URBANIZED

MODULAR AND FLEXIBLE SOLUTIONS FOR URBAN-SIZED ZERO-EMISSIONS LAST-MILE DELIVERY & SERVICES VEHICLES

# URBANIZED D3.2: Multi-layer EMS interfacing and functionalities

| | |
|---|---|
| **Primary Author(s)** | Robinson Medina, Nikos Avramis \| TNO |
| **Related Work Package** | 3 |
| **Version/Status** | 1.1 \| Final |
| **issue date** | |
| **Deliverable type** | R |
| **Dissemination Level** | Public |

| | |
|---|---|
| **Project Acronym** | URBANIZED |
| **Project Title** | modUlaR and flexible solutions for urBAN-sIzed Zero-Emissions last-mile Delivery and services vehicles |
| **Project Website** | https://urbanized.eu/ |
| **Project Coordinator** | Salvador Ruiz \| IDIADA |
| **Grant Agreement No.** | 101006943 |

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101006943.

# Co-Authors

| Name | Organisation |
|---|---|
| **Ruud Roelen** | TNO |
| **Lamberto Salvan** | ALKE |
| **Georgia Ayfantopoulou, Josep Maria Salanova Grau** | Certh |
| **Duong Tran, Mohamed Mahedi Hasan, Omar Hegazy** | VUB |

Supporters

| Name | Organisation |
|---|---|
| **Zisis Maleas** | Certh |

# Reviewers

| Version | Date | Reviewer (Name/Organisation) | Action (Checked/Approved) |
|---|---|---|---|
| **1.0** | 19/01/2022 | Steven Wilkins | Checked |
| **1.0** | 04/02/2022 | Rodrigo Lopez Martin (IDIADA) | Checked |
| **1.1** | 24/02/2022 | Victor Desmots / Salvador Ruiz | Approved |

# Document history

| Version | Date | Author(s) | Status* | Dissemination level** |
|---|---|---|---|---|

*Status: Draft, Final, Approved, Submitted (to European Commission).

**Dissemination Level: **PU**: Public; **CO**: Confidential, only for members of the consortium (including the Commission Services); **EU-RES** Classified information - restraint UE; **EU-CON**: Classified information - confidential UE; **EU-SEC**: Classified information - secret UE

# Copyright statement


## Keywords

Energy management system, fleet management tools, components interfacing, components functionalities, multi-layer EMS

# Executive summary

Urbanized develops and demonstrates the next generation of modular vehicle architectures for urban commercial e-vehicles. Such architectures are based on optimal design principles and vehicle right-sizing for a particular mission. To help on accomplishing this goal, a multi-layer Energy Management Systems (EMSs) is to be developed for the next generation modular vehicle. Such EMSs are developed in several stages throughout the project scope, as Figure 1 shows.

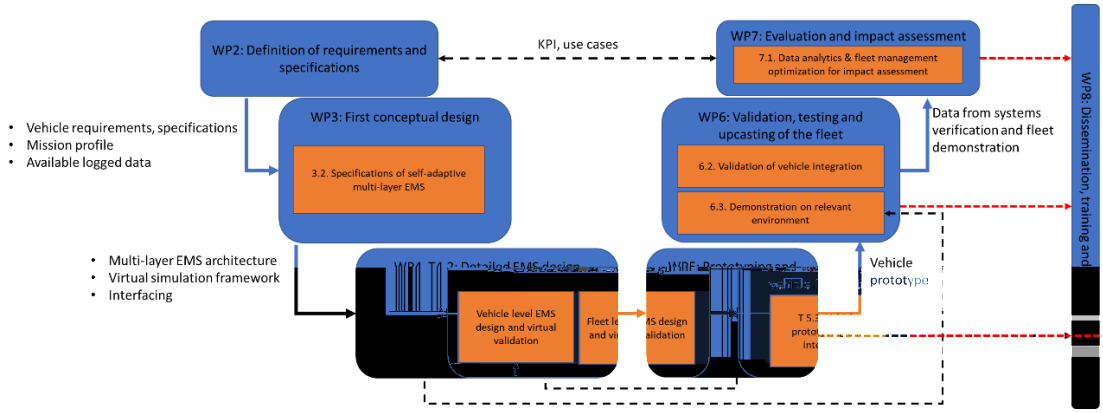


Figure 1: Work package structure of Urbanized in the context of EMS-related developments

The work presented in this report corresponds to the second step of the development of the EMSs, which is carried out in work package 3. This step is carried out in Task 3.2, which is focused on detailing the architecture and agreeing an interface and functionalities of the EMSs at vehicle and fleet level. Moreover, this task also reports the development and integration process of the multi-layer EMS into the virtual tool developed in Task 3.1. Note that the detailed development of the EMSs together with the testing and integration of these algorithms is to be carried out in next work packages.

In the context of Task 3.2, for defining the interfacing between multiple components and the vehicle architecture, an interfacing tool has been developed. Such a tool is a blend of commonly used software tools among the partners in the consortium, such as Microsoft Office and Matlab Simulink. The tool provides a visual diagram of the vehicle interfacing, as well as a comprehensive table with the details of each connection.

The document continues with an overview of the vehicle and cloud architecture and a detailed explanation of the functionality of the EMSs. Additionally, it is also provided some brief explanation of the functionality of other components in the architecture, which are related to the EMSs functions.

Using the previously described tool, the connections between each one of the components in the architecture is defined. Such connections are derived around the EMSs functions to provide the desired functionality described in the previous chapter. Each one of the connections is detailed with relevant information such as physical layer, sampling periods, data type, etc. Each

one of the connections is identified according to a unique name, which is defined according to a naming convention. The naming convention is also agreed with the involved partners.

Finally, the interfacing of the EMSs and two simulation tools is explained: the virtual simulation tool from Thessaloniki Living Lab for the cloud-based developments, and the vehicle simulator of Task 3.1 for the on-board functions.

# Table of contents

# List of figures

# List of tables

# 1. Introduction

The surge on the size of urban centres is making the management of logistics a more complex task. The difficulty of this task arises from the multiple types of deliveries and the always increasing number of daily operations. Additionally, current climate goals aim to reduce the total Green Houses Emissions by tackling first the emissions generated by the transport sector. Urbanized helps to cope with these challenges by developing a new generation of electric delivery vehicles with the following three main innovation lines:

1. High-performance e-powertrain components and control architectures, through the use of advanced co-design approaches. Such a high-performance e-powertrain helps to reduce losses in the vehicle, which in turn reduces the total energy requirement per delivery.
2. Interchangeable, plug & play cargo modules for different urban freight transport use case scenarios. Such modules help to quickly setup different cargo bodies, which allow a single vehicle to perform multiple kind of deliveries in an urban environment.
3. Integrated energy and fleet management strategies using data, connectivity and machine learning algorithms. These strategies allow for seamless integration of the electric vehicles into current logistics plans, making it more suitable for urban deliveries.

This document focuses on the third innovation line: energy and fleet management strategies, which is commonly referred to as Energy Management Systems (EMSs). To enable this innovation, four EMSs are to be developed in Urbanized:

- Eco-Driving, which provides a speed advice to the driver to optimize the energy consumption in the vehicle and reduce losses in the powertrain;
- Eco-Comfort, which provides temperature setpoints to the thermal systems in the vehicle to increase the efficiency of the thermal systems;
- Eco-Charging, which regulates the charging patters of a fleet of vehicles to reduce losses during the charging operation; and
- Eco-routing, which plans the routes of a fleet of vehicles to minimize the amount of energy used while driving.

These EMSs are to be implemented in a vehicle layer running onboard of the vehicle (Eco-Driving and Eco-Comfort), and in a fleet-level layer running on servers (Eco-Charging and Eco-routing), making the whole approach a multi-layer Energy Management System, as depicted in Figure 2.

The development and integration of such EMSs requires a clear definition of the functionality to be implemented per layer, as well as a detailed list of connections between them. Therefore, this document presents the activities conducted in Work Package 3, Task 3.2, which are the following:

- To define the functionality of each one of the EMSs, such that their application is aligned not only with the required functionality of the use cases (top down) but also with the capabilities of the vehicle being design (bottom up);
- To specify the interfaces (i.e., inputs and outputs) of the EMSs with the vehicle components (e.g., ECUs, Base Vehicle), cloud components (logistics); and between

each other. This interfacing provides clarity on the data needed, such as frequencies of transmission, formats, which are required to implement the EMSs.

- To present the interaction of the EMSs with the simulation environments available for the project: a vehicle simulation environment and a cloud simulation environment (Thessaloniki Living lab).

In the following chapters these activities are further detailed. This document is divided as follows. It starts by describing an interfacing tool developed for fulfilling the objectives of tasks in this deliverable; then, a functional specification of the each one of the EMSs to be developed and some supporting components is defined; next, the agreed interfacing between the proposed EMSs and supporting components is specified; last, the interaction between EMS and the virtual simulation environment is presented.

