscribe Context Broker GE – Orion [5]
- Complex Event Processing GE – PROTON [9]
- STH (Short Historical Term) – COMET [14]
- Front-end VM
- MySQL [15]
- CKAN OpanData Management [10]

The two public IPs provided has been assigned to the Development Environment VM (130.206.120.215) and to the NGNIX Reverse Proxy VM (http://130.206.117.164).

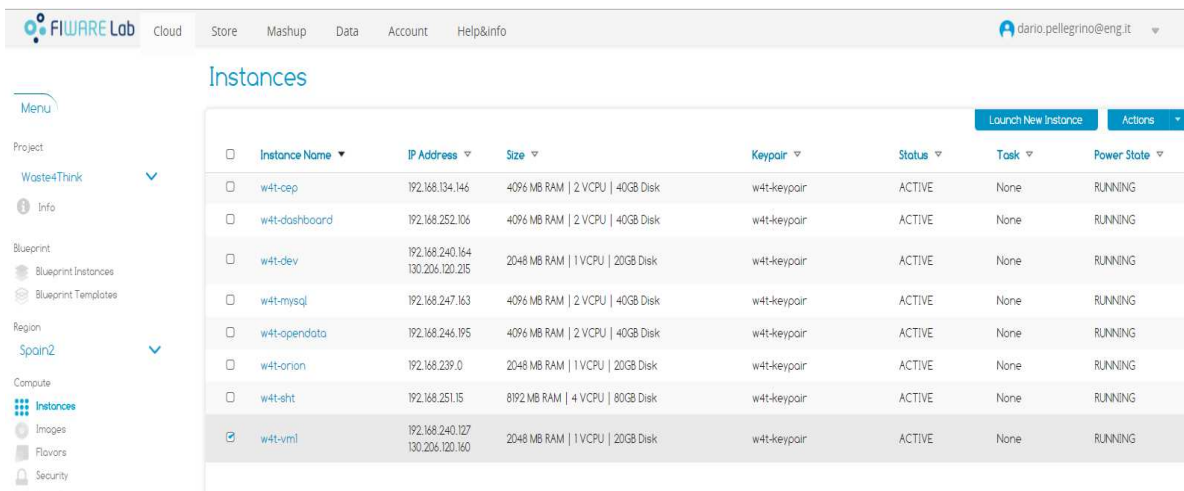


Figure 4. Waste4Think instances.



3.1. The FIWARE initiative

“FIWARE is an innovative, open cloud-based infrastructure for cost-effective creation and delivery of Future Internet applications and services, at a scale not seen before. These APIs are public and royalty-free, driven by the development of an open source reference implementation which accelerates the availability of commercial products and services based on FIWARE technologies. [1]”

In 2011 the European Commission together with the major ICT players launched an ambitious project: the realization of innovative open source technologies made in Europe, created with the aim of enabling the creation and the development of new internet-based products and services. They have created the FIWARE programme, a public-private partnership able to invest half a billion euro to stimulate the growth of entrepreneurial ecosystem, providing a set of innovative tools able to ease the realization of new ideas.

FIWARE provides cloud hosting services based on OpenStack technology and a set of components offering many added-value functions “as a service”, the Generic Enablers (GEs). The Generic Enablers are based on open standard APIs that make it easier to connect to the Internet of Things, process data and media in real-time at large scale, perform Big Data analysis or incorporate advanced features to interact with the users; FIWARE can be considered as an open alternative to existing proprietary Internet platforms.

All the Generic Enablers are listed in the FIWARE catalog, where users can find documentation, hands-on about how to create a GE instance and a reference person which is the point of contact between users and the developer of the technology and that provides support on specific aspects of each GE. All these services are offered without any lock-in, with the aim of maximizing the opportunities for developers to launch new FIWARE based products and services.

From 2011 to date, FIWARE Generic Enablers have crossed different phases of a path of development aimed at creating and testing the technologies. The first phase had the goal of define and realize the Generic Enablers. In the second phase, a number of trial sites which tested the enablers were set up. The third phase, ended in 2016, had the objective to launch them in the ecosystem of the entrepreneurs and developers.

An interesting example of implementation of projects through FIWARE is represented by FINESCE [21], a project of the second phase of the programme, which set up 7 trial sites to test application in the energy field based on FIWARE GEs. In the city of Terni, where 22% of the energy demand is covered by renewable energy sources and the problem of balancing between demand and supply of energy is high, FINESCE created a system based on a mix of demand-side management and innovative techniques of citizens’ involvement, whose technology leverage on 8 FIWARE Generic Enablers allowing the management of the large amount of data coming from the different sources to optimize the grid stability.

In September 2014, the third phase of the programme, FIWARE Accelerate, was officially launched in Munich; this phase received 80 million of euros to be distributed by 16 Accelerators projects as a free grant to start-ups willing to develop new product and services leveraging on the Generic Enablers. Moreover, the 16 Accelerators had also offered an

acceleration program to each start up selected to speed the launch of their products in the market.

Although it was born in Europe, FIWARE has been designed with a global ambition, so that benefits can spread to other regions. The FIWARE Mundus program is designed to bring coverage to this effort engaging local ICT players and domain stakeholders, and eventually liaising with local governments in different parts of the world, including Latin American, Africa and Asia.

3.2. Physical architecture

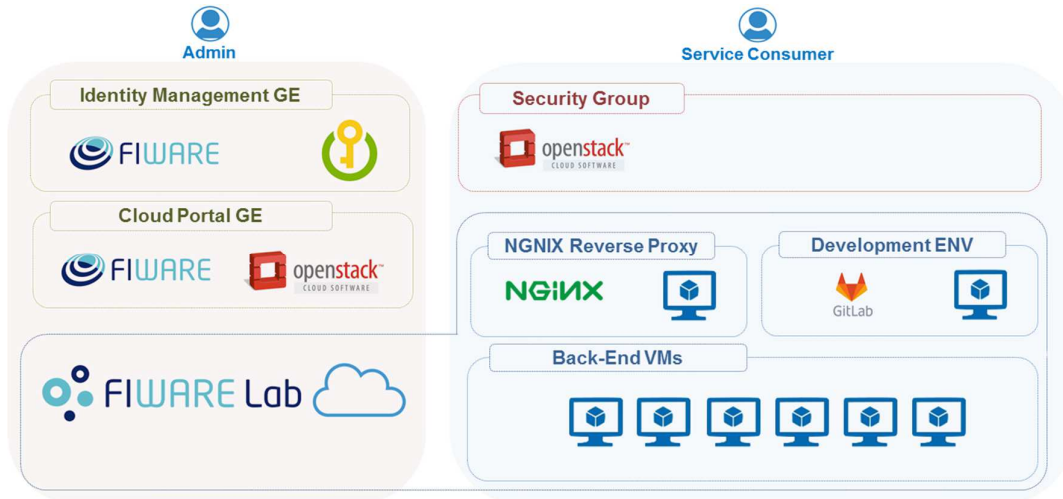


Figure 5. Physical architecture.

The physical architecture (Figure 5) includes two different FIWARE Generic Enablers:

Self Service Interfaces GE - Cloud Portal - The Self-Service Interfaces provide a support for the users of the cloud infrastructure and platform to manage their services and resources deployed in the cloud. It consists of open source implementation of a User Portal and Scripts. The User Portal is implemented in a form of a Web GUI following the same functionality as the OpenStack Dashboard. The scripts facilitate direct approach to the underlying cloud resources through a command line and is addressed for administrators [12].

Identity Management GE – KEYROCK - Identity Management covers many aspects involving users' access to networks, services and applications, including secure and private authentication from users to devices, networks and services, authorization & trust management, user profile management, privacy-preserving disposition of personal data, Single Sign-On (SSO) to service domains and Identity Federation towards applications. The Identity Manager is the central component that provides a bridge between IdM systems at connectivity-level and application-level. Furthermore, Identity Management is used for authorising foreign services to access personal data stored in a secure environment. Hereby usually the owner of the data must give consent to access the data; the consent-giving procedure also implies certain user authentication [13].

Two kinds of actors have access to the FIWARE LAB resources:



- the Admin that manage the whole FIWARE Cloud resources;
- the Service Consumer actors who are going to access each of the deployed services, in particular:
 - Waste4Think developers;
 - the sensors of the pilots;
 - APIs to retrieve data from the storage systems.

The Admin actor can access the FIWARE Cloud resources through the IDM GE by using an authorized FIWARE account and then, through the Cloud Portal GE, which is an OpenStack based application, is able to manage VMs, Security groups, Floating IPs and Keypairs. The Service Consumer actors can access the Back-End services through an NGNIX Reverse proxy. Access both to the Back-end services and development environment ports is granted with security rules defined by the Admin actor on Cloud Portal GE (see Figure 6 or the security groups defined within the FIWARE infrastructure).

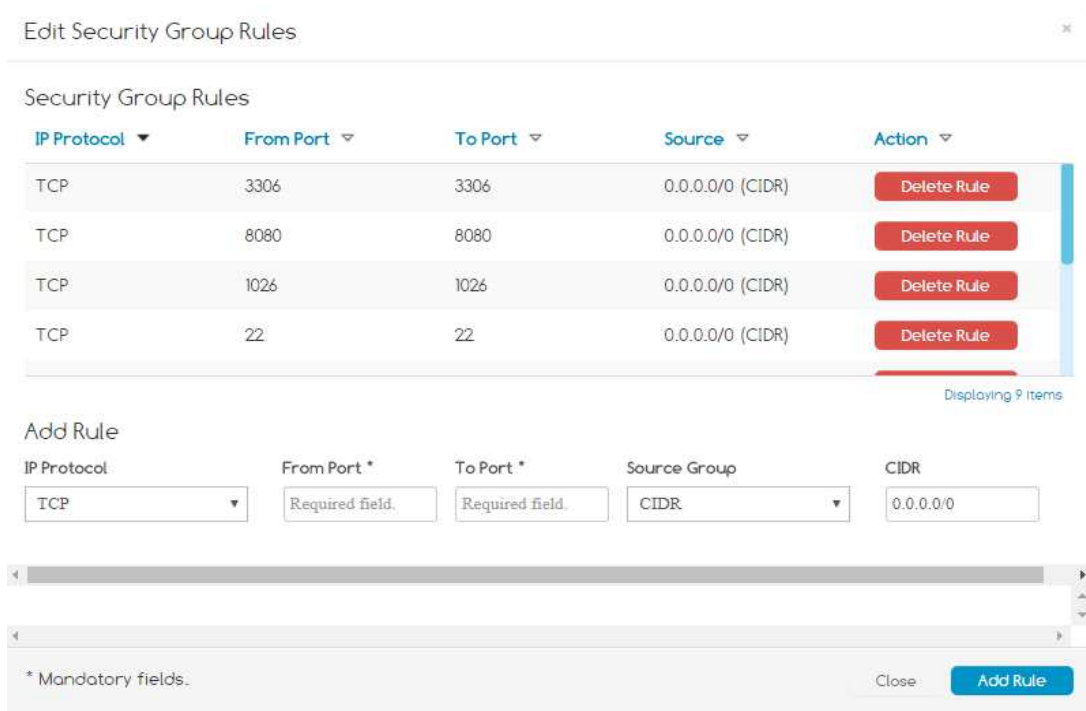


Figure 6. Security Group Rules.

3.3. Development Environment

A private instance of GitLab [7], hosted on the FIWARE Lab, has been deployed and configured to provide the Waste4Think project with its own Development Environment (see ANNEX A: GitLab installation procedure for the GITLab installation procedure). The Waste4Think GitLab instance is accessible at <http://dev.waste4think.eu/> (see Figure 7).

Waste4Think GitLab Repository



GitLab Repository for the Waste4Think project

Sign in

Username or email

Password

Remember me [Forgot your password?](#)

Figure 7. Waste4Think GitLab based development environment.

A GitLab development environment provides the following functionalities [3]:

- **Project:** organize your project repository into private, internal, or public projects;
- Built-in Continuous Integration and Continuous Delivery;
- **Issue Tracker:** quickly set the status, assignee or milestone for multiple issues at the same time or easily filter them on any properties. See milestones and issues across projects;
- **Cycle Analytics:** dashboard that lets teams measure the time it takes to go from an idea to production;
- **GitLab Pages:** easy system for hosting static sites using GitLab repositories and GitLab CI, complete with custom domains and HTTPS support;
- **Built-in Docker Registry:** GitLab Container Registry is a secure and private registry for Docker images. It allows for easy upload and download of images from GitLab CI. It is fully integrated with Git repository management;
- **Review Apps:** with Review Apps you can spin up dynamic environments for your merge requests and preview your branch in a live environment;
- **User roles:** manage access and permissions;
- **ToDos:** when a user is mentioned in or assigned to a merge request it will be included in the user ToDos, making the development workflow faster and easier to track;
- **Merge Requests:** create merge requests and @mention team members to review and safely merge your changes;
- **Merge conflict resolution:** preview merge conflicts in the GitLab UI and tell Git which version to use;
- **Activity Stream:** view a list of the latest commits, merges, comments, and team members on your project;
- **Custom Notifications:** Be notified by email, Slack, or ToDos anytime there are changes to an issue or merge request.

In the Waste4Think project, the private instance of GitLab CE [4], will be used for code versioning and sharing, collaborative documentation creation and sharing through wiki, issue tracking, continuous integration and delivery. Figure 8 shows the list of active projects at the time of writing.

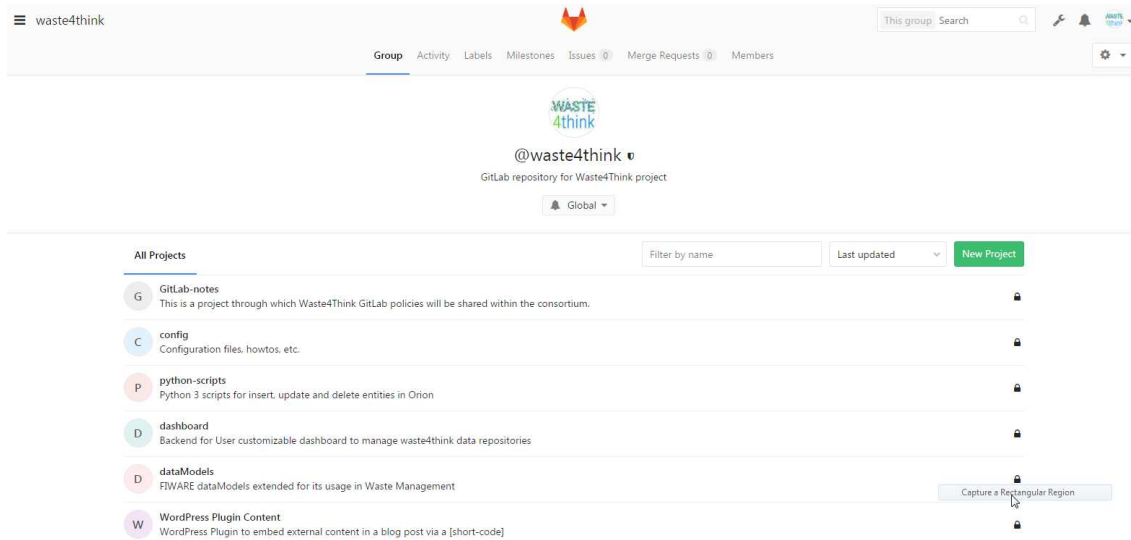


Figure 8. Waste4Think GitLab Projects.

3.4. VMs Deployment User Manual

The Back-End Virtual Machine instances are provisioned and managed using the FIWARE Cloud capabilities of the FIWARE Lab node “Spain2”.

The FIWARE Cloud is an Infrastructure as a Service platform based on OpenStack, comprising the Compute (Nova), Storage (Cinder), Network (Neutron) and Image (Glance) services.

The service FIWARE Cloud Portal (see Figure 9) has been used to provision the Virtual Machine instances by specifying the VM configuration ‘flavour’ (from a catalogue of pre-defined flavours configured by the administrator) and virtual machine image, which comprise a pre-packaged Operating System and a software stack managed by OpenStack Image (Glance) service.

The screenshot shows the 'Istanze' (Instances) page in the FIWARE Cloud Portal. The page features a navigation sidebar on the left with categories like 'Progetto', 'Compute', 'Rete', and 'Orchestrazione'. The main content area displays a table of instances with columns for Name, Image Name, IP Address, Dimensions, Key Pairs, Status, Availability Zone, Activity, Activation State, and Time since creation. Each instance has a 'Crea istantanea' (Create Snapshot) button in the 'Actions' column.

Nome istanza	Nome immagine	Indirizzo IP	Dimensione	Coppia di chiavi	Stato	Zona di disponibilità	Attività	Stato attivazione	Tempo trascorso dalla creazione	Actions
w4t-cep	cep-r5.4.3-img	192.168.200.85	m1.medium	w4t-keypair	Attivo	nova	Nessuna	In esecuzione	19 ore, 41 minuti	Crea istantanea
w4t-orion-prod	base_centos_6	192.168.193.207	m1.medium	w4t-keypair	Attivo	nova	Nessuna	In esecuzione	1 mese, 2 settimane	Crea istantanea
w4t-backend	base_ubuntu_16.04	192.168.252.106	m1.medium	w4t-keypair	Attivo	nova	Nessuna	In esecuzione	4 mesi, 2 settimane	Crea istantanea
w4t-sht	base_centos_7	192.168.251.115	m1.large	w4t-keypair	Attivo	nova	Nessuna	In esecuzione	4 mesi, 3 settimane	Crea istantanea
w4t-opendata	ckan_2.5	192.168.246.195	m1.medium	w4t-keypair	Attivo	nova	Nessuna	In esecuzione	7 mesi, 1 settimana	Crea istantanea
w4t-dev	base_centos_7	192.168.240.164 IP mobili: 130.206.120.215	m1.small	w4t-keypair	Attivo	nova	Nessuna	In esecuzione	9 mesi, 2 settimane	Crea istantanea

2015 © FIWARE. The use of FIWARE Lab services is subject to the acceptance of the [Terms and Conditions](#), [Privacy Policy](#) and [Cookies Policy](#).

Figure 9. FIWARE Cloud Portal.

Indeed, FIWARE Lab holds a rich catalogue of Virtual Machine images, including base Operating Systems (such as Ubuntu, Centos or Red Hat) as well as images comprising pre-packaged installations of the various Generic Enabler implementations (e.g., Orion Context Broker or CEP). It is possible to automatically customize the guest Operating System configuration on the first boot of the Virtual Machine, as well as to control the power state of the Virtual Machine (power on/off), suspend/resume, take snapshots, etc.

The OpenStack Storage (Cinder) service (see Figure 10) provided by FIWARE Cloud, has been used to provide persistent storage volumes, allocated from one of the storage pools configured by the administrator. These volumes can be then attached to a Virtual Machine instance, to hold persistent data of an application.

The screenshot shows the 'Volumi' (Volumes) page in the FIWARE Cloud Portal. The page features a navigation sidebar on the left. The main content area displays a table of storage volumes with columns for Name, Description, Dimensions, Status, Type, Attached to, Availability Zone, Available, Encrypted, and Actions. Each volume has a 'Modifica volume' (Modify Volume) button in the 'Actions' column.

Nome	Descrizione	Dimensione	Stato	Tipo	Collegato a	Zona di disponibilità	Arviviabile	Codificato	Actions
orion-prod-data	-	60 GB	In uso	-	Collegato a w4t-orion-prod su /dev/vdb	nova	No	No	Modifica volume
git-data-repo	gitlab data repository	50 GB	In uso	-	Collegato a w4t-dev su /dev/vdc	nova	No	No	Modifica volume

2015 © FIWARE. The use of FIWARE Lab services is subject to the acceptance of the [Terms and Conditions](#), [Privacy Policy](#) and [Cookies Policy](#).

Figure 10. FIWARE Cloud - Management Storage Volumes.



3.5. Working methodology

The development of the project will rely on the use of an Agile Software Development methodology. This methodology defines an iterative and incremental approach for working in ever changing environments. Iterations are called *sprints* and usually last two to four weeks. Each sprint begins with a meeting which establishes the objectives that must be achieved during said period. This meeting will be open to all the partners and stakeholders, but it should be considered that for these sprints to be manageable, the size of the working groups must be small (maximum 10 participants). The sprint objectives are selected among those not achieved on a previous sprint, objectives already defined in the project or any new objectives. Then, each of the developers chooses the objective on which they will work during the sprint. If their tasks are completed, they may help other members to advance in the next sprint.

A pair programming methodology will be used in critical tasks of the project. Pair programming is a technique where two programmers share a computer screen and co-work in solving the task at hand. It has been proved that pair programming has several advantages over the traditional methodology, such as: an important reduction in the number of software defects, increment of the code readability and design quality, improvement of the knowledge and communication shared between all the developers.

At the end of the sprint, incomplete objectives are analysed to decide whether it is necessary to subdivide them into several tasks for the next iteration. Objectives are built in a way that there is always a potentially usable prototype based on the completed developments of previous sprints. The use of this iterative methodology helps build a platform that addresses all the objectives set and implements controlled use cases. The implementation of the most basic operations will help verify the robustness of the development and draw conclusions for more complex stages.

4. Backend

4.1. Back-End conceptual architecture

The objective of the WP2 “Development of computer assisted integral waste management system” is the definition and development of the ICT tools for improving the short and long-term waste management at local and territorial level as well as the planning and monitoring solutions of the different Zero Waste Environments of the Waste4Think project.

To achieve this objective and, in particular, to get the results R1 “Operation and Management Module”, R2 “Collection Module”, R3 “Planning Module” and R4 “Circular Economy Model”, a generic waste management back-end system will be designed and developed.

Figure 11 represents the conceptual architecture of the waste management back-end system, which is a middleware platform based on FIWARE technologies which will be able:

- to collect context information coming from the different sensor devices that are going to be deployed in each pilot;

- to process and analyse real-time events and to trigger instantaneous predefined actions (such as alerts and/or anomalies);
- to store context information as historical data (possibly exposed as open data);
- to provide a set of public and/or private APIs to retrieve historical data;
- to provide a set of FIWARE security components to develop authentication and authorization systems both for the pilots and front-end applications.

All the FIWARE components will be provisioned and managed using the FIWARE Cloud capabilities of the FIWARE LAB node located in SPAIN.

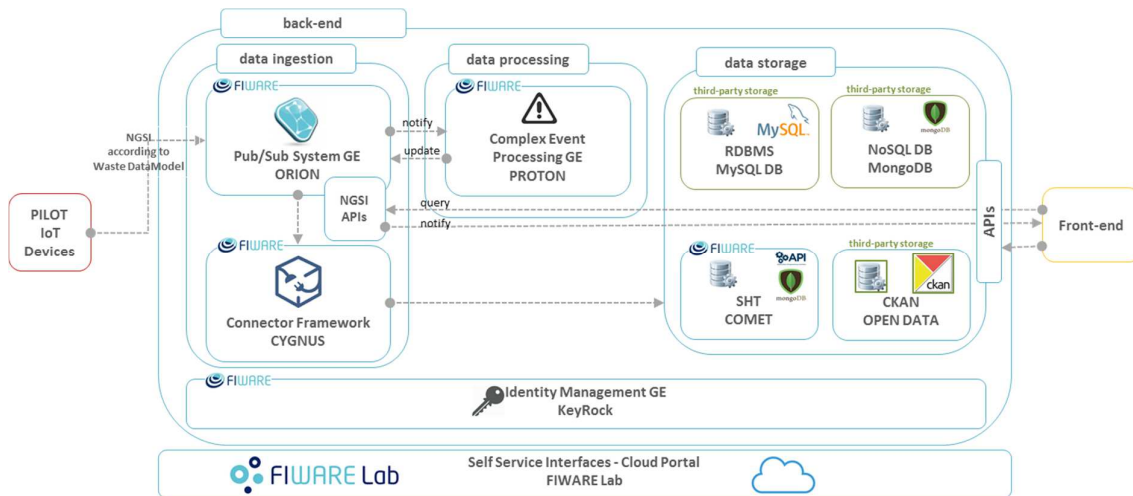


Figure 11. Conceptual back-end architecture.

The waste management back-end system will consist in three different data phases:

- data ingestion;
- data processing;
- data storage.

In the data ingestion phase, the context information coming from the different pilots' sensors will be managed, according to the Waste Management data model, via a publish/subscribe system which allows managing the whole lifecycle of context information including updates, queries, registrations and subscriptions. The data ingestion phase will also guarantee the delivery of the context information both to the data storage phase, through a connector framework, and to the data processing phase.

In the data processing phase, a Complex Event Processing system will analyse input event data in real-time and it will generate instantaneous events/alerts that reply to changing conditions.

In the data storage phase, the context information, through the connector framework, will be stored in a set of different third-party storage systems like a Relational Data Base, NoSQL data base or Open Data. The data storage phase will also provide a set of public and/or private APIs to share historical data to the external back-end world.



4.2. Back-End components

The Back-End architecture will integrate the following set of FIWARE GEs and other third-party components to cover all the functionalities provided by the system:

- FIWARE GEs
 - Publish/Subscribe Context Broker GE - ORION
- Other FIWARE components
 - Connector Framework – CYGNUS
- Third-party storage system
 - CKAN Open Data Management GE
 - MySQL (Relational DB)
 - MongoDB (NoSQL DB)

4.3. FIWARE Generic Enablers

4.3.1. Publish/Subscribe Context Broker GE – ORION

The Publish/Subscribe Context Broker GE will enable publication of context information by entities, referred as Context Producers, so that published context information becomes available to other entities, referred as Context Consumers, which are interested in processing the published context information. Applications or even other GEs in the FIWARE platform may play the role of Context Producers, Context Consumers or both. Events in FIWARE based systems refer to something that has happened or is contemplated as having happened. Changes in context information are therefore considered as events that can be handled by applications or FIWARE GEs.

The Publish/Subscribe Context Broker GE supports two ways of communications: push and pull towards both the Context Producer and the Context Consumer. It does mean that a Context Producer with a minimal or very simple logic may continuously push the context information into the Publish/Subscribe Context Broker, when the information is available or due to the internal logic of the Context Producer. The Publish/Subscribe Context Broker on its side can request the context information from Context Producers if they provide the ability to be queried (Context Producers able to act as servers are also referred as Context Providers). In a similar way, Context Consumers can pull the context information from the Publish/Subscribe Context Broker (on-request mode), while the Publish/Subscribe Context Broker can push the information to Context Consumer interested in it (subscription mode).

A fundamental principle supported by the Publish/Subscribe Context Broker GE is that of achieving a total decoupling between Context Producers and Context Consumers. On one hand, this means that Context Producers publish data without knowing which, where and when Context Consumers will consume published data; therefore, they do not need to be connected to them. On the other hand, Context Consumers consume context information of their interest, without this meaning they know which Context Producer has published a particular event: they are just interested in the event itself but not in who generated it. As a result, the Publish/Subscribe Context Broker GE is an excellent bridge enabling external applications to manage events related to the Internet of the Things (IoT) in a simpler way

hiding the complexity of gathering measures from IoT resources (sensors) that might be distributed or involving access through multiple low-level communication protocols.

All the communications between the various components of the Publish/Subscribe Context Broker high-level architecture occur via the NGSI RESTful interface interfaces/protocols. NGSI RESTful interface is inspired and based on the OMA NGSI specification [8]. This is a standard interface allowing to handle any type of data, which may include meta-data [5]. Figure 12 shows the logical architecture of the Publish/Subscribe Context Broker GE.

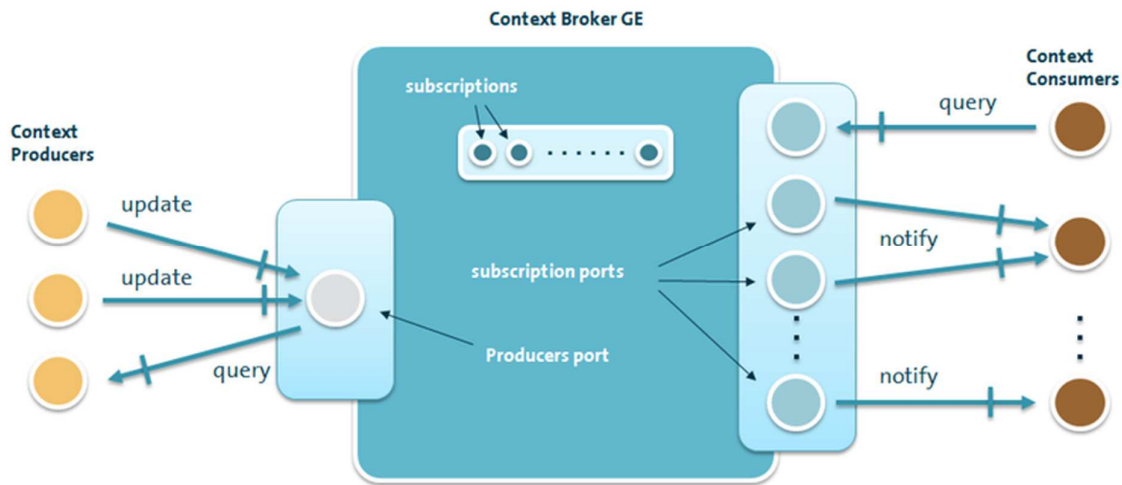


Figure 12. Logical architecture of the Publish/Subscribe Context Broker GE.

4.4. Other FIWARE components

4.4.1. Connector framework – CYGNUS

Cygnus is a connector in charge of persisting certain sources of data in certain configured third-party storages, creating a historical view of such data [17].

Internally, Cygnus is based on [Apache Flume](#) [20], a technology addressing the design and execution of data collection and persistence agents. An agent is basically composed of a listener or source in charge of receiving the data, a channel where the source puts the data once it has been transformed into a Flume event, and a sink, which takes Flume events from the channel to persist the data within its body into a third-party storage.

Cygnus is designed to run a specific Flume agent per source of data. The current stable release can persist the following sources of data in the following third-party storages:

- [HDFS](#), the [Hadoop](#) distributed file system.
- [MySQL](#), the well-known relational database manager.
- [CKAN](#), an Open Data platform.
- [MongoDB](#), the NoSQL document-oriented database.
- [FIWARE Comet \(STH\)](#), a Short-Term Historic database built on top of MongoDB.



4.5. Other components

The back-end architecture may include also other components which are not part of the FIWARE technologies but are compatible with CYGNUS Connector Framework.

4.5.1. CKAN Open Data Management GE

CKAN is the world's leading open-source data portal platform — a powerful data management system that makes data accessible – by providing tools to streamline publishing, sharing, finding and using data. CKAN is aimed at data publishers (national and regional governments, companies and organizations) wanting to make their data open and available.

Features include:

- Complete catalogue system with easy to use web interface and a powerful API
- Strong integration with third-party CMS's like Drupal and WordPress
- Data visualization and analytics
- Workflow support lets departments or groups manage their own data publishing
- Fine-grained access control
- Integrated data storage and full data API
- Federated structure: easily set up new instances with common search

4.5.2. MySQL DB

MySQL is an open source relational database management system (RDBMS) based on Structured Query Language (SQL).

MySQL runs on virtually all platforms, including Linux, UNIX, and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web-based applications and online publishing and is an important component of an open source enterprise stack called LAMP. LAMP is a Web development platform that uses Linux as the operating system, Apache as the Web server, MySQL as the relational database management system and PHP as the object-oriented scripting language. Sometimes Perl or Python is used instead of PHP [15].

4.5.3. MongoDB

MongoDB is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas. MongoDB is developed by MongoDB Inc. and is free and open-source, published under a combination of the GNU Affero General Public License and the Apache License [16].

4.5.4. Authentication module

The authentication module centralizes all aspects involving user profiles and their access to services and applications, including secure and private authentication and Single Sign-On (SSO). The authentication server is the only component that stores all the users of the platform, assigns their identifiers, relates which modules they have access to, and provides foreign services to access personal data, all the while providing a secure environment.



This module has been developed on top of Django by extending its built-in user authentication system which handles user accounts, groups, permissions and cookie-based user sessions. This reduces the effort for account creation and management, as it supports the enforcement of policies and procedures for user registration, user profile management and the modification of user accounts. This built-in user authentication system allows administrators to quickly create customized pages for users, registration of tenant applications with access to user profile data and the handling of error notifications.

For users to Single-Sign-On (SSO) across all modules, the sharing of the authorization, identification and role privileges is done through OAuth2 [26]. This is integrated within the previously described Django user system. Whenever an anonymous user tries to log in another module of the Waste4think ecosystem, the module derives the authentication process to this central login service. Doing so, full user information (e.g. name, address, phone, e-mail) is only stored in here while, remaining modules will only store a unique identifier for this user together with the locally needed attributes.

4.5.5. Statics module

This component is a global file storage built to save and retrieve any amount of data from all the Waste4Think modules. Its purpose is to centralize the location where static file resources are hosted, while providing a user security layer when uploading and making the data globally available for downloading. The Statics module is one of the most used tools by the rest of components since all web files (e.g. CSS, JS, XML) and data files (e.g. CSV, SHP, JSON) are uploaded and read from here.

4.5.6. Datasources module

Data-set registration and harmonization module, which purpose is to allow access to any piece of information in a homogeneous format, no matter how or where it is read from. The Datasources module exposes both a graphical user interface for registering new data-sets and a REST API for other modules to retrieve said data in standard JSON format. Using the GUI users can go through the registration process by selecting a datasource type, configuring its parameters and uploading a data file if needed.

Any data-set registered in the Datasources module will be globally available for reading using a single endpoint. The datasource types currently implemented are listed next:

- PostgreSQL
- PostGIS
- OpenStreetMap
- REST API
- CSV files
- JSON files

4.5.7. Sockets module

The WebSocket protocol is a widely supported open standard included in the HTML5 specification used for creating full-duplex connections. It is widely used in real-time web applications where data must be *pushed* from the server without first performing a client



request. Previous methods for simulating full-duplex connections were based on polling, a synchronous method where the client makes a request to the server to see if there is any information available. Polling is acceptable for cases with fixed time interval of message availability. However, it requires the client to open and close many unnecessary connections or holding requests open until there is some information to be sent.

The WebSocket module will benefit from the subscriptions defined by the FIWARE NGSI to visualize changes in the KPIs through the dashboard. In this way, the dashboard will be updated as soon as an element gets updated on the Context Broker.

4.5.8. Historical module

The already mentioned Sockets module provides a way of knowing the current state of a simulation the moment it takes place. This tool connects a NodeJS server to an Influx DB, a time-series platform designed to handle metrics and events. Time series are simply measurements or events that are tracked, monitored, down sampled, and aggregated over a period. The key difference with time series data from regular data is that queries will always ask questions about an element over time. With the arrival of the IoT paradigm, these databases are uniquely positioned to solve the challenges of millions of events coming in, filtering and analyzing. A time series consist of a set of values where time is a meaningful component of the data. Using a timestamp as a primary first-class attribute, Influx DB allows easily knowing and searching through the state of an element over the time. This scale has been one of the primary drivers behind the creation of specialized stores which need to face the challenges in:

4.5.9. Alerts module

Simple tool for storing emerged alerts and messages in their own database. It provides a mere endpoint where users can list received messages, discard and forward them to an email or API.

4.5.10. Complex Event Processing

A Complex Event Processing tool was designed to build data processes in a visual and intuitive way. This module can receive input event data in real-time, make a series of calculations and generate instantaneous data/alerts in reply to changing conditions. It comes in relation to reactive programming, a paradigm concerned with data flows and the propagation of change. While standard computation tends to employ synchronous, request-response interactions, reactive programming aims at creating observers which are subscribed to data changes. Whenever a change is emitted, the observer will perform calculations to process the change and obtain more meaningful information about a situation. This calculus can include analyzing the sequence of the value changed, aggregating all the values or comparing with some thresholds (e.g. for generating alerts).

The Complex Event Processing module allows creating asynchronous functions by chaining simple operations blocks. Built using NodeJS, the tool enables users to visually drag and drop blocks of code and construct what a function should compute and return. Each code block has several input and output parameters which need to be connected to their prior and post. When a function is invoked, it will go through all its blocks executing them in cascade



and passing the parameters from one to the next. These functions can be triggered in response to events, such as changes to data in the Historical module, or invoked by an HTTP.

4.6. Interaction of FIWARE components

4.6.1. Waste Management data ingestion

This part of the “Back-end architecture” puts emphasis on the waste management data coming from the IoT devices deployed on each Waste4Think project pilot.

Waste management data will be mapped to Context Broker events (“entities”) according to the Waste Management Data Model (Section 7) and will be published on the Publish/Subscribe Context Broker GE “ORION” through its NGSI REST APIs.

The IoT devices may send its own data to the back-end system in two different manner:

- via the Backend Device Management GE - IDAS [18] which is able to connect IoT devices/gateway to FIWARE platform. IDAS supports several IoT protocols with a modular architecture where modules are called “IoT Agents” that translate IoT-specific protocols (UL2.0 HTTP/MQTT, LWM2M/CoAP, JSON/MQTT, SIGFOX) into the NGSI context information protocol.
- via a NGSI custom connector which can convert IoT data in the NGSI context information.

Figure 13 shows a visual representation of the Waste Management Data Ingestion phase.

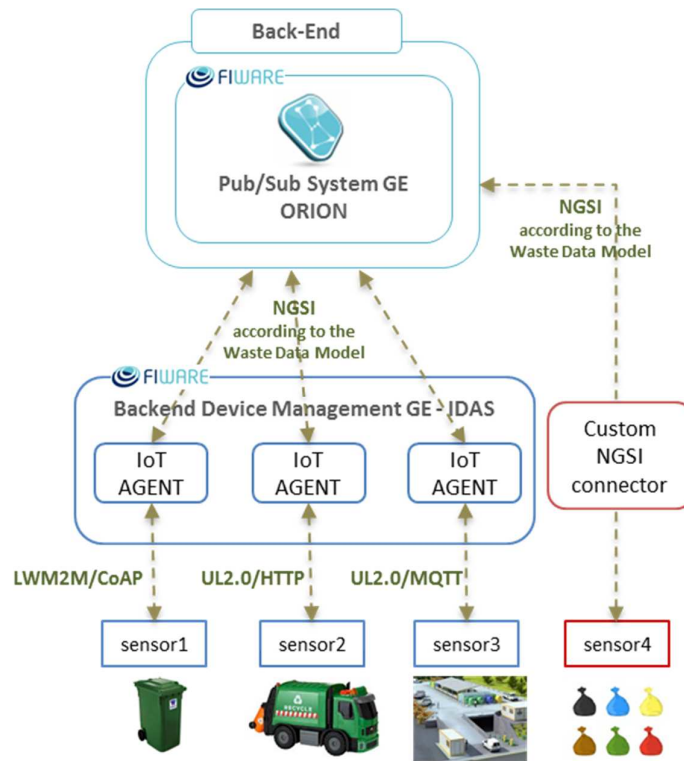


Figure 13. Waste Management data ingestion.



4.6.2. Waste Management Data storage

Waste management data mapped into the Context Broker GE ORION as context events (“entities”) are updated with a certain frequency thus representing data flowing into the back-end waste management from the pilots.

Anytime an entities attribute is updated, the Publish/Subscribe Context Broker (ORION) will notify the Connector Framework CYGNUS component; the latter acts as a sort of middleware between the Context Broker and the Storage Systems.

Figure 14 shows a visual representation of the Waste Management Data Storage phase.

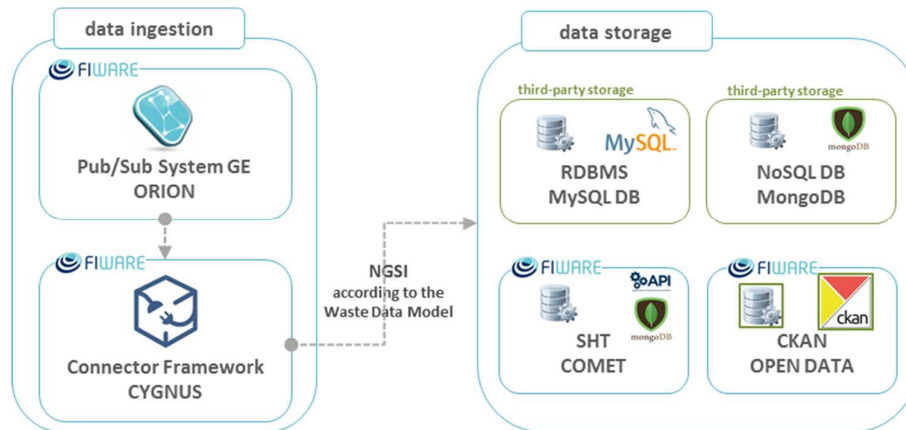


Figure 14. Waste Management Data storage.

4.6.3. Real-time processing of context events

The Context Broker GE connected to the Complex Event Processing GE “PROTON” can play two different roles: the role of Event Producer or the role of Event Consumer.

Event Producers are the source of events for event processing. Some examples of event producers are:

- external applications reporting events on user activities and on operation activities;
- sensors reporting on a new measurement.

Event Producers can provide events in two modes:

- "Push" mode: The Event Producers push events into the CEP by means of invoking a REST API.
- “Pull” mode: The Event Producer exports a REST API that the CEP can invoke to retrieve events.

Event Consumers are the destination point of events. Some examples of event consumers are:

- Dashboard: a type of event consumer that displays alarms defined when certain conditions hold on events related to some entities;



- Handling process: a type of event consumer that consumes meaningful events (such as opportunities or threats) and performs a concrete action;
- The Context Broker GE which can connect as an event consumer to the CEP and forward the events it consumes to all interested applications based on a subscription model.

The CEP sends output events to the event consumers in a “push” mode by activating their REST API.

Figure 15 shows a visual representation of the Waste Management Data Processing phase.

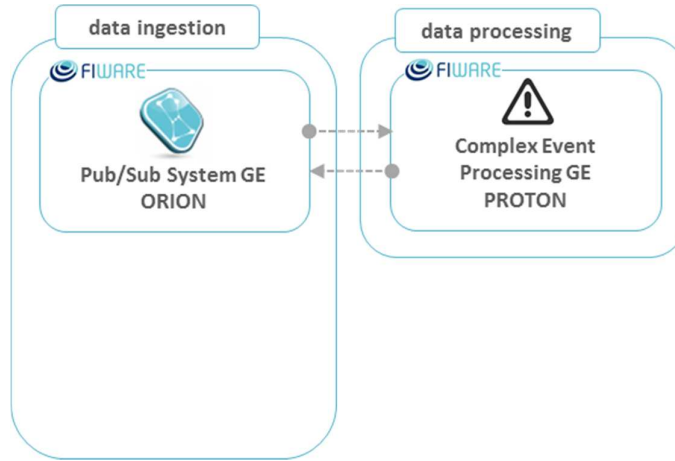


Figure 15. Real-time processing.

4.6.4. Backend security

FIWARE offers some services and tools to manage authentication and authorization in applications and backend services.

As shown in Figure 16, the Waste4Think backend provides a RESTful web service in order to expose API resources to different users. Tools such as Web Application, Rest Client, Mobile APP or other services can be used by users to communicate with the Waste4Think backend.

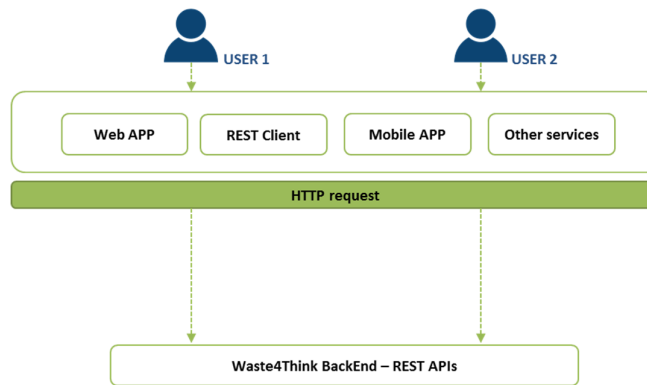


Figure 16. Waste4Think backend scenario.

In the Wast4Think context, the following FIWARE Generic Enablers (see Figure 17) will be used and combined to implement secured access to the Backend:

- The Keyrock Identity Management Generic Enabler brings support to secure and private OAuth2-based authentication of users and devices, user profile management, privacy-preserving disposition of personal data, Single Sign-On (SSO) and Identity Federation across multiple administration domains.
- The Wilma proxy Generic Enabler brings support of proxy functions within OAuth2-based authentication schemas. It also implements PEP functions within an XACML-based access control schema.
- The AuthZForce PDP/PAP Generic Enabler brings support to PDP/PAP functions within an access control schema based on the XACML standard.

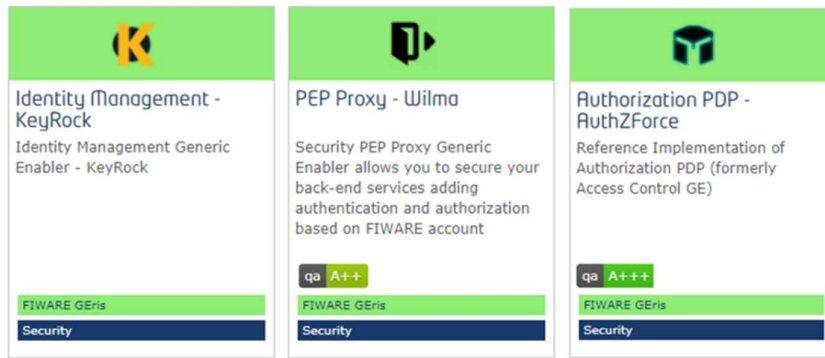


Figure 17. FIWARE Security GEs.

The following are the main steps to access a secured backend resources (Figure 18):

1. User retrieves the token from IDM Keyrock;
2. User runs an HTTP request adding the token retrieved;
3. WILMA Pep Proxy verifies the token with IDM KeyRock and AuthZForce to get the authentication/authorization user info;
4. If the user is authenticated and authorized, WILMA Pep Proxy gives the access to the Backend resource otherwise it denies the access.