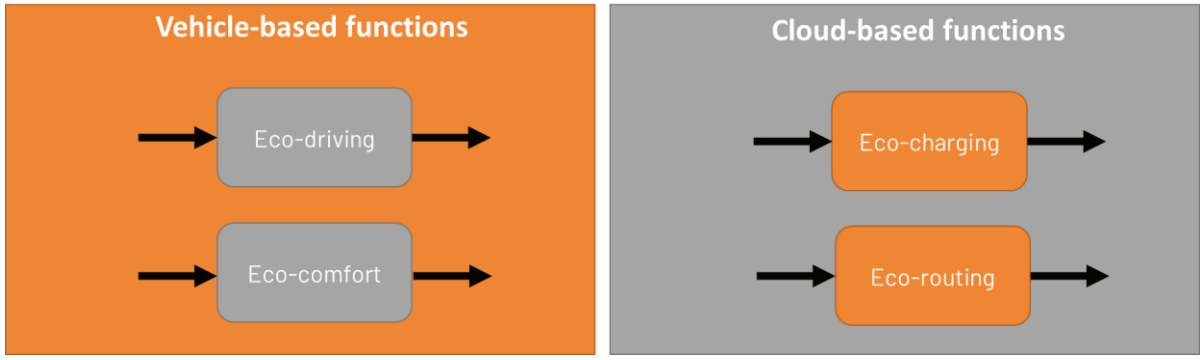


Figure 2. EMS functions to be developed

# 2. Approach: interfacing tool

In order to align the interface between software modules developed by multiple project partners, an interfacing tool has been developed. In the next subsections, such a tool is explained.

## 2.1 Motivation

The interfacing alignment between software modules developed by multiple partners is a complex task due to the multiple HW platforms (cloud, ECUs, servers, etc). functional modules (e.g., Base Vehicle), different complexities of signals, and the large amount of information that needs to be exchanged.

There are some commercial tools developed specifically to align interfaces. However, they require to purchase licenses among all the project partners, which can cause extra delays while purchasing the components on top of the added cost.

This motivates the development of a tailor-made interfacing tool, which utilizes commonly available software solutions (i.e., Microsoft office and Matlab Simulink) which are readily available among most of the project partners.

The developed tool allows the following features:

- Simultaneous online collaboration of multiple partners using a graphical user interface based on Matlab Simulink. Each users takes ownership of one component in the complete interfacing.
- Automatic connection of inputs and outputs with identical names. This allows to corroborate that an output defined by one partner is the input required by another partner(s).
- Automatic generation of an interface report. Such a report consists of tables detailing each one of the connections (sampling periods, sizes, data type, etc.) as well as issues found while trying to connect the ports (mismatches in data types, sampling periods, etc.).
- The developed interface in Matlab Simulink can be reused as a base in future work-packages to test and simulate developed software.

A detailed manual for using the interfacing tool is provided in (Medina , Avramis , & Purnot, 2021).

## 2.2 Naming convention

The developed interfacing software tool requires the same name between an input and an output port, so that a signal is created between these interfaces. Devising a structured naming convention for inputs and outputs is therefore essential for the correct working of the tool. Additionally, having a naming convention allows to provide some preliminary data about the type of information being exchanged (e.g., type of signal, description, units, etc.), as well as to guarantee that each one of the signals is only implemented once (e.g., there is only one signal providing the vehicle SoC).

Based on this, each one of the inputs and outputs in the naming convention follows the structure shown below:

*{ComponentName}_{Description}_{TypeOfSignal}_{Units}*

Where,

- {ComponentName}: corresponds to an abbreviation that identifies the source component of the signal or parameter. The list of all the component abbreviations is shown in Table 2 in the Appendix.
- {Description}: this field is a short concatenation of abbreviations of two or more keywords to identify the signal. A full list of abbreviations is shown in Table 3.
- {TypeOfSignal}: provides an abbreviation which identifies the physical nature of the signal in the vehicle, e.g., hydraulic, mechanic, electrical, etc. Table 4 shows a complete list of type of signals.
- {Units}: in this field, the unit is specified preferably according to the International System of Units (SI). A full list of used units is shown in Table 5.

Due to the architecture hierarchy, a signal might pass through several blocks until it reaches its final destination. Thus, attentiveness to the naming convention is paramount such that a signal name is changed accordingly every time the signal leaves a block and enters another. One example is shown in Figure 3, where these changes are depicted: The Field {ComponentName} changes every time based on the block that originates the signal, while the field {TypeOfSignal} changes when there is a different communication protocol between the two blocks. Also, the field {Description} is extended when it is used on a fleet level, while the field {Units} is the only one that does not change throughout the path.

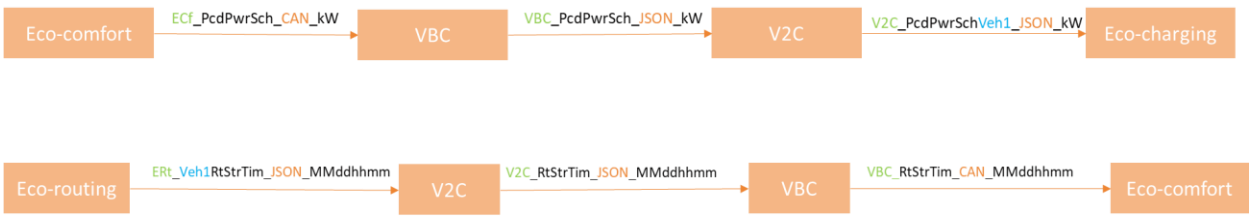


Figure 3. Naming change between blocks

## 2.3 Tool outputs

The outputs of the interfacing tool are twofold:

- A graphical interface based on a Simulink diagram, which contains all the inputs and the outputs of each component in the interfacing; An example of the Simulink diagram is shown in Figure 4. The components are grouped in different categories: running on the vehicle (Alke's vehicle), running on the cloud (cloud functions), and located in the depot

(charge infrastructure). Note that each connection has an independent colour, so that matching lines can be easily found.

- An excel file with a description of each one of the interfaces per component. The generated excel file contains one tab per component in the interfacing diagram (Figure 5). In each tab in the excel, a table reporting the inputs and the outputs of the component is shown. Table 1 shows an example of a table with the inputs for the charging point. Here, the details of the connections can be found: name, description, units, source, etc. Note that the errors column shows potential connection issues, such as unit mismatches, sampling period mismatches, etc.

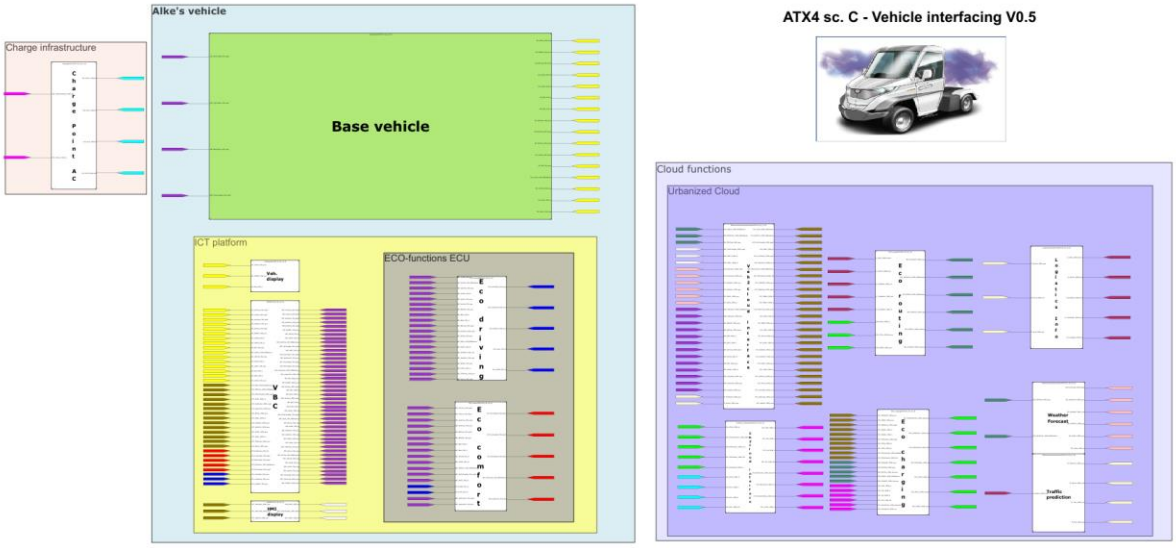


Figure 4: Interfacing tool output: diagram overview

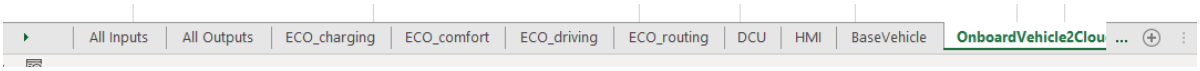


Figure 5: interfacing tool output: excel tabs

Table 1: Interfacing tool output: example of table with connections details

| Input Location | Input name | Description | Data type | Dimensions | Sample Time [s] | Unit | Source output block | Errors |
|---|---|---|---|---|---|---|---|---|
| **ECO_charging** | C2C_ChSt_JSON_dl | Chargers status, one row per charger, 0=disabled | int8 | [10 1] | 60 | dl | Ch2Cloud_Interface | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | , 1=ready, 2=error, 3=in use. | | | | | | |
| **ECO_charging** | C2C_ChU_JSON_V | Chargers output voltage, one column per phase, one row per charger | double | [10 3] | 60 | V | Ch2Cloud_Interface | |
| **ECO_charging** | C2C_ChI_JSON_A | Chargers output Current, one column per phase, one row per charger | int8 | [10 3] | 60 | A | Ch2Cloud_Interface | |
| **ECO_charging** | C2C_ChP_JSON_kW | Chargers output power, one row per charger | int8 | [10 1] | 60 | kW | Ch2Cloud_Interface | |

# 3. Functional description of main software modules in the architecture

The development of the energy-management related innovations in Urbanized, requires the interfacing of new software modules within Alke's existing vehicle at multiple levels: at cloud level and at vehicle level.