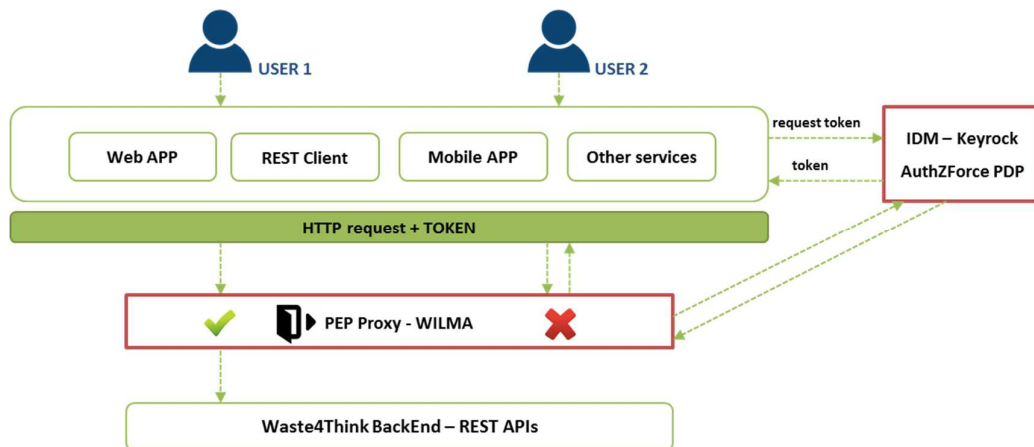


Figure 18. W4T backend security layer.



Security levels

The following security levels can be implemented to secure the Waste4Think backend:

- Level 1 “Authentication” (see Figure 19): to check if a user is a registered user in the IDM Keyrock;

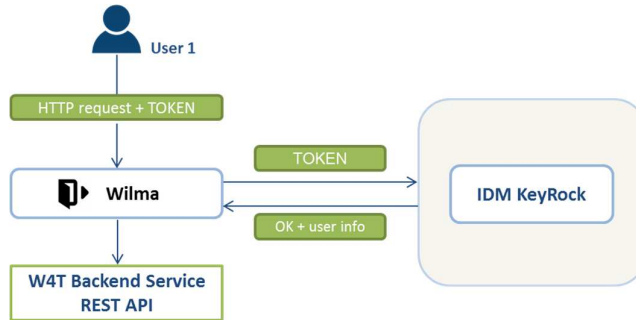


Figure 19. Security level 1 - Authentication

- Level 2 “Basic Authorization” (see Figure 20): to check if a user has permissions to access a resource (HTTP verb + resource path);

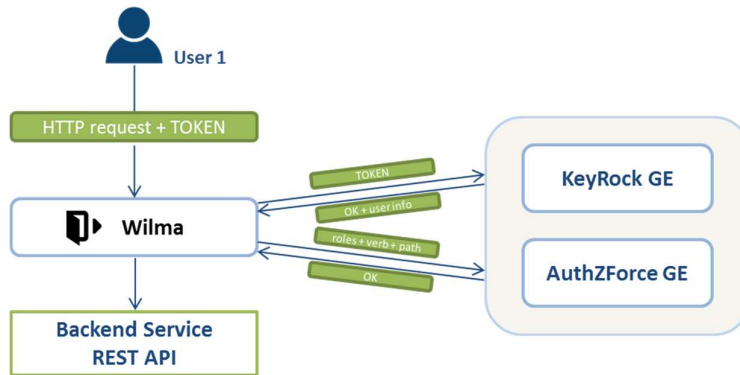


Figure 20. Security level 2 - Basic authorization

- Level 3 “Advanced Authorization” (see Figure 21): to check if a user has permissions to access a resource (XACML policies);

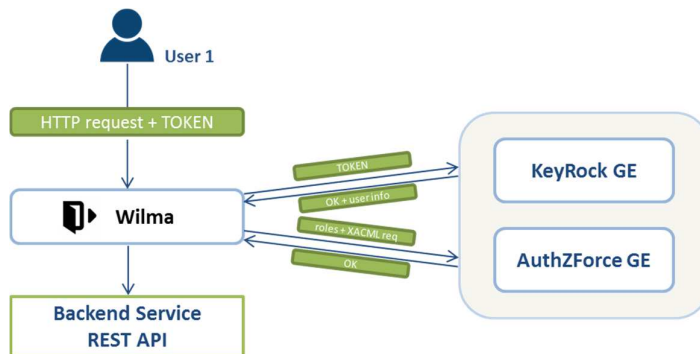


Figure 21. Security level 3 - Advanced authorization



4.7. Overview of the NGSI - REST API

This section defines the FIWARE-NGSI version 2 API. FIWARE-NGSI v2 is intended to manage the entire lifecycle of context information, including updates, queries, registrations, and subscriptions.

NGSI version 2 uses camel case (camelCase) syntax for naming properties and related artefacts used by the API.

The FIWARE NGSI (Next Generation Service Interface) API defines:

- A **data model** for context information, based on a simple information model using the notion of *context entities*;
- A **context data interface** for exchanging information by means of query, subscription, and update operations;
- A **context availability interface** for exchanging information on how to obtain context information.

4.7.1. Context data modelling and exchange

The main elements in the NGSI data model (Figure 22) are context entities, attributes and metadata. Figure 23 shows an example of a WasteContainer entity.

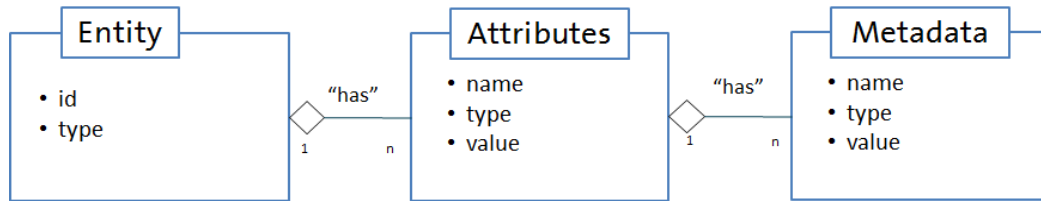


Figure 22. NGSI data model.

```

{
  id: "Wastecontainer:Example",
  type: "WasteContainer",
  category: ▶ {type: "StructuredValue", value: ["underground" ], metadata: {...}},
  dateLastEmptying: ▶ {type: "Text", value: "2016-06-21T15:05:59.408Z", metadata: {}},
  dateNextActuation: ▶ {type: "Text", value: "2016-06-28", metadata: {}},
  fillingLevel: ▶ {type: "StructuredValue", value: {value: 26.5, type: "Float",...},
  location: ▶ {type: "StructuredValue", value: {type: "Point", coordinates: [-3.16448559
  refDevice: ▶ {type: "StructuredValue", value: ["device-Fleming:12a:1" ], metadata: {...}}
  refWasteContainerIsle: ▶ {type: "Text", value: "wastecontainerisle1", metadata: {}},
  refWasteContainerModel: ▶ {type: "Text", value: "wastecontainermodell1", metadata: {}},
  serialNumber: ▶ {type: "Text", value: "ab56kj1", metadata: {}},
  status: ▶ {type: "Text", value: "ok", metadata: {}}
}
    
```

Figure 23. Example of a WasteContainer entity.



4.7.2. Context Entities

Context entities, or simply entities, are the core in the FIWARE NGSI information model. An entity represents a thing, i.e., any physical or logical object (e.g., a sensor, a person, a waste container, an issue in a ticketing system, etc.). Each entity has an **entity id**.

Furthermore, the type system of FIWARE NGSI enables entities to have an **entity type**. Entity types are semantic types; they are intended to describe the type of thing represented by the entity. For example, a context entity with id *WasteContainer:Zamudio0001* could have the type *WasteContainer*. Each entity is uniquely identified by the combination of its id and type.

4.7.3. Context Attributes

Context attributes are properties of context entities. For example, the accessibility of a waste container could be modelled as attribute *accessibility* of entity *WasteContainer:Zamudio0001*.

In the NGSI data model, attributes have an *attribute name*, an *attribute type*, an *attribute value* and *metadata*.

- The *attribute name* describes what kind of property the attribute value represents of the entity, for example *accessibility*.
- The *attribute type* represents the NGSI value type of the attribute value. Note that FIWARE NGSI has its own type system for attribute values, so NGSI value types are not the same as JSON types.
- The *attribute value* finally contains:
 - the actual data
 - optional *metadata* describing properties of the attribute value like e.g. accuracy, provider, or a timestamp

4.7.4. Context Metadata

Context metadata is used in FIWARE NGSI in several places, one of them being an optional part of the attribute value as described above. Similar to the attributes, each piece of metadata has:

- a *metadata name*, describing the role of the metadata in the place where it occurs;
- a *metadata type*, describing the NGSI value type of the metadata value;
- a *metadata value* containing the actual metadata.

The users can attach their own metadata to entity attributes. These metadata elements are processed by Orion in a transparent way: Orion simply stores them in the database at update time and retrieves them at query and notification time.

Custom metadata accepts any name except for a few reserved names, used for special metadata that are interpreted by Orion (e.g. ID or location).

Figure 24 shows an entity WasteContainer with the attribute "fillingLevel" associate to the "accuracy" metadata.

```

fillingLevel: {
  type: "StructuredValue",
  value: {
    value: 26.5,
    type: "Float",
    metadata: {
      accuracy: {
        value: 0.2,
        type: "Float"
      }
    }
  }
},

```

Figure 24. Example of a Custom Attribute Metadata.

4.7.5. API

The FIWARE-NGSI v2 API URI of the Context Broker GE installed on the Waste4Think Back-end platform are reachable at the following address:

<http://130.206.117.164/v2>

A full explanation of the main NGSI APIs (e.g. List of entities, create an entity, create a subscription, etc..) is given in ANNEX B: NGSI API.

4.8. FIWARE components considered but not used in the project

4.8.1. STH – COMET

The Short Time Historic (STH) is a component of the FIWARE ecosystem in charge of managing (storing and retrieving) historical raw and aggregated time series information about the evolution in time of context data (i.e., entity attribute values) registered in an Orion Context Broker instance [14]. All the communications between the STH and the Orion Context Broker as well as between the STH and any third party (typically for data retrieval) use standardized NGSI9 and NGSI10 interfaces.

The DoA explicitly states the use of a STH engine within the Waste4Think project. The History technology developed by the FIWARE initiative (STH-Comet) was tested, however it did not fulfil the technical requirements. One of the main problems was that the services provided were built upon v1 of the NGSI API, instead of v2 which is used by the Orion Context Broker. Also, considering the type of information that is going to be stored (waste

transactions), there is a need for a relational database such as PostgreSQL with extension for time-series, constraint checking and trigger customization (instead of a non-relational one like Mongo DB, on which STH Comet is built upon). These problems are being overcome by the development of a custom historical module, named as History.

4.8.2. Complex Event Processing (CEP) GE – PROTON

The CEP GE analyses event data in real-time, generates immediate insight and enables instant response to changing conditions. While standard reactive applications are based on reactions to single events, the CEP GE reacts to situations rather than to single events. A situation is a condition that is based on a series of events that have occurred within a dynamic time window called processing context. Situations include composite events (e.g., sequence), counting operators on events (e.g., aggregation) and absence operators. The Proactive Technology Online is an implementation of the FIWARE CEP (Complex Event Processing) GE. The Proactive Technology Online is a scalable integrated platform to support the development, deployment, and maintenance of event-driven applications. The Proactive Technology Online authoring tool allows the definition of CEP applications using a web user interface. The Proactive Technology Online engine is a runtime tool that receives information on the occurrence of events from event producers, detects situations, and reports the detected situations to external consumers [9].

The DoA explicitly states the use of a CEP engine within the Waste4Think project. All CEP technologies developed by the FIWARE initiative (Proton, Perseo and Cosmos) have been tested, however none of them fulfil the technical requirements. The main problem is the impossibility to easily keep the state between invocations (needed to perform forecasting, online calculation of KPI and tariff) and to make invocations “at will” (to consolidate the tariff and KPIs). These problems are being overcome by the development of a custom CEP, which manual can be found in Deliverable 2.4.

5. Design of the frontend

The frontend will be designed using the Model View Controller design pattern (MVC). This design pattern uses a structure that separates the data, the domain/application/business logic and the user interface in three isolated layers: model (data), controller (logic), and view (user interface).

The **model** manages the data sources of the application. It can respond to requests for information, requests to change information, and events to notify observers about changes in the information. The model in the frontend will gather information from both the Context Broker and the persistence layer. This model does not contain any information about the user interface.

The **controller** is used to communicate between classes in the model and the view. It will be implemented in separate Django apps, sharing a common model and common utilities, but working independently. This architecture is flexible enough to adapt to the specific requirements of the pilot sites, considering not all the modules are going to be deployed in each case.

The **views** will comprise a set of HTML5 interfaces, enriched with bootstrap and jQuery, representing the elements in the user interface: all the things the user can see and respond to on the screen, such as buttons, display boxes, etc.

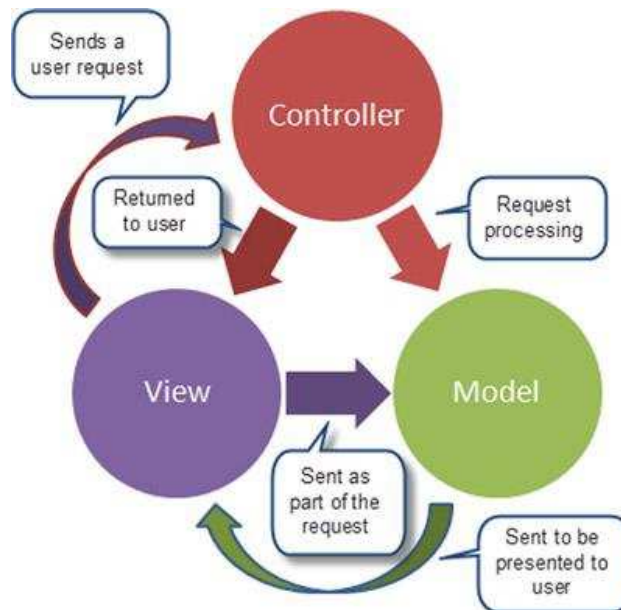


Figure 25. Conceptual diagram of the MVC pattern.

The following sections will describe the integration of the tools of R1 on a common interface within the Waste4Think final solution. Aside from the main purpose, each tool will feature a set of mock-ups to better describe their functionalities.



5.1. Integration of the different tools

As mentioned in Section 2, the W4T-Core will allow to access several applications, each one with its own objectives, and oriented towards a specific range of users and situations. To maintain uniformity across all the visual components of the platform, and avoid ending up with a set of individual applications, the next decisions were taken in terms of user interfacing:

- **Unique visual style:** Define a common style so that all the visual elements in the applications feature a uniform appearance. This will be achieved by using the same front-end component library (Bootstrap).
- **Common entry point:** All the applications will be available on a unique entry point (a web domain), will use a unique user identification system, and will be accessed through a side menu.
- **Common structure:** All the applications will use a similar visual composition with common elements (logos, icons, badges, etc.) and common solutions for similar needs.

5.1.1. W4T-Core menu

The W4T-Core Menu will be accessed just by clicking on the Waste4Think logo. This menu will be an overlay that covers a third of the tools from the right and darkens the rest of it. This menu includes the **user information**, the **logout button** and a **list of the applications available to the user**. This menu can be accessed from anywhere in the application.

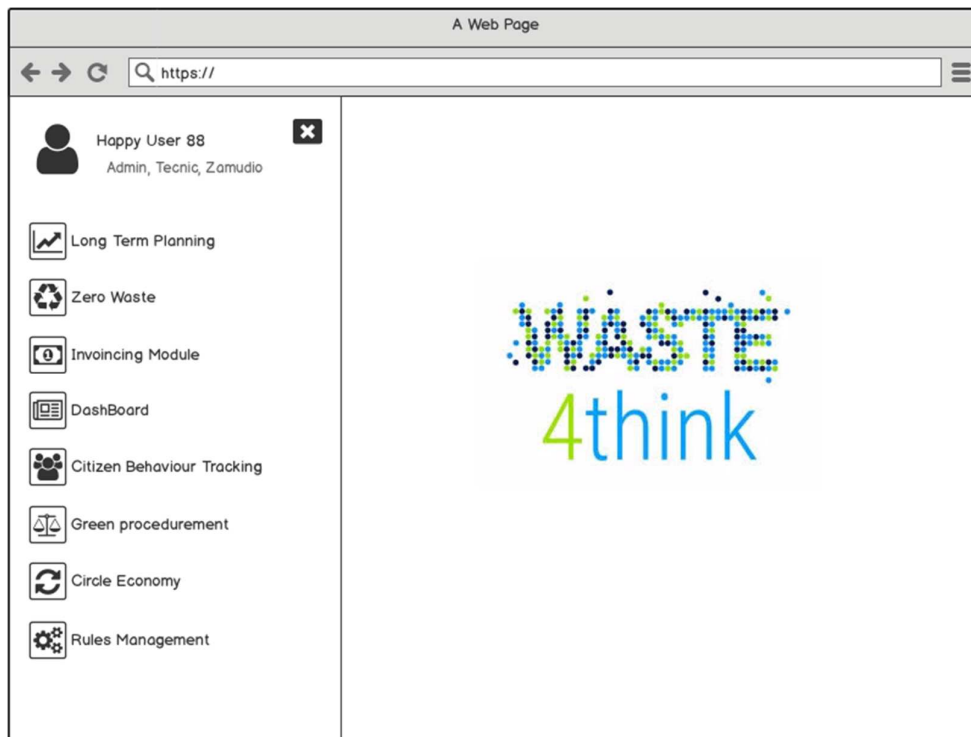


Figure 26. W4T-Core



5.2. Dashboard

The Dashboard is the visualization tool designed to provide a configurable viewport of the data collected, transformed, or produced by the whole Waste4Think ecosystem. Specifically, a dashboard comprises a set of widgets that graphically represent the information relevant to the user (Figure 27). This tool will be accessible through the side menu of the W4T-Core (Figure 26).

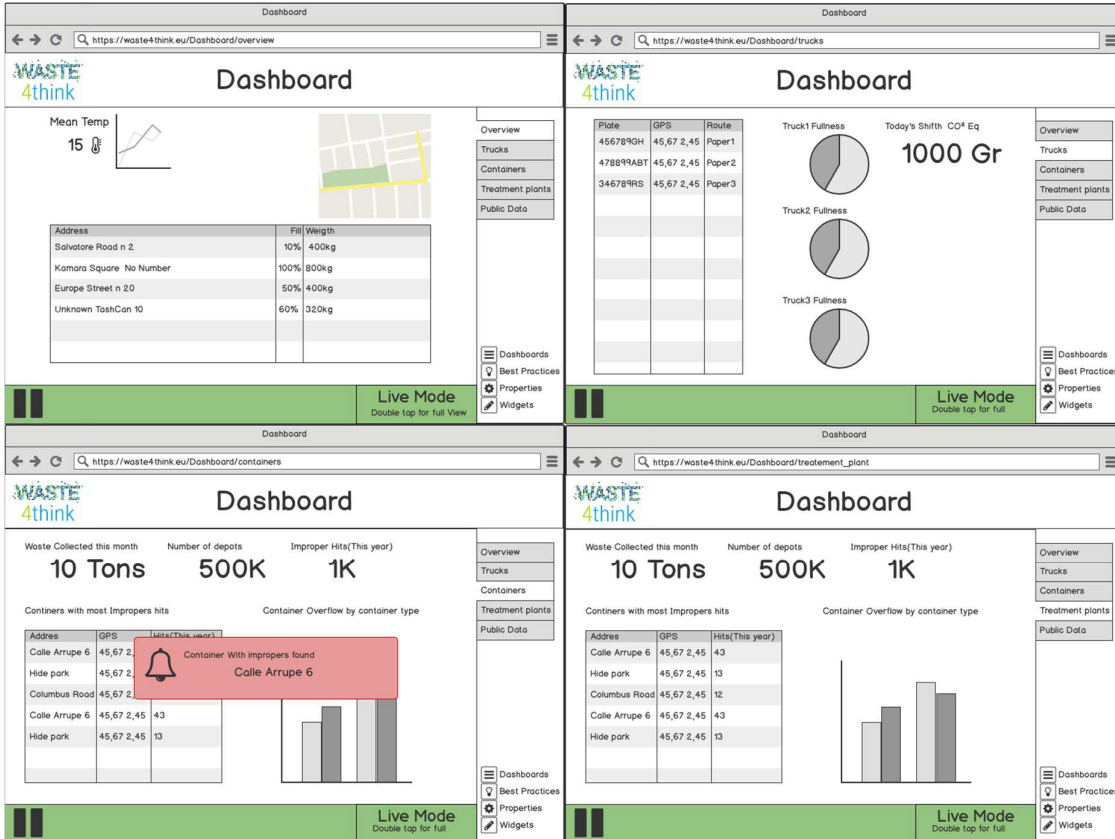


Figure 27. Examples of configurable dashboards.

5.2.1. Dashboard Manager

Each user will have access to their own dashboards, the dashboards of their own group (administrators, technicians, operators), and the public dashboards of their organization (see Section 5.2.4).

When the user accesses the Dashboard, the tool will present the list of all the dashboards created by them. The user will be able to switch from one dashboard to the other by clicking on the corresponding filter control, for example, a tabbed menu on the right side of the view (Figure 28).

The *Dashboard button* (Figure 29) opens the *Dashboard Manager* (Figure 30). This view will show a **list of all the dashboards created by the user**. At first glance, the user will be able to identify the most important aspects of each dashboard, such as the name, the owner, the visibility (which users and groups can see the dashboards, etc), and whether it should

appear on the Dashboard main view. From this view, the user may also **create** a new dashboard, and **edit** or **delete** an existing dashboard.

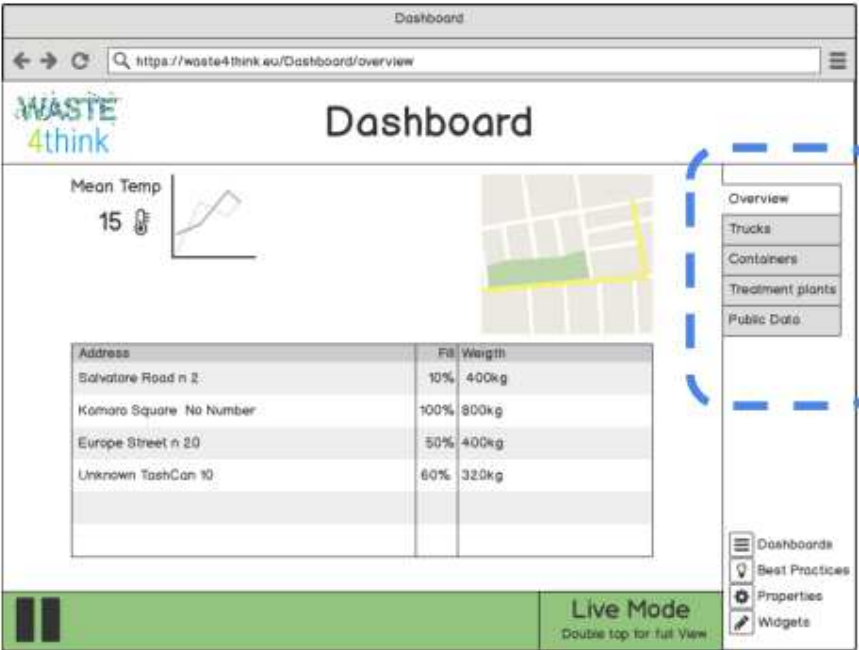


Figure 28. Tab menu to change between dashboards.

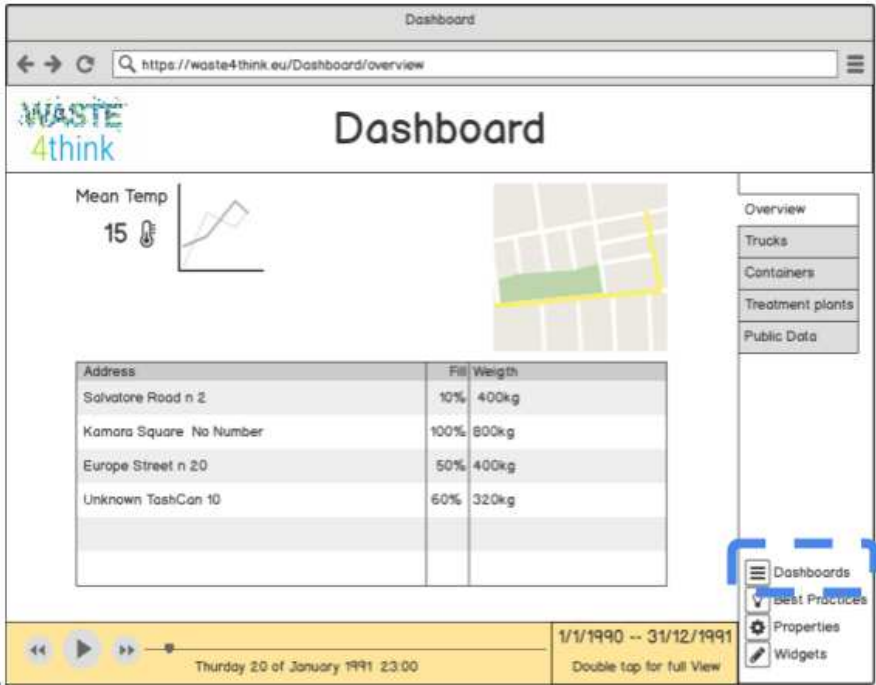


Figure 29. Dashboard button to access the Dashboard Manager.

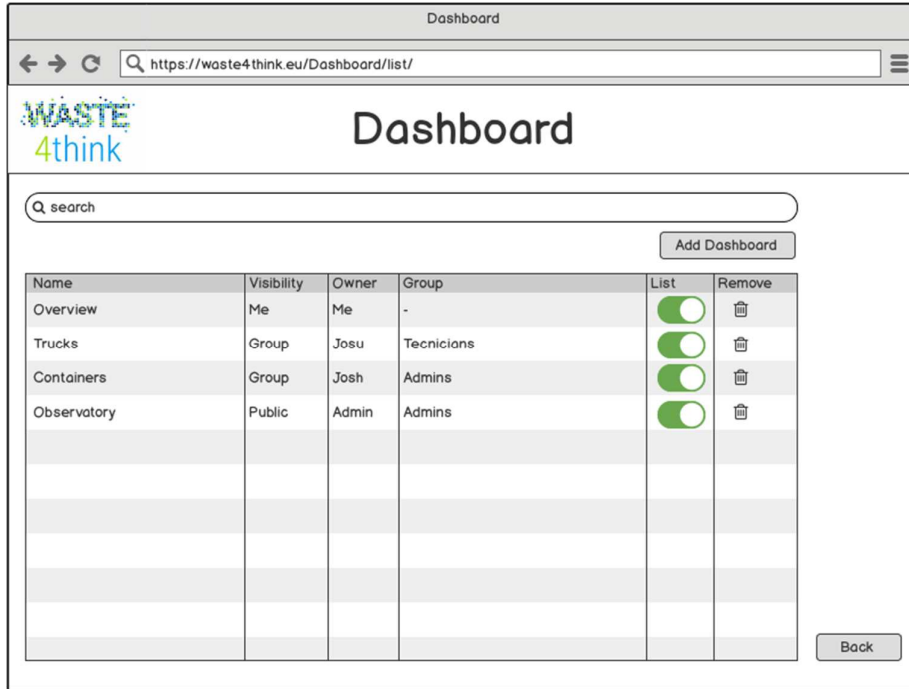


Figure 30. Dashboard Manager.

5.2.2. Dashboards properties

Each dashboard has their own set of properties that can be configured to modify its behaviour. The specific properties of a dashboard can be accessed by clicking on the *Properties button*, which will open an additional menu on the right side of the view (Figure 32 and Figure 33).

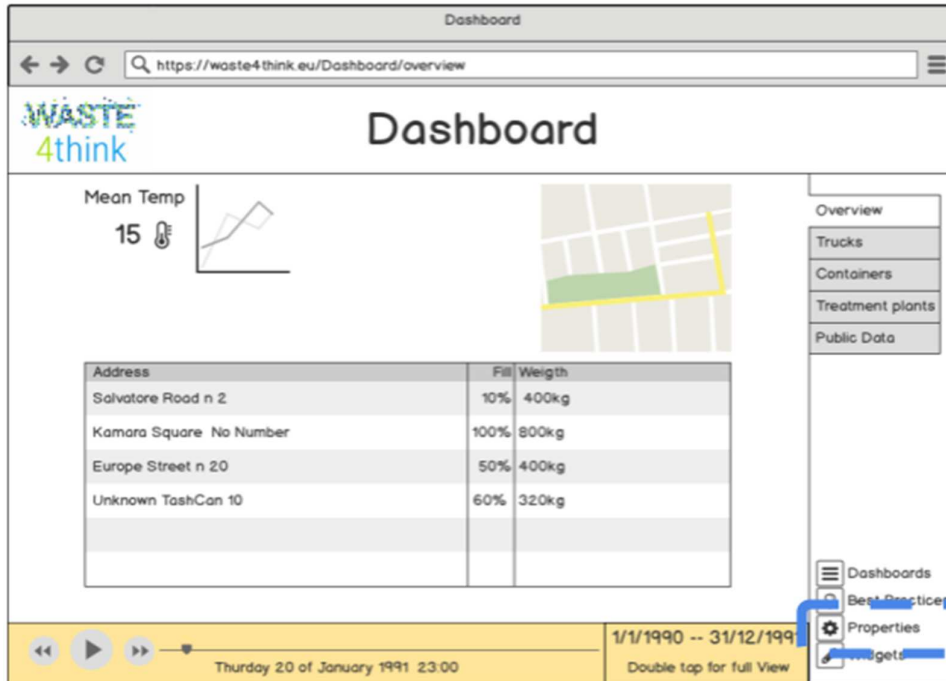


Figure 31. Properties button within the Dashboard tool.

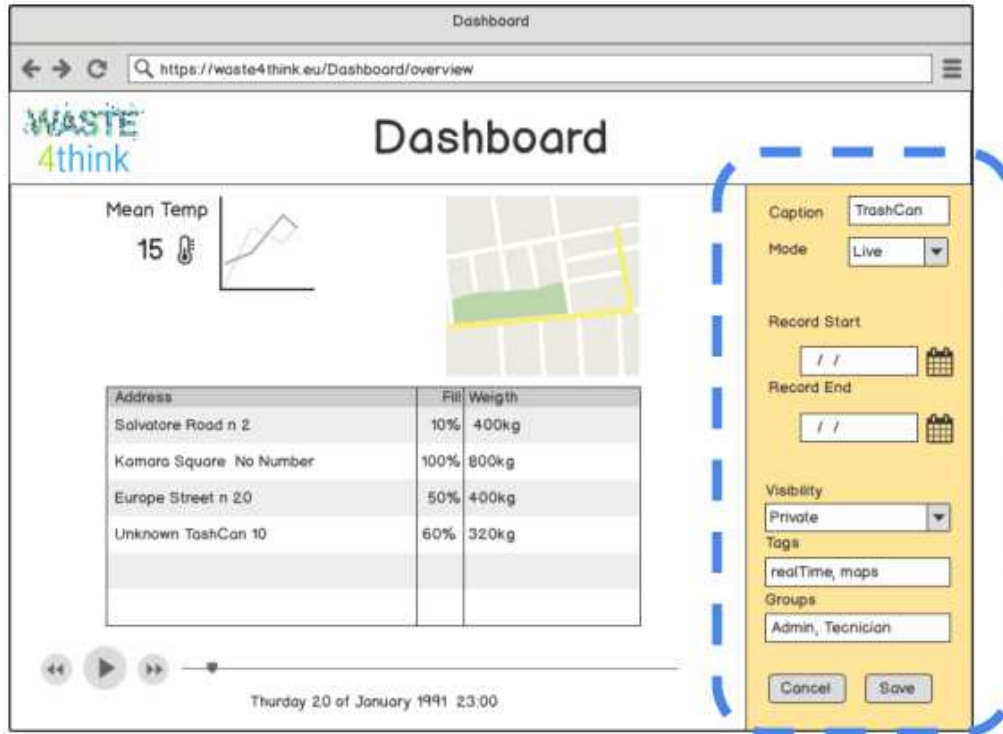


Figure 32. Dashboard properties menu.

5.2.3. Dashboards and widgets


A dashboard combines information from the different entities in the Waste4Think ecosystem and shows them on graphic elements known as **widgets**. Widgets are an essential aspect of dashboard customization. They are "at-a-glance" views of the most important data and functionality that is accessible right from the main view of the Dashboard tool. Users will be able to move widgets across their dashboards, resize them, or modify the information shown.

A preliminary list of widgets that will be available in the Dashboard tool is detailed in Table 3.

Table 3. Widgets available in the Dashboard tool.

Widget	Description	Example
Map	A map that shows Markers in the element position. Multiple elements can be passed to this widgets.	
Graph	A graph (bars, pie chart, lines, histogram) from element attributes.	



Data Grid	A data table that shows the elements, one or more, as files and attributes as columns (Or viceversa)	
Labeled	A box of text showing information on written form.	Weight: 23 Tn

A user can add widgets to a dashboard through specific widget toolbars. These toolbars are accessible by clicking on the *Widgets button* in the side menu of the dashboard (Figure 33. Widgets button). This will open a new toolbar on the bottom side of the view, where the user can access all the types of widgets available (Figure 34). To add a widget, the user will just have to **drag** the widget from the toolbar and **drop** it onto the dashboard.

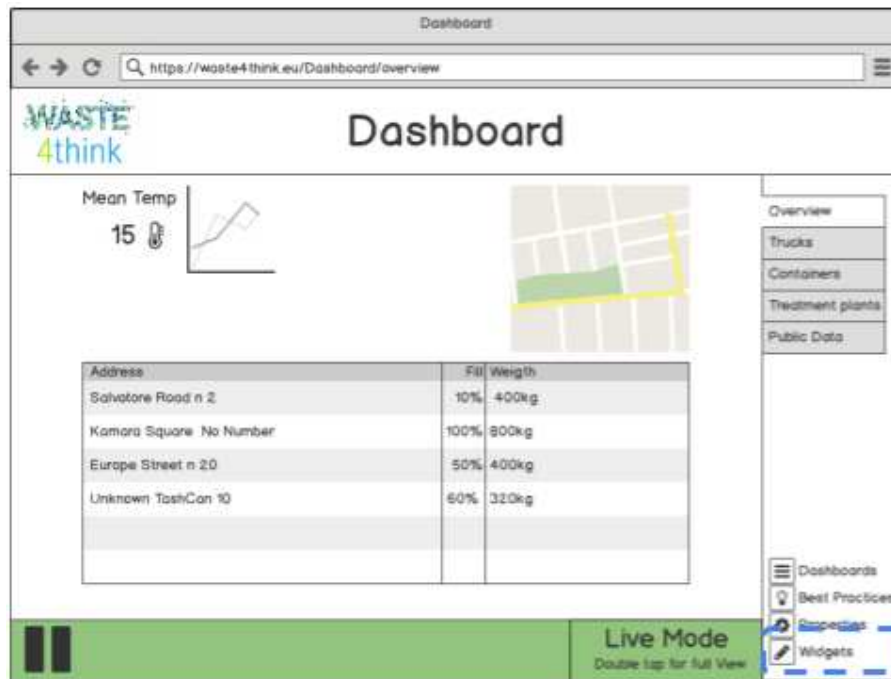


Figure 33. Widgets button.

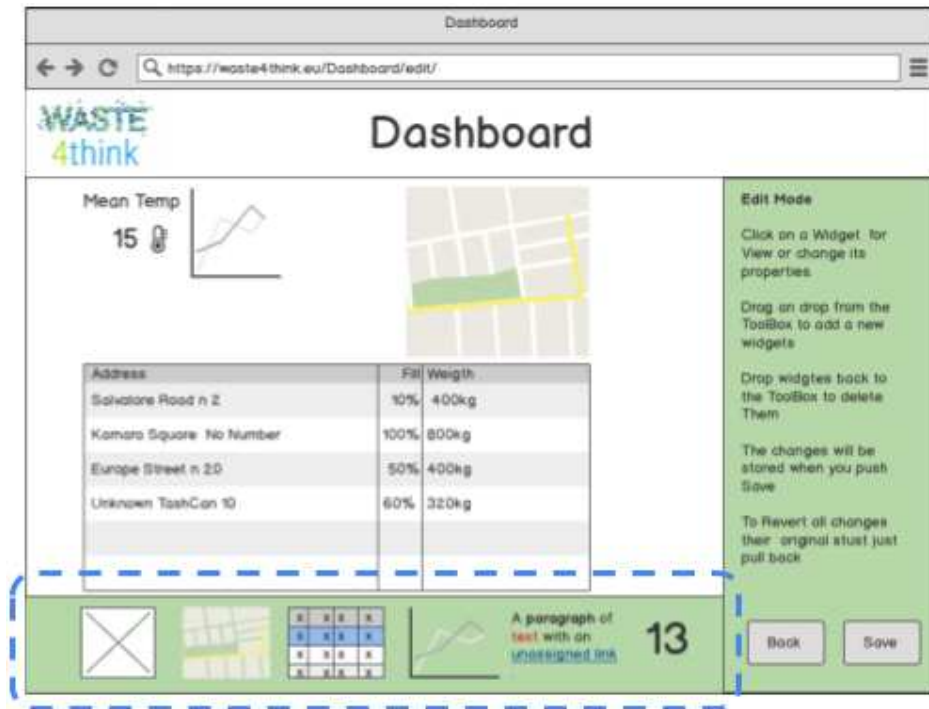


Figure 34. Widgets toolbar.

When a widget is selected, its properties are shown on a side menu. These properties are classified by nature: behaviour, appearance, entities used as data source, and entity attributes that will be shown in the widget (Figure 35 and Figure 36).

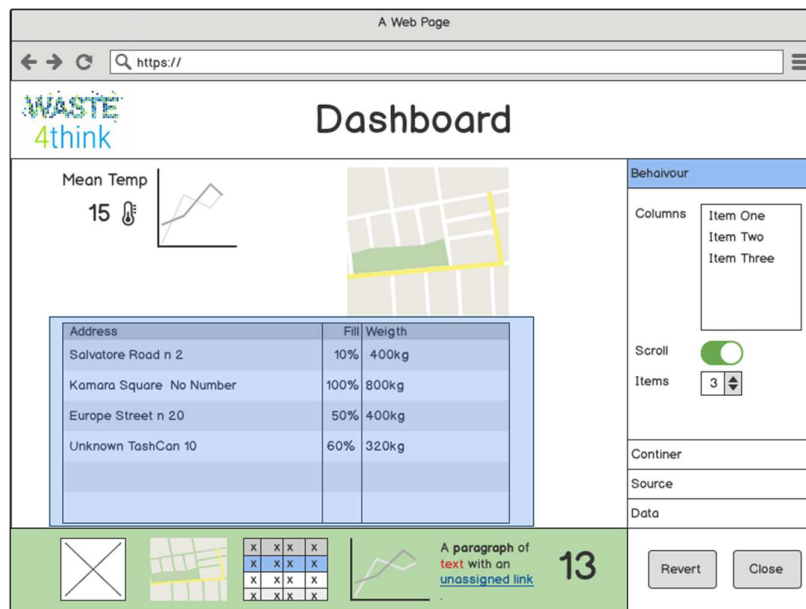


Figure 35. Behaviour properties of the widget.

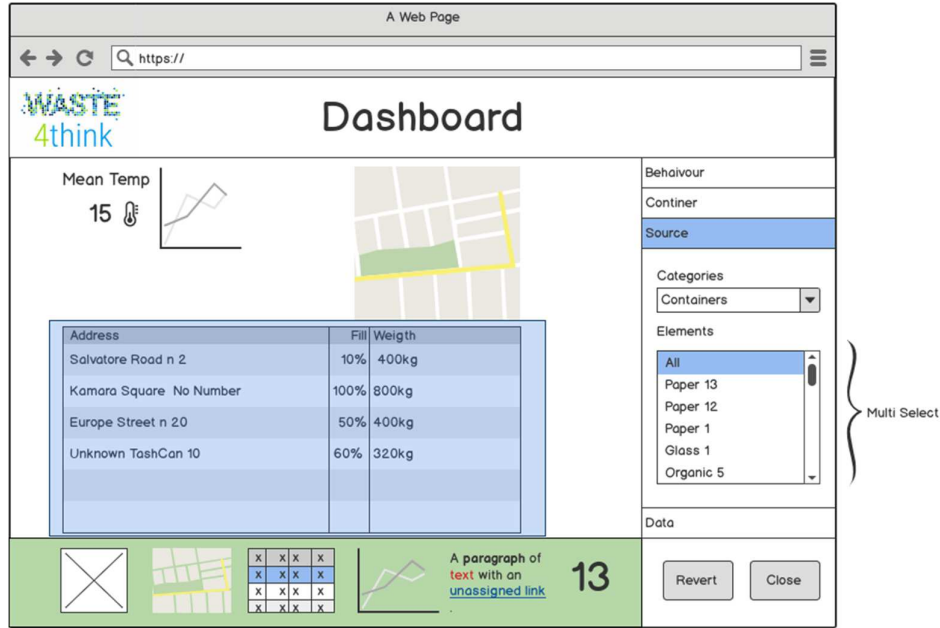


Figure 36. Data source properties of the widget.

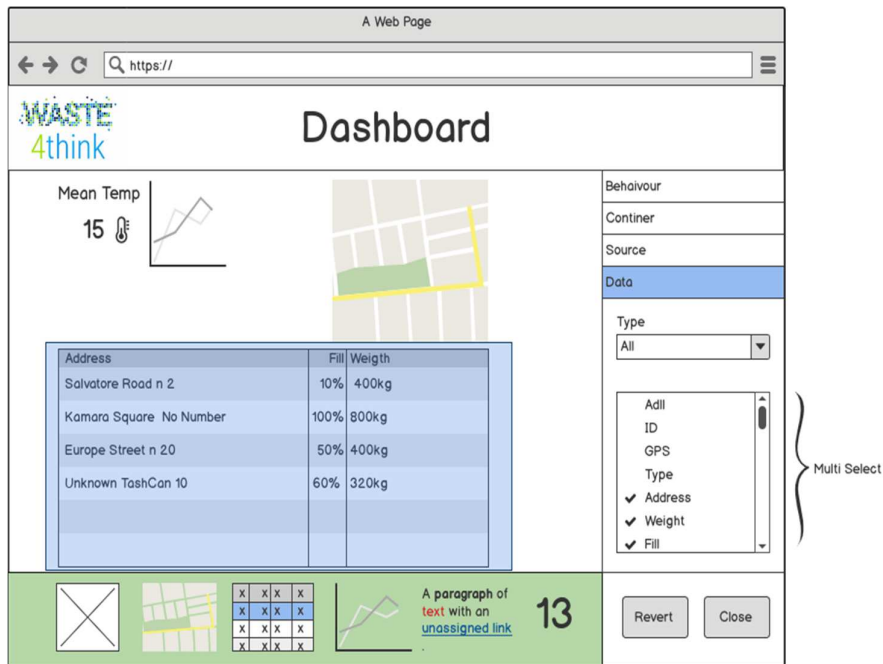


Figure 37. Data attribute properties of the widget.

5.2.4. Types of dashboards

The Dashboard tool provides different types of configurable dashboards, each one differentiated by the visual design, the controls, and their purpose. The types of dashboards available are as follows:



- a) **Live** (Figure 38). This dashboard shows the status of **the data in real-time**. The visual control provides the option to *freeze the visualization* (stop the real-time update of values) *or go back to live mode*. The **data is fed directly from de Context Broker** (see Section 4.3.1) and is not connected to any persistence layer.
- b) **Historic** (Figure 39). This dashboard shows **a historic view of the data**. The user can choose any time slice and review any moment of it. For example, replay a slice of time, or view the data in fast forward. If it is necessary, the widgets can be edited to analyze news aspect of the reviewed information. This **data is fed directly from de persistence layer**.
- c) **Scenario** (Figure 40). This dashboard shows a view of **data from a simulation scenario**. The controls are similar to those of the Historic mode, but it *provides visual indications that the data is simulated*. The data **can either be fed from the persistence layer and the simulation engine or directly from the simulation engine**.
- d) **Event** (Figure 41). This Dashboard is designed for the special monitorization of Zero-Waste Events and Zero Waste Ecosystems (see Deliverable 1.3 and 2.7). During the event period the data works like a LIVE dashboard, at the end of the period it works like a Historic dashboard. Additionally, an event dashboard has a publication period on which the dashboard becomes public, returning to its default state at the end of this period. The data is **fed from Context Broker** or from the persistence layer, depending the status of the event.

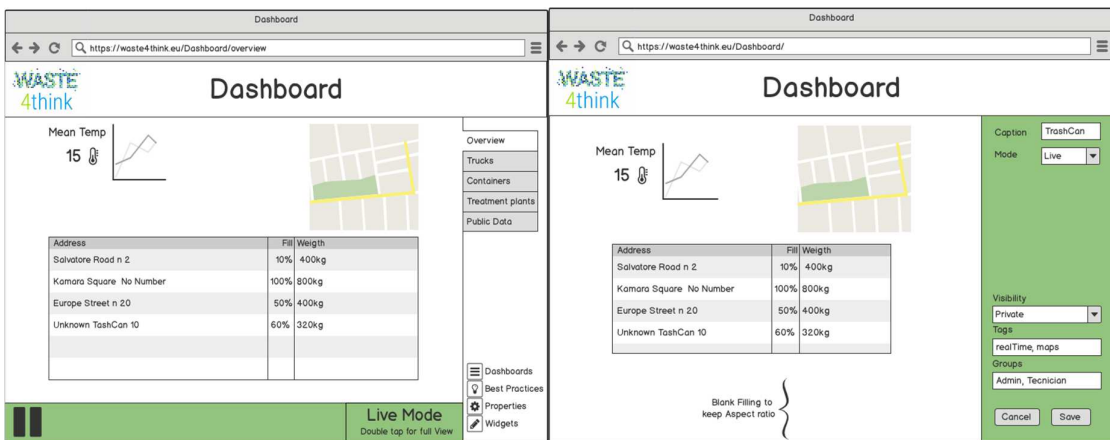


Figure 38. Live dashboard.