## 3.1 Architecture overview

An overview of the envisioned architecture is shown in Figure 6. The architecture is composed of two types of elements:

- Software modules: these elements are software blocks with a specific functionality. Examples of it are the EMSs blocks and the interfaces blocks.
- Functional modules: these elements emulate the behaviour one or multiple physical components in the architecture. Examples of it are the Base vehicle, the charging point, and the dashboards.

Additionally, notice that multiple physical layers are observed in the interfacing:

- Alke's vehicle: this level corresponds to everything to be implemented onboard of the vehicle. In this level, the existing functional modules in the vehicle are interfaced with the innovative energy management modules: Eco-Comfort and Eco-Driving. Additional functional modules are considered for displaying purposes (e.g., dashboards) and for connectivity purposes (Vehicle Body Computer).
- Cloud functions: this levels runs all the software modules related to the cloud, such as the fleet management tools. Here, two of the innovative EMSs software modules are implemented: Eco-Routing and Eco-Charging. Additional software modules are considered to interface with logistics planners (Logistics Information), to interface with the vehicle (Vehicle to Cloud Interface), and to connect with the charging point (Charger to Cloud Interface).
- Charge infrastructure: corresponds to the charging point used to charge the vehicle. This functional module provides live information about the charging point status (usability, power available, etc), such that it can be included in the logistics plan.




Figure 6: General overview of components developed

Figure 6 also shows how the cloud, charge infrastructure and vehicle components exchange information with each other using a connection to internet. The software modules that enable such a connectivity are the vehicle to charger interface and Charger to Cloud interface, in the cloud functions, as well as the VBC on board of the vehicle.

All elements in the architecture exchange several packages of information to guarantee the correct operation of the vehicle. An overview of the dependencies and general data exchange is shown in Figure 7 for the cloud, while in Figure 8 for on the on-board elements.

For the cloud elements, it is expected that they will be run at the end of the daily shifts, starting by the components that provide the day-ahead work: Logistics Information, Charger to Cloud Interface, and Vehicle to Cloud Interface. After that, the Eco-Routing algorithm will run to provide the day ahead vehicle routes. Eco-Charging will be executed next to provide a charging schedule such that the routes can be executed successfully. Notice that the Eco-Charging algorithm is executed several times during the charging process to update the progress and collect relevant information. Finally, the Vehicle to Cloud Interface and the Charger to Cloud Interface will execute to communicate the vehicle routes and charging schedules to the respective components.

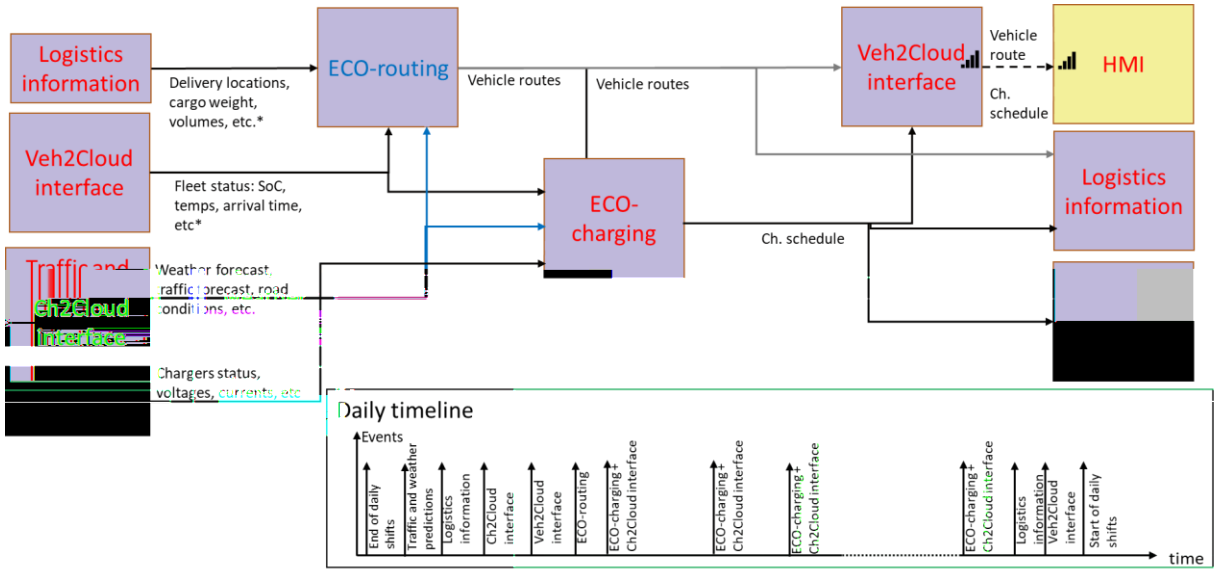


Figure 7: Execution sequence of online vehicle modules

For the onboard elements of the architecture, they will be continuously run while the vehicle is being used. First, the VBC and Base vehicle functional modules will provide all the relevant information to the Eco-Driving function. The Eco-Driving will compute an optimal speed profile to be followed by the driver. Additionally, the Eco-Driving will also provide an expected power demand resulting from using the suggested optimal speed profile, so that the Eco-Comfort can generate the desired temperature setpoints. Finally, the VBC will transmit the result of the eco-functions to the relevant functional components.

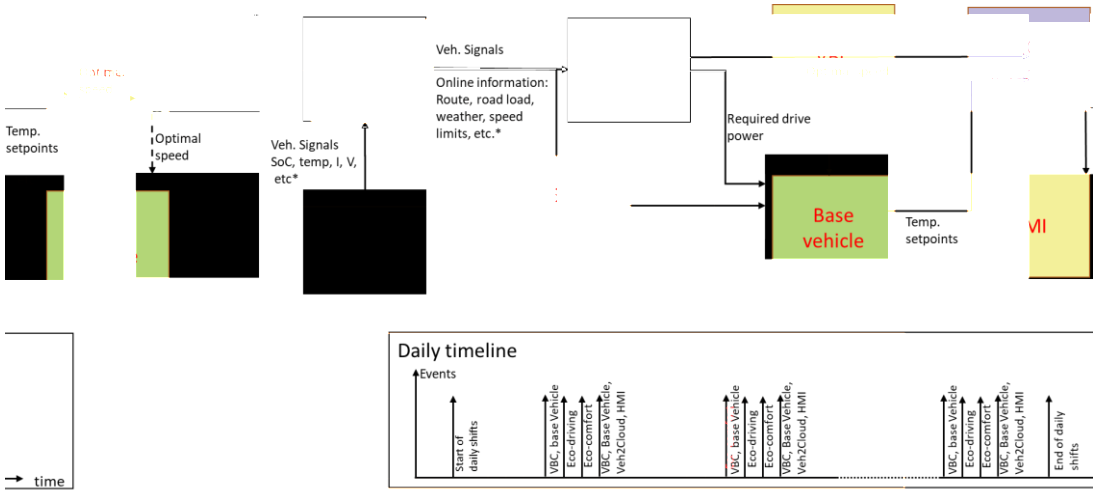


Figure 8: Execution sequence of on-board vehicle modules

In the next subsections, a detailed description of each one of the elements considered in the architecture is presented.

## 3.2 Cloud functions software modules

Cloud-based software modules are designed to be implemented in servers (e.g., via an API). Such cloud components communicate with the vehicle and among each other using an internet connection. A description of the functionality of these components is shown next:

### 3.2.1. Eco-Charging (ECh)

The Eco-Charging algorithm corresponds to one of the EMS layers running in the cloud of Urbanized and actuating over the whole fleet of electric vehicles. The Eco-Charging is the software component that provides a charging schedule for the fleet of electric vehicles. Such a schedule is composed of three items:

- when is each one of the vehicles going to be charged;
- with which power is each of the vehicles going to be charged;
- at which charger is the vehicle going to be assigned.

An example of a charging schedule is provided in Figure 9. Note that in case driver breaks are taking place in the depot location, the Eco-Charging can schedule short charging sessions, when necessary.

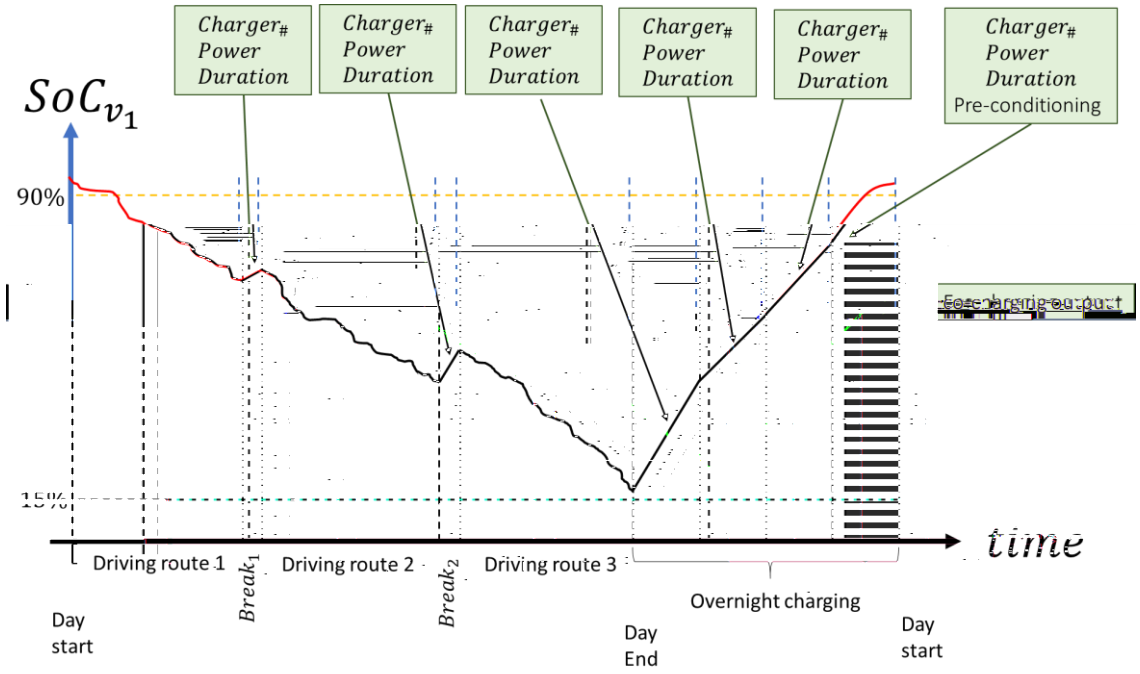


Figure 9: Eco-Charging algorithm example in one vehicle

To provide such a charging schedule, the Eco-Charging algorithm shown in Figure 10, will internally run the following steps:

- Fast route-profile generator: using trajectories (GPS routes and stop times) provided by the Eco-Routing algorithm, the Eco-Charging algorithm will use online databases to determine relevant route-related information, such as grade, speed limits, traffic flows, etc.

- Simplified vehicle models: using a simplified vehicle model, the energy requirements for the provided route will be computed. This step will be focused on determining battery status after the vehicle has driven the routes: battery state of charge, temperature, etc. Additionally, charging models are used to determine the charge time of the battery for the available charging power and the corresponding battery degradation. These simplified vehicle models will be automatically updated (in case necessary) when the vehicle body changes, making the whole approach self-adaptive.
- Optimizer: the optimizer will find the charging schedule based on a cost function and a set of constrains (see the next sections). The optimizer will use the charging model to reduce battery degradation (due to charging) and to estimate charging time. Such models are highly dependent on the battery age (e.g., the battery resistance). Therefore, a self-learning strategy will update the battery model in different battery stages to provide up-to-date predictions for the charging process.

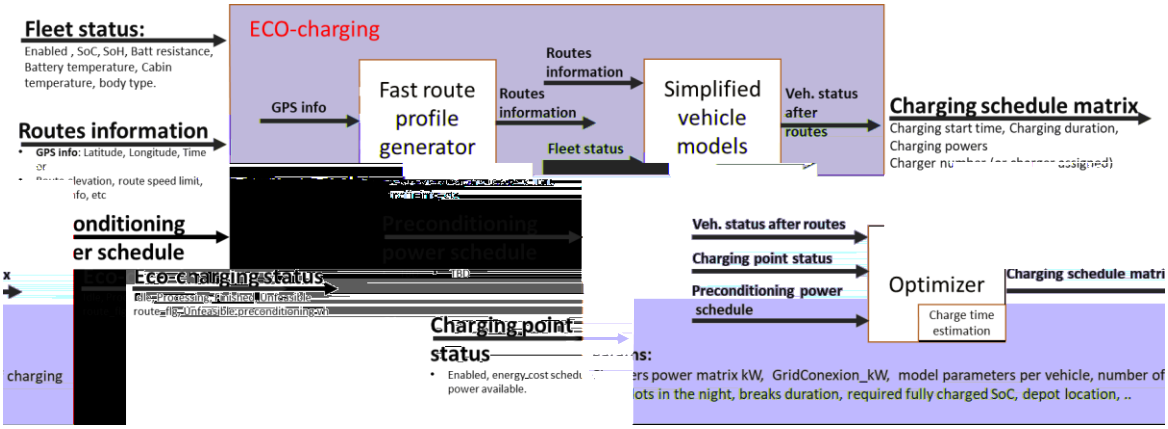


Figure 10: Eco-Charging functional description

## Eco-Charging: Objective of minimization

The objective of the cost function in the Eco-Charging is to minimize the operational cost of the fleet of vehicles. We consider only the operational cost that can be influenced with the charging process.

$$J = \min_{CS} OC$$

Where $OC$ corresponds to the operational costs [€], $CS$ is the charging schedule and $J$ is the resulting minimum cost. The operational costs are to be defined as function of the battery degradation costs, the electricity costs, and driver costs.

## Assumptions

- One charging point per vehicle is available, no reconnection of the vehicles is necessary during overnight charging. Likewise, each vehicle has a fixed charging point.
- Fleet is composed of vehicles of the same kind in different battery age states.
- Vehicles are only charged in the depot (no public charging is allowed).
- Vehicles are fixed to a specific route by the routing algorithm.

- The algorithm will run periodically overnight and during breaks if opportunity charging is allowed.
- The algorithm will include preconditioning of the thermal systems, when requested by the vehicle via the Eco-Comfort algorithm.

### Algorithm implementation

The Eco-Charging algorithm will be implemented in a server. To generate a charging schedule, an API will be developed using the interface described in the next sections.

## 3.2.2. Eco-Routing (ERt)

The Eco-routing software component is responsible for the scheduling of routes that vehicles should follow to serve a set of requests (destination points to be visited). The goal of a typical routing service is to receive a set of requests and produce the optimal sequence of actions the vehicles should take to serve the demand. In general, Vehicle Routing Problems (VRPs) are used to calculate cases of a fleet of k vehicles and n customers.

There are many variants that a VRP can take to solve more realistic instances. VRPs are usually differentiated by the constraints of the routing problem. The capacity constraints are the most known and ensure that the capacity of the vehicles will not be exceed. Furthermore, constraints like the Time Window (TW) are also common, which force an approximate arrival time, as Figure 11 shows. Another variant of VRPs lies on the dynamic or static nature of the problem. The dynamic case is used when the requests are still arriving as the vehicle starts the route. The static case solves the optimization problem before the vehicle starts the route, as it assumes that all the necessary information is predefined. Lastly, other types of factors are considered for electric vehicles, as battery SoC and recharging duration constraints are also of interest.

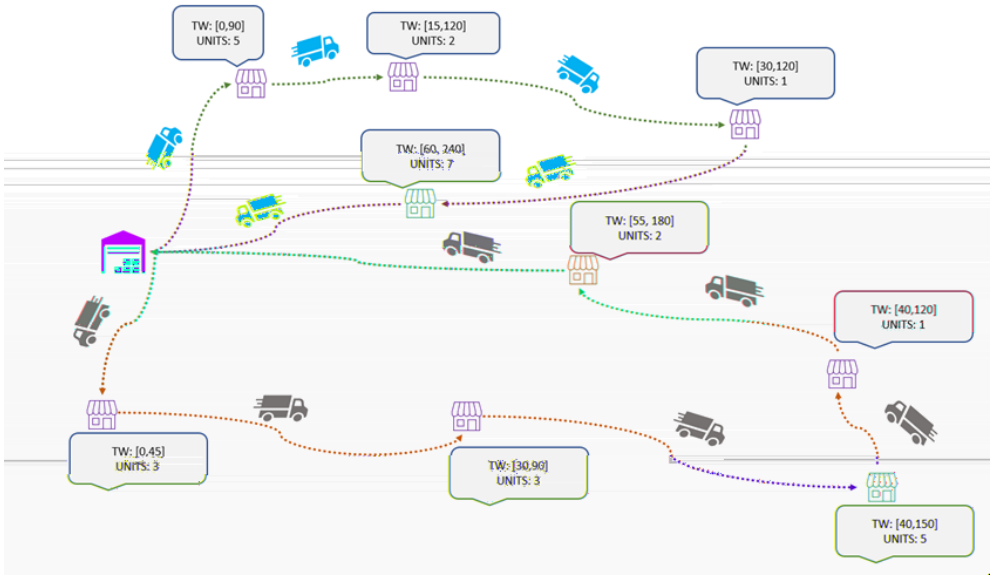


Figure 11: Typical VRT with Time Windows (TW) for arrival at each destination.

The aim of a VRP is to produce optimal sequences of actions the driver should perform, which do not violate the set of constraints. The objective function described in the equation below,

typically minimizes that total distance travelled, the total trip duration or the waiting time of the costumers.

Given:

$G = \{V, A\}$ Graph with the set of stops V and the set of Arcs (connecting the stops) A.

$$x_{ij} = \begin{cases} 1, & If\ vehicle\ moves\ from\ stop\ i\ to\ j \\ 0, & else \end{cases}$$

$$c_{ij} = The\ consuption\ if\ vehicle\ moves\ from\ stop\ i\ to\ j$$

$$\min \sum_{i \epsilon V} \sum_{j \epsilon V} c_{ij} * x_{ij} \qquad (1)$$

In Urbanized, the goal of the optimization function is to derive the most eco-friendly solution. To do so, instead of the typical measures from above, the objective function considers the energy consumption of travelling from one node to another.

The solution of the optimal problem follows Mixed Integer Programming (MIP) formulations, such as the ones described in (Ralphs, 2003) and (Kallehauge, 2005). The algorithm computes all the routes before the start of the operation. Thus, the current module implements the static case of routing problems, assuming that there will be so significant changes in the requests as vehicles operates. An overview of the routing algorithm is shown in Figure 12.