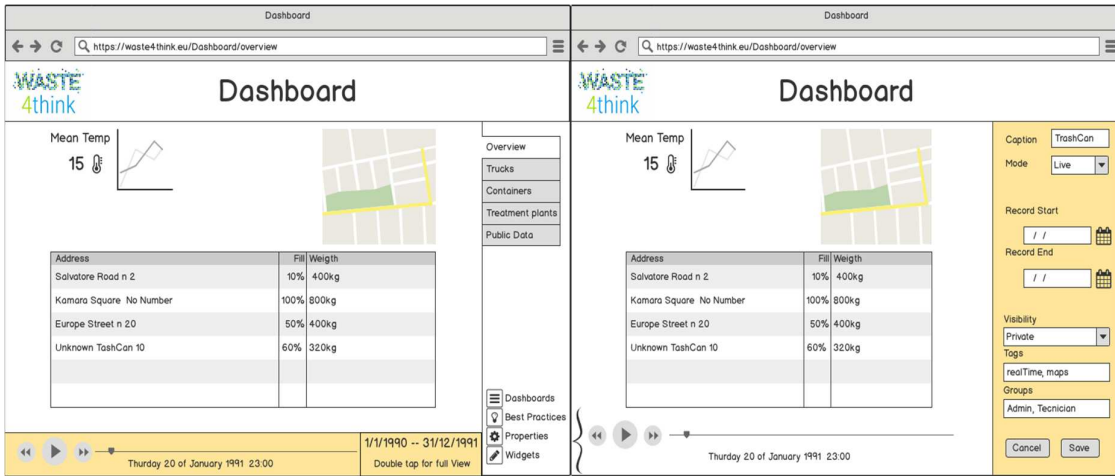


Figure 39. Historical dashboard.

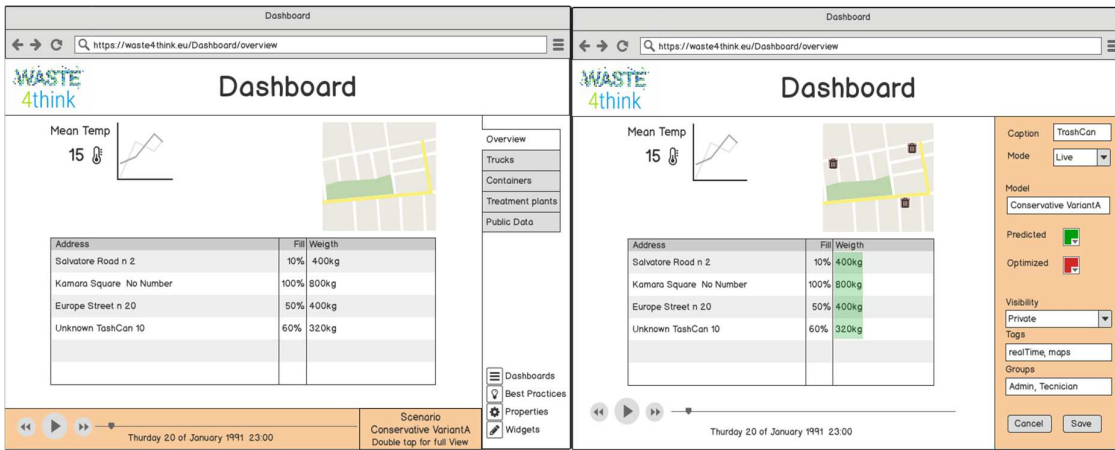


Figure 40. Scenario dashboard.

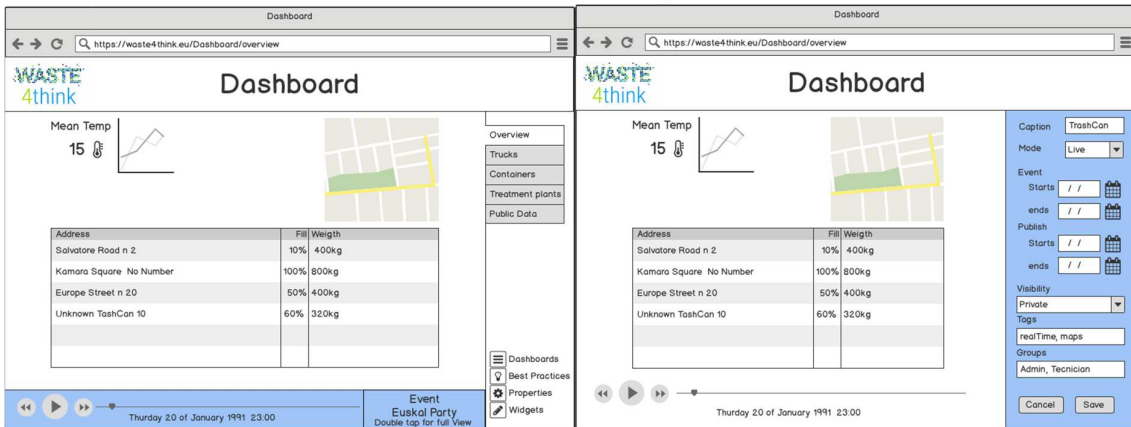


Figure 41. Event dashboard.



5.3. Rules Manager

The system uses user defined rules for create alerts, transforms data or provides complex indicators. The Rule Creator application provides an interface for CRUD operations over these rules. The tool will be accessible through the side menu of the W4T-core (Figure 26).

The Main View provides a list of the system rules (Figure 42). Each rule is show in a panel that shows the rule name the description and an icon that represent the rule type. This view also provides search of filtering of the rules showed and pagination for the lists panels.

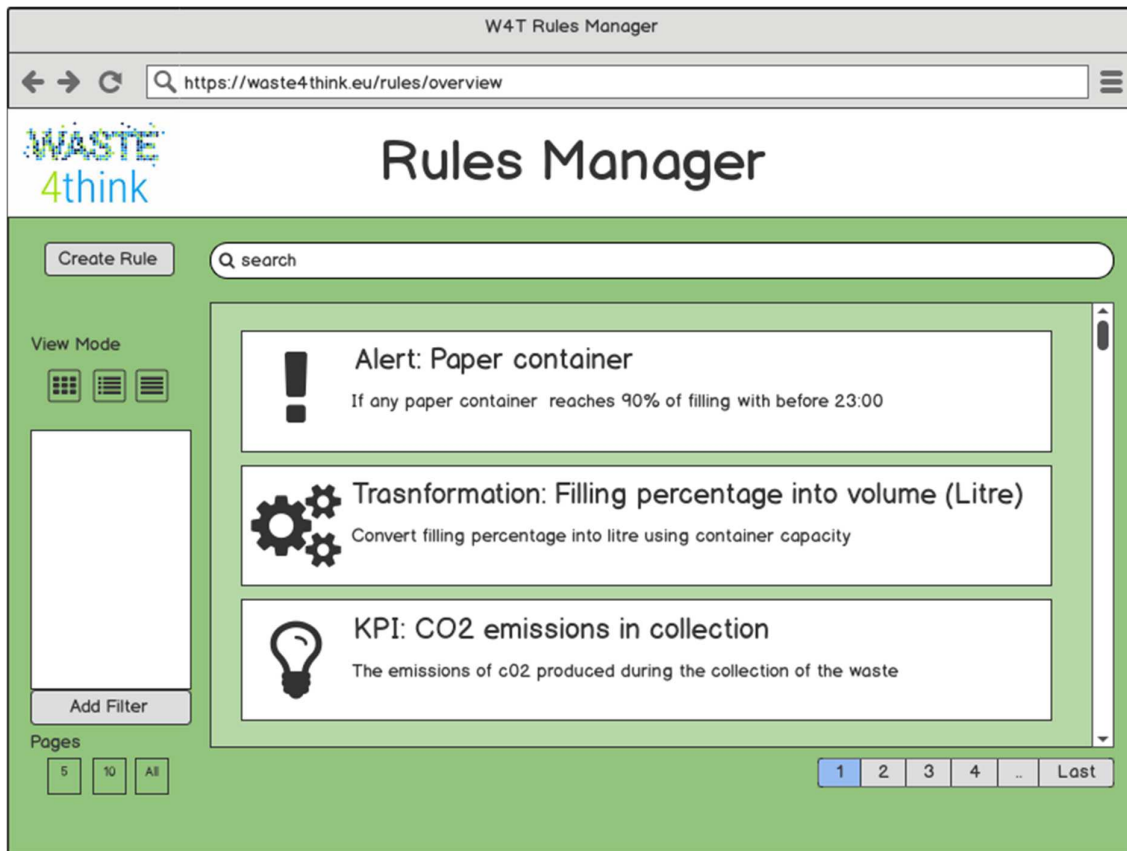


Figure 42. Rule Manager view

5.3.1. Creation of a new Rule

To create a new rule, drag and drop the elements from the toolbar (variable, operation, constant or link) to the work zone. Clicking in the new element (or other existing element) opens a specific dialog to **define the properties of the element**. Each element uses a different dialog:

1. **Operation View.** (Figure 44). This dialog shows a selector to determine the operation to perform as well as a filter and a search box to ease the search of the desired operation.
2. **Constant View.** (Figure 45). This dialog shows the value of the constant and a selector to determine the constant type. The values of the input changes (text, number) when the type is selected.



3. **Link View** (Figure 46). This dialog shows the selector of where (constant, variable, operation) a value is captured (source) and where it is deposit (target for the element and operand if the destiny has more than one input).
4. **Variable View** (Figure 47). This dialog shows a selector to determine the element to select, that admits multi-selection and a filter to ease the selection of the element. The dialog shows the attributes of the selected element that are going to be given as variable.

To **delete an element** from the rule you must drag and drop the element back into the element toolbox.

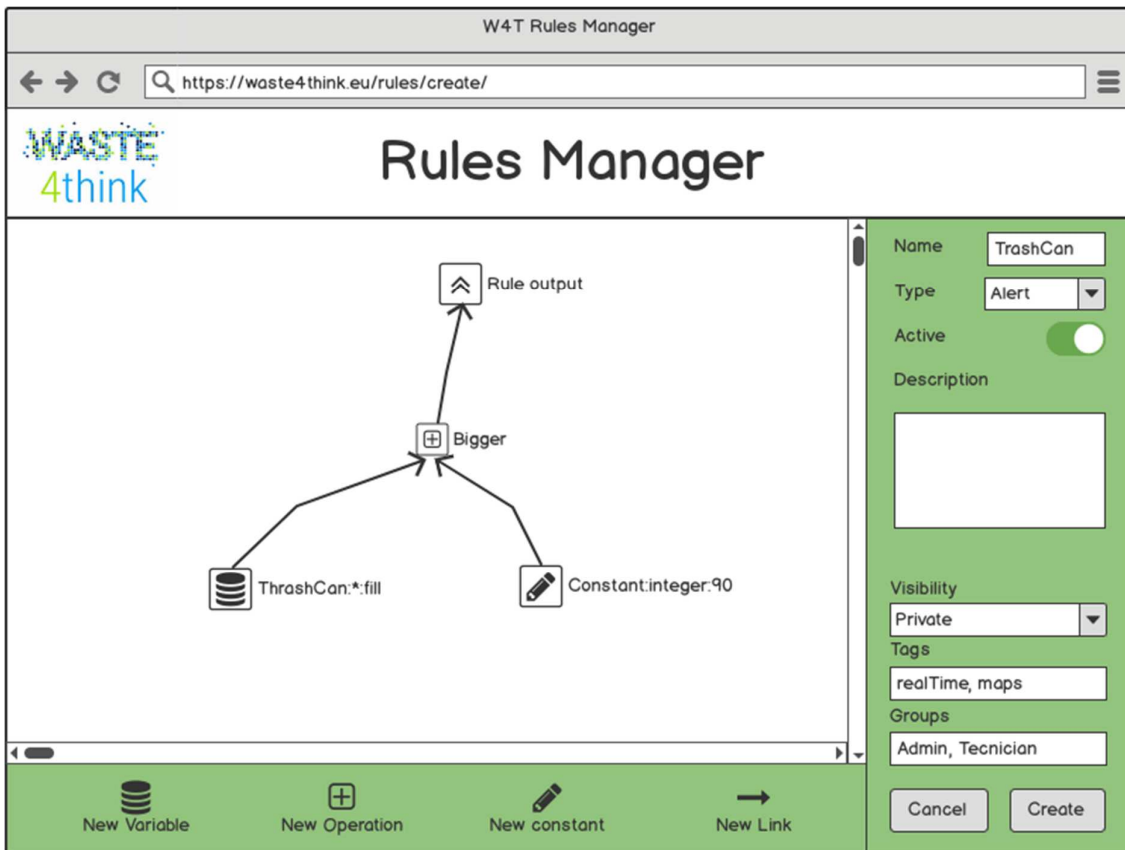


Figure 43. Example of a rule

Push a new element button (variable, operation, constant or link) opens a specific dialog to **add a new element**. Each element uses a different dialog. Double click in a deployed element show the **edit element** Dialog, that are almost the same as the create element dialog but already filled.

For **deleting an element** of the rule, you must drag and drop the element back into the element toolbox. These are the dialog used in this view:

Add Operation

Search

Type	Elements
All	Sum
Arithmetics	Max
Logical	Equal
Agregation	Predict Alpha Risk(Model)
Models	
Models(Conservative)	
Models(Agresive)	
Models(5 years)	

Cancel Add

Figure 44. View to add operation to a rule.

Add Constant

Integer String Map

90

Value input element change when type is selected

Cancel Add

Figure 45. View to add a constant to a rule.

Add Link

Source Select

Target Select

Operand Select

Cancel Add

Figure 46. View to link operations and operands.



Figure 47. Add variable view.

5.4. Public Observatory

The purpose of the Public Observatory is to provide a **public dashboard** configured to monitor, for example, the evolution of the KPIs on the pilot sites. The user will be able to compare the monitored information against a configurable baseline (previous month, previous day, previous year, etc). Any person will be able to view the Public Observatory by accessing the specific URL, without the need to provide any type of credentials. However, a Public Observatory may only be modified by the user with admin privileges to the W4T-Core.

The process of creating the dashboard for the Public Observatory follows the same steps as that of user dashboards explained in Section 5.1.

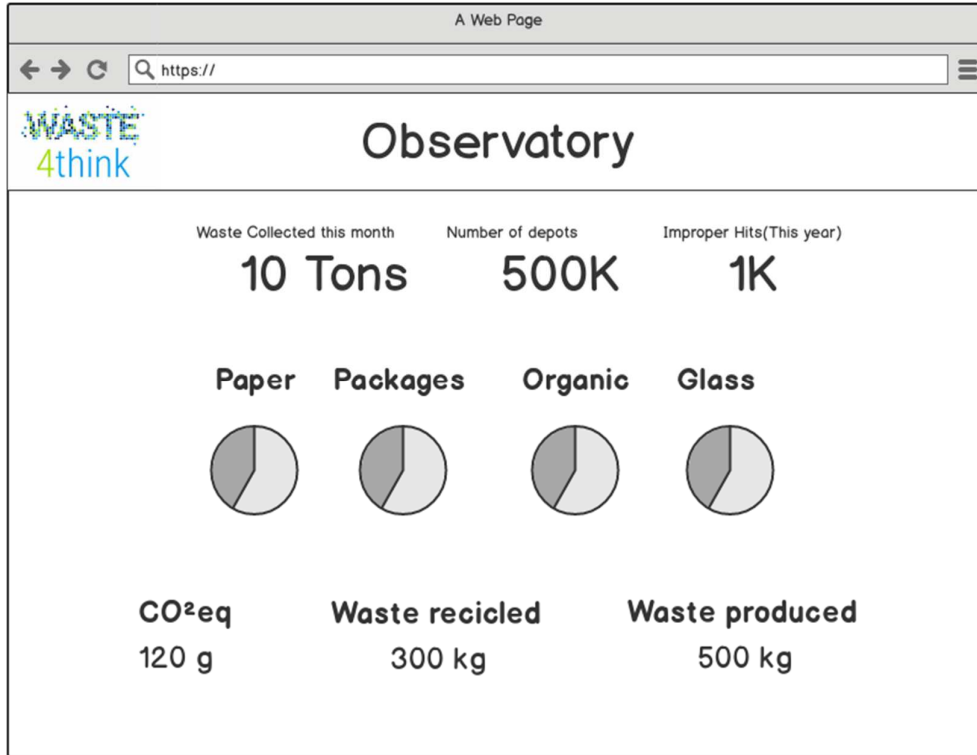


Figure 48. Main view of the Public Observatory

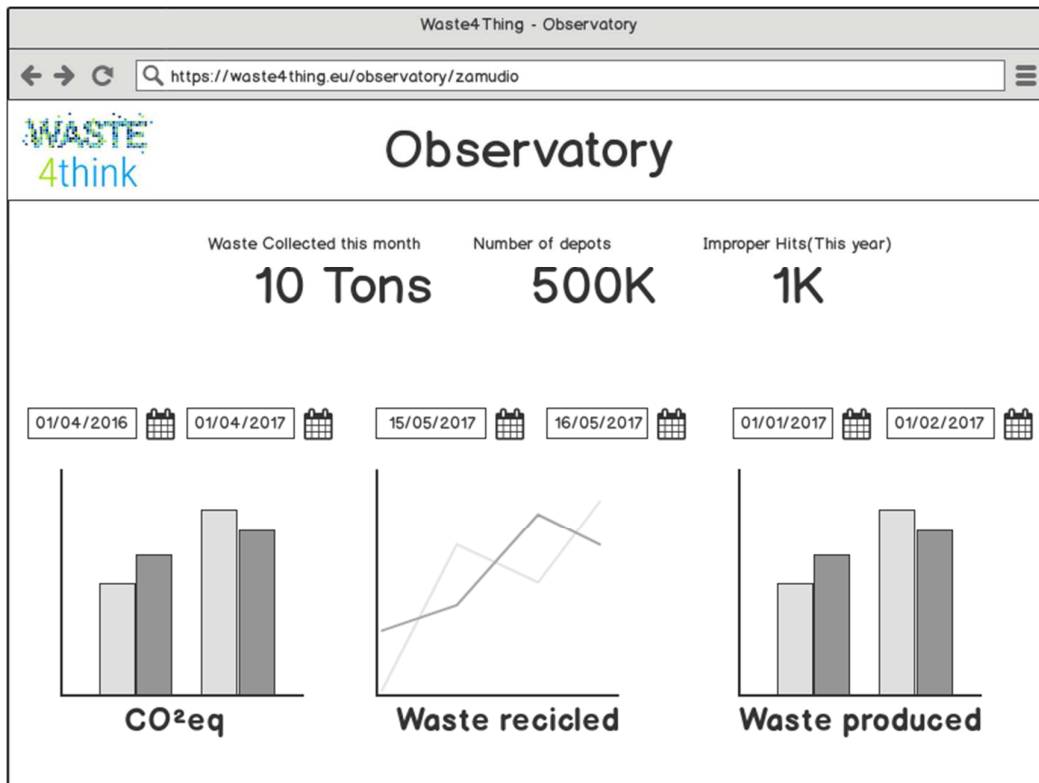


Figure 49. Baseline comparison of the Public Observatory

5.5. Citizen Behaviour Tracking

The purpose of the Citizen Behaviour Tracking tool is to provide a visual representation of the effects of certain social campaigns, in particular, and the Waste4Think project, in general. These effects will be measured by analyzing the evolution of the related KPIs over time, such as number of tweets of a certain topic, number of waste that was correctly sorted after an awareness campaign, number of users of the apps, etc.

The tool will be accessible through the side menu of the W4T-Core (Figure 26). On access, the user will be presented with a view showing the set of active dashboards, each one related to the measurement of specific KPIs under a common context (Figure 50). The user will be able to switch from one active dashboard to the other by clicking on the corresponding filter control, for example, a tabbed menu on the right side of the view.

From this view, the user will also be able to access the two main functionalities of the tool:

- Reports from Social Tracking.** Visualize the dashboards that display the evolution of KPIs for a specific campaign.
- New Social Tracking.** Create a new dashboard by defining the KPIs which evolution is going to be monitored.

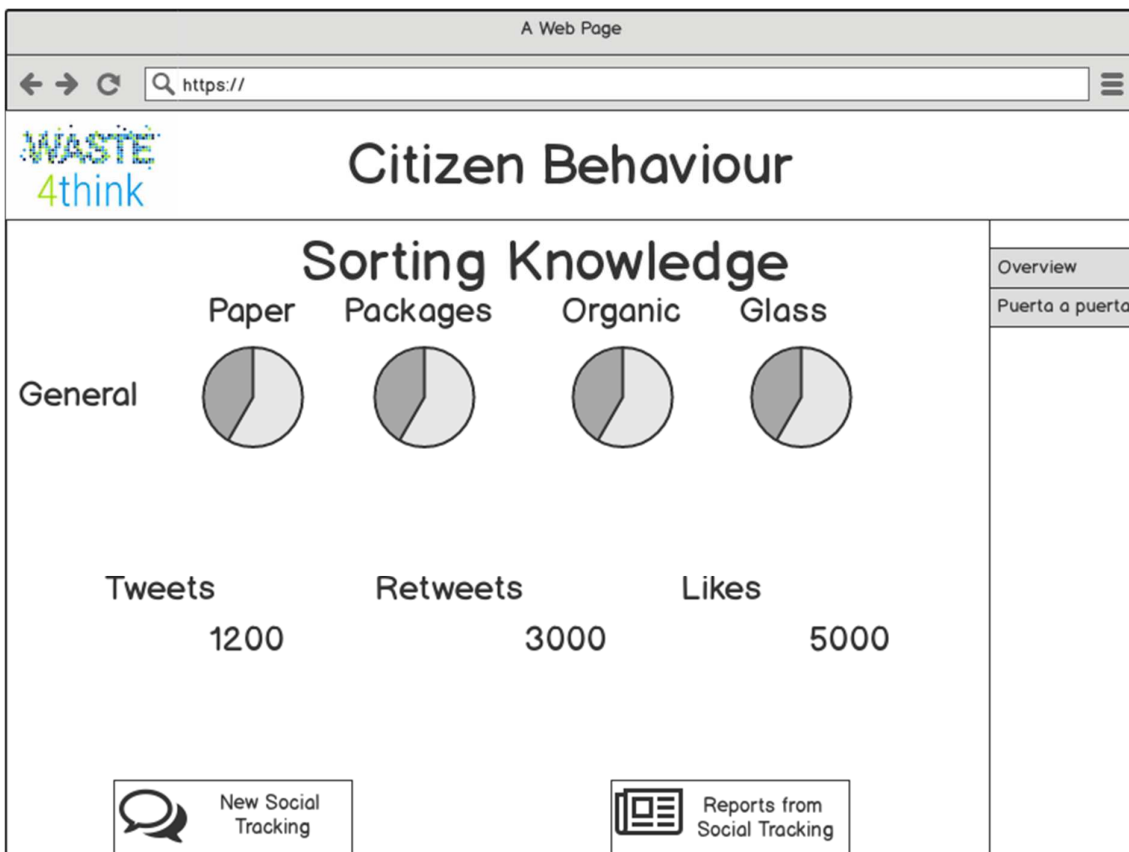


Figure 50. Main view of the Citizen Behaviour Tracking



5.5.1. Reports from Social Tracking

When the user selects the option **Reports from Social Tracking**, the tool will provide a list of all the reports created by them. At first glance, the user will be able to identify the most important aspects of each report, such as the name, the start date, the status, the visibility, etc. Specifically, the status of a tender refers to the three possible states of the report lifecycle, as follows:

- New.** The report has been created, and it may or may not be completely configured. The period of monitorization has not yet started.
- Running.** The period of time for the monitorization has already started. The related dashboard will show the evolution of the KPIs from that moment until the user decides to end the monitorization.
- Inactive.** The period of time for the monitorization has ended. The user will be able to access the historic mode of the dashboard (Section 5.2.4) to analyse the evolution of the KPI's over time.

The visibility of the dashboard lets the user filter the reports that will appear on the main view of the Citizen Behaviour Tracking tool (Figure 50):

- Visible.** When a dashboard is set to visible, it will appear on the main view of the Citizen Behaviour Tracking tool. The user will be able to select it from the filter control.
- Invisible.** When a dashboard is set to invisible, it will not appear on the main view of the Citizen Behaviour Tracking. The dashboard is not deleted, it is just hidden from view.

The screenshot shows a web browser window titled "Citizen Behaviour Tracking" with the URL "https://waste4think.eu/citizen/manage/". The page header includes the "WASTE 4think" logo and the title "Citizen Behaviour". Below the header is a search bar labeled "Select a campaign to view" with a "Q search" input field. The main content is a table with the following data:

Name	Start date	Status	Active	View	Remove
Campaign 'Puerta a puerta'	2018-01-01	New	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Campaign 'Navidades'	2018-01-02	Running	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Campaign 'Vacaciones'	2018-02-11	Inactive	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 51. List of available Citizen Behaviour Tracking reports.

5.5.2. New Social Tracking

When the user selects the option **New Social Tracking**, the tool will provide a view to define a new report (Figure 52). To correctly configure the report, the user will have to provide the following inputs.

- Name.** Introduce an identifiable title that properly recognizes the report among others.
- Social sources.** Configure the parameters for the web crawler by defining the indicators, the filters, and where to look for the information (Facebook, Twitter, forums, etc.).
- KPIs.** Add or remove the performance indicators that will be monitored in the report.

Once the user has configured all the needed parameters, the Citizen Behaviour Tracking Tool will build the dashboard with all the necessary widgets to monitor the evolution of the KPIs for a certain campaign.

The screenshot shows the 'Citizen Behaviour Tracking' web application. The browser address bar displays 'https://waste4think.eu/citizen/create/'. The page title is 'Citizen Behaviour'. The main heading is 'Create a new campaign tracker'. Below this, there is a 'Name' input field. The 'Social Sources' section is divided into three columns: 'Inputs' with an 'add' button and a list containing 'Social Net'; 'Indicators' with an 'add' button and a list containing 'Republish', 'Liked', and 'Comments'; and 'Filters' with an 'add' button and a list containing 'Not Contains bad words' and 'Contains '#ReciclaRegalo''. The 'KPIs' section features a table with columns for Name, Date, section, order, and remove. The table contains four rows of KPIs. A 'Create' button is located at the bottom right of the form.

Name	Date	section	order	remove
Kpi_Sorting_paper_correct	NOW	General	1	
Kpi_Sorting_paper_incorrect	NOW	General	2	
Kpi_Sorting_package_correct	NOW	General	3	
Kpi_Sorting_package_incorrect	NOW	General	4	

Figure 52. Creation of a new Citizen Behaviour Tracking report.

This manager provides a classic CRUD (Create, Read, Update and Delete) of the persistence layer. This interface is provided for maintenance and debugging purposes for all the entities in the database. The interface views are classic listing of elements with integrated search and filters; and formularies for visualizing or updating the entities.

5.7. Open Data Platform

An open data platform will be provided for publishing, sharing, finding, using and visualizing datasets of interest for the Waste4Think project such as socio-economic data, geographic information or sensors data.

The Waste4Think Open Data Platform is based by a customized CKAN instance (Section 4.5.1) deployed on the Waste4Think project of the FIWARE Lab Cloud Infrastructure (Section 3).

The Waste4Think Open Data Platform will offer two different types of data:

- context information coming from Waste4Think Pilots;
- pre-existing datasets.

Each type of open data will be published in a different way (see figure1):

- Waste4Think CKAN front-end: to upload pre-existing datasets;
- Waste4Think CKAN API: to upload pre-existing datasets;
- Cygnus connectors: to upload context-information coming from Waste4Think Pilots through the Context Broker.

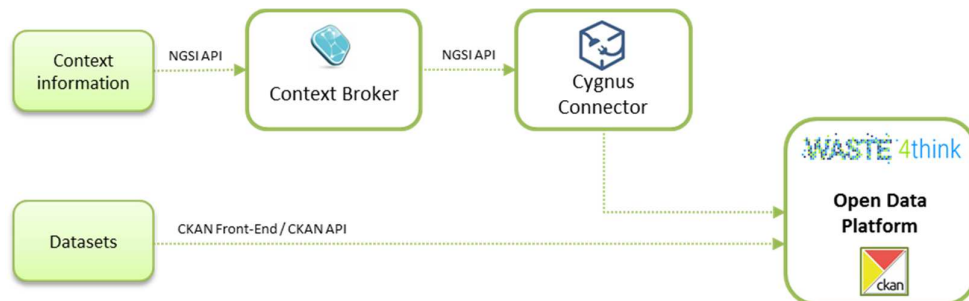


Figure 53. Types of Open Data

5.7.1. Integrating Pilot Data with the Waste4Think Open Data Platform

Information coming from the Waste4Think Pilots are mapped into the Context Broker GE ORION (Section 4.3.1) as context events (“entities”) and updated with a certain frequency.

NGSI subscriptions are created in the ORION Context Broker to one or more attributes of an entity type to notify the Connector Framework CYGNUS (Section 4.4.1) anytime these attributes will be updated.

The Connector CYGNUS is configured to accept NGSI context information coming from ORION context Broker and to write it to the Waste4Think Open data Platform.

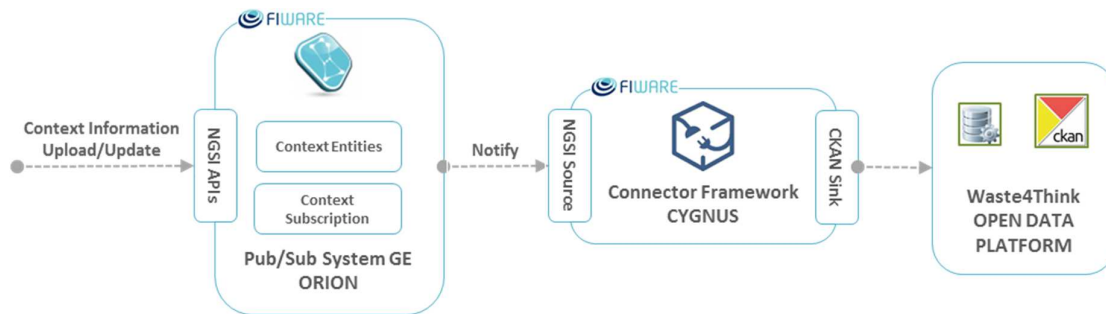


Figure 54. Connection with the Open Data Platform.

5.7.2. Waste4Think Datasets

Waste4Think organization has been created on the CKAN platform to gather datasets provided by the Waste4Think Projects. Each dataset will be sorted in categories and tagged.

At present the Open data platform provides the following open data context information coming from Zamudio Pilot:

- Deposit Point: Information about the waste collection points available in Zamudio.
- Deposit Point Type: Information about the waste collection point types available in Zamudio.

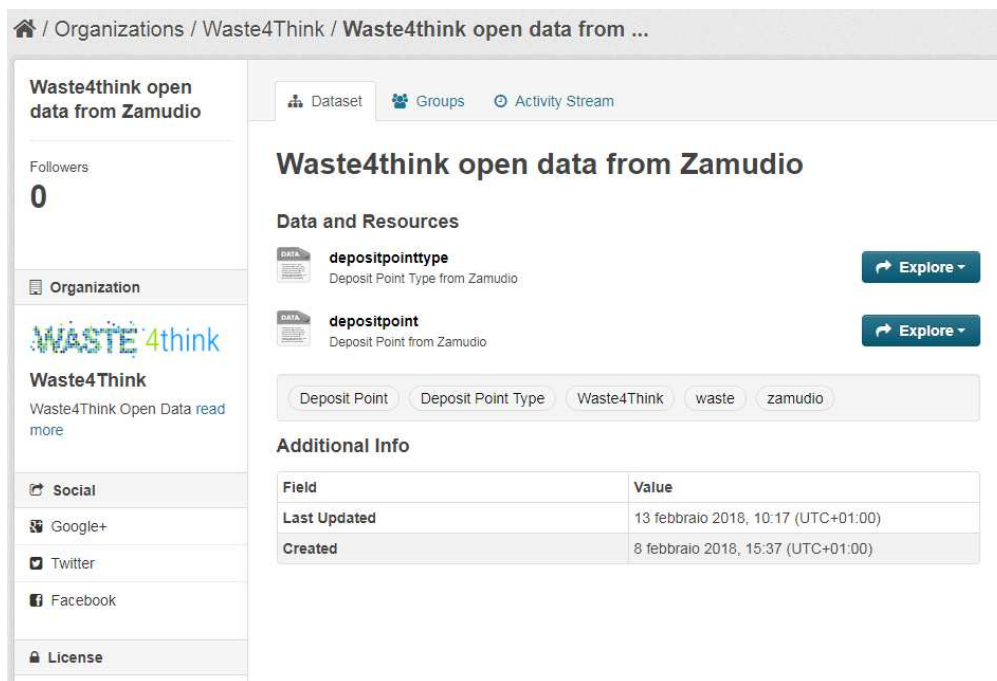


Figure 55. User interface of the Open Data Platform.



5.8. User Stories

Emilio Rivas lives in Bilbao, Spain, and he is the Head of the Waste Management Service of the city council. His responsibilities extend to the provision of all aspects of waste management: refuse collection, recycling, street cleanliness, transportation to landfill sites, and environmental waste contracts with the private sector.

Emilio arrives to his office. As usual, the first thing he does is open his Dashboards to see a broad overview of the status of the waste management system. He has previously personalized his Dashboards to see the following information:

- Alerts feed (table) of types:
 - container moved, container broken, overflow/overweight, improper waste, impossible access to the container,
 - route incidence, truck incidence,
 - special service requested
- Map with the predicted filling level of the containers
- Planned collection routes to be made today overlaid in the previous map
- Graph with the total amount of waste collected: last day, last week and last invoicing period

Emilio sees something strange in yesterday's measurements for one bin. The map widget allows the selection of one of the elements displayed and see its historical values. He does so and sees that this bin has been reporting erroneous values from some time. He opens the incidence system and creates a ticket to make someone check the sensor in this bin. Finally, he opens the CRUD tool. Then he searches for all the values of this sensor and deletes the erroneous values.

Emilio realises that he has been exclusively focusing on the technical aspects of the collection system, and now he also wants to include the monitoring of the economic, social and environmental aspects related to it.

For this end, Emilio must:

- Select the corresponding KPIs to monitor
- Select the right widget to show the information

Emilio opens the configuration interface of the Dashboard module, analyses the list of KPIs and takes a note of the KPIs he wants to track. Now, he does the same with the screen that contains the list of widgets. He writes down the following table in a paper:

	KPI	Widget
Economic	Total amount of revenues	Chart
Environmental	Total amount of GHG	Heat map
Social	Total negative quotes about waste collection system (New)	Historical graph

Now, Emilio opens the configuration interface of his own personal dashboard. There he finds an “add new widget” button. A new screen appears where he has to select the widget to visualize the information. He will start with the environmental information that he wants to track as he does not need to create any new KPI. He selects a Heat Map and configures its visualization parameters (color gradient, scale of the values, etc.) and the source of information. For this last step, Emilio opens the KPI search box and looks for the right KPI. In this case, this is not just one KPI since he wants to track the variation of a specific attribute for a list of entities. To this end, he has to write a little query where he selects the attribute “emission” of all entities “routes”. Next, he just has to drag and drop to the place where he wants to so see this map.

Then, Emilio wants to add the economic information to the Dashboard. He performs the same actions as before, but in this case, he selects a chart widget and when prompted to select a source of information, he hits the “create a new KPI” button. Now, Emilio has the visual programming tool where he can define a new KPI. In this case, the KPI is just the aggregation of all the revenues of the project, so he only needs make a couple of simple queries to select the sources of information and select the aggregation component. Then, he completes the form with the rest of information about the KPI, and he finally can continue creating the Dashboard as in the environmental information.

Next, Emilio wants to add the social information. He performs the same actions as in the previous cases, and when prompted to select a source of information he hits the “create a new KPI” button. As before, Emilio uses the visual programming tool to define this KPI. In this case, the KPI is a search of negative quotes about the waste collection, so he selects the “social behaviour tracking component” in the tool and configures it to perform an analysis of the texts published. Then, as before, he completes the form with the rest of information about the KPI and he finally can continue creating the Dashboard as in the environmental information.

Finally, he thinks that this information is quite interesting to the citizen. In order to allow this public visualization, he now goes to the Public Observatory Dashboard. This dashboard is a regular dashboard, but it is shown in the webpage and in the Citizen App. To this end, Emilio just repeats the previous actions in this Dashboard. Note that now, Emilio goes quite fast as all the KPIs have already been created, so he only has to choose the widgets, configure its shapes, link the widget to the sources of information and finally, and place it in the Public Observatory.



6. Sensor Layer technical documentation

Waste is characterized when the following questions are addressed: *When, Where, What* and *How much* waste has been generated. One of the main challenges of the project is to obtain all this information for every deposit point and for all the different collections systems in place at the pilot sites automatically (quasi real time acquisition of data). For this purpose, new hardware will be deployed at the pilot sites to be integrated among themselves and with the rest of the system. At the moment, all the pilots have identified the hardware needed and have either already purchased it or are in the process of acquiring it.

The biggest challenges related to these components are:

- for the **When** question, in order to retrieve the information needed, sensors must be deployed in the field without access to a power source and with limited connectivity. Several options with different TRL have been surveyed. Nevertheless, two problems have arisen: it has been difficult to obtain and test hardware samples, and the technology commercialized is not as robust as advertised. A solution has been found and tested, and a tender to acquire the technology is already being carried out.
- for the **Where** question, commercial system depends on the signal acquired by the GPS inside a city. Filters to correct noise measurements will be included in the platform and alerts to the changes will be used to manually decide whether further actions will be required.
- for the **What** question, the biggest challenge is that there is no technology in the market which allows to retrieve the information required during the collection of a deposit point in an automatic way. A vision-assisted characterization system to help operators in completing this task has been defined and is being tested in Zamudio.
- for the **How much** question, commercial systems have problems when measuring the weight in slopes. Moreover, there are also lixiviates that should be taken into consideration. Filters and filling level models will be implemented to correct measures and send alerts to point to potential problems.

Risk associated with all the sensors have already been identified and written in the risk section of the project (see for example Risks R6, R9, R15-R19). This explanation will be included in Section 6 of D2.1.

In this section we are presenting the design documentation of all the sensors to be deployed, which can be classified under the following categories.

- **System to monitor the generation pattern:** These sensors should allow not only to measure the amount of waste deposited in a container but also to identify the user. different technologies will be tested in the project including; electronic locks, weighting systems and filling level sensor. Section 6.1 will present the different solution tested.
- **System to characterize the waste produced:** These sensors should allow to give continuous characterization of the waste generated. Two solutions will be tested in different context: a high-resolution camera used to detect the improper use of

containers and a food waste characterization system. Section 6.2 will present the details of these sensors.

- **Treatment plant monitorization:** Several treatment plants will be developed in the project. Section 6.3 will present how it is planned to retrieve and sent the information collected by the sensors of these treatment plants to the Context Broker.
- **Real time traffic conditions:** Sensors to monitor the status of the traffic to help in the real-time routing collection were considered in the Grant Agreement but after an analysis of the state of the art, we have concluded that this is not technically viable option. Details are provided in Section 6.4.

Moreover, other inputs to the systems will be also presented here:

- **Citizen Opinion:** The opinions shared by the citizen in different context will be automatically analysed and a report will be shared. Even as it is not a sensor, this tool will provide inputs to the system so in Section 6.5 and 6.6 we will present the details of a web crawler and the information retrieved from the apps.
- **Learning Analytics:** Moreover, information about the use of the learning materials and the serious games will be collected and provided to the system to be analysed. The details of this components could be found in Section 6.7.

Please note that in this deliverable is only presented the design of the different elements of the Sensor Layer, the status of the deployment is going to be presented in the respective Monitoring Report Deliverables D1.4 to D1.7.

6.1 Identification systems

To retrieve the generation patterns of the users, in the Grant Agreement was foreseen the use of electronic lock to identify the users and a filling sensor plus a PIR or similar technology to measure the amount of waste deposited. After consultation with several experts and stakeholders, this design was discarded for several reasons:

- An individual filling sensor does not have the required precision to detect the introduction of an individual bag. For this reason, several sensors would need to be used. This solution not only was not tested but also would not be cost effective. For this reason, the use of filling sensors was discarded.
- A PIR sensors would easily detect the introduction of an individual bag, but could also have lots of false positive. Moreover, the main problem of PIR sensors is that they get dirty quite easily and lose their ability to measure anything. For these reason, PIR sensors were also discarded.

To overcome these problems, the experts consulted suggest using a locked chamber complemented with a weighting system installed in the truck (to be presented in Section 6.2). Two similar system were designed: one built for surface containers for Zamudio pilot (presented in Section 6.1.2) and one for underground containers for Cascais pilot (presented in Section 6.2.1).

6.1.1 Lock (Zamudio)

In Zamudio, a chamber system attached to street containers is proposed. No commercial solution is available so Innovative Public Procurement will be proposed. The tender is under preparation. The information of how this information reach the Context Broker will be provided in Annex G.

6.1.2 Lock (Cascais)

A commercial device has been implemented in Cascais. Technical information about it could be found in Annex A. Figure 56 shows the conceptual communication architecture of the sensors within Cascais. The truck will gather the Container ID and the container weight on its collection route through a RFID antenna. Moreover, using a radio receiver, the truck will retrieve the information stored in the electronic locks. This data will then be sent by GPRS to the MIP platform (see Deliverable 2.5 for details) along with the GPS coordinate of the waste truck. Then, from the MIP platform, a Custom NGSI API connector will access the relevant methods of the NGSI API to upload the information to the Context Broker.

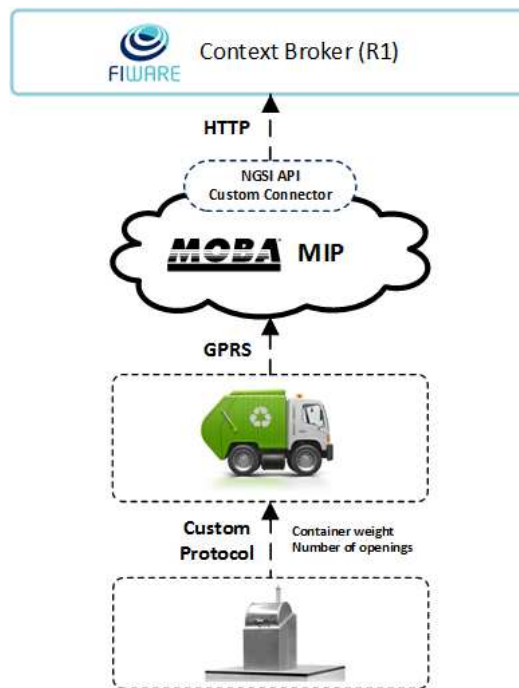


Figure 56. Conceptual communication architecture of the CASCAIS sensors.

6.2 Monitoring of containers and clean points

The monitorization of the bins was foreseen in the Grant Agreement to be made by filling sensors but for the reasons presented in Section 6.1 this was discarded. To this end, a new solution based on weighting system installed in the collection trucks was designed. Traditional weighting system perform the following actions:

- Identify the container by reading an RFID tag attached to it.
- Precisely measure the geoposition of the container using a GPS sensor.
- Measure the weight of the container.

The following section present the monitorization of the bins implemented in the different pilots' sites. Please note that in Cascais pilot a filling sensor was already deployed (see Deliverable D1.1 for details).

6.2.1 Weight sensor (Halandri)

In Halandri a commercial solution has been installed to monitor the generation pattern of the biowaste generated. The main technical characteristics are:

- The weighing error is +5% of the gross lifting load or 5 kg on the net weight of the waste.
- The elevation angles, sensor recording and processing of measurements from the controller is completed when the carriage is lifted between -35 and -10 degrees, under the horizontal position of the trolley (not the bucket). The reference (0 degrees) is the horizontal position for the hoist. These angles are variable and are adjusted depending on the construction of the hoist (may be slightly varied on each vehicle). More details of the solutions could be consulted in Annex B.

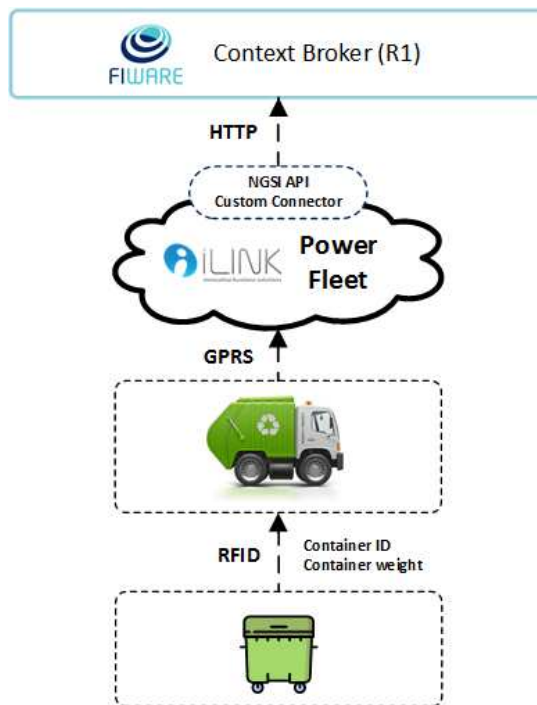


Figure 57. Conceptual communication architecture of the HALANDRI sensors.

Figure 57 shows the conceptual communication architecture of the sensors within Halandri. The truck will gather the Container ID and the container weight on its collection route through a RFID antenna. This data will then be sent to the iLink Power Fleet Software through GPRS along with the GPS coordinate of the waste truck. The iLink Power Fleet software will provide a web service to query the data in form of CSV and send it to the Context Broker through a Custom NGSI API connector that will access the relevant methods of the NGSI API.