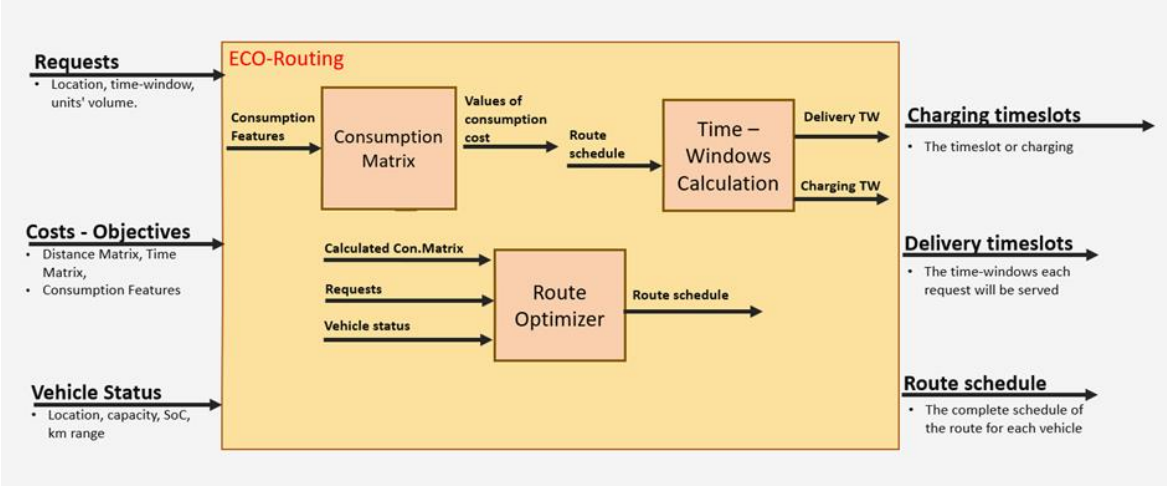


Figure 12: Eco-routing functional description.

The basic operation of the Eco-Routing algorithm can be discretized in 3 steps. The collection of requests and cost inputs, the processing, and the extraction of outputs.

The collection of requests and inputs are:

- Requests for delivery including the position of demand point, the number of necessary units, and the time window of each one.
- The distance and time between each pair of requests.
- Features that help to calculate emissions. (Congestion – traffic, temperature, elevation, mean speed, parking space, road quality, SoC)
- State of the fleet, such as capacity, SoC, km range.

The processing corresponds to:

- The consumption matrix calculation based on the consumption features.
- The core VRP optimizer that generate the route sequences.
- The sensitivity analysis tool that generates acceptable upper and lower bounds of time-frame that each request can be served.

The extraction of outputs corresponds to:

- The sequence of actions the vehicle should follow.
- The acceptable delivery timeslots.
- The required battery and the available timeslots for recharging.

The Eco-Routing modules will be run on the cloud and all the communications are handled via REST APIs in Json format.

### 3.2.3. Vehicle to Cloud (V2C) Interface

The interface between the vehicle and the cloud platform is done by the functional module Vehicle Body Computer (VBC). The VBC has an internal web client that opens an encrypted channel with the cloud to make REST requests.

### 3.2.4. Charger to Cloud (C2C) Interface

The Charger to Cloud interface is a software component bridges the Eco-Charging algorithm and the charging points. The main function of this component is to provide relevant charging information to the Eco-Charging algorithm such as energy prices, chargers availability and power, etc. Likewise, this component will communicate to the charging points the resulting charging schedule generated by the Eco-Charging.

This block was considered independent from the Eco-Charging, since it needs to consider the multiple possibilities of interfacing with charging infrastructure: e.g., DC or AC platforms, multiple communication infrastructures, communication with grid operator, etc.

This block is to be implemented online, using the same platform as the Eco-Charging algorithm.

### 3.2.5. Logistics Information (LI)

The Logistics Information software component represents all the actions and decisions that take place in the depot and management stage of the last-mile logistics operations. The main assumption is that the information about the requests is known in advance. Thus, this module requires no inputs. All the outputs are generated either via random delivery schedules (for simulation purposes), or with the use of a web service and a database system (for real-world use

cases). This module is also responsible to orchestrate the flow of operations between Eco-Routing and Eco-Charging (Controller of Eco-Routing and Eco-Charging interaction). An overview of the logistic operator is shown in Figure 13.

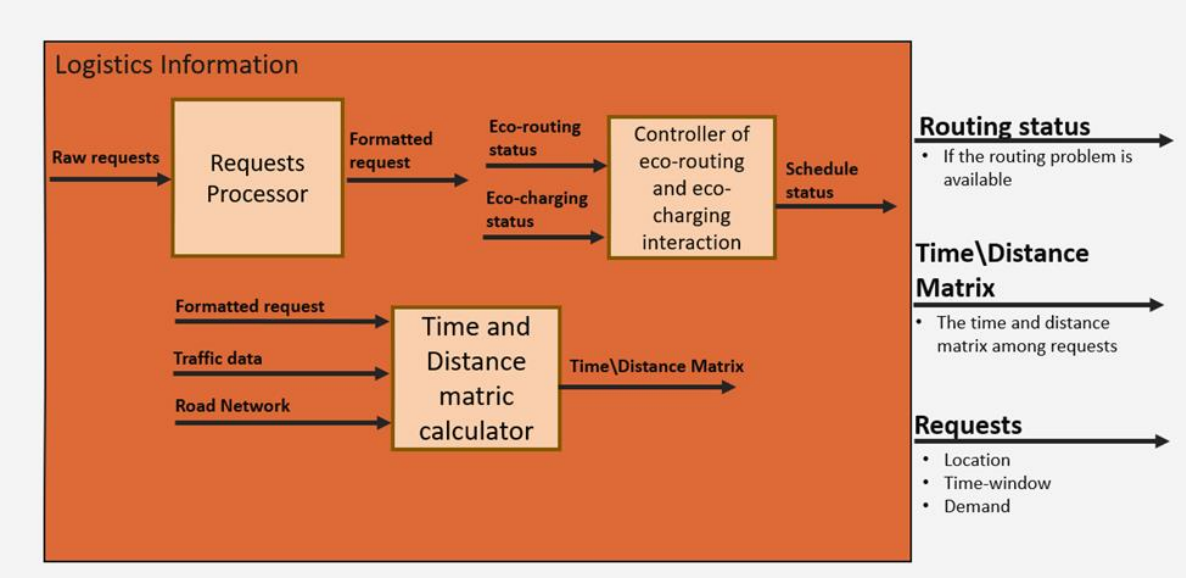


Figure 13: Logistic operator functional description.

The Logistic Information also contains one block that collects and formats the requests, as well as the time/distance matrix calculator based on real road network data and traffic data. The Logistics Information is responsible only to ensure that all the algorithms executed properly.

### 3.2.6. Weather Forecast Information

The algorithms that are to be developed in the project require information from weather web services, in order to improve their prediction functionalities. The Weather Forecast Information is a software component that will request such information from online databases via an API call and will feed it to in-vehicle functions, namely Eco-Comfort and Eco-Driving. The forecasted outputs of this module are temperature, humidity, precipitation and visibility, as well as wind speed and wind direction.

### 3.2.7. Traffic Prediction Information

Traffic Information is required by the Eco-Driving and Eco-Routing functions to create an optimal trajectory and vehicle kinematics setpoints, which minimizes the energy consumption and allow the vehicle to reach its destination within the prescribed time windows. Therefore, this software component extracts traffic information from online real-time databases that keep track of the traffic density in all of a city's road network. Also, It reads information from databases that store the legal velocity constraints (both the upper and lower speed limits) of the roads.

## 3.3 On-board vehicle components and functional modules

All the items described in the architecture as on-board are designed to be run in the vehicle while it is operating. The most relevant software modules developed for this implementation are the Eco-Driving and Eco-Comfort EMSs. Additional functional modules are

## Energy saving features of Eco-Driving

Eco-Driving saves energy using three main approaches; the first is by reducing the tractive energy demand using the methods described above, the second is by optimizing the energy extracted from regenerative braking, and the third is by activating Eco-Comfort features to reduce thermal energy demand by regulating cabin temperature in case of low energy availability. Furthermore, due to the reduction in tractive power requirements during Eco-Driving, the battery cooling requirements are also reduced, further reducing power requirements.

## Execution frequency of the Eco-Driving algorithm

The Eco-Driving algorithm, once activated, will continuously execute locally in the Eco-functions ECU. Different modules of the Eco-Driving algorithm may execute at different frequencies, with the vehicle sensor suite, which monitors the vehicle's environment, executing at a higher rate, and the getting route and weather information from the cloud executing at a lower rate.

## Eco-Driving algorithm methodology

The Eco-Driving feature is adaptable to a wide range of vehicle and cargo mass and chooses the optimally tuned control parameters regardless of the net weight of the vehicle. The algorithm should have the ability to estimate the total mass of the vehicle based on the vehicle's response to a given control command. The eco-feature can be activated by the driver using the HMI Display , which is continuously monitored for user input. If a signal is received, then the algorithm will enable Eco-Driving functionality, including:

- Retrieve weather information from the cloud, done once per hour for the geographical location of the vehicle.
- Communicate with the Eco-Routing module and receive the waypoint sequences for the next travel segment (medium frequency).
- Retrieve traffic information for the next travel segment from the cloud (low frequency).
- Sample sensor data (high frequency).
- Compute the reference vehicle speed and display on HMI.
- Monitor the battery SoC, If SoC falls below certain threshold, the eco-features, including Eco-Driving and Eco-Comfort will be activated regardless of the command from the driver, so that the vehicle range can be extended until a charging station is reached.
- Monitor the deactivation of the Eco-Driving signal. If the signal is not received, then repeat the previous six sequences.

## Eco-Driving: Objective of minimization

The objective of the cost function in the Eco-Driving is to minimize the energy consumption rate of the vehicle. The energy consumption rate depends mainly on the driving behaviour, which should satisfying the constraints placed by the traffic conditions of the route. The energy

consumption rate is defined as a function of the velocity, acceleration, grade of the road, the traffic profile of the route, and weather.

### Functional description of the Eco-Driving algorithm

Figure 15 shows a functional description of the Eco-Driving module, highlighting the data flow and signalling.

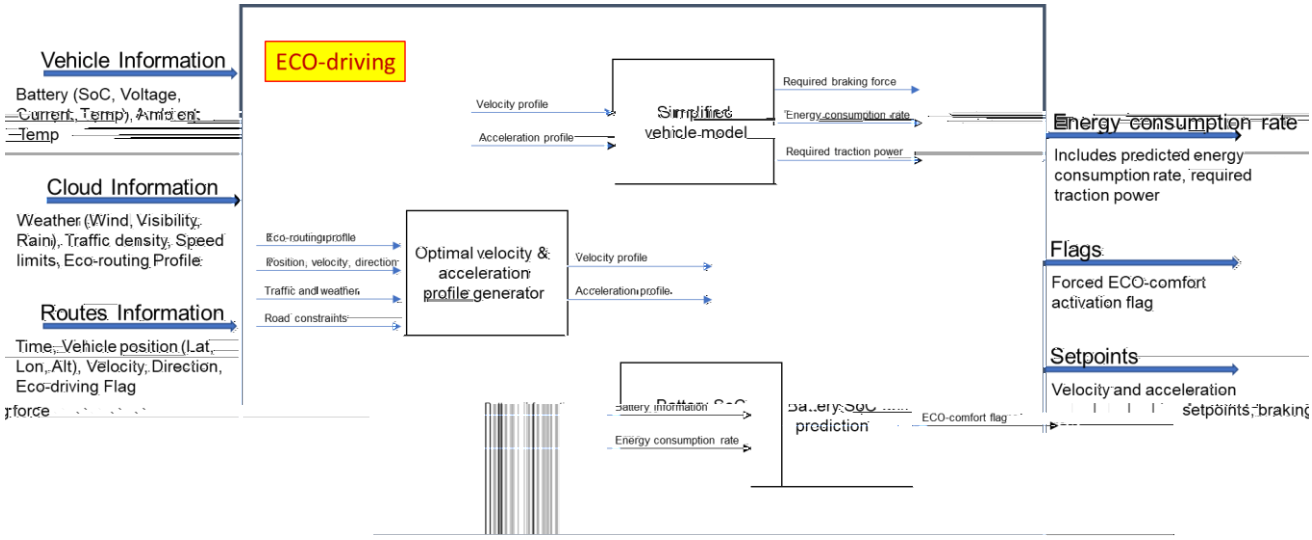


Figure 15. Eco-Driving functional description

The Eco-Driving software consists of three main functional components:

- Optimal velocity and acceleration profile generator: The Eco-Driving algorithm will take in the route information presented by the Eco-Routing algorithm, the traffic conditions gathered from the cloud, and the road elevation (grade) gathered from the GPS, to generate a modified velocity and acceleration profile that will minimize the traction energy required and maximize the braking energy recovered. Furthermore, the algorithm will take into consideration the road constraints including speed limits.
- Simplified vehicle model: using a simplified vehicle model, the energy requirements for the provided route will be computed and presented as energy consumption rate (kWh/km). The algorithm will also take the predicted energy requirements before the next charge to determine the battery status, including the final state of charge.
- Prediction: the up-to-date prediction algorithm of the battery SoC at the end of the route, so that it can be determined whether to give an activation command to the Eco-Comfort module. The activation command will be enabled if the predicted SoC falls below a pre-defined threshold.

## 3.3.2. Eco-Comfort (ECf)

The Eco-Comfort algorithm is the software component responsible for controlling the thermal behaviour of the vehicle components, aiming to minimize the energy consumption, while guaranteeing the driver comfort. The term thermal components refers to the cabin, the battery thermal circuit and the refrigerated cargo (in case it is connected to the vehicle body). The


minimization of energy consumption is achieved on two separate occasions: during driving and during charging the vehicle, where preconditioning can be enabled.

The algorithm receives inputs to estimate the available energy in the battery for heating/cooling, as well as temperature measurements from the installed sensors. Also, Weather Forecast Information is received OTA, so that the algorithm can predict thermal behaviour. Then, an optimization problem is solved to optimally distribute the available battery power, taking into account the physical constraints.

### Eco-Comfort: Objective of minimization

The objective of the cost function in the Eco-Comfort is to minimize the energy consumption of the

Optimizer: The goal of the optimizer is to compute the temperature setpoints that will lead to a minimized energy consumption of the HVAC, the battery heating/cooling circuit and the cargo freezer. These setpoints are sent to the low-level controllers of the vehicle, which will in turn implement them. Additionally, at the beginning of each delivery day, the optimizer will receive the estimated starting time of the route while the vehicle is still parked at the depot, and will preheat (or precool) the battery and the cabin. In this way, energy savings will be realized, since there will be less energy requested once the vehicle starts its schedule.

## Thermal modelling

For the driver comfort, a passenger comfort model will be developed, which allows to give freedom to the optimization process while keeping the driver comfortable. The two indexes that are used for defining thermal comfort are the Predicted Mean Vote (PMV) and Predicted Percentage Dissatisfied (PPD). The first index varies from -3 to +3 and indicates how does the passenger feel, where -3 is very cold, +3 is very warm and 0 is ideal comfort. The second index gives a prediction of the percentage of thermally dissatisfied occupants. The relationship between these two is depicted in Figure 17.

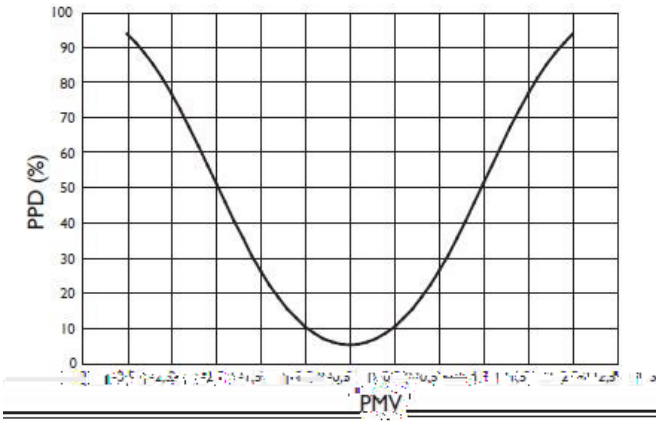


Figure 17. Thermal comfort indexes

## Preconditioning

Preconditioning is the process of conditioning the thermal systems of the vehicle, while it is still charging. The target of the preconditioning is to drive the thermal components to the optimal temperature using energy directly from the grid, while guaranteeing that the energy that is being charged is sufficient for the scheduled trip. This process can occur both for preheating and precooling, depending on the ambient temperature and weather forecast.

## Algorithm implementation

The Eco-Comfort algorithm is one of the software modules that will be implemented inside the "Eco-functions ECU". Such ECU is a separate controller that will be placed inside the vehicle for the URBANIZED project, specifically for Eco-Driving and Eco-Comfort. This controller can

communicate directly via CAN only with the Vehicle Body Computer (VBC), which is presented later in this chapter. The algorithm can be enabled in 2 ways: 1) from the driver, through the HMI Display Dashboard or 2) from the Eco-Driving module, if the measured SoC of the vehicle drops below a defined threshold.

### 3.3.3. Base Vehicle (BV)

The Base vehicle represents the ALKE vehicle block seen as vehicle drivetrain, battery, BMS, OBC, thermal system, lighting system and all other vehicle features excluding the ICT platform. This block generates most of vehicle status and driving data shared with the cloud platform and other eco-oriented blocks.

### 3.3.4. Vehicle Body Computer (VBC)

The Vehicle Body Computer is a functional module that has data storage capabilities and it is the communication bridge between the vehicle CAN bus network and the cloud platform.

### 3.3.5. Vehicle Display Dashboard (VDD)

This is the main display located on the vehicle dashboard just behind the steering wheel it shows all drive-oriented data. Besides displaying the real-time parameters and any anomalies (special indicator lights), it also manages troubleshooting and maintenance activities that can be triggered by special keys designed for interactive operation of various functions. Figure 18 shows an example of such a dashboard.

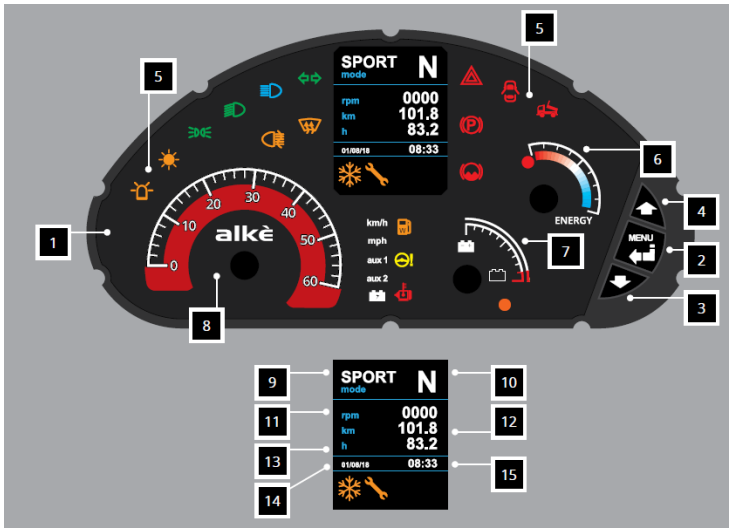


Figure 18: ALKE vehicle display installed on the baseline vehicle (ATX3)

### 3.3.1. HMI Display Dashboard

This is a functional module that represents a second display positioned on the dashboard, which will be running an Android device to allow the simultaneous use of a dedicated interface to display specific vehicle data and to allow the use of Bpost's logistics apps. It is the input user interface for entering information regarding destination, preferences on eco-oriented settings and more. At the same time, it will allow the display of additional information

to that already shown on the primary display addressed to the driver at the level of vehicle status or recommendations to optimise parameters or operational efficiency behaviours. Figure 19 shows an example of such a dashboard.