6.2.2 Number of bags (Seveso)

Even as this system is already deployed in Seveso, we present here how the information from these sensors will reach the Context Broker. Figure 58 shows the conceptual communication architecture of the sensors within Seveso pilot site. The truck will gather the number of bags and the ID of the bag owners on its collection route through RFID antenna. This data will then be sent to the Softline Ge.R.A (further detailed in Deliverable 2.7) along with the GPS coordinate of the waste truck. The Softline Ge.R.A software will then send the data to the Context Broker through a Custom NGS API connector that will access the relevant methods of the NGS API.

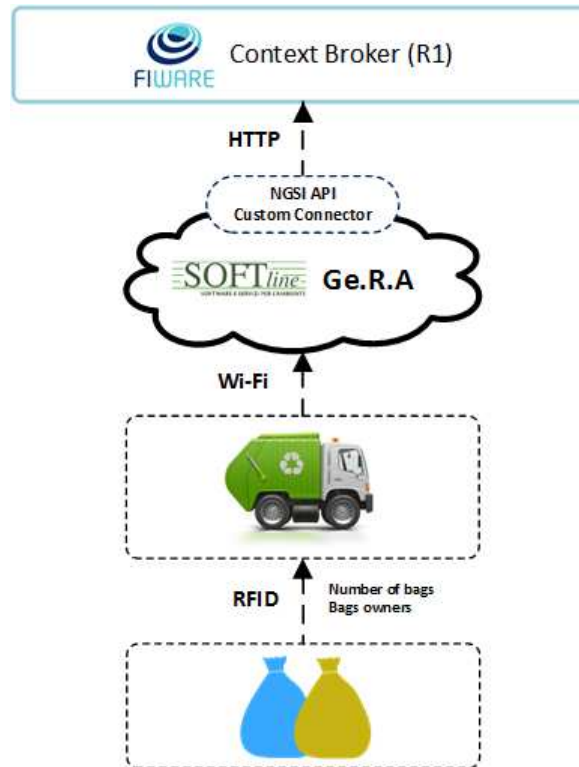


Figure 58. Conceptual communication architecture of the SEVESO sensors.

6.2.3 Truck data app (confidential)

In Zamudio, an innovative truck control system was designed. This system will integrate the measurements of the weighting system, the collection of the logs from the lock system and an innovative system to detect the presence of improper waste in the container. As the solution is currently under consideration to be patented, this section is considered confidential (Annex G).

6.2.4 Kitchen data app (confidential)

Finally, a food surplus monitoring system was developed. As the solution is currently under consideration to be patented, this section is considered confidential (Annex H).



6.3 Monitoring of the treatment plants

6.3.1 Community composting plants (Zamudio)

Both community composting plants and in situ composting plants were planned to be monitored. Even as there are sensors available, the experts in the field do not recommend using them as its cost-benefit relation is very weak. It is planned to test this in only one community composting plant with a commercial solution. Details of this sensors could be seen in Annex C. Moreover, an interface to provide the information retrieved from analog sensors will be provided. This interface will be used in the Learning Materials.

6.3.2 R19: Biomass production from food/kitchen waste

This section will explain how the information retrieved from the sensors of the Biomass Production Waste Treatment plant were introduced. Details of the treatment plant could be consulted in Deliverable D3.1.

Regarding the activity, the PLC that controls the operation of the treatment plant is connected to a SCADA system through which the operators control the operating variables of the system. The data gathered is automatically saved by the PLC and can be exported as spreadsheet files.

```

timestamp, id, nutria, d, CO2(%), mg O2
timestamp1, id1, nutria1, CO21, mg O21
timestamp2, id2, nutria1, CO22, mg O22
timestamp3, id3, nutria1, CO23, mg O23
timestamp4, id4, nutria1, CO24, mg O24
...
timestampn, idn, nutrian, CO2n, mg O2n

```

The communication between R19 (Biomass production from food/kitchen waste treatment plant) and FIWARE is explained in Figure 59. The data sent from the SCADA will be gathered in a single PC. A simple script will be responsible for parsing the data, adjusting it to the data model, and sending it to FIWARE over the HTTP protocol through the defined API.

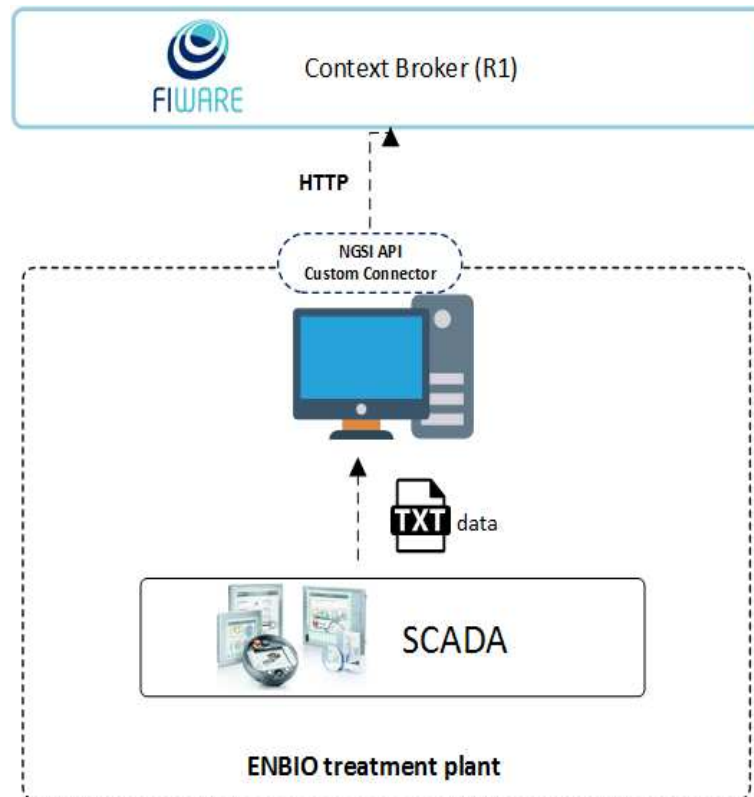


Figure 59. Conceptual communication architecture of the ENBIO treatment plant.

6.3.3 R20: Biofuel and Compost production from disposable nappies

This section will explain how the information retrieved from the sensors of the Biofuel and Compost Treatment Plant were introduced. Details of the treatment plant could be consulted in Deliverable D3.3.

The two-pilot stage plant managed by GreenTech will provide data about the number of gaseous biofuels and compost from nappies and expired dairy products and foods from supermarkets at pilot-scale via co-digestion. Regarding the activity, the PLC that controls the operation of the treatment plant is connected to a SCADA system through which the operators control the operating variables of the system (mixing frequency), operating mode (single or two stage), operation of feed pumps, etc. Specifically, the variables that will be measured from the treatment process are as follows:

- Biogas:** The pilot plant has two biogas flow meters which communicate with a PC via a pulse generator and record the biogas flow and temperature in daily basis. The recording period (per minutes, seconds or hours) depends on the monitoring settings specified on the software that accompanies the flow meters. The monitored values are originally stored in HST files (files used to log a history of system events), but are then converted to plain text files for parsing using the software provided by the sensor brand. The format of the file is as follows:

<i>User</i>					
<i>Filepath</i>					
<i>Start Date</i>					
<i>Start Time</i>					
<i>Gas Meter</i>					
<i>Pulse Generator</i>					
<i>Port</i>					
<i>Smoothing Factor</i>					
<i>Sampling Time Interval [min]</i>					
<i>Comment</i>					
<i>End of Measurement</i>					
<i>Serial Number</i>					
<i>Date Time</i>	<i>Runtime [min]</i>	<i>Volume [ltr]</i>	<i>Flow Rate [ltr/h]</i>	<i>Pulse [-]</i>	
<i>datetime₁</i>	<i>runtime₁</i>	<i>volume₁</i>	<i>flowrate₁</i>	<i>pulse₁</i>	
<i>datetime₂</i>	<i>runtime₂</i>	<i>volume₂</i>	<i>flowrate₂</i>	<i>pulse₂</i>	
<i>datetime₃</i>	<i>runtime₃</i>	<i>volume₃</i>	<i>flowrate₃</i>	<i>pulse₃</i>	
<i>datetime₄</i>	<i>runtime₄</i>	<i>volume₄</i>	<i>flowrate₄</i>	<i>pulse₄</i>	
...					
<i>datetime_n</i>	<i>runtime_n</i>	<i>volume_n</i>	<i>flowrate_n</i>	<i>pulse_n</i>	

- **pH and temperature:** The pH and temperature values of two reactors is measured by an HACH-Lange controller. The monitored values are saved in plain text files in the following format.

<i>Time, pH Methanogenic Reactor, temp Methanogenic Reactor, pH Acidogenic Reactor, temp Acidogenic Reactor</i>
<i>time₁, pH_methanR₁, temp methanR₁, pH acidogenicR₁, temp acidogenicR₁</i>
<i>time₂, pH_methanR₂, temp methanR₂, pH acidogenicR₂, temp acidogenicR₂</i>
<i>time₃, pH_methanR₃, temp methanR₃, pH acidogenicR₃, temp acidogenicR₃</i>
<i>time₄, pH_methanR₄, temp methanR₄, pH acidogenicR₄, temp acidogenicR₄</i>
...
<i>time_n, pH_methanR_n, temp methanR_n, pH acidogenicR_n, temp acidogenicR_n</i>

The communication between R20 (Biofuel and Compost production from disposable nappies treatment plant) and FIWARE is explained in Figure 60. The data sent from both the pulse the generator and the HACH-controller will be gathered in a single PC. A simple script will be responsible for parsing the data, adjusting it to the data model, and sending it to FIWARE over the HTTP protocol through the defined API.

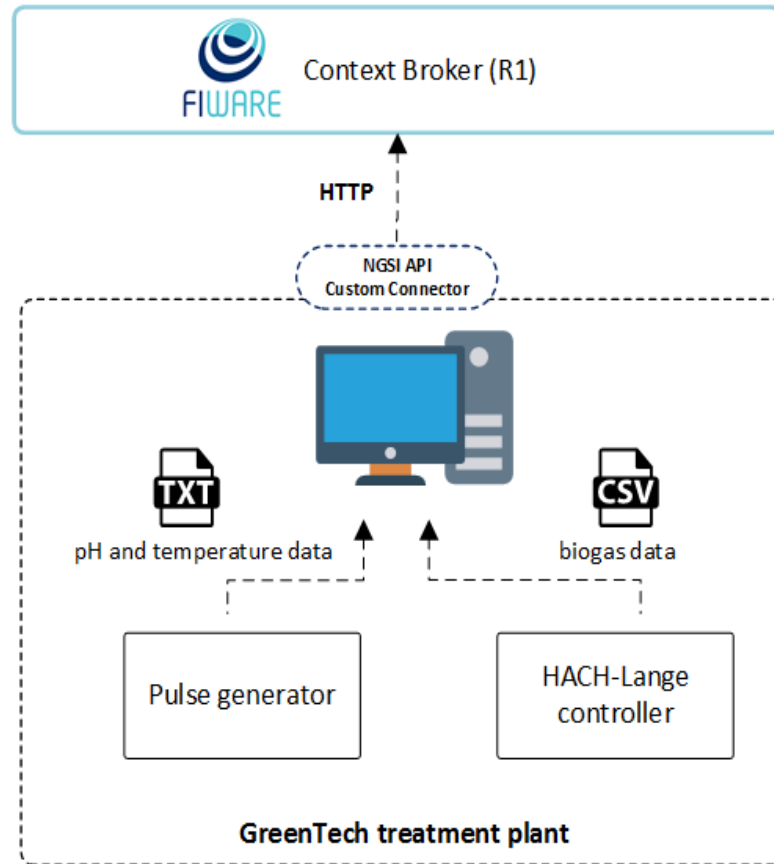


Figure 60. Conceptual communication architecture of the GreenTech treatment plant.

6.4 Real time traffic condition

The use of sensors for retrieving the real-time traffic condition was foreseen in the Grant Agreement but after consultation with several stakeholders and experts it was decided that this is not feasible to be made in a cost-effective way. For this reason, it was decided that it will be used third-party services to provide this information or models built from historical information.

6.5 Web Crawler

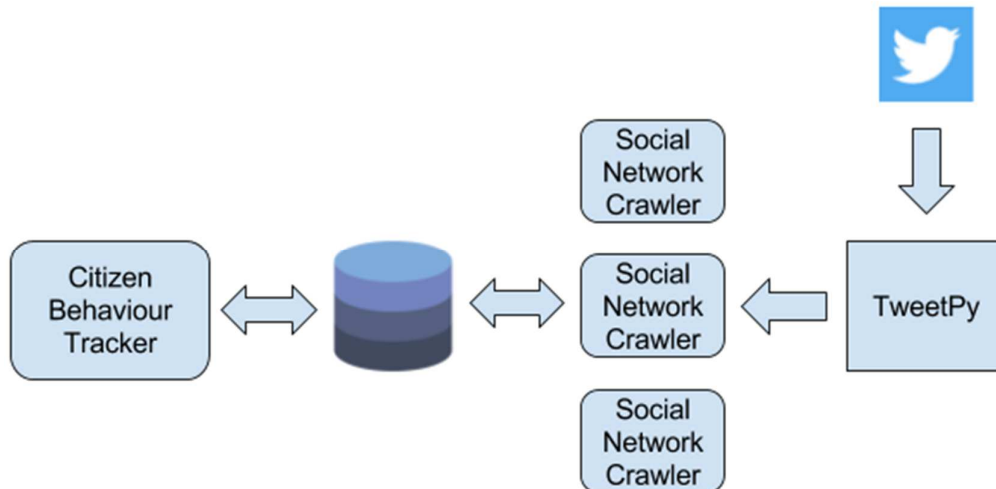
The Web Crawler objective is to measure the impact of the campaigns and the status of several topics in the social networks. We are going to produce the following status metrics:

- **Rediffusion status:** The number of times the elements are republished or reprinted in other nets. Our measure is the sum of retweets, likes and responses to all campaign adverts
- **Reputation status:** The number of followers and direct mentions
- **Impact (of a campaign) status:** rediffusion + reputation change during campaign

These statuses will be monitored before and after the campaigns and uses for visualizing the campaign lifecycle in the social networks. We are going to use **data-centric architecture**

where a pool of social crawlers will harvest the social network in base of the campaigns stored by citizen behaviour.

The Citizen behaviour tracker will introduce campaign tracking information in the Database and retrieves (and process) the generated data for the visualization. Meanwhile our social network crawlers will find the campaigns data and will ask twitter for the raw data and store it until the campaign requirements are fulfilled.



6.6 Apps

The apps will be used as input device in several aspects:

- Input of good practices in local trades
- Input of incidences in the waste collection system
- Both sides of transactions of food surpluses
- Surveys

6.7 Learning Analytics

The learning analytics will retrieve information from the use of the games and the digital learning materials. As both serious games and learning materials have not reach enough maturity and their concept have evolved significantly during this month only information for one serious game could be provided. In any case, the main questions we want to answer in this module are:

- Do you carry your own bags and / or shopping cart / container to avoid using single use containers?
- Which is the impact if you do not separate and classify domestic waste to a greater extent?
- Do you know where the debris will end up when the truck picks them up?
- Give your opinion about the ease of use of the citizen card for the deposit of waste
- what are the main problems regarding waste management in your locality today?
- Do you know the rate of waste you pay?
- Which do you think is the most expensive waste to be managed?
- Do you think we should all pay the same garbage rate?



- What amount of cooked food waste do you generate?
- Do you know about composting and its environmental potential?
- Do you think recycling is necessary?
- Do you know the colours of each container?
- Do you deposit the remains of fruits and vegetables in the brown container or in your auto-compost to make compost?

More information about this could be found in Deliverable D4.5 and detailed information will be provided in Deliverables D4.3, D4.4 and D4.6 to D4.9.

6.7.1 Learning Analytics for R9: Sorting Game (Ways2Sort)

In this section we provide the preliminary list of information that will be retrieved in the Sorting Game. Please take into consideration that this list is currently under discussion.

- Efficiency to learn about sorting:
 - Measure the knowledge about sorting in general: mean hits and fails per object
 - *Measure the level of learning along a session*: Differences between the hit and fails per sorting type between the start of session and the end
 - Level of learning along time:
 - Citizen in general: Differences between the hit and fails per sorting type between the sessions considering the dates to assess the effect of social campaigns
 - Schools: specific experiment design to be defined by VIRTUALWARE.
 - Identify the objects in doubt: Movements along the belts
- Social factors and before starting the game):
 - *Age and gender* (taken during the registration)
 - *Awareness* (taken at the beginning and at the end of a game):
 - In which grade do you recycle? or
 - Which percentage of wastes do you think you recycle at home?
- In play information:
 - User
 - Session
 - Object movements:
 - Origin
 - Destination (corrected and played)
 - Intermediate movements
 - Timestamp
 - Velocity of the conveyor belt
 - Number of conveyor belt



7. Data model

7.1 FIWARE data models

In this section we present the work carried out to model the information that is going to be shared in the platform. As one of the main guidelines of Waste4Think is to build over existing technologies and data models, we have built our contributions over the FIWARE data models. To ease the understanding of our contributions, we present here just a short description of the works carried out. For this, for every new or modified entity in the data model we will present here:

- a short introduction with the main concepts modelled
- an UML like model to clarify its relationship with other entities
- several examples

The final documentation is created automatically from the source code using the Readthedocs (<https://readthedocs.org/>) documentation system. In <https://github.com/deusto-tech/dataModels> a snapshot of the documentation built over our latest changes can be found. Please note that this webpage includes also the previous contributions of the original developers of the data model.

7.1.1 FIWARE main guidelines for data model

The FIWARE community has started a series of initiatives to enable a new generation of smarter applications which exploit large scale, real-time ‘context information’. These initiatives were mainly focused to harmonize both APIs and data models. In detail, the NGSI version 2 API (<http://fiware.github.io/context.Orion/api/v2/>) is aimed at making developer’s life easier, by providing simpler but powerful RESTful interfaces. The combination of NGSI version 2 and harmonized data models enables the creation of portability at the data layer.

Harmonizing data models means creating a shared vocabulary of terms and relationships that provide uniformity on the representation of different concepts: waste management, parking, public transport, weather, etc. In the FIWARE context, existing vocabularies, especially Schema.org (<https://schema.org/>), have been adopted and leveraged, to define the FIWARE data models.

FIWARE provides to developers a data model development document at <http://fiware-datamodels.readthedocs.io/en/latest/guidelines/index.html> with the guidelines and the recommendations related to the syntax, reuse of data model already existing, data types / attribute definitions / units / relatives value supported, and more, with the aim of creating additional data models or extending those already existing.

The FIWARE data models documentation is hosted in the related GitHub repository <https://github.com/Fiware/dataModels>. Such documentation is currently written in markdown format and published to a Readthedocs site at the following link <http://fiware-datamodels.readthedocs.io/en/latest/>.

The FIWARE data model repository contains:



- code that allows to expose different harmonized datasets useful for different applications. Such datasets are exposed through the FIWARE NGSI version 2 API (query).
- JSON Schemas and documentation on harmonized data models for smart cities, developed jointly with OASC (<http://www.oascities.org/>) and other domains.

7.2 Waste4Think improvements over FIWARE Waste Management Data Model

The aim of this section is to extend several entities from the existing FIWARE Waste data model, which currently only covered waste containers and their main attributes as Filling level, Temperature, cargoWeight (see [7][1], <http://fiware-datamodels.readthedocs.io/en/latest/WasteManagement/doc/introduction/index.html>) to fit the greatest amount possible of waste management scenarios and methodologies. Please note that this extension will not just consider the scenarios of Waste4Think but tries to be universal, so it would be able to cope with scenarios such as:

- Characterization of sorting types for different countries with their waste materials and primary categories.
- A wide range of different municipal waste collection system including on street / buried containers, door2door collection, deposit areas, clean points, industrial waste management using specialized deposit points, etc.
- Fixed and on time scheduling of routes for waste collection.
- Different rates for the waste collection services including a universal way to define PAYT and Bonuses schemas.
- Management the different social actions in the territory.

The following subsections will provide a summary of the new entities proposed.

7.2.1 Waste Characterization

Introduction of the waste characterization concept by extending the entities Litter and LitterCategory defined in the FIWARE Waste Data model. These characterizations allow a fine-grained definition of how wastes are grouped and managed in different scenarios.

- **Waste** (formerly called *Litter*): The material discarded by citizens or business' after primary use. A waste can be almost any material which is worthless for the user. Nevertheless, it can be useful for another agent as input. Its main attributes are:
 - id: Unique identifier.
 - type: Entity type. It must be equal to 'Waste'.
 - name: Name given to the waste.
 - name:en: Name given to the waste in English.
 - name:eus: Name given to the waste in Basque.
 - name:pt: Name given to the waste in Portuguese.
 - name:it: Name given to the sorting type in Italian.
 - name:gr: Name given to the waste in Greek.
 - description: Explanation of the waste.

- description:es: Explanation of the waste in Spanish.
- description:eus: Explanation of the waste in Basque.
- description:pt: Explanation of the waste in Portuguese.
- description:it: Explanation of the waste in Italian.
- description:gr: Explanation of the waste in Greek.
- refCategory: Reference to the WasteCategory entity.
- definitionSource: Source where this characterization comes from.
- image: URL of the waste photo.
- wasteCode: LER waste code.
- **WasteCategory** (formerly called *LitterCategory*): A higher level category that describes some features all child wastes share such as origin, material, fabrication process, etc. Its main attributes are:
 - id: Unique identifier.
 - type: Entity type. It must be equal to 'WasteCategory'.
 - name: Name given to the waste category
 - name:en: Name given to the waste category in English.
 - name:eus: Name given to the waste category in Basque.
 - name:pt: Name given to the waste category in Portuguese.
 - name:it: Name given to the waste category in Italian.
 - name:gr: Name given to the waste category in Greek.
 - description: Explanation of the waste category.
 - description:es: Explanation of the waste category in Spanish.
 - description:eus: Explanation of the waste category in Basque.
 - description:pt: Explanation of the waste category in Portuguese.
 - description:it: Explanation of the waste category in Italian.
 - description:gr: Explanation of the waste category in Greek.
 - refWastes: List of Waste entities that compose this category.
- **SortingType**: Grouping of wastes that are collected together in a country or service area. Sorting types usually vary according to legislation, collecting company or technical requirements. Traditional sorting types are usually light packaging, glass bottles, paper and cardboard, organic waste and refuse. Its main attributes are:
 - id: Unique identifier.
 - type: Entity type. It must be equal to 'SortingType'.
 - name: Name given to the sorting type.
 - name:en: Name given to the sorting type in English.
 - name:eus: Name given to the sorting type in Basque.
 - name:pt: Name given to the sorting type in Portuguese.
 - name:it: Name given to the sorting type in Italian.
 - name:gr: Name given to the sorting type in Greek.
 - description: Explanation of the waste category.
 - description:es: Explanation of the sorting type in Spanish.
 - description:eus: Explanation of the sorting type in Basque.
 - description:pt: Explanation of the sorting type in Portuguese.
 - description:it: Explanation of the sorting type in Italian.
 - description:gr: Explanation of the sorting type in Greek.

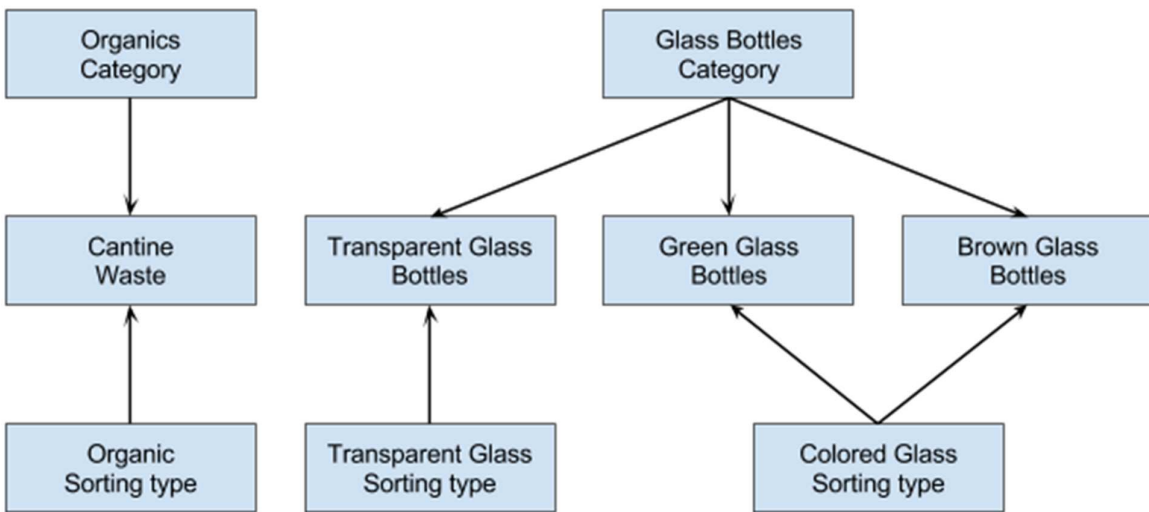


- regulation: Regulation under which the sorting type is operating.
- refResources: List of Waste entities that are collected by this sorting type.
- shape: If the shape of the container is very relevant or representative for the sorting, specify the shape of the container.
- color: Color associated to the sorting type.
- annotations: A field reserved for annotations (incidences, remarks, etc.).
- wasteCharacterization: Characterization of the sorting type.
- areaServed: Higher level area to which the sorting type belongs to. It can be used to define the area where the sorting type is applied, etc.

7.2.1.1 UML



7.2.1.2 Example



```

{
  "id": "WasteCategory:9",
  "type": "WasteCategory",
  "name": "Glass bottles",
  "description": "Glass bottles including white, green and brown glass",
  "refWastes": ["Waste:15", "Waste:85", "Waste:6"]
}
{
  "id": "waste:15",
  "type": "Waste",

```

```

    "name": "Brown glass bottles",
    "refWasteCategory": " WasteCategory:9",
    "description": "Bottles made of brown glass"
  }
  {
    "id": "waste:85",
    "type": "Waste",
    "name": "Green glass bottles",
    "refWasteCategory": " WasteCategory:9",
    "description": "Bottles made of green glass"
  }
  {
    "id": "waste:6",
    "type": "Waste",
    "name": "Transparent glass bottles",
    "refWasteCategory": " WasteCategory:9",
    "description": "Bottles made of white glass"
  }
  {
    "id": "sortingtype:12a",
    "type": "SortingType",
    "name": "Color glass collection",
    "description": "Collection of colored glass bottles",
    "color": "green",
    "refWastes": ["Waste:15", " Waste:85"]
  }
  {
    "id": "sortingtype:12a",
    "type": "SortingType",
    "name": "White glass collection",
    "description": "Collection of transparent glass bottles",
    "color": "white",
    "refWastes": ["Waste:6"]
  }
}

```

7.2.2 Deposit points

Generalization of the *WasteContainer* entity defined in the FIWARE Waste Data model to cover other ways of waste collection. The *DepositPoint* entity supersedes *WasteContainer* to consider collection methods like door2door, pneumatic, clean points, etc. Moreover, we have added several important features that were missing from deposit points.

- **DepositPoint** (formerly called *WasteContainer*): Place where waste is deposited for its collection. It covers from traditional containers to less usual methods such as a

personal plastic bags (in door2door systems), pneumatic waste dumpsters, fixed collection centers, mobile collection centers, etc. Its main attributes are:

- id: Unique identifier.
- type: Entity type. It must be equal to 'DepositPoint'.
- serialNumber: Serial number of the container.
- refSortingType: Reference to the SortingType entity.
- description: Description of the container.
- refType: Reference to the DepositPointType entity.
- storedWasteOrigin: Origin of the waste stored (household, municipal, industrial, construction, hostelry, agriculture, other)
- location: Container's location represented by a GeoJSON Point.
- address: Civic address where the container is located.
- fillingLevel: Filling level of the container (percentage, expressed in parts per one). When the container is full it must be equal to `1.0`. When the container is empty it must be equal to `0.0`. If it is not possible to determine the current filling level, it must be equal to `null`.
- cargoWeight: Weight of the container load.
- temperature: Temperature inside the container.
- methaneConcentration: Methane (CH₄) concentration inside the container.
- regulation: Regulation under which the container is operating.
- responsible: Responsible for the container, i.e. entity in charge of actuating (emptying, collecting etc.).
- owner: Container's owner.
- dateServiceStarted: Date at which the container started giving service.
- dateLastEmptying: Timestamp which represents when the container was emptied last time.
- nextActuationDeadline: Deadline for next actuation to be performed (emptying, picking up, etc.).
- actuationHours: Hours suitable for performing actuations over the container. (See <http://schema.org/openingHours>)
- openingHours: Hours when the point is available. (See <http://schema.org/openingHours>)
- dateLastCleaning: When the container was cleaned last time.
- nextCleaningDeadline: Deadline for next cleaning.
- refDepositPointIsle: Isle where the container is placed if any.
- status: Container's status from the point of view of safety.
- color: Container's color
- image: A URL with a photo of the container.
- annotations: A field reserved for annotations (incidences, remarks, etc.).
- areaServed: Higher level area to which the container belongs to. It can be used to group containers per responsible, district, neighbourhood, etc.
- dateModified: Last update timestamp of this entity
- refDevice: Reference to the device(s) used to monitor this container.
- **DepositPointType** (formerly called *WasteContainerModel*): Container, bag, dumpster, etc. models with their features that are common to all DepositPoints from

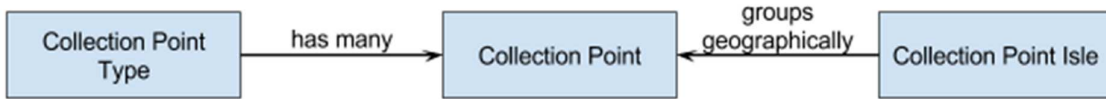
this type such as waste it contains, capacity, color, height, width, price, etc. Its main attributes are:

- id: Unique identifier.
- type: Entity Type. It must be equal to `DepositPointType`.
- name: Name given to the point type
- refInputs: List of Waste entities that are accepted by the container.
- refOutputs: List of Waste entities that are emitted by the container.
- width: Width of the collection container, bag, etc. (Metadata: units)
- height: Height of the collection container, bag, etc. (Metadata: units)
- depth: Depth of the collection container, bag, etc. (Metadata: units)
- weight: Weight of the collection container, bag, etc. (Metadata: units)
- cargoVolume: Total volume the point can hold. (Metadata: units)
- maximumLoad: Maximum load the point can hold safely. (Metadata: units)
- recommendedLoad: Manufacturer recommended load for the points. (Metadata: units)
- category: Category of the deposit point type.
 - `trashCan`
 - `bulk`
 - `wheelieBin`
 - `bag`
 - `fixed collection centers`
 - `mobile collection centers`
 - `underground`. The container is placed underground.
 - `pneumatic`. Pneumatic collection boxes.
 - `portable`. The container can be moved to a certain extent.
 - `fixed`. The container is fixed to a wall, support or handle.
 - Any other value meaningful for the application
- insertHolesNumber: Number of insert holes the container has.
- insertHoleWidth: Width of the hole or lid.
- insertHoleHeight: Height of the hole or lid.
- loadType: Deposit point loading type:
 - `side`
 - `upper`
 - `front`
 - Any other value meaningful for the application
- madeOf: Material the container is made of.
- madeOfCode: Material Code as per standard tables.
- brandName: Name of the brand.
- modelName: Name of the model.
- manufacturerName: Name of the manufacturer.
- colors: Available colors, even if it is transparent for inside visibility.
- image: A URL with a a photo of the container type.
- compliantWith: A list of standards to which the container is compliant with (ex. `UNE-EN 840-2:2013`)
- accessLimitation: Access limitation type to the container type.

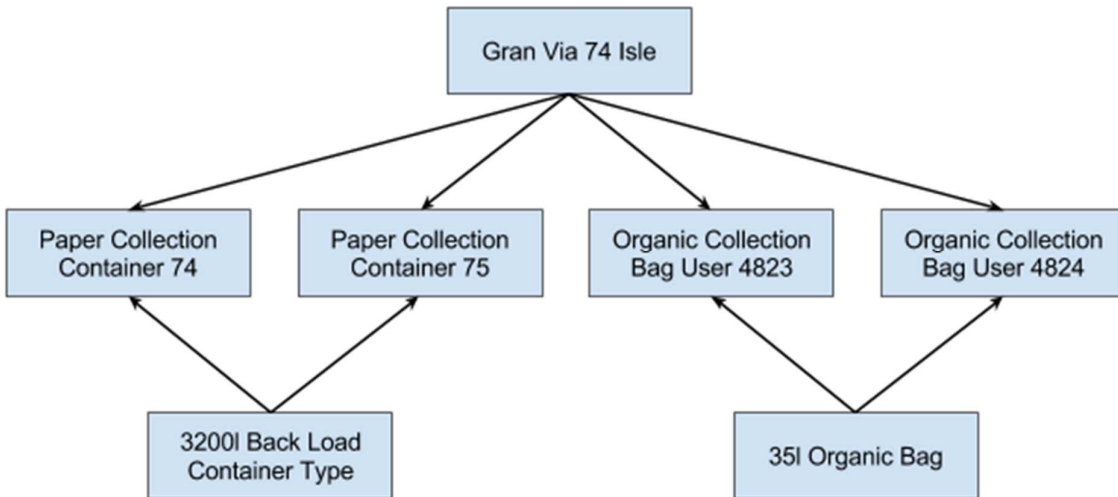
- `lock`
 - Any other value meaningful for the application.
 - userIdentification: User identification type to the deposit point type.
 - `nfc`
 - `rfid`
 - `qr code`
 - inputControl: Input control type of the container type.
 - `chamber`
 - `weight`
 - `volume`
 - Any other value meaningful for the application.
 - maximumInputVolume: Maximum volume a user can deposit. (Metadata: units)
 - features: A list of container features.
 - `wheels`
 - `handAperture`
 - `feetAperture`
 - `lid`
 - `roundedLid`
 - `insertHoles`
 - `lockable`
 - `accessible holes`
 - Any other value meaningful for the application.
- **DepositPointIsle** (formerly called *WasteContainerIsle*): A group of DepositPoints that belong together in a geographic area. This can be useful for calculations at DepositPointIsle level where there are more than one DepositPoint in a place and users tend to use them indifferently resulting in irregular measurements. Its main attributes are:
 - id: Unique identifier.
 - type: Entity type. It must be equal to `DepositPointIsle`.
 - location: Location of the isle represented by a GeoJSON Polygon.
 - address: Civic address where the isle is located.
 - name: Name given to the isle
 - description: Description about the isle.
 - features: A list of features provided by the isle.
 - `containerFix`. Allows to fix containers to a permanent position.
 - `fenced`. The isle is properly fenced.
 - `locked`. Locked isle, where a lock needs to be opened for access. (Implies being fenced).
 - `underground`. The isle allows to hold buried containers.
 - Any other value meaningful to the application.
 - refDepositPoint: List of DepositPoint entities present in the isle.
 - areaServed: Higher level area to which the isle belongs to. It can be used to group isles per responsible, district, neighbourhood, etc.
 - dateModified: Last update timestamp of this entity.

- o dateCreated: Creation timestamp of the isle (This might different than the entity creation time)

7.2.2.1 UML



7.2.2.2 Example



```

{
  "id": "wastecontainertype:c1",
  "type": "WasteContainerType",
  "width": 0.50,
  "height": 0.80,
  "depth": 0.40,
  "cargoVolume": 150,
  "brandName": "Contenedores Ejemplo",
  "modelName": "C1",
  "compliantWith": ["UNE-EN 840-2:2013"],
  "madeOf": "plastic",
  "features": ["wheels", "lid"],
  "category": ["dumpster"]
}
{
  "id": "depositpoint:Fleming:12a",
  "type": "DepositPoint",
  "refDepositPointType": "depositpointtype:c1",
  "refDepositPointIsle": "depositpointisle:Fleming:12",
  "serialNumber": "ab56kjl",
}

```

```

    "location": {
      "type": "Point",
      "coordinates": [ -3.164485591715449, 40.62785133667262 ]
    },
    "fillingLevel": 0.4,
    "dateLastEmptying": "2016-06-21T15:05:59.408Z",
    "dateNextActuation": "2016-06-28",
    "status": "ok",
    "category": ["underground"],
    "refDevice": ["device-Fleming:12a:1"]
  }
}

  "id": "depositpointisle:Fleming:12",
  "type": "DepositPointIsle",
  "location": {
    "type": "Polygon",
    "coordinates": [
      [
        [ -3.164485591715449, 40.62785133667262 ],
        [ -3.164445130316209, 40.627871567372239 ],
        [ -3.164394553567159, 40.627772099765778 ],
        [ -3.164424899616589, 40.62775018317452 ],
        [ -3.164485591715449, 40.62785133667262 ]
      ]
    ]
  },
  "address": {
    "streetAddress": "Calle Dr. Fleming, 12",
    "addressLocality": "Guadalajara",
    "addressCountry": "ES"
  },
  "features": ["underground"],
  "name": "Dr. Fleming 12, Esquina Manuel Paez Xaramillo",
  "description": "Container isle located downtown",
  "points": ["depositpoint:Fleming:12a", "depositpoint:Fleming:12b"]
}

```

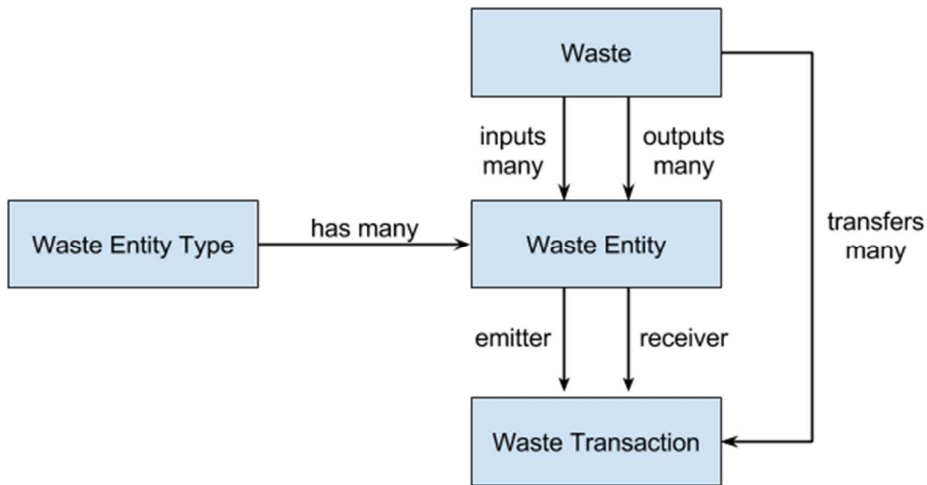
7.2.3 Waste flow

Introduction of new agents in the FIWARE Waste data model to correctly monitor the waste flow along the waste management chain. In this sense, we have modelled: generators, treatments and endPoints as a unique Waste Entity that can have waste inputs and outputs. Over those new entities, we have defined the wasteTransaction concept that we use to record all transactions among these entities.

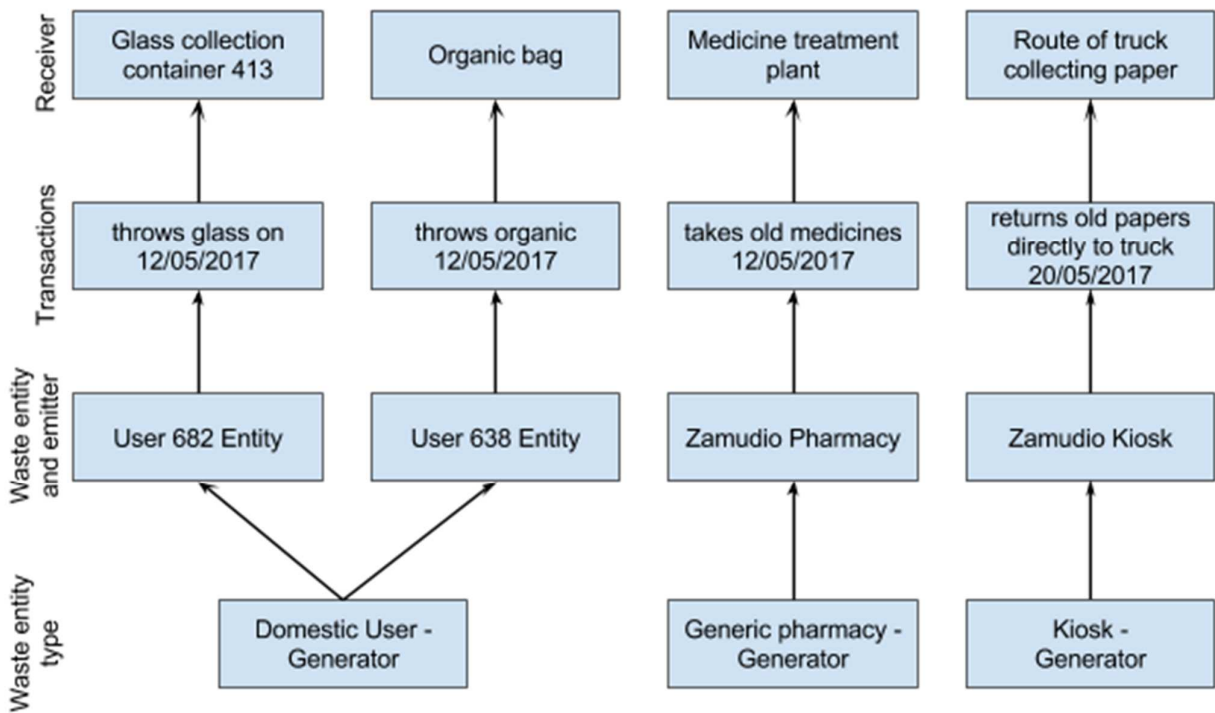
- **WasteEntity:** Each instance where waste is created, transformed or consumed, belonging to a WasteEntityType and containing the location of the entity together with specific features. Its main attributes are:
 - id: Unique identifier.
 - type: Entity type. It must be equal to `WasteEntity`.
 - location: Entity's location represented by a GeoJSON Point.
 - address: Civic address where the entity is located.
 - refWasteEntityType: Reference to a WasteEntityType entity.
- **WasteEntityType:** Description of a group of entities that will take part in the waste flow, generating, transforming or consuming waste. This generic model will have an optional list of input wastes and optional list of output wastes. The main attributes for a WasteEntityType entity are:
 - id: Unique identifier.
 - type: Entity Type. It must be equal to `WasteEntityType`.
 - refInputs: Reference to the list of Input WasteEntity entities. List of structured values containing entity Waste id and a unit of estimated quantity per day.
 - refOutputs: Reference to the list of Output WasteEntity entities. List of structured values containing entity Waste id and a unit of estimated quantity per day.
 - name: Name given to the waste entity type.
 - description: Description of the model.
 - category.
 - Treatment category.
 - Business Identification: CNAE or similar identification code.
 - ...
 - Any other category relevant to the application.
- **WasteTransaction:** Waste movements between two entities. Keeps track of the emitter, receiver, optional device that measured the transaction and the amount of waste transferred:
 - id: Unique identifier.
 - type: Entity type. It must be equal to `WasteTransaction`.
 - refEmitter: Transaction's emitter entity.
 - String value containing entity type (WasteEntity, DepositPoint, Route, etc.) and its id.
 - refReceiver: Transaction's receiver entity.
 - String value containing entity type (WasteEntity, DepositPoint, Route, etc.) and its id.
 - refCapturer: Reference to the Id of the Device defined in the FIWARE Device Data Model that measured this transaction, if exists.
 - date: Timestamp which represents when the transaction was made.
 - emittedResources. List of Waste/SortingType entities emitted in this transaction by the refEmitter.
 - receivedResources. List of Waste/Sorting Type entities emitted in this transaction by the refResources.

- Incidence: Default to null or non existing. In order not to delete any transaction, if one is incorrect to mark it as incorrect and do not use it in calculations, but keep it in the database to know.
- IncidenceReason: "Explanation if necessary of why it is incorrect or what changes have been handmade.

7.2.3.1 UML



7.2.3.2 Example



```

{
  "id": "wasteentitytype:tp1",
  "type": "WasteEntityType",
  "refWasteInputs": [ "waste:053", "waste:123"],
  "refWasteOutputs": [ "waste:6723"],
  "name": "Incinerator plant"
}
{
  "id": "wasteentity:zabalgardi",
  "type": "WasteEntity",
  "refWasteEntityType": "wasteentitytype:tp1",
  "location": {
    "type": "Point",
    "coordinates": [ -3.164485591715449, 40.62785133667262 ]
  },
  ...
}
{
  "id": "wastetransaction:5832952",
  "type": "WasteTransaction",
  "refEmitter": "WasteContainer:92" ,
  "refReceiver": { entityType: "Route" , id: "route:503" },
  "date": "2017-04-07T11:21:41.309Z",
  "refTransferredWastes": [ "waste:6723" , "waste:2342" , "waste:581" , "waste:12" ],
  "transferredLoad": {value: 946, unitCode: "KGM" },
}

```

7.2.4 Route

Introduction of the *route* entity in the data model to be able to monitor and manage itineraries of vehicles.

- **Route:** This entity models a particular transport route model, including all properties which are common to multiple itinerary instances belonging to such model. A route is a way to identify a journey that is regularly made by a given transport.
 - **id:** Entity's unique identifier.
 - **type:** Entity type. It must be equal to ``Route``.
 - **shortName:** Short name of a route, often a short and abstract identifier like "7", "34", or "Blue" that riders use to identify a route.
 - **longName:** Full name of a route, more descriptive than the shortName and often include the route's destination or stop
 - **description:** Description about this route.
 - **vehicleType:** Describes the type of transportation used on a route
 - ``tram``: Any light rail or street level system within a metropolitan area.
 - ``subway``: Any underground rail system within a metropolitan area.
 - ``rail``: Used for intercity or long-distance travel.
 - ``bus``: Used for short- and long-distance bus routes.

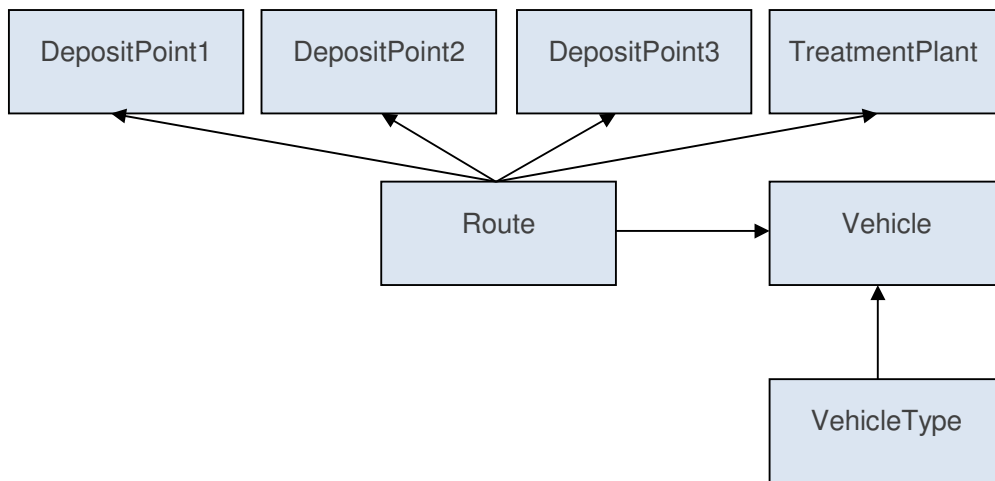
- `ferry`: Used for short- and long-distance boat service.
 - `cablecar`: Used for street-level cable cars where the cable runs beneath the car.
 - `gondola`: Suspended cable car. Typically used for aerial cable cars where the car is suspended from the cable.
 - `funicular`: Any rail system designed for steep inclines.
 - `car`: Particular vehicle.
 - `van`
 - `truck`
 - `electriccar`
 - Any other value meaningful for the scenario
- scheduledPath: Itinerary's scheduled path to be performed by the vehicle. Each value is formed by latitude, longitude and timestamp.
- realPath: Itinerary's real path that was performed by the vehicle. Each value is formed by latitude, longitude and timestamp.
- refAssignedVehicle: Vehicle that was really assigned to the route.
- refScheduledVehicle: Timestamp which captures when the driver started the route.
- arrivalPoint: Arrival location represented as a GeoJSON point.
- departurePoint: Departure location represented as a GeoJSON point.
- realArrivalTimestamp: Timestamp that captures when the driver finished the route.
- realDepartureTimestamp: Timestamp which captures when the driver started the route.
- scheduledDepartureTimestamp: Timestamp which represents when the departure should be made.
- scheduledArrivalTimestamp: Timestamp which represents when the arrival should be made.
- realStops: List of Agent on which the vehicle stopped on its route.
- scheduledStops: List of Agent on which the vehicle stopped on its route.
- refResourceTypes: List of resources gathered/delivered by the route.
- **Vehicle:** This entity models a transport. A vehicle will follow a specific route during service hours.
 - id: Entity's unique identifier.
 - type: Entity type. It must be equal to `Vehicle`.
 - vehiclePlateIdentifier: The Vehicle Identification Number (VIN) is a unique serial number used by the automotive industry to identify individual motor vehicles.
 - Name: Local identifier of the vehicle.
 - location: Vehicle's location represented by a GeoJSON Point
 - refType: Reference to entity of type VehicleType.
 - refInputs: Reference to the list of Input Resource entities. List of structured values containing entity id and a unit of estimated quantity per day.
 - refOutputs: Reference to the list of Output Resource entities. List of structured values containing entity id and a unit of estimated quantity per day.



- owner: Owner of the vehicle.
- category: Vehicle category(ies) from an external point of view. This is different than the vehicle type (car, lorry, etc.) represented by the `vehicleType` property. (`public`, `private`, `municipalServices`, `specialUsage`, `tracked`, `nonTracked`)
- speed: Denotes the magnitude of the horizontal component of the vehicle's current velocity and is specified in Kilometers per Hour. If provided, the value of the speed attribute must be a non-negative real number. `null` *MAY* be used if `speed` is transiently unknown for some reason.
- cargoWeight: Current weight of the vehicle's cargo
- purchaseDate: The date the item e.g. vehicle was purchased by the current owner.
- mileageFromOdometer: The total distance travelled by the particular vehicle since its initial production, as read from its odometer.
- vehicleConfiguration A short text indicating the configuration of the vehicle, e.g. '5dr hatchback ST 2.5 MT 225 hp' or 'limited edition'
- color Vehicle's color
- features: Indicated the technical/physical features of the vehicle: `gps`, `airbag`, `overspeed`, `abs`, `wifi`, `backCamera`, `proximitySensor`, `disabledRamp`, `alarm`, `internetConnection` or any other needed by the application.
- serviceProvided; Indicates the purpose of the vehicle: `garbageCollection`, `parksAndGardens`, `construction`, `streetLighting`, `roadSignalling`, `cargoTransport`, `urbanTransit`, `maintenance`, `streetCleaning`, `wasteContainerCleaning`, `auxiliaryServices`, `goodsSelling`, `fairground`, `specialTransport`) or any other value needed by an specific application.
- vehicleSpecialUsage: Indicates whether the vehicle is been used for special purposes, like commercial rental, driving school, or as a taxi. The legislation in many countries requires this information to be revealed when offering a car for sale. (`taxi`, `ambulance`, `police`, `fireBrigade`, `schoolTransportation`, `military`)
- areaServed: Higher level area served by this vehicle. It can be used to group vehicles per responsible, district, neighbourhood, etc.
- serviceStatus: Vehicle status (from the point of view of the service provided). `parked`, `onRoute`, `broken`, etc.
- **VehicleType:** Models the physical and technical characteristics of a vehicle that are common to vehicles that fall under the same category.
 - Id: Entity's unique identifier.
 - type: Entity type. It must be equal to `VehicleType`.
 - refInputs: Reference to the list of Input Resource entities. List of structured values containing entity id and a unit of estimated quantity per day.
 - refOutputs: Reference to the list of Output Resource entities. List of structured values containing entity id and a unit of estimated quantity per day.
 - name: Name given to the vehicle type.
 - description: Description about the model.

- vehicleType. Type of vehicle from the point of view of its characteristics. 'car','truck',etc.
- brandName: Identifier of the brand.
- numberOfAxes: Number of axes of the vehicle type.
- maxCargoPerAxe: Maximum cargo supported by each axe of the vehicle type.
- engineType: Vehicle engine type.
- enginePower: Vehicle engine power.
- tireTypes: Types of tires used by the vehicle.
- modelName: Model of the vehicle type.
- manufacturerName: Name of the manufacturer.
- vehicleModelDate: Name of the model
- maxCargoWeight: Maximum cargo weight supported by the vehicle type in kg.
- fuelDepositCapacity: Total capacity of the fuel deposit, in liters.
- compactingRatio: Compacting ratio of the vehicle type, in percentage.
- fuelType: Type of fuel(s) accepted by the vehicle type.
- fuelConsumption: Amount of fuel consumption per kilometer.
- height: height of the vehicle type in meters.
- width: width of the vehicle type in meters.
- Depth: depth of the vehicle type in meters.
- weight: weight of the vehicle type in meters.
- loadType: Loading type of the waste into the vehicle.

7.2.4.1 UML



7.2.4.2 Example

```

{
  "id": "223451",
  "type": "Route",
  "arrivalPoint": { "type": "Point", "coordinates": [ -9.33871, 38.7353 ] },
  "departurePoint": { "type": "Point", "coordinates": [ -9.38682, 38.7376 ] },
  "description": ""
}
    
```

```

"longName": "",
"realArrivalTimestamp": "2018-12-10T21:57:42+00:00",
"realDepartureTimestamp": "2018-12-10T20:55:56+00:00",
"realPath": [
  {
    "timestamp": "2018-12-10T20:55:5600:00",
    "latitude": 38.7376,
    "longitude": -9.38682
  },
  {
    "timestamp": "2018-12-10T21:43:3400:00",
    "latitude": 38.7235,
    "longitude": -9.33601
  },
  {
    "timestamp": "2018-12-10T21:57:1200:00",
    "latitude": 38.7353,
    "longitude": -9.3387
  },
  {
    "timestamp": "2018-12-10T21:57:4200:00",
    "latitude": 38.7353,
    "longitude": -9.33871
  }
]],
"realStops": [
  {
    "refStop": "DepositPoint:12233",
    "stopPoint": { "type": "Point", "coordinates": [ -9.34098, 38.703 ] },
    "arrivalTimestamp": "2018-12-10T21:21:5100:00",
    "departureTimestamp": "2018-12-10T21:21:5100:00"
  },
  {
    "refStop": "DepositPoint:10509",
    "stopPoint": { "type": "Point", "coordinates": [ -9.33992, 38.7021 ] },
    "arrivalTimestamp": "2018-12-10T21:24:2800:00",
    "departureTimestamp": "2018-12-10T21:21:5100:00"
  }
]],
"refAssignedVehicle": ""Vehicle:65"
"refScheduledVehicle": "Vehicle:65",
"scheduledArrivalTimestamp": "",
"scheduledDepartureTimestamp": "",
"scheduledPath": [],
"scheduledStops": [
  {
    "refStop": "DepositPoint:10500",

```

```

    "stopPoint": { "type": "Point", "coordinates": [ -9.34105, 38.7031 ] },
    "arrivalTimestamp": "",
    "departureTimestamp": ""
  },
  {
    "refStop": "DepositPoint:10503",
    "stopPoint": { type": "Point", "coordinates": [ -9.3407, 38.7027 ] },
    "arrivalTimestamp": "",
    "departureTimestamp": ""
  }
],
"shortName": "",
"vehicleType": "",
"refResourceTypes": []
}

```

7.2.5 KeyPerformanceIndicators

Extension of the existing KeyPerformanceIndicator entity.

- **KeyPerformanceIndicator:** A type of performance measurement. KPIs evaluate the success of an organization or of an activity in which it engages. The present data model defines a type of NGSI entity which captures the value and associated details of a key performance indicator. The data model is flexible enough to accommodate different usage scenarios: An entity per KPI calculation or a unique entity per KPI which value evolves over time. Please note that in the latter case a historical module, such as the STH would have to take care of the KPI evolution.
 - id: Entity's unique identifier.
 - type: It must be `KeyPerformanceIndicator`.
 - name: Indicator's name which should be meaningful in the context of a project or organization. Example `KPI-2016-2018-Incidences-Street`.
 - alternateName: An alias for the KPI.
 - organization: Subject organization evaluated by the KPI.
 - process: Subject process evaluated by the KPI.
 - product: Subject *product or service* evaluated by the KPI.
 - provider: Provider of the product or service, if any, that this KPI evaluates.
 - businessTarget: For informative purposes, the business target to which this KPI is related to.
 - description: Indicator's description.
 - calculationFrequency: How often the KPI is calculated.
 - `hourly`
 - `daily`
 - `weekly`
 - `monthly`
 - `yearly`
 - `quarterly`
 - `bimonthly`
 - `biweekly`

- Any other value meaningful for the application and not covered by the above list.
- category: Indicator's category.
 - `quantitative`
 - `qualitative`
 - `leading`
 - `lagging`
 - `input`
 - `process`
 - `output`
 - `practical`
 - `directional`
 - `actionable`
 - `financial`
 - Any other value meaningful to the application and not covered by the above list.
- calculatedBy: The organization in charge of calculating the KPI.
- calculationMethod: The calculation method used.
- calculationFormula: For informative purposes, the formula used for calculating the indicator.
- aggregatedData: Entity(ies) and attribute(s) aggregated by the KPI.
 - Attribute type: List of StructuredValue
 - `entityType`: Entity type which data is aggregated.
 - `attrs`: Attributes which value is aggregated.
- calculationPeriod: KPI's period of time.
- currentStanding: The KPI's current standing as per its `kpiValue`. Values: very good, good, fair, bad, very bad.
- kpiValue: Value of the KPI. (Metadata: Units)
- dateCreated: The date on which the organization created this KPI. This date might be different than the entity creation date.
- referenceValue: A reference this KPI should be similar to.
- expectedValue: Expected value for the KPI according to the reference.
- desirableTendency: A text describing the desirable tendency of the KPI.
- dateNextCalculation: Date on which a new calculation of the KPI should be available.
- dateExpires: The date on which the KPI will be no longer necessary or meaningful.
- dateModified: Last update date of the KPI data. This can be different than the last update date of the KPI's value.
- location: Location of the area to which the KPI refers to.
- address: Civic address of the area to which the KPI refers to.
- area: For organizational purposes, it allows to add extra textual geographical information such as district, borough, or any other hint which can help to identify the KPI coverage.



7.2.6 Social Actions

Introduction to the *strategy* and *action* concepts. These two concepts model actions made to influence the values of one or several KPIs. This way, a *strategy* will consist of a series of *actions* that have a similar goal. On the other hand, with an *action*, we understand things like a marketing campaign, the introduction of a new tax, etc.

- **Strategy:** Global concept covering several actions that are carried out to achieve one or more goals. Strategy generally involves determining the target group, the area served, objectives and actions to achieve them.
 - id: Entity's unique identifier.
 - type: It must be `Strategy`.
 - name: Strategy's name which should be meaningful in the context of a project or organization.
 - description: Strategy's description.
 - startDate: When the social action starts.
 - endDate: When the social action ends.
 - keyMessages: Main messages that are to be transmitted with this strategy.
 - promoter: entity or organization in charge of organizing the strategy
 - executor: entity or organization that will carry out the strategy.
 - refActions: Reference to entities of type Action that belong to this strategy.
 - refTargetGroups: Groups towards which the strategy is oriented (young adults, high school students, tourists, etc.).
 - refWasteStages: Parts of the hierarchy (prevention, communication, reuse) in which the strategy wants to make a difference.
 - refWasteStreams: Waste streams that are covered by this strategy.
 - resultsFeedback: Feedback gained from the application of this strategy.
 - url: URL with more information about the strategy.
- **Action:** Actions taken to try to have an influence on a target group according to a predefined strategy. Actions generally involve a specific methodology applied on a predefined target group.
 - id: Entity's unique identifier.
 - type: It must be `Action`.
 - name: Social Action's name which should be meaningful in the context of a project or organization.
 - refStrategy: Strategy to which the actions belongs.
 - description: Action's and methodology description.
 - success: 0 to 5 number describing the success of the action once implemented.
 - status: Current level of the action. Expected values are:
 - `started`
 - `ongoing`
 - `finished`
 - areaServed: Location of the area to which the Social Action is applied. It can be used to define the area where the strategy is applied, etc.

- areaDescription: For organizational purposes, it allows to add extra textual geographical information such as district, borough or any other hint which can help to identify the Social Action coverage.
 - successKeyPoints: Information about the activities that contributed to the success or failure of the action.
 - lessonsLearnt: Text about the lessons learned with the action.
 - startDate: Datetime when the action starts.
 - endDate: Datetime when the action ends.
 - refTargetGroups: Groups towards which the action is oriented (young adults, high school students, tourists, etc.).
 - refWasteStreams: Waste streams that are covered by this action.
 - refWasteStages: Parts of the hierarchy (prevention, communication, reuse) in which the action wants to make a difference. Expected values:
 - `recycle`
 - `reuse`
 - `valorization`
 - `ecodesign`
 - `prevention`
 - Any other value meaningful for the application
 - KPIs: Aggregation of all the indicators that are measured by the different implementations belonging to an action (number of participants, amount of waste generated, etc.)
 - participationRate: Amount of people that really participated in the action against the expected number of attendants.
 - monitoringMethodology: Description of the monitoring methodology description: surveys, waste weighting, etc.
 - communicationTools: List of URLs to Communication tools developed during the action.
 - trainingTools: List of URLs to training tools developed during the action.
 - otherTools: List of URLs to other tools developed during the action.
 - socialImpact: Text describing the social impact.
 - economicImpact: Text describing the economic impact.
 - environmentalImpact: Text describing the environmental impact.
 - otherImpact: Text describing other, not contemplated, impact that was measured by the action.
 - costs: Total cost of the action once implemented.
 - otherInformation: Text with other information that may be interesting to underline.
 - documents: List of documents produced for the action (leaflets, flyers, news on local magazines, etc).
 - pictures: URLs to pictures of the action.
 - isBestPractice: Denotes whether the action is considered as best practice.
 - keywords: Text describing the keywords associates to the action.
 - url: URL with more information about the action.
- **Implementation:** Iterations of repetitions of a specific action along time.

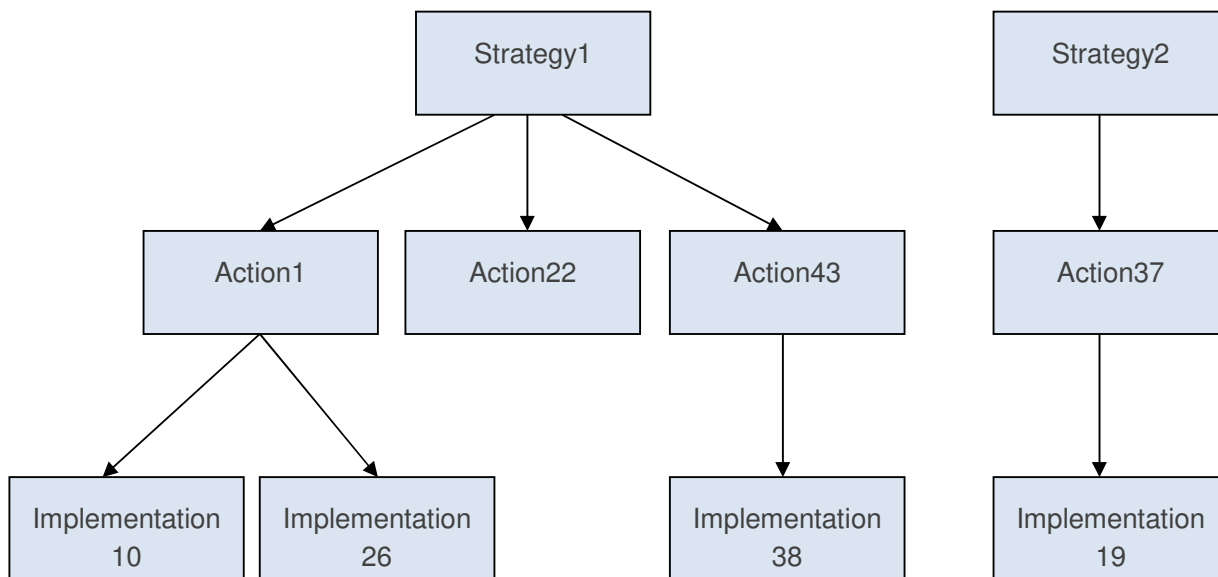


- id: Entity's unique identifier.
- type: It must be `Action`.
- name: Implementation's name which should be meaningful in the context of a project or organization.
- refAction: reference to entity of type Action to which this implementation belongs.
- success: 0 to 5 number describing the success of the action once implemented.
- description: Explanation of the implementation.
- areaServed: Location of the area to which the implementation is applied. It can be used to define the area where the strategy is applied, etc.
- startDate: Datetime when the implementation starts.
- endDate: Datetime when the implementation ends.
- participationRate: Amount of people that really participated in the action against the expected number of attendants.
- costs: Total cost of the implementation.
- mediaURLs: URLs to published media during the implementation.
- KPIs: Aggregation of all the indicators that are measured by the different implementations belonging to an action (number of participants, amount of waste generated, etc.)
- refTargetGroups: Groups towards which the action is oriented (young adults, high school students, tourists, etc.).
- refWasteStreams: Waste streams that are covered by this action.
- refWasteStages: Parts of the hierarchy (prevention, communication, reuse) in which the action wants to make a difference. Expected values:
 - `recycle`
 - `reuse`
 - `valorization`
 - `ecodesign`
 - `prevention`
 - Any other value meaningful for the application
- documents: List of documents produced for the implementation (leaflets, flyers, news on local magazines, etc).
- pictures: URLs to pictures of the implementation.
- otherInformation: Text with other information that may be interesting to underline.
- otherQualitativeResults: Qualitative results of the implementation.
- keywords: Text describing the keywords associates to the action.
- **BestPractice:** Refer to a method or technique that has been generally accepted as superior to any alternatives because it produces results that are superior to those achieved by other means or because it has become a standard way of doing things.
 - id: Best practice's unique identifier.
 - type: It must be `BestPractice`.
 - stage: List of stages in which the practice was applied:
 - `recycle`

- `reuse`
 - `valorization`
 - `ecodesign`
 - `prevention`
 - Any other value meaningful for the application
- refMunicipality: Id of the municipality where the best practice was applied.
- municipalityType: List of types of municipalities where the best practice was applied:
 - rural
 - urban
 - semiurban
 - Any other value meaningful for the application
- municipalityActivities: List of activities that are carried out in the municipality where the best practice was applied:
 - tourism
 - industry
 - services
 - agriculture
 - Any other value meaningful for the application
- municipalityCollectionType: Waste collection system the municipality has.
- refEnterprise: ID of the company that implemented the best practice.
- enterpriseActivities: List of CNAE codes describing the enterprise activities.
- refWasteInputs: Reference to Wastes the enterprises at the municipality or the enterprise that implemented the best practice can have as inputs. List of structured values containing entity Waste id and a unit of estimated quantity per day.
- refWasteOutputs: Reference to Wastes the enterprises at the municipality or the enterprise that implemented the best practice can have as outputs. List of structured values containing entity Waste id and a unit of estimated quantity per day.
- enterpriseBillingAmount: Amount of money the factory makes a year.
- enterpriseEmployeesAmount: Number of employees of the enterprise.
- scale: At which territorial scale has the best practice been implemented. Expected values:
 - local
 - regional
 - european
 - Any other value meaningful for the application
- viability: 1 to 5 value of how feasible it is to replicate the best practice.
- cost: Final cost of the best practice
- impact: Environmental impact improved by the best practice.



7.2.6.1 UML



7.2.7 Tariff

Tariffs are normally defined for specific target groups (please do not confuse them with the targets groups of the project), therefore a municipality can hold a variety of different tariffs at the same time. For example, a specific tariff can be defined for the residential sector in downtown, while a different tariff can be defined for the industrial park or the residential sector in the outskirts of a city.

In general, a tariff is composed of three parts: a fixed part, a variable part and bonuses. The fixed part should only depend on the attributes of the target group, while the variable part should only depend on the use of the waste collection system. Finally, the bonuses will be linked to the achievement of a KPI. In general, the tariff is the sum of fixed and variable part minus the bonuses, but other strategies can be applied.

In some cases, a set of bonuses can be defined along with the tariff to foster a particular good behaviour. The use of bonuses is more frequent with non-PAYT tariffs, but it can also be used in conjunction with PAYT tariffs to further increase its impact. Please, take into consideration that a fine is just a bonus of the opposite sign.

Bonus should have a clear target group and are subject to the achievement of a certain milestone in a KPI on a certain time interval (typically a year).

Some of the concepts taken into consideration to build the fixed part of the tariff are as follows:

- Geographical localization of the property.
- Surface of the property.
- Number of persons registered, or number of persons employed.



On the other hand, some of the concepts taken into consideration to build the variable part are:

- Water consumption (or other proxies).
- Collection frequency or number of uplifts.
- Weight or volume collected.

Some of the typical concepts that are prone to be included as bonus are as follows:

- Use of the organic collection system.
- Use of in-situ treatments like auto-composting or other reuse techniques.
- Participation on social campaigns.

The above concepts can be encapsulated in the following abstract formula:

$$\text{tariff}_T(t_{ini}, t_{end}; c, d, e, k; a, u, w) := h(f(c; a), v(d; u(t_{ini}, t_{end})), b(e; k; w(t_{ini}, t_{end})))$$

where T identify the target group, t_{ini} and t_{end} represent the initial and end date of the bill, f , v and b are three functions that represent the fixed and the variable part respectively (in general they are just the scalar product $f(c; a) = c \cdot a$ and b the bonus earned, h is another function that represents the final tariff (in general just the sum of the previous two minus the third one), c , d and e are the fixed parameters (with respect to a , u , k and w but that can depend on external factors) of f , v and b (that typically represent the unitary cost) and, a denotes the values of the attributes of the taxable person, $u(t_{ini}, t_{end})$ the number of uses of the system made during the billing period and k denotes the parameters of the bonuses and $w(t_{ini}, t_{end})$ the values of the KPIs used to.

Please note that b could be further expanded to:

$$b(e; k; w(t_{ini}, t_{end})) := \sum_{i \in B} \beta(e_i; k_i; w_i(t_{ini}, t_{end}))$$

where B denotes the set of potential bonuses, β is a function that gives the bonus earned from the results achieved, k is the set of parameters used to determine if the milestone has been achieved, e denotes the parameters of the bonus earned, and w denotes the values achieved in the KPI associated with the bonus over a specific time period. Please note that β usually takes the following form:

$$\text{if}(w_i(t_{ini}, t_{end}) > k_i) \text{ then } e_i \text{ else } 0$$

Under this model, a tariff is defined by the four functions f , g , b and h . An easy way of storing this function in a database is by means of a Straight Line Program (SLP) [22]. This data structure has been found to have great properties to efficiently store and compute algebraic problems [23] as well as artificial intelligence problems [22]. An slp consists of a finite sequence of computational assignments where each assignment is obtained by applying some functional to a set of arguments that can be variables, constants or pre-computed results. Figure 61 contains an example of an SLP that compute the function $2y(x + 1)^2 - 2y$.

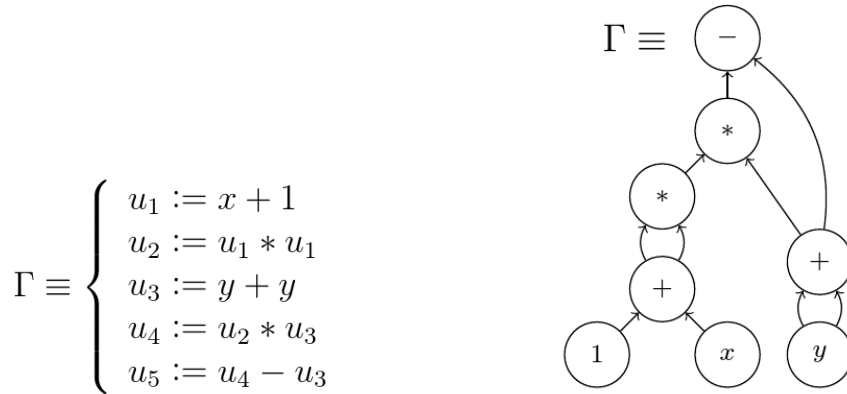


Figure 61. Linear and graph representation of an SLP to evaluate the function $2y(x + 1)^2 - 2y$

As seen in the left side of Figure 61, this function representation can be easily stored in a database as a list of triplets. In this project, we propose to store these three functions as three SLP that can be easily compiled as CEP functions. The functional used to build the SLPs will be restricted to the arithmetic functions plus the min and max functions. This way, an SLP can code any piecewise rational function. The variables on which the SLP will operate will be of the following type:

- **fixed constants** (with respect to both a , the set of attributes of the taxable person and u , the uses of the collection system) (c and d in the previous notation).
- **pointers to attributes** of a taxable person (a in the previous notation).
- **pointers to the results of the CEP** calculation of the uses in a certain period of time (u in the previous notation).

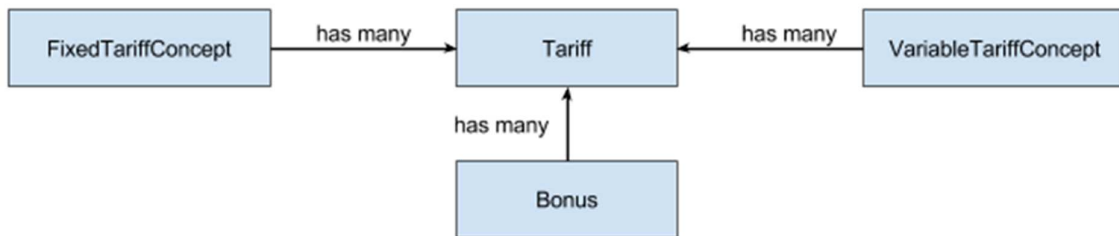
Therefore, following the mentioned schema the data model for Tariff would be as follows:

- **FixedTariffConcept**: It will receive the target group entity and will calculate some fixed concepts according to the static properties of the entity.
 - id: Unique identifier.
 - type: Entity Type. It must be equal to `FixedTariffConcept`.
 - name: Name given to the tariff concept.
 - description: Description of the tariff concept.
- **VariableTariffConcept**: It will receive a time serie of elements related to the target group entity and will calculate some variable concepts according to those elements.
 - id: Unique identifier.
 - type: Entity Type. It must be equal to `VariableTariffConcept`.
 - name: Name given to the tariff concept.
 - description: Description of the tariff concept.
- **Bonus**: Bonuses decrease the tariff fixed or variable concepts subtracting a value or by multiplying the tariff by a number lower than 1. It is very important to store both the sources of the funding and the total budget of the bonus to be able to manage its implementation.
 - id: Unique identifier.
 - type: Entity type. It must be equal to `Bonus`.



- name: Name given to the bonus concept.
- description: Description of the bonus and how to apply it.
- **Tariff:** The tariff is composed of a fixed part and a variable part. Additionally, it has the following attributes:
 - id: Unique identifier.
 - type: Entity Type. It must be equal to `Tariff`.
 - name: Name given to the tariff
 - description: Text describing the tariff.
 - targetGroup: Text describing the entities that are the target of this strategy.
 - periodicity: How often the tariff is calculated.
 - unit: Units of the tariff.
 - fixedConcepts: List of `FixedTariffConcept` that describe the fixed part of the tariff
 - variableConcepts: List of `VariableTariffConcept` that describe the variable part of the tariff

7.2.7.1 UML



7.2.8 FoodApp

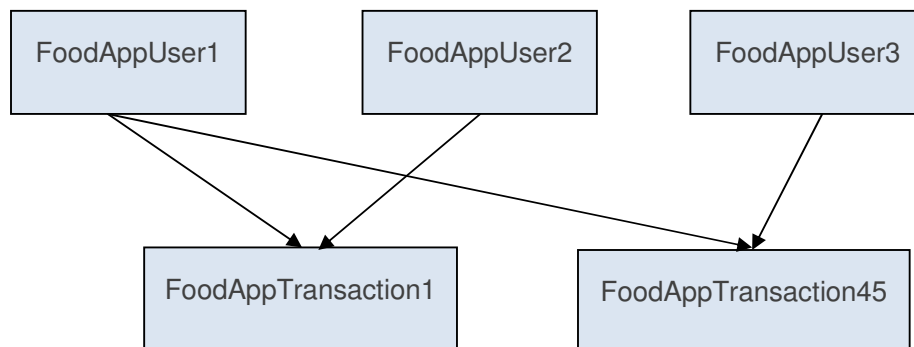
With the support of a real-time characterization system, the Food App (R5) will allow citizens and enterprises to advertise the products that continue to be in good condition but are going to be thrown and to implement cost-effective processes for their collection and retrieval by social kitchens, for example. With a prevention approach the app will also give information to the generators (large generators as supermarkets or restaurants, but also citizens in general) about their daily generation patterns. As such, the data model also contemplates a specific set of entities to describe the interactions of between users of the app and subsequent food transactions. More information about the FoodApp is available in Deliverable 4.11 Technical Documentation of R5, R6 and R7: Apps.

- **FoodAppUser:** This entity describes the user registered in the food app whether donor or donee:
 - id: Entity's unique identifier.
 - type: It must be `FoodAppUser`.
 - profile: Whether the user is donor, donee, or both.



- refResourceSurplus: List of surpluses that are published by the user. This list changes dynamically according to the transactions (donations performed) and the publication of new. Each surplus will feature the following attributes:
 - foodCategory
 - weight
 - weightUnit
 - metadata with additional information about the food surplus
- economicalActivities: economic activity code of the user.
- **FoodAppTransaction:** This entity describes the process of the donation of food between donor and donee:
 - id: Entity's unique identifier.
 - type: It must be 'FoodAppTransaction'.
 - family: It must be 'Transaction',
 - refEmitter: Id of the FoodAppUser that has published the surplus
 - refReceiver: Id of the FoodAppUser that received the surplus
 - refCapturer: Application that captures the transaction. Must be 'FoodApp',
 - date: Date of the transaction,
 - emittedResources: List of surpluses that were emitted on this transaction by the donor. Each surplus will feature the following attributes:
 - foodCategory
 - weight
 - weightUnit
 - metadata with additional information about the food surplus
 - receivedResources: List of surpluses that were given on this transaction to the donor. Each surplus will feature the following attributes:
 - foodCategory
 - weight
 - weightUnit
 - metadata with additional information about the food surplus

7.2.8.1 UML





8. Namespaces

The data described in the data model and stored in the Context Broker will be managed using the namespace convention provided by FIWARE [24].

The namespace is defined as a hierarchy where the root level contains all elements in the Context Broker. Each element can be created within a specific name subspace that allows not only to organize, but also, limits the visibility of the element. This means that an element is visible for any command (query, delete or update) invoked on its namespace or a higher-level namespace.

The usage of namespaces has three main advantages:

- **Isolation between pilot sites.** Data in a specific namespace cannot be reached from a different namespace. Therefore, defining a namespace for each different pilot allows to use the same infrastructure for multiple pilots transparently.

*Example: All containers of Zamudio are under the **/Zamudio** namespace. All apps deployed for Zamudio will use the namespace in their commands. Therefore, even in the case that a Zamudio module fails, the data from the other pilot sites is secure from any interference.*

- **Mirroring elements:** Using namespaces allows to mirror different status of the same element while preserving its original ID.

*Example: When run, a Zamudio simulation will clone all the data from the waste trucks of **/Zamudio/Real** to **/Zamudio/SimTruck1**. Every graphical element, such as the widgets, and the Rules Manager, which need the element ID to operate, will continue to correctly identify the object, but will be able to retrieve its values either from the Real or the Simulated namespace.*

- **Information granularity.** Using namespaces provides different levels of isolation inside the same data environment.

*Example: All the data from Zamudio is stored in the namespace **/Zamudio**. The waste collection data is under **/Zamudio/WasteCollection** and the social data is under **/Zamudio/Food**, an app can be set to only access **/Zamudio/Waste**. Adding an additional level of privacy and data security.*



8.1 Naming convention

The naming convention is defined as a five-level structure where each level categorizes the elements by a different semantic. Figure 62. Namespace structure for the Context Broker. contain a graphical representation of the five levels.

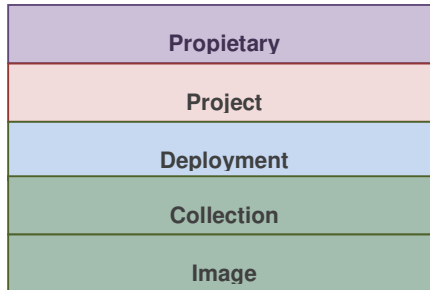


Figure 62. Namespace structure for the Context Broker.

- **Proprietary.** Organization in charge of the management of data. Example: FDEUSTO.
- **Project.** A group of applications that work together to provide a solution or a service. Example: Waste4Think.
- **Deployment.** Each of the different deployments of the project. Example: Zamudio, Cascais, Halandria, or Seveso.
- **Collection.** Group of entities that share a common scope and purpose. WasteCollection, Social, KPIs, etc.
- **Image:** Special category that allows to organize different status of the same entities. Example: Real, Simulation1, Simulation2, Expected, etc.

Examples:

/FDEUSTO/Waste4Think/Zamudio/WasteCollection/Container1/Real

/FDEUSTO/Waste4Think/Zamudio/WasteCollection/Container1/Simulation1

/FDEUSTO/Waste4Think/Cascais/Social/Real

/FDEUSTO/Waste4Think/Cascais/Social/Simulated1



Category	Category description	Entity	Entity description
	Introduction of the waste characterization concept by extending the entities Litter and LitterCategory defined in the FIWARE Waste Data model. The characterizations allow a fine-grained definition of how wastes are grouped and managed in different scenarios.	Resource	The material discarded by citizens or business' after primary use. A waste can be almost any material which is worthless for the user. Nevertheless, it can be useful for another agent as input.
		ResourceCategory	A higher-level category that describes the main characteristics of different waste types such as their material composition, hazardous properties, etc.
		SortingType	Grouping of entities that are collected together in a recycling or sorting area according to their material composition, hazardous properties, etc. The material is used for recycling, energy recovery, or other purposes.

Figure 64. Pilots data document - Entities

- entities are defined at attribute level (Figure 65);

Entity	Entity description	Entity Attributes	Attribute description	Attribute Type	Attribute mandatory	Eco-solution that uses the attribute
Resource	The material discarded by citizens or business' after primary use. A waste can be almost any material which is worthless for the user. Nevertheless, it can be useful for another agent as input.	type	Resource type. In this case fixed to "Waste"	Entity Type	Yes	R1.2, R1.4, R3.2, R3.3, R4, R5.1, R7.2, R8, R9,
		name	Waste Name. Example "Green glass bottle"	Text	Yes	
		description	Waste description. Example "Bottle made of green glass"	Text	No	
		refCategory	Reference to category entity this belongs. Example [wastecategory:9]	WasteCategory ID	Yes	
		definitionSource	Where this characterization comes from	Text	No	
		wasteCode	LER waste code.	Text	No	

Figure 65. Pilots data document - Attributes

- each sheet has been structured with a static part that provide description information about entities and attributes (Figure 66) and an editable part filled by each pilot (Figure 67);

Category	Category description	Entity	Entity description	Entity Attributes	Attribute description	Attribute Type	Attribute mandatory	Eco-solution that uses the attribute	Attribute provided
Waste Characterization	Introduction of the waste characterization concept by extending the entities Litter and LitterCategory defined in the FIWARE Waste Data model. The characterizations allow a fine-grained definition of how wastes are grouped and managed in different scenarios.	Resource	The material discarded by citizens or business' after primary use. A waste can be almost any material which is worthless for the user. Nevertheless, it can be useful for another agent as input.	id	Unique identifier. Example "waste:6"	Entity ID	Yes	R1.2, R1.4, R3.2, R3.3, R4, R5.1, R7.2, R8, R9,	
				family	Fixed to "Resource"	Text	Yes		
				type	Resource type. In this case fixed to "Waste"	Entity Type	Yes		
				name	Waste Name. Example "Green glass bottle"	Text	Yes		
				description	Waste description. Example "Bottle made of green glass"	Text	No		
				refCategory	Reference to category entity this belongs. Example [wastecategory:9]	WasteCategory ID	Yes		
				definitionSource	Where this characterization comes from	Text	No		

Figure 66. Pilots data document - Descriptions



Cascais			Seveso			Zamudio				Hal	
Attribute provided	Attribute F(ixed) / U(updatable)	How data is provided CSV: CSV File POST: Send entity directly to the ContextBroker	Attribute provided	Attribute F(ixed) / U(updatable)	How data is provided CSV: CSV File POST: Post request to the ContextBroker	Attribute provided	Attribute F(ixed) / U(updatable)	How data is provided CSV: CSV File POST: Post request to the ContextBroker	Origin of the information	Attribute provided	Attribute F(ixed) / U(updatable)
	F	Description of how data (both entites and attributes update) are sent to the context broker (i.e HTTP POST using NGSiv2 API, CSV, HTTP specific REST API to be developed).	Yes	F		Yes	F		NonBota2er, Waste2Sort (database already retrieved), ILC (food), easetech	Yes	F
	F		Yes			Yes	F			Yes	
	F		Yes	F		Yes	F			Yes	F
						Yes	F				
						Yes	F				
						Yes	F				

Figure 67. Pilots data document – Editable part

9.2. Uploading pilot data to Waste4Think Backend

Waste4think pilots can opt for different methods to upload their data (sensors, legacy system, etc.) to the backend (see Figure 68. Uploading systems). These systems provide pilots different approaches to handle uploading data, with the aim to cover several scenarios such as near real-time process, batch process and upload process that require a user interaction.

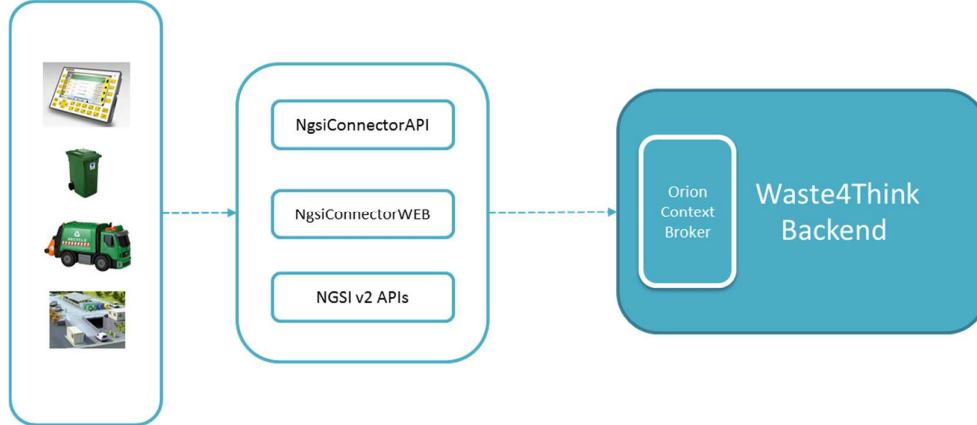


Figure 68. Uploading systems

NgsiConnectorAPI

The NGSConnectorAPI (Figure 70) is a RESTful web service that allows authorised Waste4Think users to create, update and retrieve Waste4Think context entities in the backend system compliant with Data Model defined in the Waste4Think project. In particular two POST methods have been developed for a massive uploading/updating of entities from a CSV file compliant with Waste4Think Data Model. The NGSConnectorAPI will be secured by using the security level 2 “Basic Authorization” defined in Section 4.6.4 .



At the time of writing NgsiConnectorAPI provides the following APIs:

- POST /entities: to create entities from user supplied CSV compliant with Waste4Think Data Model;
- POST /entities/update: to update entities from user supplied CSV compliant with Waste4Think Data Model;
- GET /entities: to retrieve a list of entities by specifying Service and ServicePath;
- GET /entities/{entityId}: to retrieve an entity by specifying EntityId, Service and ServicePath;
- GET /entities/type/{typeId}: to retrieve a list of Entities by specifying EntityType, Service and ServicePath;

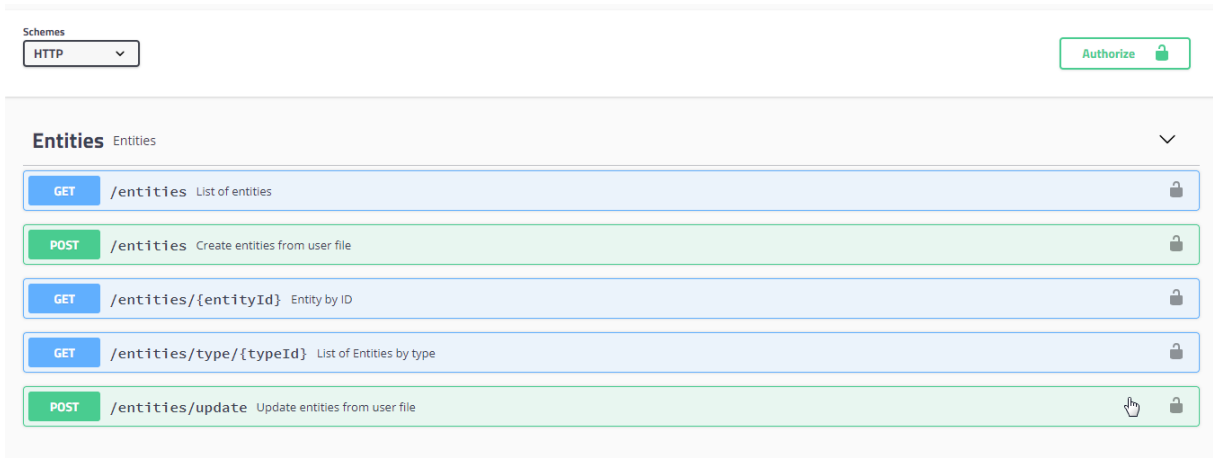


Figure 69. NgsiConnectorAPI

NGSICConnectorWEB

NGSICConnectorWEB has been created on top of the NGSICConnectorAPI in order to provide Waste4Think users a WEB User interface to execute the APIs exposed by NgsiConnectorAPI. Only authorized Waste4Think users can access NgsiConnectorWEB service.

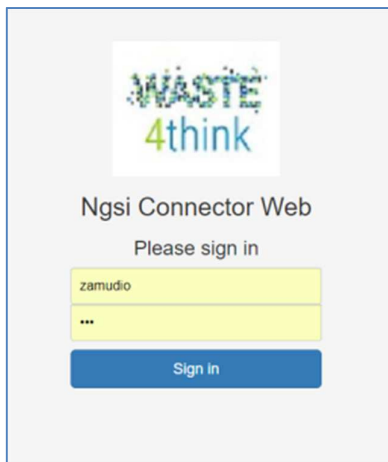


Figure 70. NgsiConnectorWEB - Login



The NgsiConnectorWEB provides the following functionalities (Figure 71):

- List entities: to retrieve a list of entities by specifying Service and ServicePath.
- Create entities (Figure 72): to create entities from user supplied CSV compliant with Waste4Think Data Model.
- Update entities: to update entities from user supplied CSV compliant with Waste4Think Data Model.