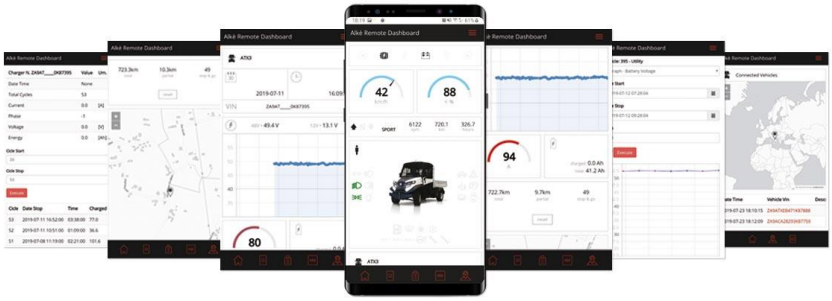

Figure 19: Example of Android app in line with the expected HMI interface

## 3.4 Charge Infrastructure, Charging Point AC

This functional module is external to the vehicle and represents the charging infrastructure present in the warehouse or depot, where the vehicle is usually parked during off-peak periods for charging. It also represents roadside charging stations. This charging point provides the possibility of charging with AC voltage plugs compatible with the connector supplied with the vehicle and connected to the on-board charger (OBC). An example of a charging point is shown in Figure 20.

Figure 20: Example of charging point used by ALKE ATX3 electric vehicles

# 4. Interfacing of Modules

To establish the information (signals) to be actively shared between multiple elements of the architecture of Section 3, the interfacing tool described in Section 2 is used. In the next subsections, the main interfacing components are summarized. A complete overview of the interfaces is found in Excel in (TNO, VuB, ALKE, CERTH, 2021). The Simulink output of the interfacing is shown in Figure 4, and it can be found in (TNO, VuB, ALKE, CERTH, 2021).

## 4.1 Cloud functions modules

The cloud functions components run all the fleet management functions and communicate with the vehicle using an internet connection. In the next subsections, each one of the interfaces is summarized

### 4.1.1. Eco-Charging

The Eco-Charging receives its inputs from three main sources:

- The Vehicle to Cloud interface: this group of inputs provides all the vehicle-fleet information relevant to the charging process. Examples of such information are the State of Charge, battery temperature per vehicle in the fleet, etc. per vehicle in the fleet
- The Charger to Cloud Interface: this group of inputs provides all the relevant information from the charging points. Examples of such information are the chargers status (busy, free), available charging power per charger, measured charging power, etc. Additional information provided by this interface is related to the grid status: grid price schedule and grid limitation.
- The Eco-Routing algorithm: this algorithm provides information about the day-ahead routes of each of the vehicles in the fleet. Likewise, this input collects the expected required energy by each vehicle to fulfil such a route.

The Eco-Charging provides as outputs:

- The charging schedule: The charging schedule corresponds to a set of maximum charging powers that each charger can output per time stamp. Such a schedule is sent to the Charger to Cloud Interface, so that it is distributed to each chargers in the fleet.
- Eco-Charging summary output: these signals provide relevant metrics about the charging schedule. Examples of such outputs are the expected fleet-level charging cost, expected charging energy, and expected vehicle SoC at the end of the charging process.

A graphical overview of all the inputs and outputs of the Eco-Charging

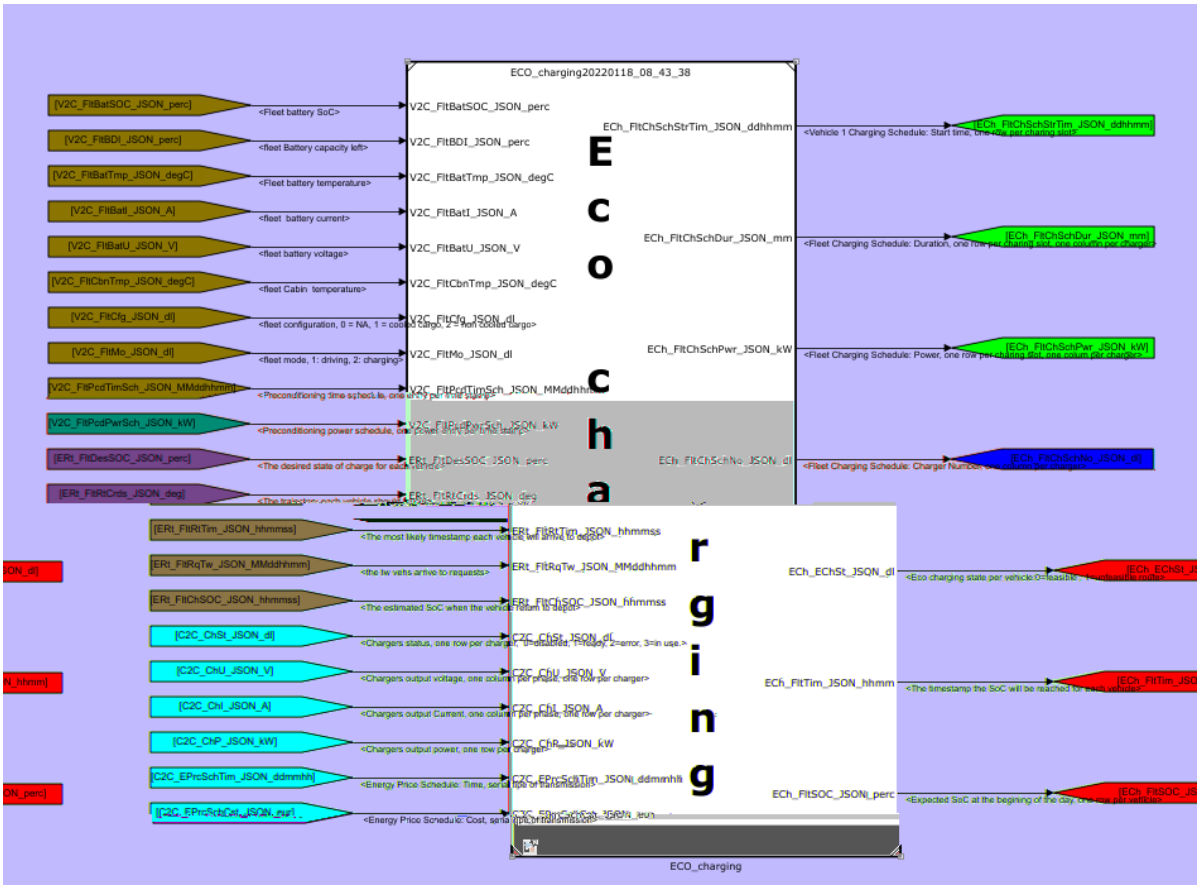


Figure 21: Eco-Charging interfacing

### 4.1.2. Eco-Routing

The Eco-Routing takes inputs from 2 modules:

- Logistics Information : this module provides the input data for the requests of the logistics operator (e.g., destination points), the constraints that each request has (e.g., arrival time), and the time and distances matrices that are needed for optimizations purposes. Moreover, it provides the energy consumption matrix that is used by the Eco-Routing as the main objective for minimization.
- Eco-Charging information: this module provides input data regarding the time that each vehicle will be available for a requested SoC. In addition, it provides the time frame that chargers are expected to be available, so that the routing algorithm manages the arrivals properly.

The Eco-Routing provides the following outputs:

- Eco-Charging information: This output provides the SoC that should be reached per vehicle to complete the planned routes; the timeframe that charging operation should take place; and the expected SoC per vehicle when it returns to the depot.
- The features that are related with the Vehicle to Cloud communication module: Those outputs include the trajectory the vehicle should follow and the sequence of stops that should take to serve the requests. The Eco-Driving module uses such

trajectory to compute its output. Likewise, the upper and lower timeframe for each vehicle should be completing each request.

A graphical overview of all inputs and outputs of the Eco-Charging algorithm is shown in Figure 22. A brief description of each one of the inputs and outputs is shown in Appendix B, Table 8 for the inputs and Table 9 for the outputs.