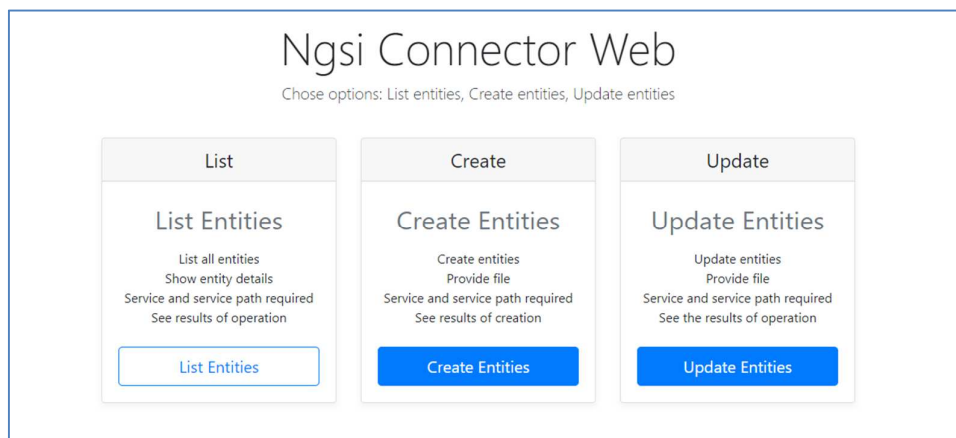


Figure 71. NgsiConnectorWEB - Home page

Figure 72. NgsiConnectorWEB – Create entities page

NGSI v2 APIs

NGSI v2 APIs handles the entire lifecycle of context information in the Waste4Think backend and more specifically in the FIWARE component Orion Context Broker, including updates, queries, registrations, and subscriptions.

Authorised Waste4Think users could execute upload or update entities, by using NGSI v2 APIs exposed by the FIWARE Orion Context Broker.



ORION Context Broker will be secured by using the security level 2 “Basic Authorization” defined in the Section 4.6.4.

Create entity

The NGSI v2 POST method http://orion_instance/v2/entities create an entity in the Orion Context Broker. The request payload is an object representing the entity to be created (Figure 73. NGSI Create Entity).

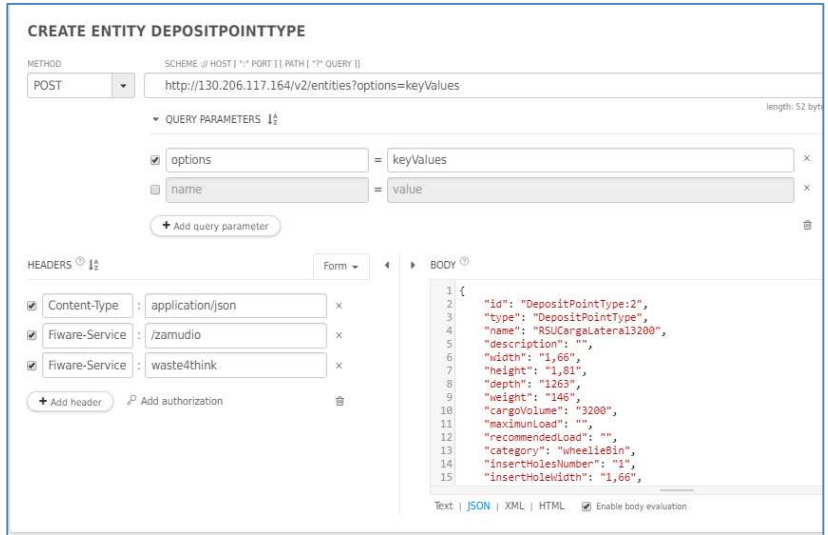


Figure 73. NGSI Create Entity

Update entity attribute

The NGSI v2 PUT method http://orion_instance/v2/entities/{entities}/attrs/{attribute} update an entity attribute in the Orion Context Broker. The request payload is an object representing the new attribute data (Figure 74).

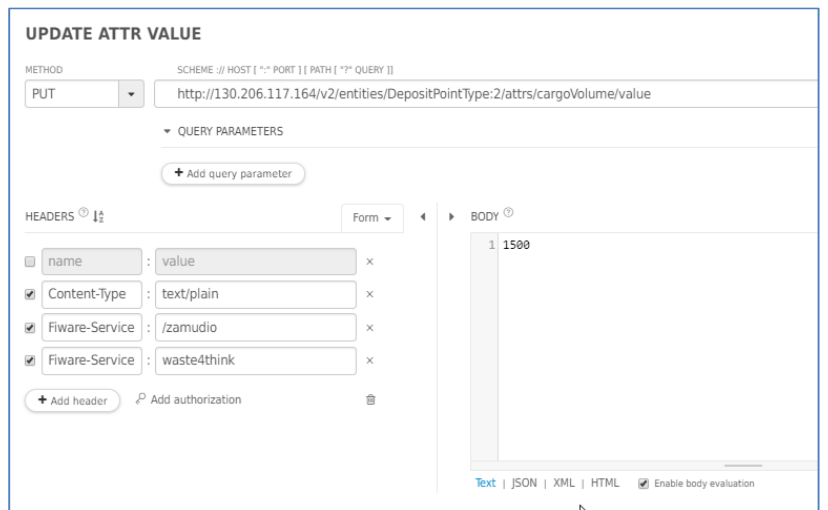


Figure 74. NGSI Update Entity attribute



9. References

1. FIWARE.org: About us - <https://www.fiware.org/about-us/>
2. Description of action – Innovation Action - NUMBER — 688995 — Waste4Think
3. GitLab Documentation <https://docs.gitlab.com/ce/README.html>
4. GitLab Community Edition <https://about.gitlab.com/features/>
5. FIWARE Context Broker GE – ORION
<https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Data.ContextBroker>
6. FIWARE User community -
http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_LAB_Terms_and_Conditions#User_Policy
7. GitLab - <https://about.gitlab.com/>
8. Open Mobile Alliance (OMA) Next Generation Services Interface (NGSI) Specification - <http://www.openmobilealliance.org/wp/>
9. Complex Event Processing (CEP) - Proactive Technology Online – <https://catalogue.fiware.org/enablers/complex-event-processing-cep-proactive-technology-online>
10. GitLab - <https://about.gitlab.com/>
11. CKAN - <https://catalogue.fiware.org/enablers/ckan>
12. Self Service Interfaces GE - Cloud Portal - <https://catalogue.fiware.org/enablers/self-service-interfaces-cloud-portal>
13. Identity Management KeyRock - <https://catalogue.fiware.org/enablers/identity-management-keyrock>
14. STH Comet - <https://fiware-sth-comet.readthedocs.io/en/latest/>
15. MySQL - <https://www.mysql.com/>
16. MongoDB - <https://www.mongodb.com/>
17. Cygnus - <https://github.com/telefonicaid/fiware-cygnus>
18. Backend Device Management IDAS - <https://catalogue.fiware.org/enablers/backend-device-management-idas>
19. NGNIX - <https://www.nginx.com/resources/admin-guide/reverse-proxy/>
20. Apache Flume - <https://flume.apache.org/>
21. FINESCE Project – <http://www.finesce.eu/>
22. CÉSAR L. ALONSO, JOSÉ LUIS MONTAÑA, JORGE PUENTE, and CRUZ ENRIQUE BORGES International Journal on Artificial Intelligence Tools 2009 18:05, 757-781, <https://doi.org/10.1142/S0218213009000391>
23. M. Giusti, J. Heintz, J.E. Morais, J. Morgenstem, L.M. Pardo, Straight-line programs in geometric elimination theory, In Journal of Pure and Applied Algebra, Volume 124, Issues 1–3, 1998, Pages 101-146, ISSN 0022-4049, [https://doi.org/10.1016/S0022-4049\(96\)00099-0](https://doi.org/10.1016/S0022-4049(96)00099-0).
24. FIWARE: Entity Service Paths - http://fiware-orion.readthedocs.io/en/master/user/service_path/
25. Docker - <https://www.docker.com/>
26. Oauth2 - <https://oauth.net/2/>



APPENDIXES OF
D2.1 Technical Documentation of R1: Operation and Management Module
 WP2 Development of Computed Assisted Integral Waste Management Systems
 FDEUSTO
 Report
 Public
 Reviewers: xx, xx, xx

Version	Date	Description of main changes	Author
1.0	27/11/2017	Frist draft	All



Annex A: Deployment methodology

A1. Git Lab Deployment

The following procedure has been used to install GitLab Community Edition on top of a Linux Centos VM deployed on the FIWARE Lab.

STEP 1 - Install and configure the necessary dependencies

```
sudo yum install curl policycoreutils openssh-server openssh-clients
sudo systemctl enable sshd
sudo systemctl start sshd
sudo yum install postfix
sudo systemctl enable postfix
sudo systemctl start postfix
sudo firewall-cmd --permanent --add-service=http
sudo systemctl reload firewalld
```

STEP 2 - Add the GitLab package server and install the package

```
curl -sS https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh |
sudo bash
sudo yum install gitlab-ce
```

STEP 3 - Configure and start GitLab

```
sudo gitlab-ctl reconfigure
```

STEP 4 - Change GitLab data repository

Create a volume on FIWARE Lab (50 GB) and attach it to the VM Dev (see)

Volumes

<input type="checkbox"/>	Name ▾	Description ▾	Size (GB) ▾	Status ▾	Attachments ▾
<input type="checkbox"/>	git-data-repo	gitlab data repository	50	in-use	1

Figure 75. GitLab Volume

```
sudo fdisk -l
sudo fdisk /dev/vdb/ → : p → : w
Create ext3 file system
sudo mkfs -t ext3 /dev/vdb
```

Create directory

```
sudo mkdir /data/gitlab/git-data
```

Mount volume

```
sudo mount /dev/vdb /data
```

Modify configuration file gitlab.rb

```
Add → git_data_dirs($default => /data/gitlab/git-data
```

```
sudo gitlab-ctl stop
```

```
sudo rsync -av /var/opt/gitlab/git-data/repositories /data/gitlab/git-data/
```

```
sudo gitlab-ctl upgrade
```

A2. W4T Core Deployment

The deployment of the software solutions within the Waste4Think project includes the installation of the software packages and its dependencies as well as the configuration of specific ports, file sources, logging, and environmental variables on the dedicated VM configured to be the backend (Section 3.4). This process can be repetitive and, if done manually, can lead to many mistakes and delays in the development. Therefore, to ease and unify the deployment process, the installation procedures will rely on the use of Docker [25].

Docker is an open source software platform to create, deploy and manage virtualized application containers on a common operating system (OS), with an ecosystem of allied tools. Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession. An example of Dockerfile is shown below:

```
FROM python:3  
  
RUN mkdir -p /var/www/ && mkdir -p /var/log/gunicorn  
  
WORKDIR /var/www/  
  
# For use pg_isready in postgres server status checking  
  
RUN apt-get update && apt-get install -y \  
    postgresql-client \  
    gettext-base \  
    && apt-get clean && rm -rf /var/lib/apt/lists/*  
  
# Prepare python  
  
COPY docker/requirements.txt .
```

```

RUN pip install --no-cache-dir -r requirements.txt

# Copy config and scripts

COPY docker/logging.yaml .

COPY docker/*.sh ./

# Copy APP

COPY ./src .

# Set configs

ENV LOG_CONFIG "logging.yaml"

ENV DEBUG False

EXPOSE 7000

ENTRYPOINT ["sh"]

CMD ["/run.sh"]

```

The deployment of the software solutions will entail two different branches: development branch and production branch. This can also be automated with Docker by creating a different YAML configuration file for each purpose.

a) Development

The development branch contains new code that has not been tested yet on a mock server that imitates the production server. Developers work on bugs and features, these get committed and pushed to a stable development branch. Then development merges into production when the code is stable enough to be deployed. An example of YAML Docker configuration file for development purposes is as follows:

```

version: '3.1'
services:
  django:
    image: '${REGISTRY}/${IMAGE}:${TAG}'
    ports:
      - ${DJANGO_HOST_PORT}:8000
    env_file:
      - ${STACK}.env
    depends_on:
      - postgres
    volumes:
      - ${DJANGO_STATIC_HOST_PATH}:/var/www/static
      - ${DJANGO_MEDIA_HOST_PATH}:/var/www/media
      - ${DJANGO_GUNICORN_LOG}:/var/log/gunicorn

  postgres:
    image: postgres:10
    ports:

```

```

- ${POSTGRES_HOST_PORT}:5432
env_file:
- ${STACK}.env
volumes:
- postgres_data:/var/lib/postgresql/data

volumes:
postgres_data:

```

b) Production

The production branch are the "live" or public version of a site once a stable build of the software has been achieved. The production server usually runs with different error logging and warning message suppression. An example of Docker YAML configuration file for production purposes is as follows:

```

version: '3.1'
services:
  django:
    image: '${REGISTRY}/${IMAGE}:${TAG}'
    ports:
      - ${DJANGO_HOST_PORT}:8000
    env_file:
      - ${STACK}.env
    depends_on:
      - postgres
    volumes:
      - ${DJANGO_STATIC_HOST_PATH}:/var/www/static
      - ${DJANGO_MEDIA_HOST_PATH}:/var/www/media
      - ${DJANGO_GUNICORN_LOG}:/var/log/gunicorn

  postgres:
    image: postgres:10
    env_file:
      - ${STACK}.env
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:

```



Annex B: NGSi API

B.1. Retrieve API Resources

This resource does not have any attributes. Instead it offers the initial API affordances in the form of the links in the JSON body. It is recommended to follow the “url” link values, Link or Location headers where applicable to retrieve resources. Instead of constructing your own URLs, to keep your client decoupled from implementation details.

Table 4. Retrieve API Resources.

REQUEST	<pre>curl -i -X GET \ 'http://130.206.120.160/v2/'</pre>
RESPONSE	<pre>{ "entities_url": "/v2/entities", "types_url": "/v2/types", "subscriptions_url": "/v2/subscriptions", "registrations_url": "/v2/registrations" }</pre>

B.2. List entities

Retrieves a list of entities that match different criteria by id, type, pattern matching (either id or type) and/or those which match a query or geographical query (see Simple Query Language and Geographical Queries).

Table 5. List of entities.

REQUEST	<pre>curl -i -X GET \ 'http://130.206.120.160/v2/entities/'</pre>
RESPONSE	<pre>[{ "id": "WasteContainer:Zamudio0001", "type": "WasteContainer", "accessibility": { "type": "Text", "value": "Mala --peligrosidad al estar cerca de la carretera, etc--", "metadata": { } }, }, "anchor": { "type": "Text", "value": "Sin anclaje --acera--", "metadata": { } } ]</pre>



Table 6. List of entities by id and type.

REQUEST	<pre>curl -i -X GET \ http://130.206.120.160/v2/entities?type=WasteContainer&id=WasteContainer%3AZamudio0001'</pre>
RESPONSE	<pre>{ { "id": "WasteContainer:Zamudio0001", "type": "WasteContainer", "accessibility": { "type": "Text", "value": "Mala --peligrosidad al estar cerca de la carretera, etc--", "metadata": { } }, "anchor": { "type": "Text", "value": "Sin anclaje --acera--", "metadata": { } } } }</pre>

B.3. Create an entity

The payload is an object representing the entity to be created.

Table 7. Create an entity.

REQUEST	<pre>curl -i -X POST \ -H "Content-Type:application/json" \ -d \ '{ "id": "wastecontainer1", "type": "WasteContainer", "refWasteContainerModel": "wastecontainermodel1", "refWasteContainerIsle": "wastecontainerisle1", "serialNumber": "ab56kjl", "location": { "type": "Point", "coordinates": [-3.164485591715449, 40.62785133667262] }, "fillingLevel" : 0.4, "dateLastEmptying": "2016-06-21T15:05:59.408Z", "dateNextActuation": "2016-06-28", "status": "ok", "category": ["underground"], "refDevice": ["device-Fleming:12a:1"] }' \ 'http://130.206.120.160/v2/entities?options=keyValues'</pre>
RESPONSE	201 CREATED



B.4. Update an attribute value

The request payload is the new attribute value.

The payload MIME type in the request is specified in the Content-Type HTTP header.

Table 8. Update an entity.

REQUEST	<pre>curl -i -X PUT \ -H "Content-Type:text/plain" \ -d \ '0.565' \ 'http://130.206.120.160:1026/v2/entities/wastecontainer:seveso:0001/attrs/fillingLevel/value'</pre>
RESPONSE	204 No Content

Create a subscription

The Context Consumers can also get context information in an asynchronous way using notifications. The Context Broker GE is “programmed” to send notifications upon given conditions (specified in the subscription request).

A subscription is represented by a JSON object with the following fields:

- *id*: Subscription unique identifier. Automatically created at creation time.
- *description* (optional): A free text used by the client to describe the subscription.
- *subject*: An object that describes the subject of the subscription.
- *notification*: An object that describes the notification to send when the subscription is triggered.
- *expires*: Subscription expiration date in ISO8601 format. Permanent subscriptions must omit this field.
- *status*: Either active (for active subscriptions) or inactive (for inactive subscriptions). If this field is not provided at subscription creation time, new subscriptions are created with the active status, which can be changed by clients afterwards. For expired subscriptions, this attribute is set to expired (no matter if the client updates it to active/inactive).
- *throttling*: Minimal period of time in seconds which must elapse between two consecutive notifications. It is optional.

Table 9. Create subscription.

REQUEST	<pre>curl -i -X POST \ -H "Content-Type:application/json" \ -d \ '{</pre>
---------	---



	<pre>"description": "fillingLevel Subscription for the WasteContainer entity", "subject": { "entities": [{ "idPattern": ".*", "type": "WasteContainer" }], "condition": { "attrs": ["fillingLevel"] } }, "notification": { "http": { "url": "http://localhost:5050/notify" } }, "expires": "2017-03-29T14:00:00.00Z", "throttling": 5 } \ 'http://130.206.120.160/v2/subscriptions'</pre>
RESPONSE	201 CREATED



Annex C: Cascais Lock System

SOTKON
SOTKIS ACCESS CONTROL

- SMART TAG QUADRO -

1. SYSTEM OFFERED

Sotkon with this offer intends to propose a solution that provides the control over the residents regarding the waste deposition equipments available in the development. The technical proposal that is presented in detail below has been developed to provide a solution consisting of a control device of innovative access, developed on the basis of solid knowledge acquired during many years of experience gained with the application of the technology to control the rejection of waste, of which is undisputed pioneer this solution.

Sotkon, therefore, presents this offer with the certainty that is proposing a device that represents an effective response to the demands of the Hallsville Quarter Managing Agent, aimed at maximum efficiency of urban waste collection and a high level of service to the user/client.

The main services offered can be summarized as follows:

- A. Control devices, integrated in the intake receptacles, for the transfer of waste by authorized users;
- B. FOBS to access the devices;
- C. Full preventative and corrective maintenance service;
- D. Full access to web based server and platform, where the devices can be managed and controlled.



References

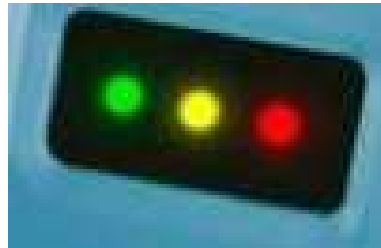
This technology is present on the Italian territory with 6.753 machines divided into 181 municipalities for a total of 326,000 users, equivalent to more than 17,000,000 of waste depositions per year. The first volume control devices installed on the Italian territory (Trentino) were placed at the site in 1999 and are still in perfect working order.



Features of the system

COMPONENTS:

- Smart TAG (with RFID and regions defined)
- 3 LED Display
- Tag slot mechanism
- Battery
- Solar Panels
- Electro-Mechanic Lock and electronic board
- ISM Communication box



OPENING SYSTEM

The opening of the device is simple and intuitive as specified below, and it can take place only when the software recognizes the fob permitted in a given area.

DEPOSITION OF THE WASTE

As soon as the user enters the key in the slot located at the right side of the device by means of a REED contact system is automatically activated. With the aim of reducing the energy required, the system always stays in a disabled state. If the software recognizes user (key) as a "enabled user" given the consent to the opening of the device. The movement of the system is released, and the user can move it opening the lid manually, giving access to deposit the bag with waste. The user completes the operation by closing the lid. The key is released the mechanism and may be removed by the user.

Availability of the system

The device is activated immediately when inserting the FOB. The yellow light will turn on for 1 second while the FOB is verified, the Green LED will light up and the lid can be easily and quickly opened.

Instructions on how to use the system

Detailed information about the status of the device, as well as useful information for the correct usage is provided through three LEDs on the front of the device. The information provided is simple, direct, not requiring interpretation skills and they have been designed for those who do not have any particular familiarity with computerized electronic systems. Instead of written messages, it was considered appropriate to opt for the simple language of colors of light that we all know and are accustomed to seeing and traffic use: red for stop, green to indicate the approval and the yellows to give advance notice of the imminence of a state change. The device uses LEDs with these three colors to reproduce the information on the status of access.

Through the three LEDs the following messages will be provided:

- messages of support for the implementation of the deposition;
- Indication enabling the provision of a particular user (key);
- Indication of errors or anomalies in the system; Any temporary inhibition (set in a certain period of time for security or other requirements)

The FOB is inserted below the LED display:

Checking the FOB:



Yellow LED lights for 1 second

Authorization, start the deposition:



Green LED lights permanently

The deposition was made correctly and the FOB can be removed:



Green LED flashes during 15 seconds.

If the FOB is not authorized to be used in a specific container:



Red LED lights permanently.

If the device is out of service:



Red LED flashes

5 steps to use the device:

- ❑ Insert the FOB
- ❑ Open the device
- ❑ Deposit the waste
- ❑ Close the device
- ❑ Remove the FOB

SPECIFICATIONS of the FOBS



The key provided to the user is the only mode of authorized access to the access control system. It consists of a RFID transponder "Smart Tag" that is not powered (passive) having the working frequency equal to 13.56 MHz.

- Material: ABS
- Weight: 5,51g
- Bar code on the outer surface: code128

- Degree of protection: IP68
- Mechanical stability: 500 N (10 s static)
- Resistance to water spray pressure: 45 bar (10h)
- Dimensions: 54,5mm x 30 mm x 4 mm
- Working frequency: 13.56 MHz
- Compliance: ISO / IEC 15693-2-3; ISO / IEC 18000-3 (reader compatible with ISO 15695)
- Total storage capacity: 40 Byte
- Identification code to 8Bytes: unique, irreproducible, manipulated or not copy able, imprinted with laser procedure on the outer surface of the key to reading through light pen
- Operating temperature: -20 ° C + 60 ° C

ELECTRO-MECHANIC LOCK

The electromechanical lock has the following main features:

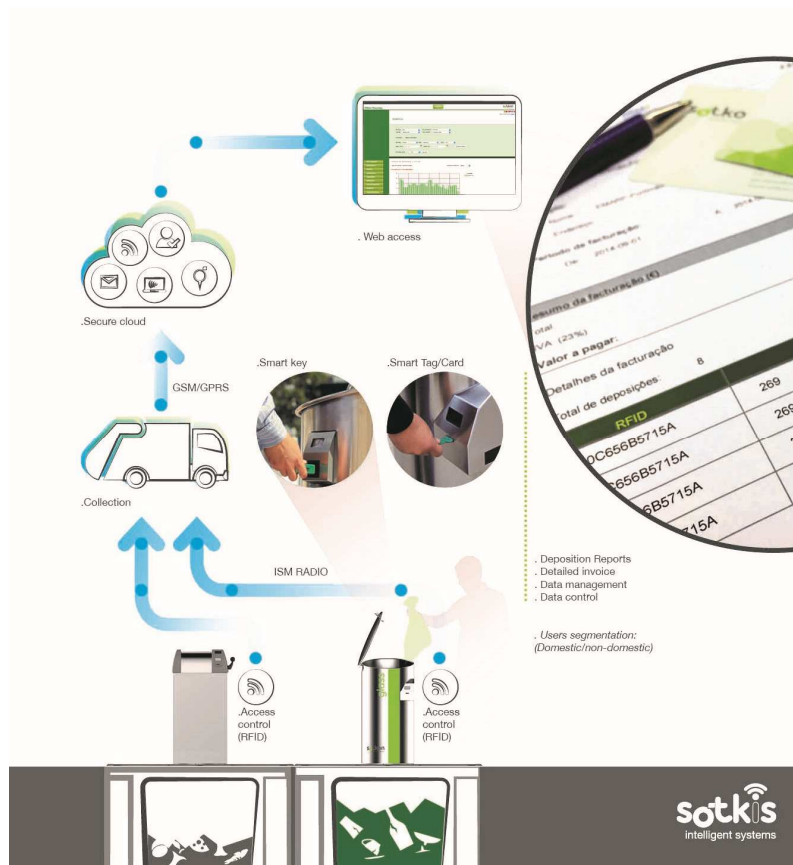
- Power supply: 3.0 V;
- Standby power consumption: 13 µA
- 2 MB of external memory accounting data with a high degree of security (storage capacity capable of recording and storing 50,000 contributions equal to a period of continuous use average of about 15 years.)
- Real Time Clock (RTC), to record the date and time (year, month, day, hour, minute, second) run by the user presentation;
- 3 LED high contrast to the formation of the messages and user guide for maintenance personnel;
- Internal code unique serial ID 6 alphanumeric characters;
- Up to 3 internal area codes to 9 alphanumeric characters can be adjusted to the possible definition of the group / users who can access the service in a special container
- Possibility of setting times and times when the grant is inhibited (eg, special events, etc.)
- Ability to inhibit access to the lost accidentally keys (400 keys / device)
- Software can be updated at any time without dismantling of electronic on-site and remote
- Operating Temperature: -20 ° C + 60 ° C

SAFETY FROM INFRINGEMENT

The code of each key is unique and irreproducible within the system implemented. Security against manipulation of data is typical for a key (region code, key type, etc..). The cloning of the keys themselves is ensured by a specific algorithm that verifies the authenticity of each bit and / or byte in order to any changes that occur.

The key, given to each authorized user, allows access to the system in order the deposition of the waste. The association between the key and the user is performed using procedures that categorically exclude the repetition of codes and or assignments of the same key to multiple users.

DATA TRANSFER – OBC (On Board Computer)



The data is downloaded automatically from any device to any emptying of the container using the ISM (Industrial Scientific and medical) short distance transmission protocol on free frequency band. ISM technology is characterized by high standards of security against intruders and is completely independent of the cover of the mobile phone network where the application should be since it uses the 2.4 Ghz frequency band.

This solution is distinguished by its low cost, since it is simply a single SIM module data receiver/transmitter that requires a SIM for any data transfer device. But another huge benefit determined by the use of this solution lies in the fact that it is necessary, for each transmission, amounts of electricity well 9 times lower than those required by a GSM transmission. By adopting this type of data transmission great attention is paid to two aspects which, notoriously, were developed because of the required contraction of transmissions via GSM network:

- **the costs of transmission**
- **the energy that this type of transmission needs.**

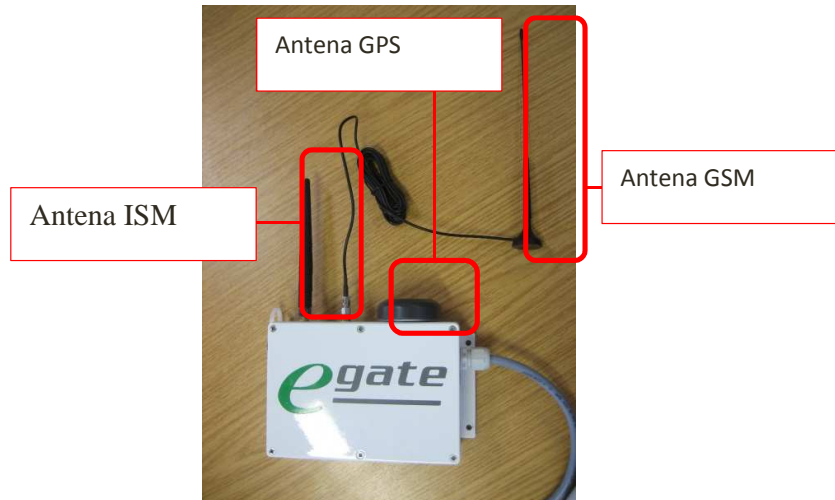
Data transfers will be stored in the electronic time control device and layout will be automatically transmitted to the vehicle making sure to empty the container and the central server.

The operation of data transmission by each specific control of the receptor-transmisor module installed in the vehicle that carries the collection will be fully automatic because the device will be able to recognize the presence of the medium and a transmitter receiver module that has been identified and has been recognized. Data collected from the module will then be transmitted receiver/transmitter on the same mobile network with GSM or GPRS standard transmission to the central data server. The data will be collected automatically by the server using software that will make the necessary process. The reception and transmission system will be able to ensure a two-way communication between the device and vice versa between medium and device.

The transfer of data collected through electronic system in question can be viewed by clicking on the central data server and then imported and viewed on PC in standard formats (for example, txt, csv, xls, etc.).

The system of two-way communication between the web server and the device described is patented and therefore cannot be used by other manufacturers of control of personal devices.

The following image shows the functional bidirectional data transmission system diagram.



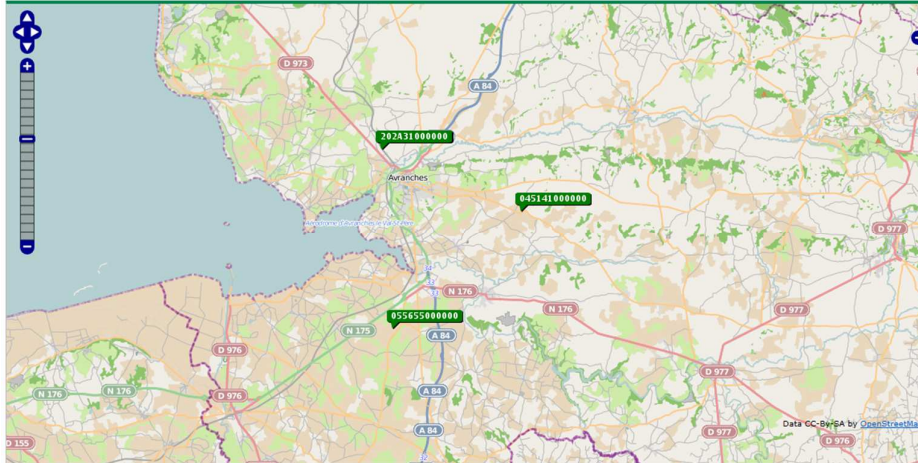
Features of the SOFTWARE needed for the data processing

Web based data portal

For the purpose of detection, processing and data management of the individual components of the system is mandatory the use of a Web portal. Through the Web portal you can also request the assignment of the keys and change the parameters of individual devices and data acquisition tools. The portal is generally based on standards and needs only a browser with Java Script and cookies-support to access it.

The portal offers the following characteristics:

- Device management;
- Management of keys; -
- Management of data transfers;
- Statistical analysis.



Devices management

It is possible to select and assign different areas by user within the territory through the web platform.

egate ... it counts.

Regioni Calotte Chiavi utente chiave Amministrazione

Filtro:

Regione: 999990005 EMZ_Test

ID	IMEI	Codice regione	Regione	Conferimenti	Conferimenti (totali)	Call service	Data readout	Ricarica batteria
FFFFF0000000		999990005	EMZ_Test	0 / 85	5		1-feb-2012 15:38:34	<input type="text" value="7"/>
FFFFF0000000		999990005	EMZ_Test	0 / 85	12		1-feb-2012 14:47:11	<input type="text" value="7"/>
FFFFF0000000		999990005	EMZ_Test	0 / 90	2		24-nov-2011 13:33:19	<input type="text" value="7"/>
FFFFF0000000		999990005	EMZ_Test	0 / 80	8		2-mag-2012 11:47:30	<input type="text" value="7"/>

+

Details viewing

The details view is a tool to perform checks and optimizations of the entire operating costs. With a maximum of three clicks you can choose and represented in the mode summary any information that you want. The information that appears always can be exported directly into MS Excel and ASCII.

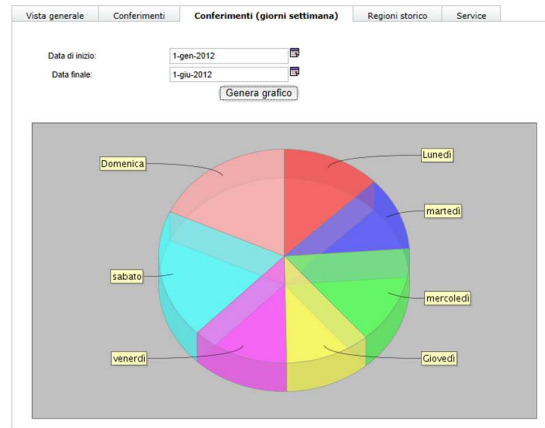
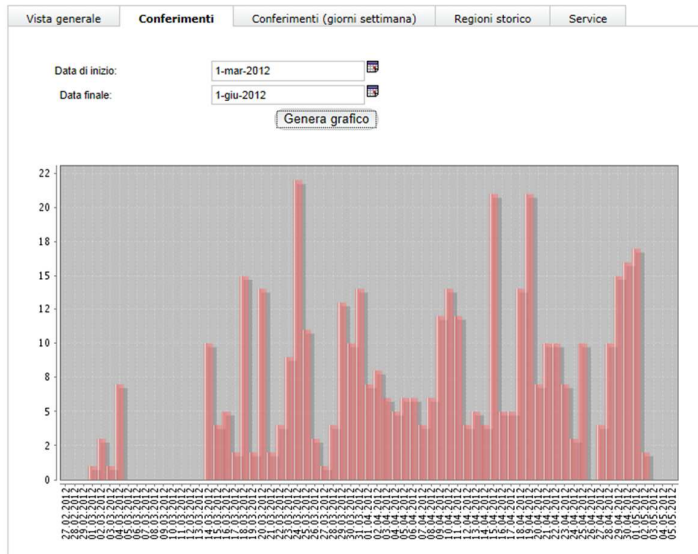
Vista generale	Conferimenti	Conferimenti (giorni settimana)	Regioni storico	Service
ID:	<input type="text" value="FFFFED000000"/>			
Alias:	<input type="text" value="EMZ_Test"/>			
Ricarica batteria:	<input type="text" value="7"/>			
Conferimenti:	0 / 85			
Conferimenti (totali):	5			
Data readout:	1-feb-2012 15:38:34		Nessuna immagine disponibile	
IMEI:				
Ultima connessione:				
Segnale GSM:				
Cambio regione				
Regione:	<input type="text" value="999990005 EMZ_Test"/>			
Commento:	<input type="text"/>			
<input type="button" value="Modifica immagine"/> <input type="button" value="Indietro"/> <input type="button" value="Salva"/>				

Data transfer and export screen

You can see easily the details of the depositions of each device by clicking on the device which is displayed on the map or by clicking on the ID code of the device that appears in the corresponding submenu. In addition, the data and configuration of each device system can be displayed:

- Device and its parameters;
- Containers
- Alarms
- List of users and black list
- Time of cancelation of the fobs
- Systems and additional parameters

All the relevant characteristics of the elements in question are included in the software, displayed and updated. In the case of a device such as: internal identification code, region code, battery fill level limit, inhibition time LED configuration, geographic location, or even a picture of the position of the box. If necessary, the configuration parameters can be reviewed and modified. All data can be exported in CSV or XLS format.



Archive of registers

All big changes made to a device (setting, by moving into territory etc.) are stored in a database log file so that any changes can be documented at the time.

FOB management

The user has a screen of keys in the system. Each key is identified by a unique code of 12 alphanumeric characters that represents permission to access a certain number of devices. The depositions made by a user will be associated to the territory defined for that same key. Includes a table that summarizes the main information of each individual key (internal code, area, etc.). Clicking on an icon allows seeing more detailed information.

The details view allows to:

1. Change the parameters of each key;
2. Change the status (active, inactive, blocked, destroyed)

3. Change the regions of each system;
 4. Configuracion of the system;
- All operations in each FOB is registered and it is shown in the view of key details.

The screenshot shows the egate system interface with the 'Regioni' menu selected. A table displays notification details:

ID	Calotta	Data di notifica	Tipo	stato	Descrizione	notificato da	Tecnico
00000012	000000000000	24-apr-2012 15:02:04	test	test		info@emz-kanauer.com	info@emz-kanauer.com
00000011	FFFFF0000000	24-apr-2012 7:54:04	test	test		info@emz-kanauer.com	info@emz-kanauer.com

Additional interface elements include a search filter, a sidebar with 'Vista generale', 'Importazione', 'Service', and 'Importare lettura', and a footer with login information and version details.

The screenshot shows the 'Service' view for a specific notification. The table below contains the details:

ID	Data di notifica	Tipo	stato	Descrizione
00000011	24-apr-2012 7:54:04	test	test	

Below the table is a 'Descrizione:' input field and an 'aggiungere' button.

The screenshot shows the 'Regioni' menu selected, displaying a list of regions. The table below contains the data:

ID	Codice regione	Nome della regione	Conferimenti	Stato	utente chiave
07F19E040000	99990320	Nozay Test	0	Inattivo	
0E489F040000	99990320	Nozay Test	0	Inattivo	
12399F040000	99990320	Nozay Test	0	Inattivo	
13399F040000	99990320	Nozay Test	0	Inattivo	
13759F040000	99990320	Nozay Test	0	Inattivo	
1C589F040000	99990320	Nozay Test	0	Inattivo	
214A9F040000	99990320	Nozay Test	0	Inattivo	
23649F040000	99990320	Nozay Test	0	Inattivo	
28829F040000	99990320	Nozay Test	0	Inattivo	
29299F040000	99990320	Nozay Test	0	Inattivo	
2D289F040000	99990320	Nozay Test	0	Inattivo	
30EA9E040000	99990320	Nozay Test	0	Inattivo	
30EF9E040000	99990320	Nozay Test	0	Inattivo	
31579F040000	99990320	Nozay Test	0	Inattivo	
3E249F040000	99990320	Nozay Test	0	Inattivo	
3EEF9E040000	99990320	Nozay Test	0	Inattivo	
42379F040000	99990320	Nozay Test	0	Inattivo	
44D29F040000	99990320	Nozay Test	0	Inattivo	
47489F040000	99990320	Nozay Test	0	Inattivo	
53649F040000	99990320	Nozay Test	0	Inattivo	
53E09F040000	99990320	Nozay Test	0	Inattivo	
541C9F040000	99990320	Nozay Test	0	Inattivo	
67529F040000	99990320	Nozay Test	0	Inattivo	
690E9A040000	99990320	Nozay Test	0	Inattivo	
6E5D9F040000	99990320	Nozay Test	0	Inattivo	
6F929F040000	99990320	Nozay Test	0	Inattivo	
70E49E040000	99990320	Nozay Test	0	Inattivo	
74829F040000	99990320	Nozay Test	0	Inattivo	
7E919E040000	00000000	Nozay Test	0	Inattivo	

Registration archive

All main exchanges made to the FOBs are kept in an archive inside the data base.

Access to the data resulting from the depositions

As mentioned above the access to data on the depositions made to a single device can be done in 2 different ways, both characterized by the extreme ease of use. The data can be filtered by different criteria:

1. Shell Code;
2. FOB code;
3. Date.

Data export

Deposition lists can be exported in desired formats. Each entry exported is marked in order to avoid double exports.

Tasks related to data management

In regards to the Exchange of data stored in the device, the following tasks will be executed:

- 1) Download of all the depositions stored routines;
- 2) Transference of the data to data base;
- 3) Creation of a data base containing the identification number of all the devices and unique codes of the FOBs;
- 4) All that is necessary to assure that all electronic information are well converted and are compatible with agreed formats;
- 5) Verification of data inconsistency;
- 6) Management of shared black lists;
- 7) All collected data will be stored in devices for a minimum period of 12 months and are available to the client.

Data property

All the gathered and generated data through the Access control devices are sole property of the client.

AUTONOMY OF THE SYSTEM

The system is powered by an internal battery that has the following characteristics: tipo Li-SoCl₂ (cloruro de tionilo Li)

- voltage: 3 .6V
- 19000 m
- wheigt: 100 g
- size of cylindrical shape: 33.2 * 58, 7 mm
- Estimated duration: 2,5 years
- Temperature ranges: -55 ° + 85 ° C /C
- Conformity with European Directive 2006/66/CE

Certifications

The system of Access control proposed in the present offer is:

- CE branded;
- In conformity with Directive 2001/95/CE, regarding general safety of products;
- In conformity with Directive 99/5/CE regarding radioelectric and telecommunication devices
- In conformity with Directive 2002/95/CE (RoHS) regarding the restriction on the use of dangerous materials
- In conformity with UNI EN 12574 regarding the technical and mechanical features of containers

GENERAL CONDITIONS

Warranty

Devices, access keys, and any other device that comes as a result of this procedure will be guaranteed altogether for a period of 1 year from the date of delivery.

The applicable conditions to validate the referred warranty are:

Maintenance revision (every 6 months)

Semi-annual review of maintenance will consist of the implementation of the controls and the following activities:

Verification of operation of the device

Do a "reset" with the master key

Components control

- **Deposit and opening mechanism**
Integrity and function of the structure
Internal components
Locks
Fixation of the system
Signs of corrosion
- **mechanics**
General operation
Opening and closing adjustments
FOB slot
Internal mechanisms

Electric and electronic components

- LEDs
- Time and date settings
- Errors memory
- Seal integrity

Malfunctions repairation

All the malfunctions due to ruptures or safety and usage problems will be repaired.

Restoring functionality

When a device does not operate correctly the system can be temporally usable without access control, in order to allow no service disruptions. It is assured an intervention in 1 week from recognised notification.

SOFTWARE WARRANTY

During the first year, it is warranted all problems that may occur with the operation of the software, that affects the normal operation of the system, and all the available updates to the software will be provided.

To validate this warranty it is specified:

Actions of ordinary and extraordinary maintenance of all software installed on devices and data acquisition systems will be scheduled and organized to allow the correct operation of the same.



Annex D: Halandri Weight Sensor

Technical Datasheet

TWID-20-SRE

Weighing & RFID data controller

The TWID-20 controller is part of the broader family of communications & signal-processing products manufactured by Trinity Systems. It is designed and customized for use in refuse-collection vehicles and applications that require automated weighing & electronic identification of waste containers (e.g. Intelligent Waste Management applications). The controller includes an integrated RFID reader and external ports for connecting to weighing sensors and the vehicle's on-board telematics unit (if required for sending real-time data to command & control center over 3G/GPRS networks).

- ▷ **Designed for heavy-duty vehicles**
- ▷ **Increased electrical protection & EM immunity**
- ▷ **Serial communication with external devices (e.g. telematics unit)**
- ▷ **Digital I/Os & Analog sensor inputs**
- ▷ **Real Time Clock / 16 MByte Flash memory**
- ▷ **3-axis accelerometer**
- ▷ **Includes customized weighing and diagnostics application firmware**
- ▷ **Embedded UHF Gen2 RFID reader** (reading/encoding tags)


Technical Specifications

Power Supply	10 – 36V DC
Power consumption	400 mA @ 24V (max @ RFID transmit)
Communications	2x RS232 serial ports (supporting customized protocols)
RFID reader	UHF 865-870MHz (EPC Gen2 / ISO 18000-6C) / range 9m / RF out +30dBm
I/O Ports	<ul style="list-style-type: none"> - 4 optically isolated general purpose digital inputs - 4 optically isolated general purpose digital outputs - 6 analog inputs (10bit A/D resolution) - Auxiliary Power Supply
Connectors – Indicators	2x SMA coaxial (female) for external RFID antennas 2x D9-SUB (male, female) 2x Molex Micro-Fit, 10p 1x Molex Mini-Fit, 4p 1x 3-Color LED
Dimensions / Weight	180 x 112 x 32 mm / 450 gr
Operating Temperatures	-25° C ... +75° C
Enclosure	Anodized aluminum (highly durable)
Approvals	CE, RoHS, ETSI EN 302 208/EN 300 220/EN 301 489-1 & 3/EN 60950-1

TWID-20-SRE Rev.4-EN / 28-JUL-2016

specifications are subject to change without any prior notice



Annex E: Community Composting Plant Sensors

21.6. Smart Agriculture

The Smart Agriculture models allow to monitor multiple environmental parameters involving a wide range of applications. It has been provided with sensors for air and soil temperature and humidity, solar visible radiation, wind speed and direction, rainfall, atmospheric pressure, etc.

The main applications for this Waspote Plug & Sense! model are precision agriculture, irrigation systems, greenhouses, weather stations, etc. Refer to [Libelium website](#) for more information.

Two variants are possible for this model, normal and PRO. Next section describes each configuration in detail.



Figure: Smart Agriculture Waspote Plug & Sense! model

21.6.1. Normal

Sensor sockets are configured as shown in the figure below.

Sensor Socket	Sensor probes allowed for each sensor socket	
	Parameter	Reference
A	Weather Station WS-3000 (anemometer + wind vane + pluviometer)	9256-P
B	Soil Moisture 1	9248-P, 9324-P, 9323-P
C	Soil Moisture 3	9248-P, 9324-P, 9323-P
D	Soil Temperature	86949-P
	Temperature + Humidity + Pressure	9370-P
	Luminosity (Luxes accuracy)	9325-P
	Ultrasound (distance measurement)	9246-P
E	Leaf Wetness	9249-P
	Soil Moisture 2	9248-P, 9324-P, 9323-P
F (digital bus)	Temperature + Humidity + Pressure	9370-P
	Luminosity (Luxes accuracy)	9325-P
	Ultrasound (distance measurement)	9246-P

Figure: Sensor sockets configuration for Smart Agriculture model

Note: For more technical information about each sensor probe go to the [Development section](#) on the Libelium website.

21.6.2. PRO

Sensor sockets are configured as shown in the figure below.

Sensor Socket	Sensor probes allowed for each sensor socket	
	Parameter	Reference
A	Weather Station WS-3000 (anemometer + wind vane + pluviometer)	9256-P
B	Soil Moisture 1	9248-P, 9324-P, 9323-P
	Solar Radiation (PAR)	9251-P
	Ultraviolet Radiation	9257-P
C	Soil Moisture 3	9248-P, 9324-P, 9323-P
	Dendrometers	9252-P, 9253-P, 9254-P
D (digital bus)	Soil Temperature (Pt-1000)	9255-P
	Temperature + Humidity + Pressure	9370-P
	Luminosity (Luxes accuracy)	9325-P
	Ultrasound (distance measurement)	9246-P
E	Leaf Wetness	9249-P
	Soil Moisture 2	9248-P, 9324-P, 9323-P
F (digital bus)	Temperature + Humidity + Pressure	9370-P
	Luminosity (Luxes accuracy)	9325-P
	Ultrasound (distance measurement)	9246-P

Figure: Sensor sockets configuration for Smart Agriculture PRO model

* Ask Libelium [Sales Department](#) for more information.

Note: For more technical information about each sensor probe go to the [Development section](#) on the Libelium website.

4.2. Temperature, Humidity and Pressure Sensor Probe

The BME280 is a digital temperature, humidity and atmospheric pressure sensor developed by Bosch Sensortec.

Specifications

Electrical characteristics

Supply voltage: 3.3 V

Sleep current typical: 0.1 μ A

Sleep current maximum: 0.3 μ A

Temperature sensor

Operational range: -40 ~ +85 $^{\circ}$ C

Full accuracy range: 0 ~ +65 $^{\circ}$ C

Accuracy: ± 1 $^{\circ}$ C (range 0 $^{\circ}$ C ~ +65 $^{\circ}$ C)

Response time: 1.65 seconds (63% response from +30 to +125 $^{\circ}$ C).

Typical consumption: 1 μ A measuring

Humidity sensor

Measurement range: 0 ~ 100% of relative humidity (for temperatures < 0 $^{\circ}$ C and > 60 $^{\circ}$ C see figure below)

Accuracy: < $\pm 3\%$ RH (at 25 $^{\circ}$ C, range 20 ~ 80%)

Hysteresis: $\pm 1\%$ RH

Operating temperature: -40 ~ +85 $^{\circ}$ C

Response time (63% of step 90% to 0% or 0% to 90%): 1 second

Typical consumption: 1.8 μ A measuring

Maximum consumption: 2.8 μ A measuring



Figure: Image of the Temperature, Humidity and Pressure Sensor Probe

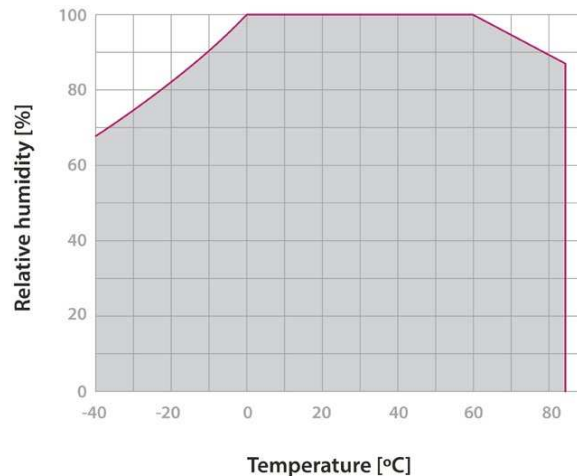


Figure: Humidity sensor operating range

Pressure sensor

Measurement range: 30 ~ 110 kPa

Operational temperature range: -40 ~ +85 $^{\circ}$ C

Full accuracy temperature range: 0 ~ +65 $^{\circ}$ C

Absolute accuracy: ± 0.1 kPa (0 ~ 65 $^{\circ}$ C)

Typical consumption: 2.8 μ A measuring

Maximum consumption: 4.2 μ A measuring



Annex F: Zamudio Citizen Card

MF1S70yyX

MIFARE Classic 4K - Mainstream contactless smart card IC
for fast and easy solution development

Rev. 3.0 — 2 May 2011
196430

Product data sheet
COMPANY PUBLIC

1. General description

NXP Semiconductors has developed the MIFARE Classic MF1S70yyX to be used in a contactless smart card according to ISO/IEC 14443 Type A.

The MIFARE Classic 4K MF1S70yyX IC is used in applications like public transport ticketing and can also be used for various other applications.

1.1 Anticollision

An intelligent anticollision function allows to operate more than one card in the field simultaneously. The anticollision algorithm selects each card individually and ensures that the execution of a transaction with a selected card is performed correctly without interference from another card in the field.

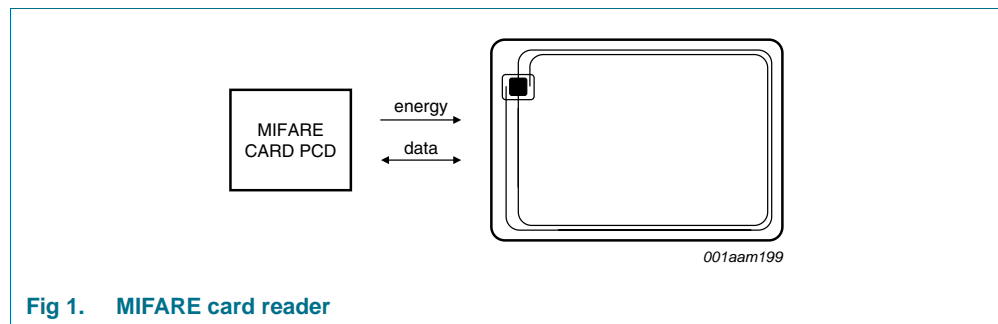


Fig 1. MIFARE card reader

1.2 Simple integration and user convenience

The MF1S70yyX is designed for simple integration and user convenience which allows complete ticketing transactions to be handled in less than 100 ms.

1.3 Security

- Manufacturer programmed 7-byte UID or 4-byte NUID identifier for each device
- Random ID support
- Mutual three pass authentication (ISO/IEC DIS 9798-2)
- Individual set of two keys per sector to support multi-application with key hierarchy



1.4 Delivery options

- 7-byte UID, 4-byte NUID
- bumped die on wafer
- MOA4 and MOA8 contactless module

2. Features and benefits

- Contactless transmission of data and supply energy
- Operating frequency of 13.56 MHz
- Data integrity of 16-bit CRC, parity, bit coding, bit counting
- Typical ticketing transaction time of < 100 ms (including backup management)
- Operating distance up to 100 mm depending on antenna geometry and reader configuration
- Data transfer of 106 kbit/s
- Anticollision

2.1 EEPROM

- 4 kB, organized in 32 sectors of 4 blocks and 8 sectors of 16 blocks (one block consists of 16 byte)
- Data retention time of 10 years
- User definable access conditions for each memory block
- Write endurance 100000 cycles

3. Applications

- Public transportation
- Electronic toll collection
- School and campus cards
- Internet cafés
- Access management
- Car parking
- Employee cards
- Loyalty

4. Quick reference data

Table 1. Quick reference data

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
C_i	input capacitance	[1]	14.9	16.9	19.0	pF
f_i	input frequency		-	13.56	-	MHz
EEPROM characteristics						
t_{ret}	retention time	$T_{amb} = 22\text{ }^{\circ}\text{C}$	10	-	-	year
$N_{Endu(W)}$	write endurance	$T_{amb} = 22\text{ }^{\circ}\text{C}$	100000	200000	-	cycle

[1] LCR meter, $T_{amb} = 22\text{ }^{\circ}\text{C}$, $f_i = 13.56\text{ MHz}$, 2 V RMS.

5. Ordering information

Table 2. Ordering information

Type number	Package		Version
	Name	Description	
MF1S7001XDUD	FFC Bump	8 inch wafer, 120 μm thickness, on film frame carrier, electronic fail die marking according to SECS-II format), Au bumps, 7-byte UID	-
MF1S7001XDUF	FFC Bump	8 inch wafer, 75 μm thickness, on film frame carrier, electronic fail die marking according to SECS-II format), Au bumps, 7-byte UID	-
MF1S7000XDA4	MOA4	plastic leadless module carrier package; 35 mm wide tape, 7-byte UID	SOT500-2
MF1S7000XDA8	MOA8	plastic leadless module carrier package; 35 mm wide tape, 7-byte UID	SOT500-4
MF1S7031XDUD	FFC Bump	8 inch wafer, 120 μm thickness, on film frame carrier, electronic fail die marking according to SECS-II format), Au bumps, 4-byte non-unique ID	-
MF1S7031XDUF	FFC Bump	8 inch wafer, 75 μm thickness, on film frame carrier, electronic fail die marking according to SECS-II format), Au bumps, 4-byte non-unique ID	-
MF1S7030XDA4	MOA4	plastic leadless module carrier package; 35 mm wide tape, 4-byte non-unique ID	SOT500-2
MF1S7030XDA8	MOA8	plastic leadless module carrier package; 35 mm wide tape, 4-byte non-unique ID	SOT500-4

6. Block diagram

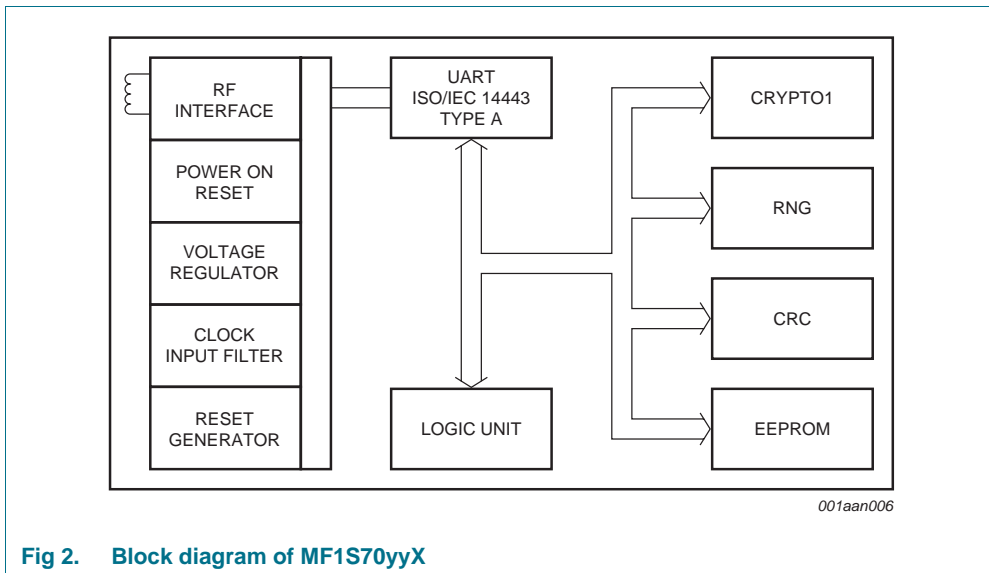


Fig 2. Block diagram of MF1S70yyX

7. Pinning information

7.1 Pinning

The pinning for the MF1S70yyXDAX is shown as an example in [Figure 3](#) for the MOA4 contactless module. For the contactless module MOA8, the pinning is analogous and not explicitly shown.

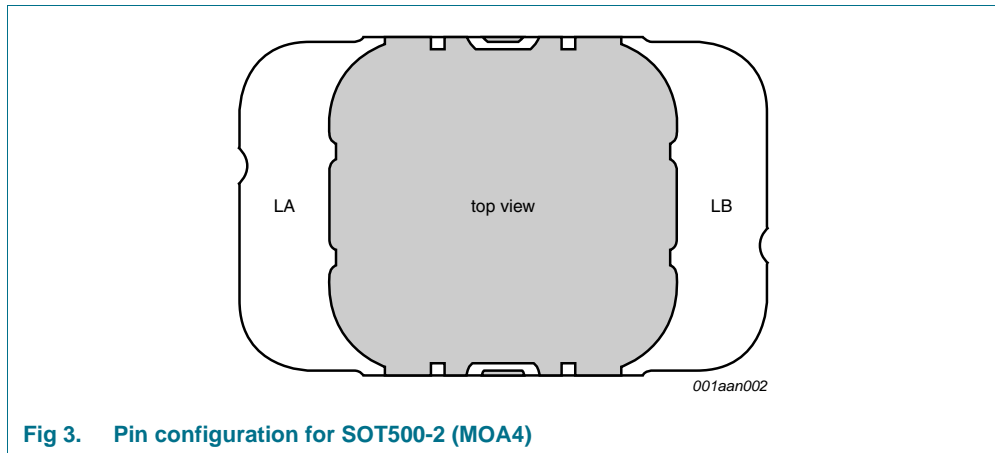


Fig 3. Pin configuration for SOT500-2 (MOA4)

Table 3. Pin allocation table

Pin	Symbol	
LA	LA	Antenna coil connection LA
LB	LB	Antenna coil connection LB

8. Functional description

8.1 Block description

The MF1S70yyX chip consists of a 4 kB EEPROM, RF interface and Digital Control Unit. Energy and data are transferred via an antenna consisting of a coil with a small number of turns which is directly connected to the MF1S70yyX. No further external components are necessary. Refer to the document [Ref. 1](#) for details on antenna design.

- RF interface:
 - Modulator/demodulator
 - Rectifier
 - Clock regenerator
 - Power-On Reset (POR)
 - Voltage regulator
- Anticollision: Multiple cards in the field may be selected and managed in sequence
- Authentication: Preceding any memory operation the authentication procedure ensures that access to a block is only possible via the two keys specified for each block
- Control and Arithmetic Logic Unit: Values are stored in a special redundant format and can be incremented and decremented
- EEPROM interface
- Crypto unit: The CRYPTO1 stream cipher of the MF1S70yyX is used for authentication and encryption of data exchange.
- EEPROM: 4 kB is organized in 32 sectors of 4 blocks and 8 sectors of 16 blocks. One block contains 16 bytes. The last block of each sector is called “trailer”, which contains two secret keys and programmable access conditions for each block in this sector.

8.2 Communication principle

The commands are initiated by the reader and controlled by the Digital Control Unit of the MF1S70yyX. The command response is depending on the state of the IC and for memory operations also on the access conditions valid for the corresponding sector.

8.2.1 Request standard / all

After Power-On Reset (POR) the card answers to a request REQA or wakeup WUPA command with the answer to request code (see [Section 9.4](#), ATQA according to ISO/IEC 14443A).

8.2.2 Anticollision loop

In the anticollision loop the identifier of a card is read. If there are several cards in the operating field of the reader, they can be distinguished by their identifier and one can be selected (select card) for further transactions. The unselected cards return to the idle state and wait for a new request command. If the 7-byte UID is used for anticollision and selection, two cascade levels need to be processed as defined in ISO/IEC 14443-3.

Remark: For the 4-byte non-unique ID product versions, the identifier retrieved from the card is not defined to be unique. For further information regarding handling of non-unique identifiers see [Ref. 6](#).

8.2.3 Select card

With the select card command the reader selects one individual card for authentication and memory related operations. The card returns the Select Acknowledge (SAK) code which determines the type of the selected card, see [Section 9.4](#). For further details refer to the document [Ref. 2](#).

8.2.4 Three pass authentication

After selection of a card the reader specifies the memory location of the following memory access and uses the corresponding key for the three pass authentication procedure. After a successful authentication all memory operations are encrypted.

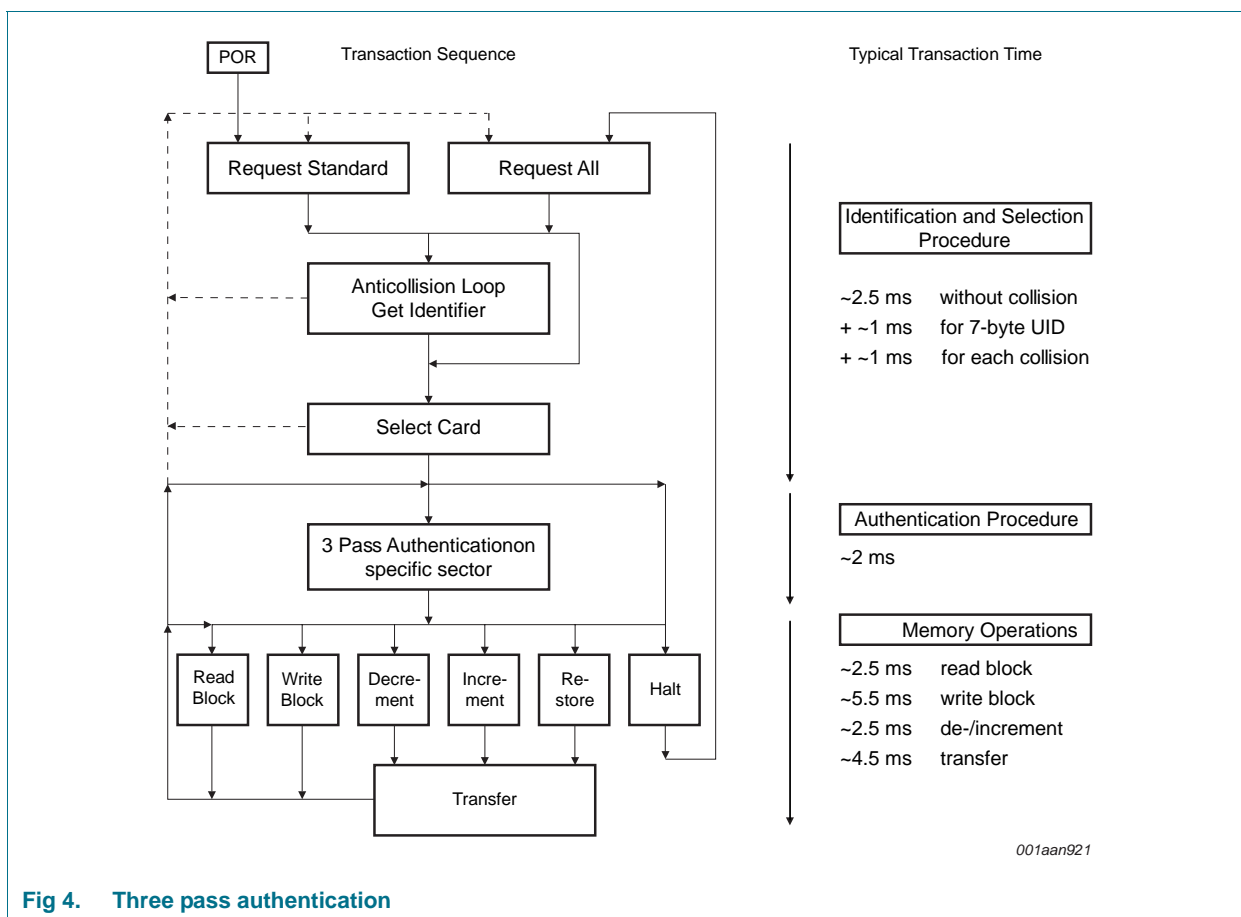


Fig 4. Three pass authentication

8.2.5 Memory operations

After authentication any of the following operations may be performed:

- Read block
- Write block
- Decrement: Decrements the contents of a block and stores the result in an internal data-register
- Increment: Increments the contents of a block and stores the result in an internal data-register
- Restore: Moves the contents of a block into an internal data-register
- Transfer: Writes the contents of the temporary internal data-register to a value block

8.3 Data integrity

Following mechanisms are implemented in the contactless communication link between reader and card to ensure very reliable data transmission:

- 16 bits CRC per block
- Parity bits for each byte
- Bit count checking
- Bit coding to distinguish between “1”, “0” and “no information”
- Channel monitoring (protocol sequence and bit stream analysis)

8.4 Three pass authentication sequence

1. The reader specifies the sector to be accessed and chooses key A or B.
2. The card reads the secret key and the access conditions from the sector trailer. Then the card sends a number as the challenge to the reader (pass one).
3. The reader calculates the response using the secret key and additional input. The response, together with a random challenge from the reader, is then transmitted to the card (pass two).
4. The card verifies the response of the reader by comparing it with its own challenge and then it calculates the response to the challenge and transmits it (pass three).
5. The reader verifies the response of the card by comparing it to its own challenge.

After transmission of the first random challenge the communication between card and reader is encrypted.

8.5 RF interface

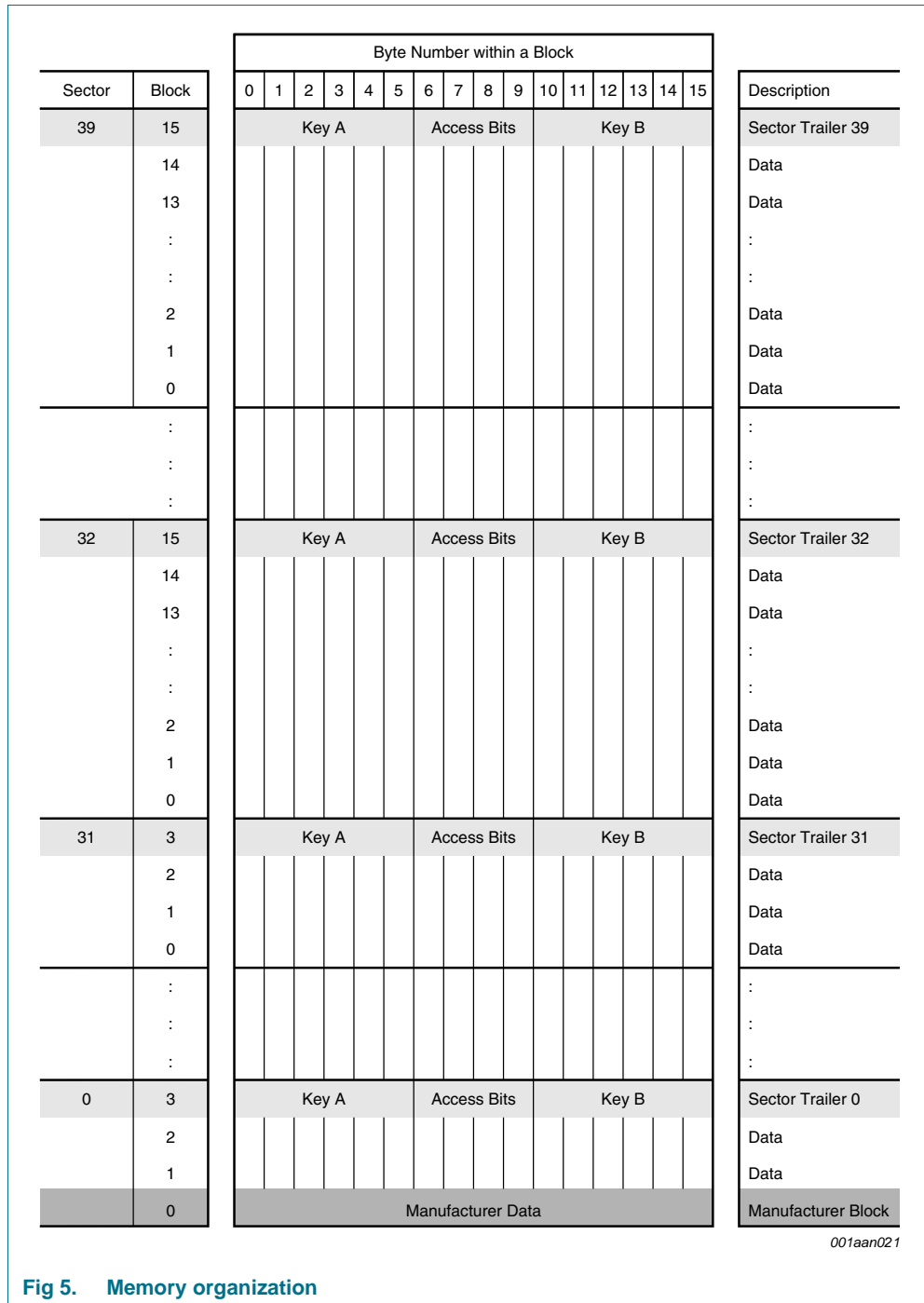
The RF-interface is according to the standard for contactless smart cards ISO/IEC 14443A.

For operation, the carrier field from the reader always needs to be present (with short pauses when transmitting), as it is used for the power supply of the card.

For both directions of data communication there is only one start bit at the beginning of each frame. Each byte is transmitted with a parity bit (odd parity) at the end. The LSB of the byte with the lowest address of the selected block is transmitted first. The maximum frame length is 163 bits (16 data bytes + 2 CRC bytes = $16 \times 9 + 2 \times 9 + 1$ start bit).

8.6 Memory organization

The 4096×8 bit EEPROM memory is organized in 32 sectors of 4 blocks and 8 sectors of 16 blocks. One block contains 16 bytes.



8.6.1 Manufacturer block

This is the first data block (block 0) of the first sector (sector 0). It contains the IC manufacturer data. This block is programmed and write protected in the production test. The manufacturer block is shown in [Figure 6](#) and [Figure 7](#) for the 4-byte NUID and 7-byte UID version respectively.

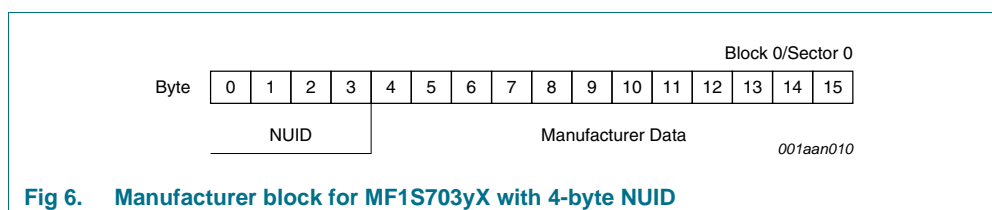


Fig 6. Manufacturer block for MF1S703yX with 4-byte NUID

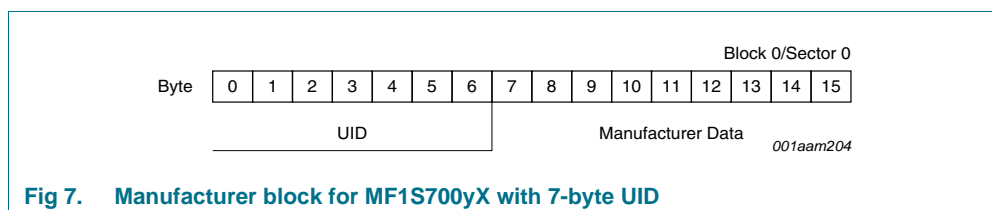


Fig 7. Manufacturer block for MF1S700yX with 7-byte UID

8.6.2 Data blocks

One block consists of 16 bytes. The first 32 sectors contain 3 blocks and the last 8 sectors contain 15 blocks for storing data (Sector 0 contains only two data blocks and the read-only manufacturer block).

The data blocks can be configured by the access bits as

- read/write blocks
- value blocks

Value blocks can be used for e.g. electronic purse applications, where additional commands like increment and decrement for direct control of the stored value are provided

A successful authentication has to be performed to allow any memory operation.

Remark: The default content of the data blocks at delivery is not defined.

8.6.2.1 Value blocks

Value blocks allow performing electronic purse functions (valid commands are: read, write, increment, decrement, restore, transfer). Value blocks have a fixed data format which permits error detection and correction and a backup management.

A value block can only be generated through a write operation in value block format:

- Value: Signifies a signed 4-byte value. The lowest significant byte of a value is stored in the lowest address byte. Negative values are stored in standard 2's complement format. For reasons of data integrity and security, a value is stored three times, twice non-inverted and once inverted.

- **Adr:** Signifies a 1-byte address, which can be used to save the storage address of a block, when implementing a powerful backup management. The address byte is stored four times, twice inverted and non-inverted. During increment, decrement, restore and transfer operations the address remains unchanged. It can only be altered via a write command.

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	value			value			value			adr	$\overline{\text{adr}}$	adr	$\overline{\text{adr}}$			

001aan018

Fig 8. Value blocks

An example of a valid value block format for the decimal value 1234567d and the block address 17d is shown in Table 4. First, the decimal value has to be converted to the hexadecimal representation of 0012D687h. The LSByte of the hexadecimal value is stored in Byte 0, the MSByte in Byte 3. The bit inverted hexadecimal representation of the value is FFED2978h where the LSByte is stored in Byte 4 and the MSByte in Byte 7.

The hexadecimal value of the address in the example is 11h, the bit inverted hexadecimal value is EEh.

Table 4. Value block format example

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	value			value			value			adr		$\overline{\text{adr}}$	adr	$\overline{\text{adr}}$		
Values [hex]	84	D6	12	00	78	29	ED	FF	84	D6	12	00	11	EE	11	EE

8.6.3 Sector trailer

The sector trailer is always the last block in one sector. For the first 32 sectors this is block 3 and for the remaining 8 sectors it is block 15. Each sector has a sector trailer containing the

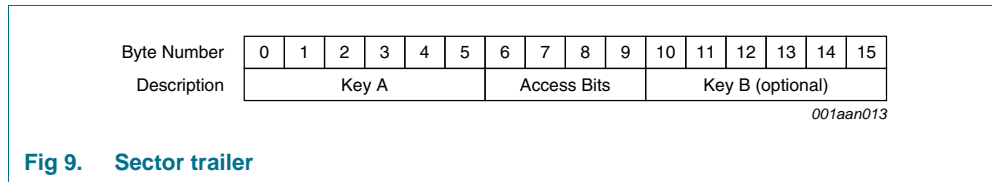
- secret keys A (mandatory) and B (optional), which return logical “0”s when read and
- the access conditions for the blocks of that sector, which are stored in bytes 6...9. The access bits also specify the type (data or value) of the data blocks.

If key B is not needed, the last 6 bytes of the sector trailer can be used as data bytes. The access bits for the sector trailer have to be configured accordingly, see Section 8.7.2.

Byte 9 of the sector trailer is available for user data. For this byte the same access rights as for byte 6, 7 and 8 apply.

When the sector trailer is read, the key bytes are blanked out by returning logical zeros. If key B is configured to be readable, the data stored in bytes 10 to 15 is returned, see Section 8.7.2.

All keys are set to FFFF FFFF FFFFh at chip delivery.



8.7 Memory access

Before any memory operation can be done, the card has to be selected and authenticated as described in [Section 8.2](#). The possible memory operations for an addressed block depend on the key used during authentication and the access conditions stored in the associated sector trailer.

Table 5. Memory operations

Operation	Description	Valid for Block Type
Read	reads one memory block	read/write, value and sector trailer
Write	writes one memory block	read/write, value and sector trailer
Increment	increments the contents of a block and stores the result in the internal data register	value
Decrement	decrements the contents of a block and stores the result in the internal data register	value
Transfer	writes the contents of the internal data register to a block	value
Restore	reads the contents of a block into the internal data register	value

8.7.1 Access conditions

The access conditions for every data block and sector trailer are defined by 3 bits, which are stored non-inverted and inverted in the sector trailer of the specified sector.

The access bits control the rights of memory access using the secret keys A and B. The access conditions may be altered, provided one knows the relevant key and the current access condition allows this operation.

Remark: With each memory access the internal logic verifies the format of the access conditions. If it detects a format violation the whole sector is irreversibly blocked.

Remark: In the following description the access bits are mentioned in the non-inverted mode only.

The internal logic of the MF1S70yyX ensures that the commands are executed only after a successful authentication.

Table 6. Access conditions

Access Bits	Valid Commands	Block (sectors 0 - 31)	Block(s) (sectors 32-39)	Description
C1 ₃ , C2 ₃ , C3 ₃	read, write	→ 3	15	sector trailer
C1 ₂ , C2 ₂ , C3 ₂	read, write, increment, decrement, transfer, restore	→ 2	10-14	data block(s)
C1 ₁ , C2 ₁ , C3 ₁	read, write, increment, decrement, transfer, restore	→ 1	5-9	data block(s)
C1 ₀ , C2 ₀ , C3 ₀	read, write, increment, decrement, transfer, restore	→ 0	0-4	data block(s)

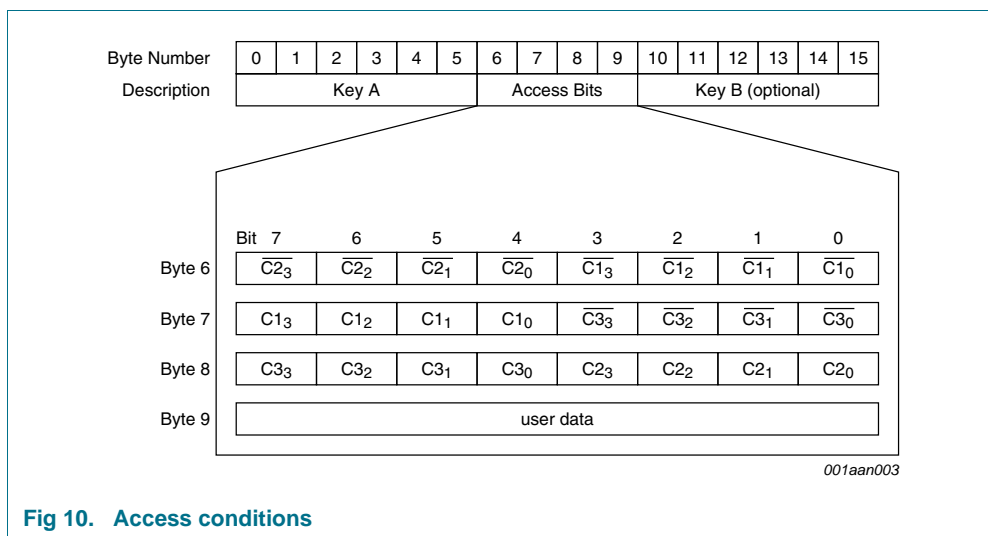


Fig 10. Access conditions

8.7.2 Access conditions for the sector trailer

Depending on the access bits for the sector trailer (block 3, respectively block 15) the read/write access to the keys and the access bits is specified as 'never', 'key A', 'key B' or key A|B' (key A or key B).

On chip delivery the access conditions for the sector trailers and key A are predefined as transport configuration. Since key B may be read in the transport configuration, new cards must be authenticated with key A. Since the access bits themselves can also be blocked, special care has to be taken during the personalization of cards.

Table 7. Access conditions for the sector trailer

Access bits			Access condition for						Remark
			KEYA		Access bits		KEYB		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read ^[1]
0	1	0	never	never	key A	never	key A	never	Key B may be read ^[1]
1	0	0	never	key B	key A B	never	never	key B	
1	1	0	never	never	key A B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration ^[1]
0	1	1	never	key B	key A B	key B	never	key B	
1	0	1	never	never	key A B	key B	never	never	
1	1	1	never	never	key A B	never	never	never	

[1] For this access condition key B is readable and may be used for data

8.7.3 Access conditions for data blocks

Depending on the access bits for data blocks (blocks 0...2) the read/write access is specified as 'never', 'key A', 'key B' or 'key A|B' (key A or key B). The setting of the relevant access bits defines the application and the corresponding applicable commands.

- Read/write block: the operations read and write are allowed.
- Value block: Allows the additional value operations increment, decrement, transfer and restore. With access condition '001' only read and decrement are possible which reflects a non-rechargeable card. For access condition '110' recharging is possible by using key B.
- Manufacturer block: the read-only condition is not affected by the access bits setting!
- Key management: in transport configuration key A must be used for authentication

Table 8. Access conditions for data blocks

Access bits			Access condition for				Application
C1	C2	C3	read	write	increment	decrement, transfer, restore	
0	0	0	key A B	key A B	key A B	key A B	transport configuration ^[1]
0	1	0	key A B	never	never	never	read/write block ^[1]
1	0	0	key A B	key B	never	never	read/write block ^[1]
1	1	0	key A B	key B	key B	key A B	value block ^[1]
0	0	1	key A B	never	never	key A B	value block ^[1]
0	1	1	key B	key B	never	never	read/write block ^[1]
1	0	1	key B	never	never	never	read/write block ^[1]
1	1	1	never	never	never	never	read/write block

[1] If key B may be read in the corresponding Sector Trailer it cannot serve for authentication (see grey marked lines in [Table 7](#)). As a consequence, if the reader authenticates any block of a sector which uses such access conditions for the Sector Trailer and using key B, the card will refuse any subsequent memory access after authentication.

9. Command overview

The MIFARE Classic card activation follows the ISO/IEC 14443 Type A. After the MIFARE Classic card has been selected, it can either be deactivated using the ISO/IEC 14443 Halt command, or the MIFARE Classic commands can be performed. For more details about the card activation refer to [Ref. 4](#).

9.1 MIFARE Classic command overview

All MIFARE Classic commands use the MIFARE CRYPTO1 and require an authentication.

All available commands for the MIFARE Classic are shown in [Table 9](#).

Table 9. Command overview

Command	ISO/IEC 14443	Command code (hexadecimal)
Request	REQA	26h (7 bit)
Wake-up	WUPA	52h (7 bit)
Anticollision CL1	Anticollision CL1	93h 20h
Select CL1	Select CL1	93h 70h
Anticollision CL2	Anticollision CL2	95h 20h
Select CL2	Select CL2	95h 70h
Halt	Halt	50h 00h
Authentication with Key A	-	60h
Authentication with Key B	-	61h
Personalize UID Usage	-	40h
MIFARE Read	-	30h
MIFARE Write	-	A0h
MIFARE Decrement	-	C0h
MIFARE Increment	-	C1h
MIFARE Restore	-	C2h
MIFARE Transfer	-	B0h

All commands use the coding and framing as described in [Ref. 3](#) and [Ref. 4](#) if not otherwise specified.

9.2 Timings

The timing shown in this document are not to scale and values are rounded to 1 μ s.

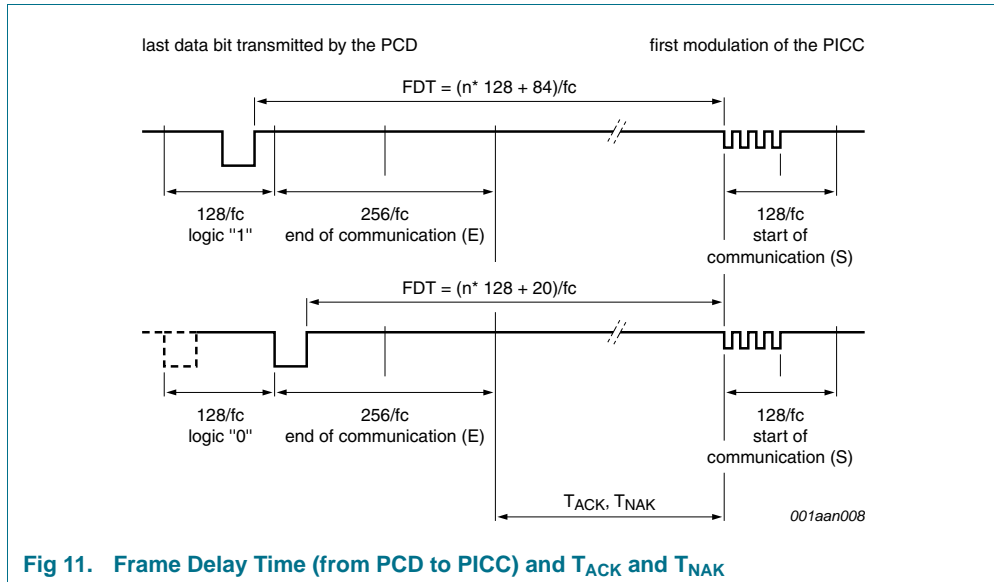
All given times refer to the data frames including start of communication and end of communication, but do not include the encoding (like the Miller pulses).

Consequently a data frame sent by the PCD contains the start of communication (1 “start bit”) and the end of communication (one logic 0 + 1 bit length of unmodulated carrier).

A data frame sent by the PICC contains the start of communication (1 “start bit”) and the end of communication (1 bit length of no subcarrier).

All timing can be measured according to ISO/IEC 14443-3 frame specification as shown for the Frame Delay Time in [Figure 11](#). For more details refer to [Ref. 3](#) and [Ref. 4](#).

The frame delay time from PICC to PCD must be at least 87 μs.



Remark: Due to the coding of commands, the measured timings usually excludes (a part of) the end of communication. This needs to be considered, when comparing the specified with the measured times.

9.3 MIFARE Classic ACK and NAK

The MIFARE Classic uses a 4 bit ACK / NAK as shown in [Table 10](#).

Table 10. MIFARE ACK and NAK

Code (4-bit)	ACK/NAK
Ah	Acknowledge (ACK)
0h to 9h, Bh to Fh	NAK

9.4 ATQA and SAK responses

For details on the type identification procedure please refer to [Ref. 2](#).

The MF1S70yyX answers to a REQA or WUPA command with the ATQA value shown in [Table 11](#) and to a Select CL1 command (CL2 for the 7-byte UID variant) with the SAK value shown in [Table 12](#).

Table 11. ATQA response of the MF1S70yyX

Sales Type	Hex Value	Bit Number															
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
MF1S700yX	00 42h	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
MF1S703yX	00 02h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 12. SAK response of the MF1S70yyX

Sales Type	Hex Value	Bit Number							
		8	7	6	5	4	3	2	1
MF1S70yyX	18h	0	0	0	1	1	0	0	0

10. UID Options and Handling

The MF1S70yyX product family offers two delivery options for the UID which is stored in block 0 of sector 0.

- 7-byte UID
- 4-byte NUID (Non-Unique ID)

This section describes the MIFARE Classic MF1S70yyX operation when using one of the 2 UID options with respect to card selection, authentication and personalization. See also [Ref. 6](#) for details on how to handle UIDs and NUIDs with MIFARE Classic products.

10.1 7-byte UID Operation

All MF1S70yyX products are featuring a 7-byte UID. This 7-byte UID is stored in block 0 of sector 0 as shown in [Figure 7](#). The behaviour during anti-collision, selection and authentication can be configured during personalization for this UID variant.

10.1.1 Personalization Options

The 7-byte UID variants of the MF1S70yyX can be operated with four different functionalities, denoted as UIDFn (UID Functionality n).

1. UIDF0: anti-collision and selection with the double size UID according to ISO/IEC 14443-3
2. UIDF1: anti-collision and selection with the double size UID according to ISO/IEC 14443-3 and optional usage of a selection process shortcut
3. UIDF2: anti-collision and selection with a single size random ID according to ISO/IEC 14443-3
4. UIDF3: anti-collision and selection with a single size NUID according to ISO/IEC 14443-3 where the NUID is calculated out of the 7-byte UID

The anti-collision and selection procedure and the implications on the authentication process are detailed in [Section 10.1.2](#) and [Section 10.1.3](#).

The default configuration at delivery is option 1 which enables the ISO/IEC 14443-3 compliant anti-collision and selection. This configuration can be changed using the 'Personalize UID Usage' command. The execution of this command requires an authentication to sector 0. Once this command has been issued and accepted by the PICC, the configuration is automatically locked. A subsequently issued 'Personalize UID Usage' command is not executed and a NAK is replied by the PICC.

Remark: As the configuration is changeable at delivery, it is strongly recommended to send this command at personalization of the card to prevent unwanted changes in the field. This should also be done if the default configuration is used.

Remark: The configuration only becomes effective only after PICC unselect or PICC field reset.

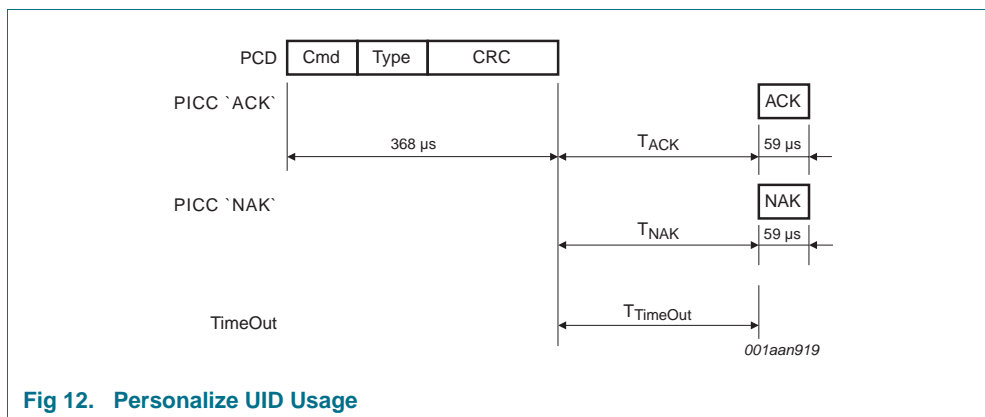


Fig 12. Personalize UID Usage

Table 13. Personalize UID Usage command

Name	Code	Description	Length
Cmd	40h	Set anti-collision, selection and authentication behaviour	1 byte
Type	-	Encoded type of UID usage: UIDF0: 00h UIDF1: 40h UIDF2: 20h UIDF3: 60h	1 byte
CRC	-	CRC according to Ref. 4	2 bytes
ACK, NAK	see Table 10	see Section 9.3	4-bit

Table 14. Personalize UID Usage timing

These times exclude the end of communication of the PCD.

	T _{ACK min}	T _{ACK max}	T _{NAK min}	T _{NAK max}	T _{TimeOut}
Personalize UID Usage	71 µs	T _{TimeOut}	71 µs	T _{TimeOut}	10 ms

10.1.2 Anti-collision and Selection

Depending on the chosen personalization option there are certain possibilities to perform anti-collision and selection. To bring the MIFARE Classic into the ACTIVE state according to ISO/IEC 14443-3, the following sequences are available.

Sequence 1: ISO/IEC 14443-3 compliant anti-collision and selection using the cascade level 1 followed by the cascade level 2 SEL command

Sequence 2: using cascade level 1 anti-collision and selection procedure followed by a Read command from block 0

Sequence 3: ISO/IEC 14443-3 compliant anti-collision and selection using the cascade level 1 SEL command

Remark: The Read from Block 0 in Sequence 2 does not require a prior authentication to Sector 0 and is transmitted in plain data. For all other sequences, the readout from Block 0 in Sector 0 is encrypted and requires an authentication to that sector.

Table 15. Available activation sequences for 7-byte UID options

UID Functionality	Available Activation Sequences
UIDF0	Sequence 1
UIDF1	Sequence 1, Sequence 2
UIDF2	Sequence 3
UIDF3	Sequence 3

10.1.3 Authentication

During the authentication process, 4-byte of the UID are passed on to the MIFARE Classic Authenticate command of the contactless reader IC. Depending on the activation sequence, those 4-byte are chosen differently.

Table 16. Input parameter to MIFARE Classic Authenticate

UID Functionality	Input to MIFARE Classic Authenticate Command
Sequence 1	CL2 bytes (UID3...UID6)
Sequence 2	CL1 bytes (CT, UID0...UID2)
Sequence 3	4-byte NUID/RID (UID0...UID3)

10.2 4-byte UID Operation

All MF1S703yXDyy products are featuring a 4-byte NUID. This 4-byte NUID is stored in block 0 of sector 0 as shown in [Figure 6](#).

10.2.1 Anti-collision and Selection

The anti-collision and selection process for the product variants featuring 4-byte NUIDs is done according to ISO/IEC 14443-3 Type A using cascade level 1 only.

10.2.2 Authentication

The input parameter to the MIFARE Classic Authenticate command is the full 4-byte UID retrieved during the anti-collision procedure. This is the same as for the activation Sequence 3 in the 7-byte UID variant.

11. MIFARE Classic commands

11.1 MIFARE Authentication

The MIFARE authentication is a 3-pass mutual authentication which needs two pairs of command-response. These two parts, MIFARE authentication part 1 and part 2 are shown in [Figure 13](#), [Figure 14](#) and [Table 17](#).

[Table 18](#) shows the required timing.

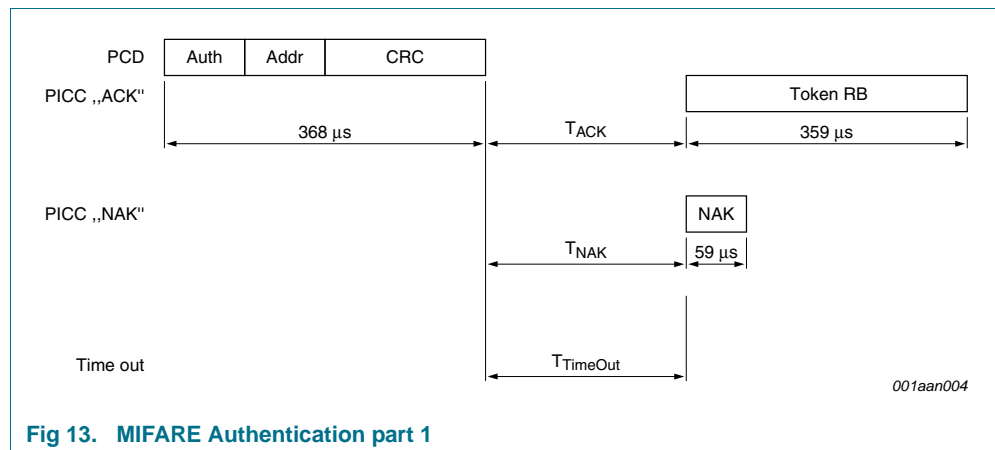


Fig 13. MIFARE Authentication part 1

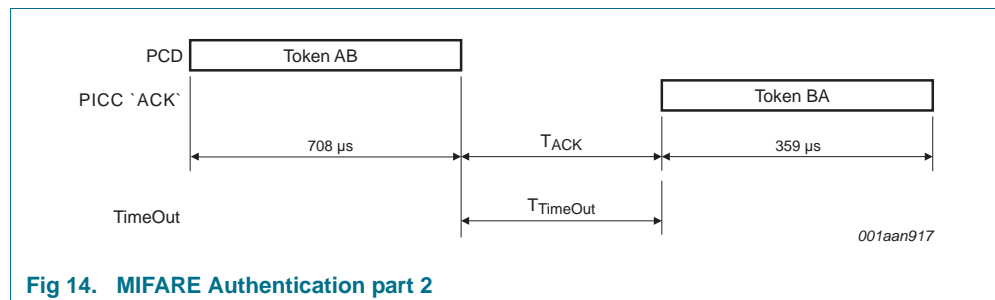


Fig 14. MIFARE Authentication part 2

Table 17. MIFARE authentication command

Name	Code	Description	Length
Auth (with Key A)	60h	Authentication with Key A	1 byte
Auth (with Key B)	61h	Authentication with Key B	1 byte
Addr	-	MIFARE Block address (00h to FFh)	1 byte
CRC	-	CRC according to Ref. 4	2 bytes
Token RB	-	Challenge 1 (Random Number)	4 bytes
Token AB	-	Challenge 2 (encrypted data)	8 bytes
Token BA	-	Challenge 2 (encrypted data)	4 bytes
NAK	see Table 10	see Section 9.3	4-bit

Table 18. MIFARE authentication timing

These times exclude the end of communication of the PCD.

	T _{ACK min}	T _{ACK max}	T _{NAK min}	T _{NAK max}	T _{TimeOut}
Authentication part 1	71 μs	T _{TimeOut}	71 μs	T _{TimeOut}	1 ms
Authentication part 2	71 μs	T _{TimeOut}			1 ms

Remark: The minimum required time between MIFARE Authentication part 1 and part 2 is the minimum required FDT according to [Ref. 4](#). There is no maximum time specified.

Remark: The MIFARE authentication and encryption requires an MIFARE reader IC (e.g. the CL RC632). For more details about the authentication command refer to the corresponding data sheet (e.g. [Ref. 5](#)). The 4-byte input parameter for the MIFARE Classic Authentication is detailed in [Section 10.1.3](#) and [Section 10.2.2](#).

11.2 MIFARE Read

The MIFARE Read requires a block address, and returns the 16 bytes of one MIFARE Classic block. The command structure is shown in [Figure 15](#) and [Table 19](#).

[Table 20](#) shows the required timing.

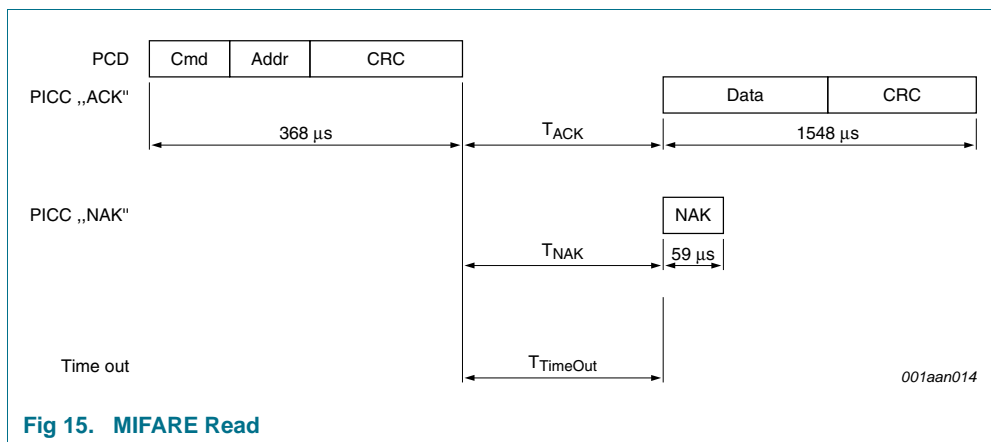


Fig 15. MIFARE Read

Table 19. MIFARE Read command

Name	Code	Description	Length
Cmd	30h	Read one block	1 byte
Addr	-	MIFARE Block address (00h to FFh)	1 byte
CRC	-	CRC according to Ref. 4	2 bytes
Data	-	Data content of the addressed block	16 bytes
NAK	see Table 10	see Section 9.3	4-bit

Table 20. MIFARE Read timing

These times exclude the end of communication of the PCD.

	T _{ACK min}	T _{ACK max}	T _{NAK min}	T _{NAK max}	T _{TimeOut}
Read	71 μs	T _{TimeOut}	71 μs	T _{TimeOut}	5 ms

11.3 MIFARE Write

The MIFARE Write requires a block address, and writes 16 bytes of data into the addressed MIFARE Classic 4K block. It needs two pairs of command-response. These two parts, MIFARE Write part 1 and part 2 are shown in [Figure 16](#) and [Figure 17](#) and [Table 21](#).

[Table 22](#) shows the required timing.

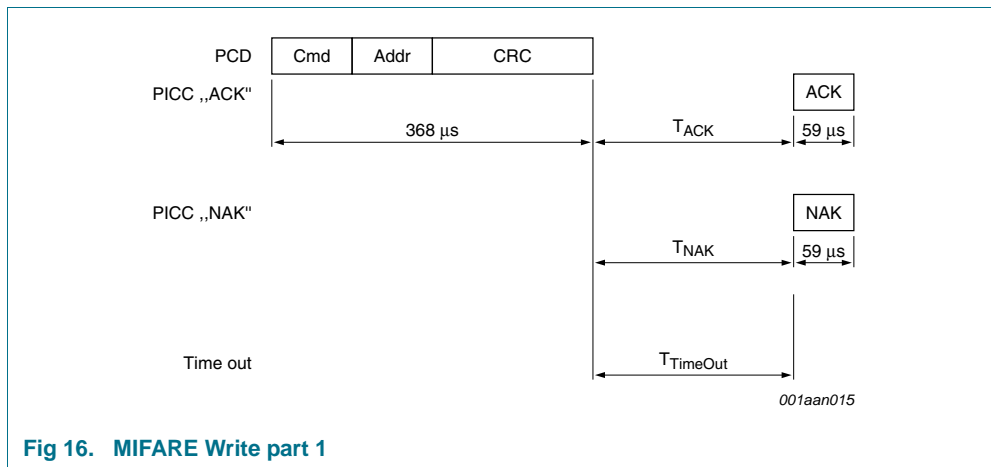


Fig 16. MIFARE Write part 1

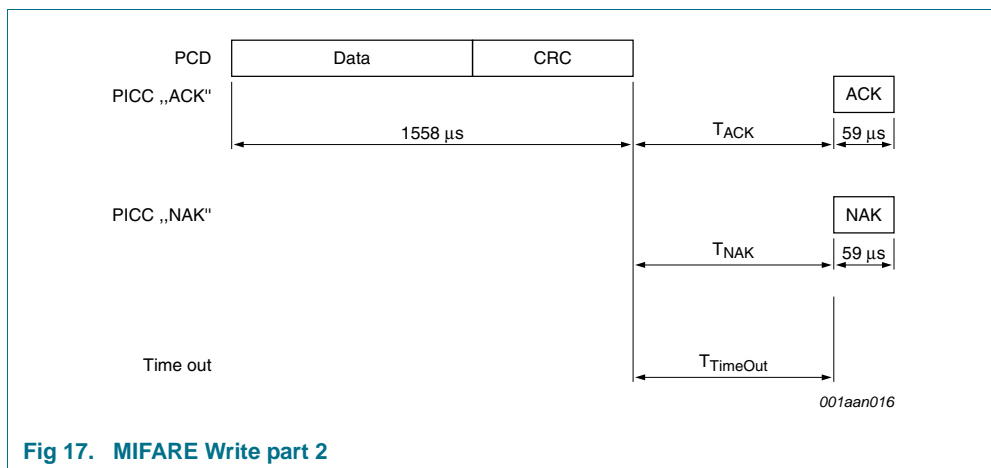


Fig 17. MIFARE Write part 2

Table 21. MIFARE Write command

Name	Code	Description	Length
Cmd	A0h	Write one block	1 byte
Addr	-	MIFARE Block or Page address (00h to FFh)	1 byte
CRC	-	CRC according to Ref. 4	2 bytes
Data	-	Data	16 bytes
NAK	see Table 10	see Section 9.3	4-bit

Table 22. MIFARE Write timing

These times exclude the end of communication of the PCD.

	T _{ACK min}	T _{ACK max}	T _{NAK min}	T _{NAK max}	T _{TimeOut}
Write part 1	71 μs	T _{TimeOut}	71 μs	T _{TimeOut}	5 ms
Write part 2	71 μs	T _{TimeOut}	71 μs	T _{TimeOut}	10 ms

Remark: The minimum required time between MIFARE Write part 1 and part 2 is the minimum required FDT according to [Ref. 4](#). There is no maximum time specified.

11.4 MIFARE Increment, Decrement and Restore

The MIFARE Increment requires a source block address and an operand. It adds the operand to the value of the addressed block, and stores the result in a volatile memory.

The MIFARE Decrement requires a source block address and an operand. It subtracts the operand from the value of the addressed block, and stores the result in a volatile memory.

The MIFARE Restore requires a source block address. It copies the value of the addressed block into a volatile memory.

All three commands are responding with a NAK to the first command part if the addressed block is not formatted to be a valid value block, see [Section 8.6.2.1](#).

The two parts of each command are shown in [Figure 18](#) and [Figure 19](#) and [Table 23](#).

[Table 24](#) shows the required timing.

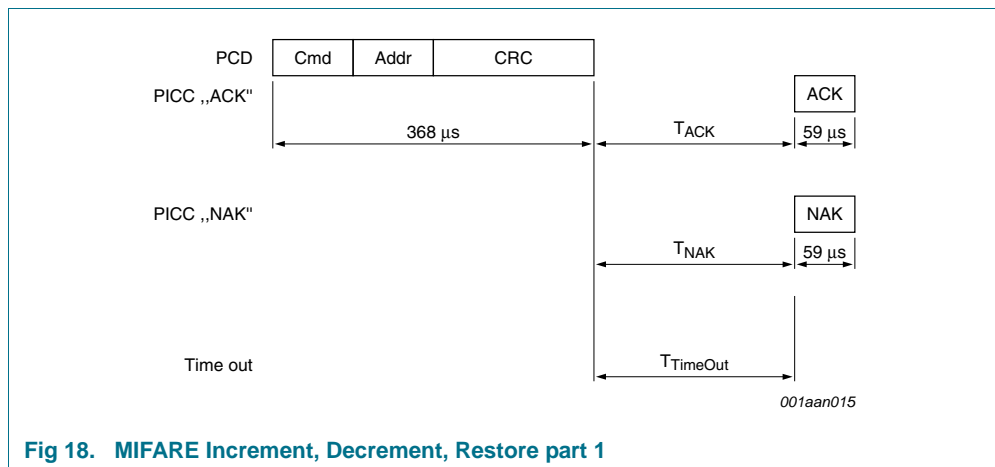


Fig 18. MIFARE Increment, Decrement, Restore part 1

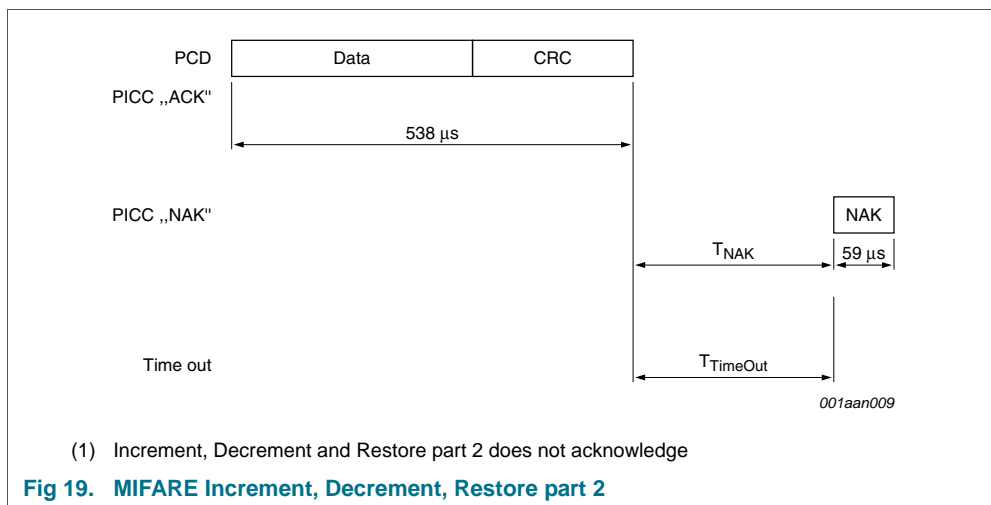


Table 23. MIFARE Increment, Decrement and Restore command

Name	Code	Description	Length
Cmd	C1h	Increment	1 byte
Cmd	C0h	Decrement	1 byte
Cmd	C2h	Restore	1 byte
Addr	-	MIFARE source block address (00h to FFh)	1 byte
CRC	-	CRC according to Ref. 4	2 bytes
Data	-	Operand (4 byte signed integer)	4 bytes
NAK	see Table 10	see Section 9.3	4-bit

Table 24. MIFARE Increment, Decrement and Restore timing

These times exclude the end of communication of the PCD.

	T _{ACK min}	T _{ACK max}	T _{NAK min}	T _{NAK max}	T _{TimeOut}
Increment, Decrement, and Restore part 1	71 μs	T _{TimeOut}	71 μs	T _{TimeOut}	5 ms
Increment, Decrement, and Restore part 2	71 μs	T _{TimeOut}	71 μs	T _{TimeOut}	5 ms

Remark: The minimum required time between MIFARE Increment, Decrement, and Restore part 1 and part 2 is the minimum required FDT according to [Ref. 4](#). There is no maximum time specified.

Remark: The MIFARE Increment, Decrement, and Restore commands require a MIFARE Transfer to store the value into a destination block.

Remark: The MIFARE Increment, Decrement, and Restore command part 2 does not provide an acknowledgement, so the regular time out has to be used instead.

11.5 MIFARE Transfer

The MIFARE Transfer requires a destination block address, and writes the value stored in the volatile memory into one MIFARE Classic block. The command structure is shown in [Figure 20](#) and [Table 25](#).

[Table 26](#) shows the required timing.

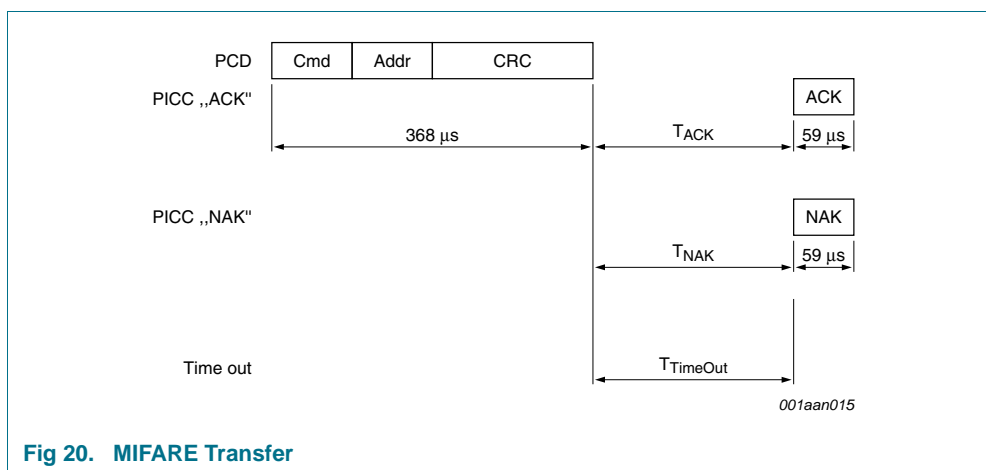


Fig 20. MIFARE Transfer

Table 25. MIFARE Transfer command

Name	Code	Description	Length
Cmd	B0h	Write value into destination block	1 byte
Addr	-	MIFARE destination block address (00h to FFh)	1 byte
CRC	-	CRC according to Ref. 4	2 bytes
NAK	see Table 10	see Section 9.3	4-bit

Table 26. MIFARE Transfer timing

These times exclude the end of communication of the PCD.

	T _{ACK} min	T _{ACK} max	T _{NAK} min	T _{NAK} max	T _{TimeOut}
Transfer	71 µs	T _{TimeOut}	71 µs	T _{TimeOut}	10 ms

12. Limiting values

Stresses above one or more of the limiting values may cause permanent damage to the device. Exposure to limiting values for extended periods may affect device reliability.

Table 27. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134).

Symbol	Parameter	Min	Max	Unit
I_i	input current	-	30	mA
$P_{tot}/pack$	total power dissipation per package	-	120	mW
T_{stg}	storage temperature	-55	125	°C
T_{amb}	ambient temperature	-25	70	°C
V_{ESD}	electrostatic discharge voltage on LA/LB [1]	2	-	kV
I_{lu}	latch-up current	±100	-	mA

[1] ANSI/ESDA/JEDEC JS-001; Human body model: C = 100 pF, R = 1.5 kΩ

13. Characteristics

Table 28. Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
C_i	input capacitance	[1]	14.9	16.9	19.0	pF
f_i	input frequency		-	13.56	-	MHz
EEPROM characteristics						
t_{ret}	retention time	$T_{amb} = 22\text{ °C}$	10	-	-	year
$N_{endu(W)}$	write endurance	$T_{amb} = 22\text{ °C}$	100000	200000	-	cycle

[1] LCR meter, $T_{amb} = 22\text{ °C}$, $f_i = 13.56\text{ MHz}$, 2 V RMS.

14. Wafer specification

For more details on the wafer delivery forms see [Ref. 9](#).

Table 29. Wafer specifications MF1S70yyXDUy

Wafer	
diameter	200 mm typical (8 inches)
maximum diameter after foil expansion	210 mm
thickness	MF1S70yyXDUD 120 $\mu\text{m} \pm 15 \mu\text{m}$
	MF1S70yyXDUF 75 $\mu\text{m} \pm 10 \mu\text{m}$
flatness	not applicable
Potential Good Dies per Wafer (PGDW)	est. 66264
Wafer backside	
material	Si
treatment	ground and stress relieve
roughness	R_a max = 0.5 μm R_t max = 5 μm
Chip dimensions	
step size ^[1]	x = 659 μm y = 694 μm
gap between chips ^[1]	typical = 19 μm minimum = 5 μm
Passivation	
type	sandwich structure
material	PSG / nitride
thickness	500 nm / 600 nm
Au bump (substrate connected to VSS)	
material	> 99.9 % pure Au
hardness	35 to 80 HV 0.005
shear strength	> 70 MPa
height	18 μm
height uniformity	within a die = $\pm 2 \mu\text{m}$ within a wafer = $\pm 3 \mu\text{m}$ wafer to wafer = $\pm 4 \mu\text{m}$
flatness	minimum = $\pm 1.5 \mu\text{m}$
size	LA, LB, VSS, TEST ^[2] = 66 $\mu\text{m} \times 66 \mu\text{m}$
size variation	$\pm 5 \mu\text{m}$
under bump metallization	sputtered TiW

[1] The step size and the gap between chips may vary due to changing foil expansion

[2] Pads VSS and TESTIO are disconnected when wafer is sawn.

14.1 Fail die identification

Electronic wafer mapping covers the electrical test results and additionally the results of mechanical/visual inspection. No ink dots are applied.

14.2 Package outline

For more details on the contactless modules MOA4 and MOA8 please refer to [Ref. 7](#) and [Ref. 8](#).

PLLMC: plastic leadless module carrier package; 35 mm wide tape

SOT500-2

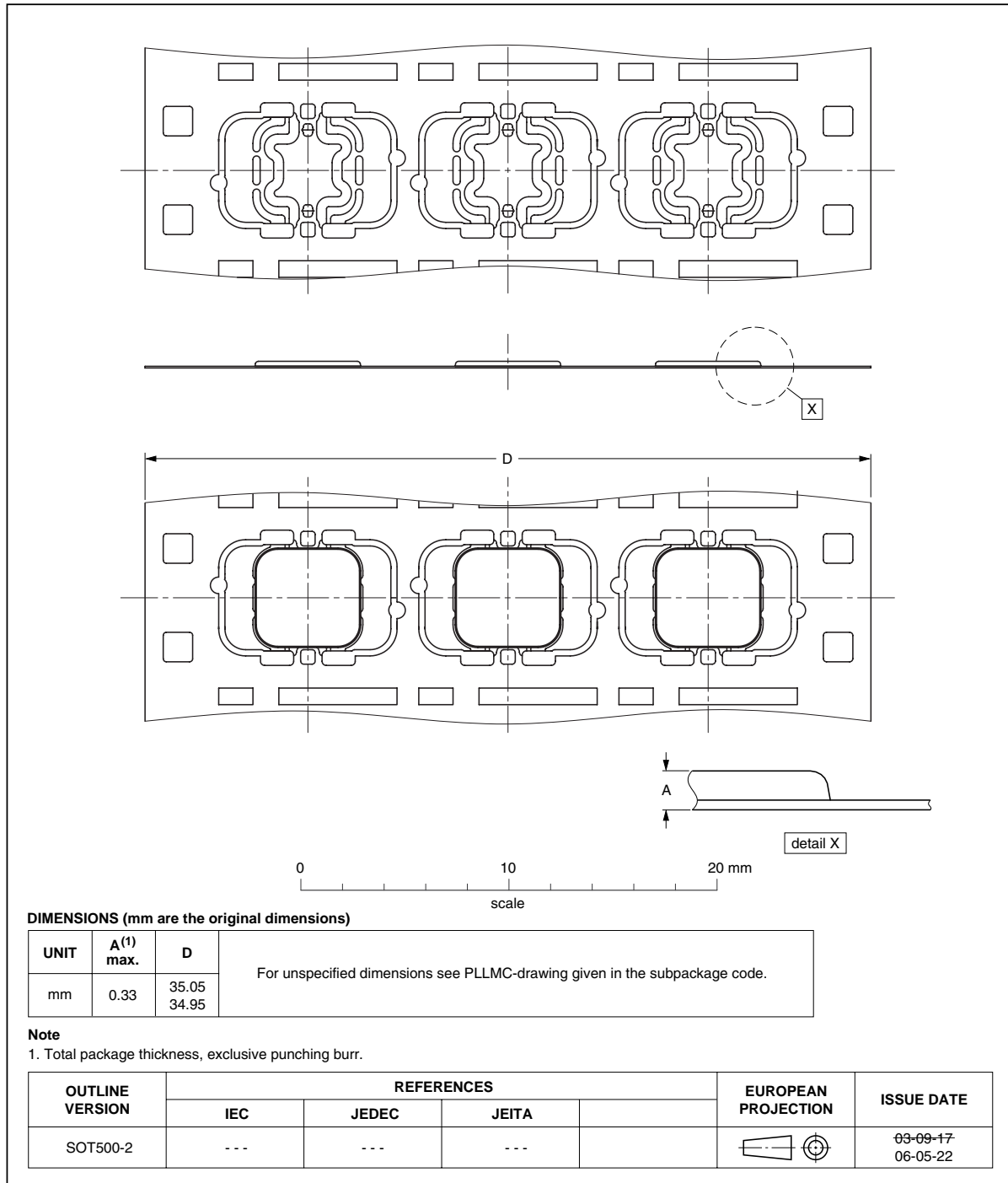


Fig 21. Package outline SOT500-2

PLLMC: plastic leadless module carrier package; 35 mm wide tape

SOT500-4

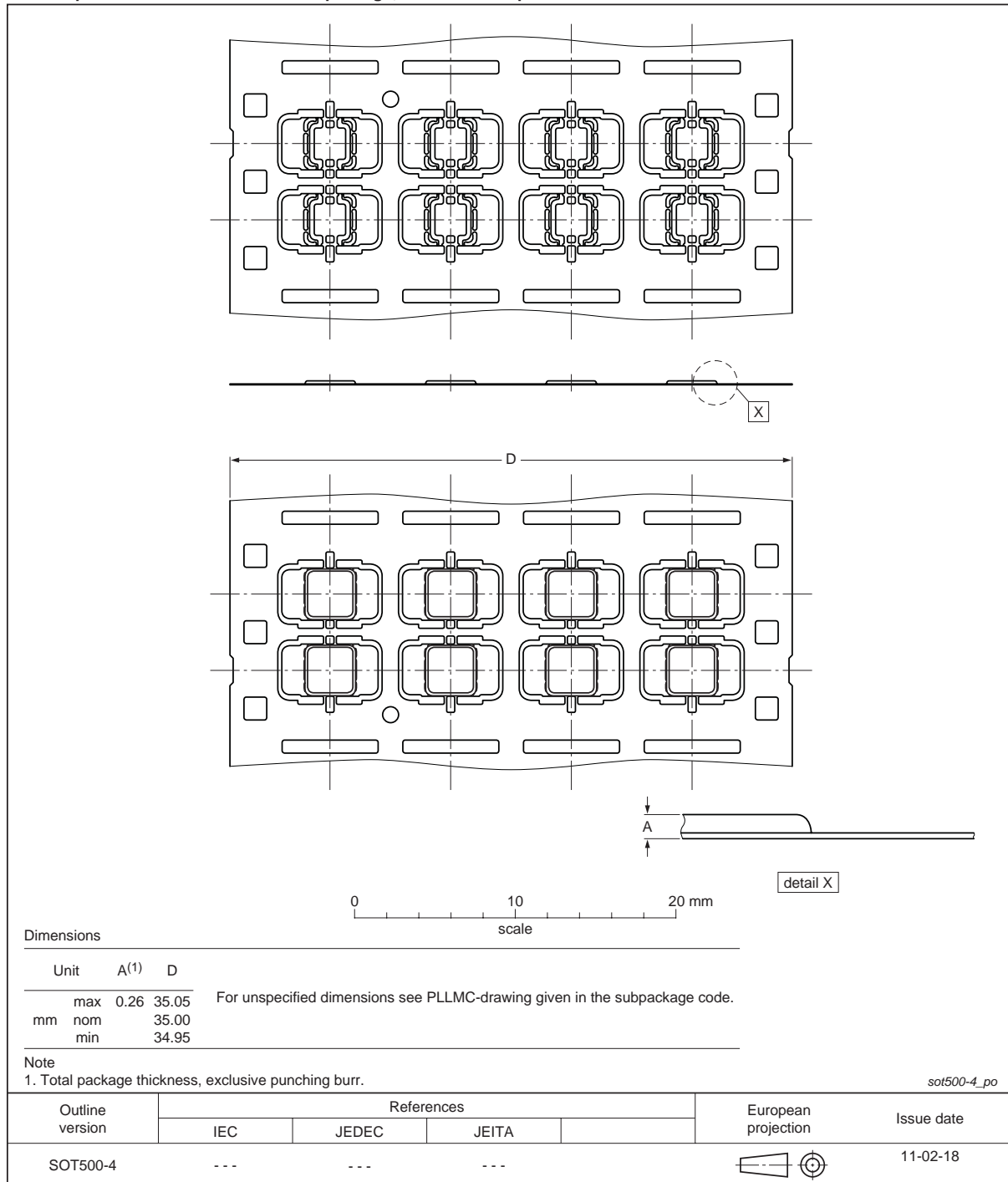


Fig 22. Package outline SOT500-4

14.3 Bare die outline

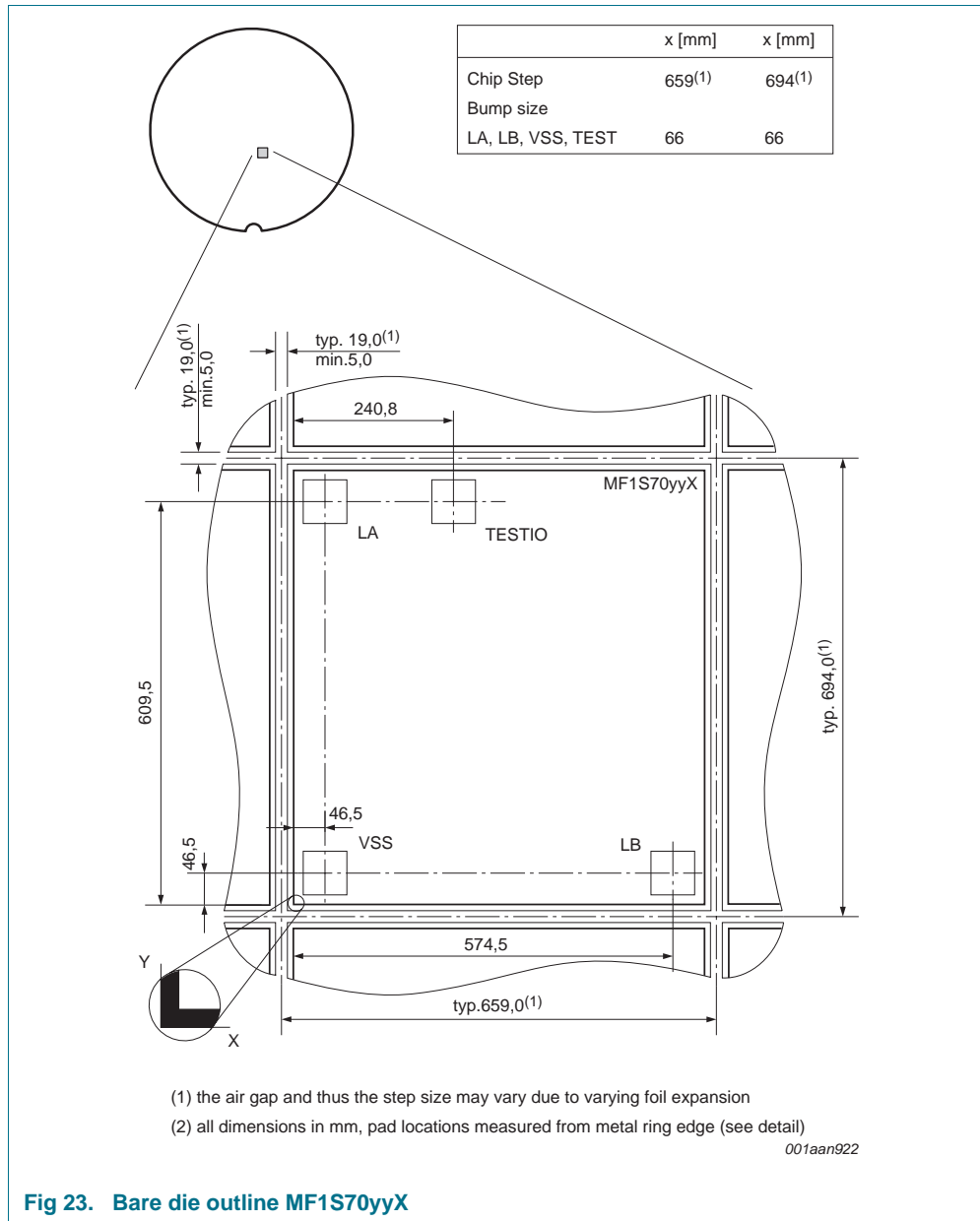


Fig 23. Bare die outline MF1S70yyX

15. Abbreviations

Table 30. Abbreviations and symbols

Acronym	Description
ACK	ACKnowledge
ATQA	Answer To reQuest, Type A
CRC	Cyclic Redundancy Check
CT	Cascade Tag (value 88h) as defined in ISO/IEC 14443-3 Type A
EEPROM	Electrically Erasable Programmable Read-Only Memory
FDT	Frame Delay Time
FFC	Film Frame Carrier
IC	Integrated Circuit
LCR	L = inductance, Capacitance, Resistance (LCR meter)
LSB	Least Significant Bit
NAK	Not AcKnowledge
NUID	Non-Unique IDentifier
NV	Non-Volatile memory
PCD	Proximity Coupling Device (Contactless Reader)
PICC	Proximity Integrated Circuit Card (Contactless Card)
REQA	REQuest command, Type A
RID	Random ID
RF	Radio Frequency
RMS	Root Mean Square
SAK	Select AcKnowledge, type A
RNG	Random Number Generator
SECS-II	SEMI Equipment Communications Standard part 2
TiW	Titanium Tungsten
UID	Unique IDentifier
WUPA	Wake-Up Protocol type A

16. References

- [1] **MIFARE (Card) Coil Design Guide** — Application note, BU-ID Document number 0117**[1](#)
- [2] **MIFARE Type Identification Procedure** — Application note, BU-ID Document number 0184**[1](#)
- [3] **ISO/IEC 14443-2** — 2001
- [4] **ISO/IEC 14443-3** — 2001
- [5] **MIFARE & I-CODE CL RC632 Multiple protocol contactless reader IC** — Product data sheet
- [6] **MIFARE and handling of UIDs** — Application note, BU-ID Document number 1907**[1](#)
- [7] **Contactless smart card module specification MOA4** — Delivery Type Description, BU-ID Document number 0823**[1](#)
- [8] **Contactless smart card module specification MOA8** — Delivery Type Description, BU-ID Document number 1636**[1](#)
- [9] **General specification for 8" wafer on UV-tape; delivery types** — Delivery Type Description, BU-ID Document number 1005**[1](#)

1. ** ... document version number

17. Revision history

Table 31. Revision history

Document ID	Release date	Data sheet status	Change notice	Supersedes
MF1S70YYX v.3.0	20110502	Product data sheet	-	MF1S70YYX v.2.0
Modifications:	• General update			
MF1S70YYX v.2.0	20101027	Preliminary data sheet	-	-

18. Legal information

18.1 Data sheet status

Document status ^{[1][2]}	Product status ^[3]	Definition
Objective [short] data sheet	Development	This document contains data from the objective specification for product development.
Preliminary [short] data sheet	Qualification	This document contains data from the preliminary specification.
Product [short] data sheet	Production	This document contains the product specification.

[1] Please consult the most recently issued document before initiating or completing a design.

[2] The term 'short data sheet' is explained in section "Definitions".

[3] The product status of device(s) described in this document may have changed since this document was published and may differ in case of multiple devices. The latest product status information is available on the Internet at URL <http://www.nxp.com>.

18.2 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

Short data sheet — A short data sheet is an extract from a full data sheet with the same product type number(s) and title. A short data sheet is intended for quick reference only and should not be relied upon to contain detailed and full information. For detailed and full information see the relevant full data sheet, which is available on request via the local NXP Semiconductors sales office. In case of any inconsistency or conflict with the short data sheet, the full data sheet shall prevail.

Product specification — The information and data provided in a Product data sheet shall define the specification of the product as agreed between NXP Semiconductors and its customer, unless NXP Semiconductors and customer have explicitly agreed otherwise in writing. In no event however, shall an agreement be valid in which the NXP Semiconductors product is deemed to offer functions and qualities beyond those described in the Product data sheet.

18.3 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or

malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at <http://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

Non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Bare die — All die are tested on compliance with their related technical specifications as stated in this data sheet up to the point of wafer sawing and are handled in accordance with the NXP Semiconductors storage and transportation conditions. If there are data sheet limits not guaranteed, these will be separately indicated in the data sheet. There are no post-packing tests performed on individual die or wafers.

NXP Semiconductors has no control of third party procedures in the sawing, handling, packing or assembly of the die. Accordingly, NXP Semiconductors assumes no liability for device functionality or performance of the die or systems after third party sawing, handling, packing or assembly of the die. It is the responsibility of the customer to test and qualify their application in which the die is used.

All die sales are conditioned upon and subject to the customer entering into a written die sale agreement with NXP Semiconductors through its legal department.

18.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

MIFARE — is a trademark of NXP B.V.

19. Contact information

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: salesaddresses@nxp.com

20. Tables

Table 1.	Quick reference data	2
Table 2.	Ordering information	3
Table 3.	Pin allocation table	4
Table 4.	Value block format example	11
Table 5.	Memory operations	12
Table 6.	Access conditions	13
Table 7.	Access conditions for the sector trailer	14
Table 8.	Access conditions for data blocks	15
Table 9.	Command overview	16
Table 10.	MIFARE ACK and NAK	17
Table 11.	ATQA response of the MF1S70yyX	18
Table 12.	SAK response of the MF1S70yyX	18
Table 13.	Personalize UID Usage command	20
Table 14.	Personalize UID Usage timing	20
Table 15.	Available activation sequences for 7-byte UID options	21
Table 16.	Input parameter to MIFARE Classic Authenticate	21
Table 17.	MIFARE authentication command	22
Table 18.	MIFARE authentication timing	23
Table 19.	MIFARE Read command	23
Table 20.	MIFARE Read timing	23
Table 21.	MIFARE Write command	24
Table 22.	MIFARE Write timing	25
Table 23.	MIFARE Increment, Decrement and Restore command	26
Table 24.	MIFARE Increment, Decrement and Restore timing	26
Table 25.	MIFARE Transfer command	27
Table 26.	MIFARE Transfer timing	27
Table 27.	Limiting values	28
Table 28.	Characteristics	28
Table 29.	Wafer specifications MF1S70yyXDUy	29
Table 30.	Abbreviations and symbols	33
Table 31.	Revision history	35

21. Figures

Fig 1.	MIFARE card reader	1
Fig 2.	Block diagram of MF1S70yyX	3
Fig 3.	Pin configuration for SOT500-2 (MOA4)	4
Fig 4.	Three pass authentication	6
Fig 5.	Memory organization	9
Fig 6.	Manufacturer block for MF1S703yX with 4-byte NUID	10
Fig 7.	Manufacturer block for MF1S700yX with 7-byte UID	10
Fig 8.	Value blocks	11
Fig 9.	Sector trailer	12
Fig 10.	Access conditions	13
Fig 11.	Frame Delay Time (from PCD to PICC) and T_{ACK} and T_{NAK}	17
Fig 12.	Personalize UID Usage	20
Fig 13.	MIFARE Authentication part 1	22
Fig 14.	MIFARE Authentication part 2	22
Fig 15.	MIFARE Read	23
Fig 16.	MIFARE Write part 1	24
Fig 17.	MIFARE Write part 2	24
Fig 18.	MIFARE Increment, Decrement, Restore part 1	25
Fig 19.	MIFARE Increment, Decrement, Restore part 2	26
Fig 20.	MIFARE Transfer	27
Fig 21.	Package outline SOT500-2	30
Fig 22.	Package outline SOT500-4	31
Fig 23.	Bare die outline MF1S70yyX	32

22. Contents

1	General description	1	11	MIFARE Classic commands	22
1.1	Anticollision	1	11.1	MIFARE Authentication	22
1.2	Simple integration and user convenience	1	11.2	MIFARE Read	23
1.3	Security	1	11.3	MIFARE Write	24
1.4	Delivery options	2	11.4	MIFARE Increment, Decrement and Restore	25
2	Features and benefits	2	11.5	MIFARE Transfer	27
2.1	EEPROM	2	12	Limiting values	28
3	Applications	2	13	Characteristics	28
4	Quick reference data	2	14	Wafer specification	29
5	Ordering information	3	14.1	Fail die identification	29
6	Block diagram	3	14.2	Package outline	30
7	Pinning information	4	14.3	Bare die outline	32
7.1	Pinning	4	15	Abbreviations	33
8	Functional description	5	16	References	34
8.1	Block description	5	17	Revision history	35
8.2	Communication principle	5	18	Legal information	36
8.2.1	Request standard / all	5	18.1	Data sheet status	36
8.2.2	Anticollision loop	5	18.2	Definitions	36
8.2.3	Select card	6	18.3	Disclaimers	36
8.2.4	Three pass authentication	6	18.4	Trademarks	37
8.2.5	Memory operations	7	19	Contact information	37
8.3	Data integrity	7	20	Tables	38
8.4	Three pass authentication sequence	7	21	Figures	39
8.5	RF interface	8	22	Contents	40
8.6	Memory organization	8			
8.6.1	Manufacturer block	10			
8.6.2	Data blocks	10			
8.6.2.1	Value blocks	10			
8.6.3	Sector trailer	11			
8.7	Memory access	12			
8.7.1	Access conditions	13			
8.7.2	Access conditions for the sector trailer	14			
8.7.3	Access conditions for data blocks	15			
9	Command overview	16			
9.1	MIFARE Classic command overview	16			
9.2	Timings	16			
9.3	MIFARE Classic ACK and NAK	17			
9.4	ATQA and SAK responses	18			
10	UID Options and Handling	19			
10.1	7-byte UID Operation	19			
10.1.1	Personalization Options	19			
10.1.2	Anti-collision and Selection	20			
10.1.3	Authentication	21			
10.2	4-byte UID Operation	21			
10.2.1	Anti-collision and Selection	21			
10.2.2	Authentication	21			

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© NXP B.V. 2011.

All rights reserved.

For more information, please visit: <http://www.nxp.com>
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 2 May 2011
196430



Annex G: Truck Data App (confidential)

This confidential section will be provided in Deliverable 3.1.



Annex H: Kitchen Data App (confidential)

This confidential section will be provided in Deliverable 3.1.



Annex I: Data Ingestion Template (confidential)

This confidential section will be provided in Deliverable 3.1.



Comments from external reviewers

External reviewer 1

DATE: 4.12.2017

Issue	Yes	No	Score (1=low to 5=high)	Comments
Is the format of the document correct?	X		4	Only minor comments (see text)
Does the format of the document meet the objectives of the work done?	X		5	
Does the index of the document Collect precisely the tasks and issues that need to be reported?	X		5	
Is the content of the document clear and well described?	X		5	
Does the content of each section describe the advance done during the task development?	X		5	
Does the content have sufficient Technical description to make clear the research and development performed?	X		5	
Are all the figures and tables numerated and described?	X		5	
Are the indexes correct?	X		5	
Is the written English correct?	X		3	Only minor changes/corrections (see text)
Main technical terms are correctly referenced?	X		5	
Glossary present in the document?		X	5	Terms are well described no need for glossary

Name: Michael Kornaros

Email: kornaros@chemeng.upatras.gr

Partner: UPATRAS

External reviewer 2

DATE: 11.12.2017

Issue	Yes	No	Score (1=low to 5=high)	Comments
Is the format of the document correct?	X		4	The colors of the inserted tables could be different.
Does the format of the document meet the objectives of the work done?	X		5	
Does the index of the document Collect precisely the tasks and issues that need to be reported?	X		5	
Is the content of the document clear and well described?	X		5	
Does the content of each section describe the advance done during the task development?	X		5	
Does the content have sufficient Technical description to make clear the research and development performed?	X		4	
Are all the figures and tables numerated and described?	X		5	
Are the indexes correct?	X		5	
Is the written English correct?	X		5	Really easy to read.
Main technical terms are correctly referenced?	X		5	
Glossary present in the document?		X		

Name: Konstatinos Nikakis**Email: kostasnikakis@hotmail.com****Partner: ENBIO**

External Reviewer 3

DATE 11.12.2017

Issue	Yes	No	Score (1=low to 5=high)	Comments
Is the format of the document correct?	X		4	Cross-references in the whole document are missing
Does the format of the document meet the objectives of the work done?	X		5	
Does the index of the document Collect precisely the tasks and issues that need to be reported?	X		5	
Is the content of the document clear and well described?	X		4	More similar images are inserted in the section 5 and it has not been referenced in the text.
Does the content of each section describe the advance done during the task development?	X		5	
Does the content have sufficient Technical description to make clear the research and development performed?	X		4	Technical details in the section 6 are missing.
Are all the figures and tables numerated and described?		X	3	Some figures description and numeration are missing. Table of figures is missing.
Are the indexes correct?	X		4	Cross-references in the whole document are missing
Is the written English correct?	X		5	
Main technical terms are correctly referenced?	X		5	
Glossary present in the document?		X		

Name: Dario Pellegrino**Email: Dario.pellegrino@eng.it****Partner: ENG**