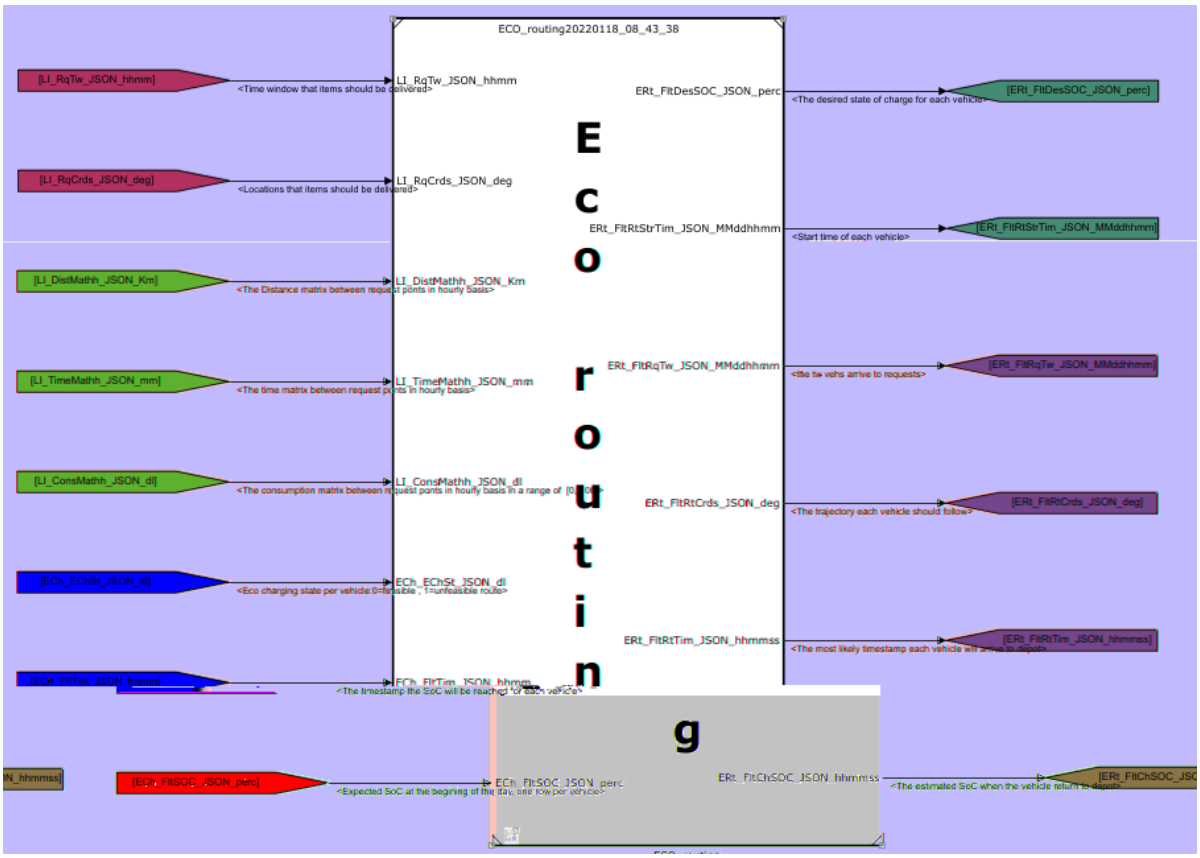


Figure 22: Eco-Routing interfacing.

### 4.1.3. Offboard Vehicle to Cloud Interface

The Offboard Vehicle to Cloud Interface receives its inputs from these main sources:

- The Eco-Routing algorithm: this algorithm provides information about the day-ahead routes of each of the vehicles in the fleet. This input collects the expected window the vehicles arrive to requests, vehicle route starting time and the routed vehicle trajectory.
- The HMI: this interface provides information from the driver about eco-functions data such as cabin temperature setpoint, Eco-Comfort function status and Eco-Driving function status.
- The Weather forecast provider: this group of inputs provides weather information from an external provider. Examples of such information are temperature forecast profile, humidity forecast profile, wind speed forecast, wind direction forecast, rainfall forecast and visibility distance.

- The VBC (Vehicle Body Computer): this module collects most vehicle data before sharing it with the cloud functions. Information shared in this context are preconditioning time schedule, preconditioning power schedule, SoC, battery temperature, current and voltage, cabin temperature, cargo body configuration, battery capacity left, vehicle charging status, vehicle velocity setpoint, vehicle acceleration setpoint and vehicle braking force.
- The Traffic profile: this modules provides data about road traffic density and default speed limit of the road.

The Offboard Vehicle to Cloud Interface provides outputs to:

- The VBC (Vehicle Body Computer): this module collects most vehicle data before sharing with the cloud functions. Examples of information shared in this specific context are humidity forecast profile, temperature forecast profile, cabin temperature setpoint, Eco-Comfort status, Eco-Driving status, rainfall forecast, the time window when the vehicle arrives to requests, the routed vehicle trajectory, vehicle route starting time, default speed Limit of Road, Road Traffic Density, Visibility distance, Wind direction forecast and Wind speed forecast.
- The Eco-Charging: this group of outputs provides all the vehicle-fleet information relevant to the charging process that is shared with the charging algorithm. Examples of such information are battery capacity left, battery current, battery SoC, battery temperature, battery voltage, cabin temperature, vehicle configuration, vehicle mode, preconditioning power schedule and preconditioning time schedule, for each vehicle in the fleet.
- The HMI: this module visualizes information to the driver about eco-functions data. In this context, the data shared corresponds to vehicle acceleration setpoint, vehicle Braking setpoint and vehicle velocity setpoint.

A graphical overview of all the inputs and outputs of the Offboard Vehicle to Cloud Interface is shown in Figure 23 .

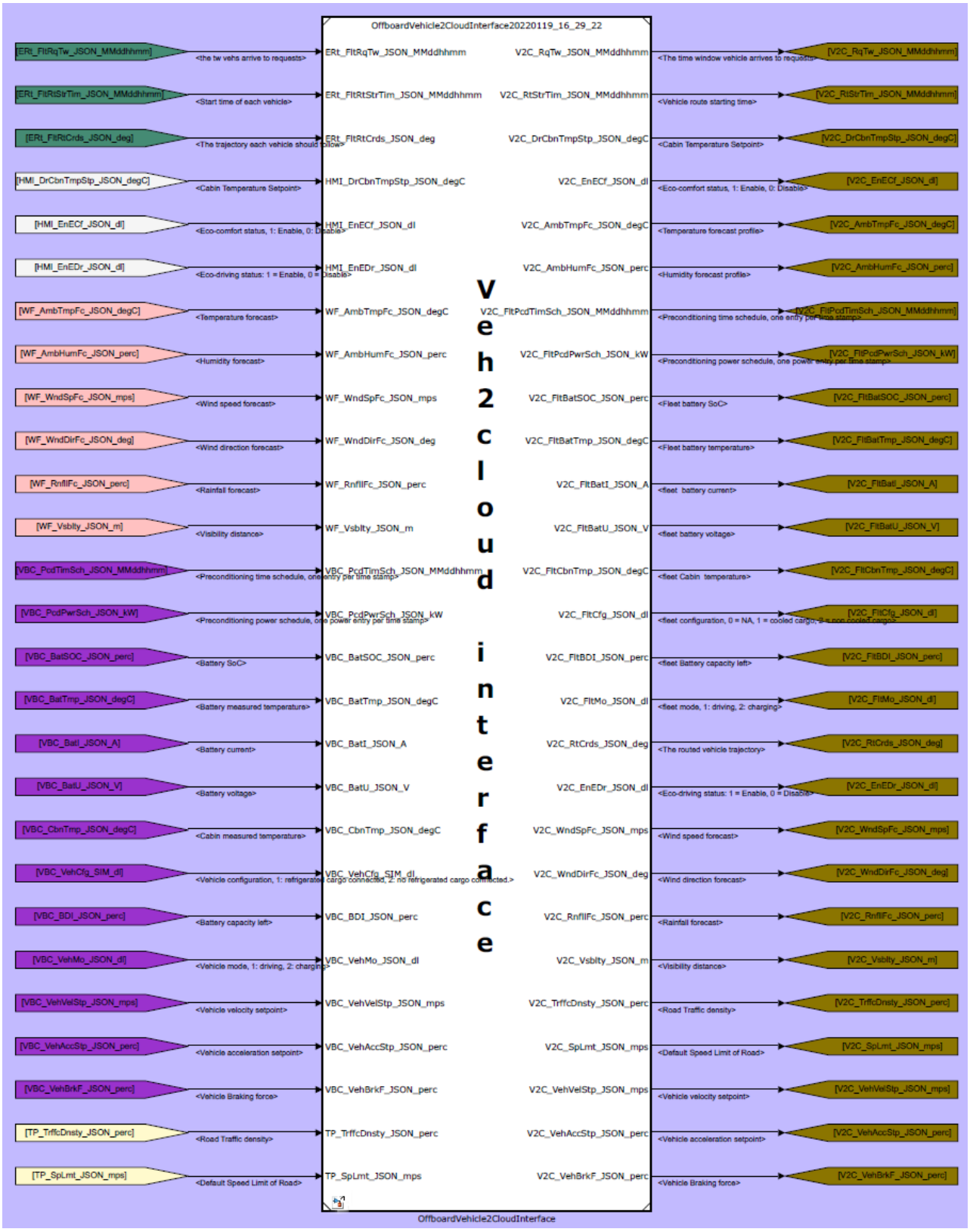


Figure 23: Vehicle to Cloud Interface interfacing

### 4.1.4. Charger to Cloud (C2C) interface

The charging to cloud interface software component provides connectivity between the Eco-Charging algorithm and the charging points (or chargers) and several online resources. Therefore, the following groups of inputs are observed:

- From the charging points: relevant information about the charging points is collected in the C2C interface. Examples of such information are charging point status (e.g., available or busy), charging point maximum power, etc.
- From the Eco-Charging algorithm: charging schedule of all chargers.

Likewise, the following outputs are observed:

- To each charger: the charging schedule. Such a charging schedule is composed of a maximum power per charger, and a time schedule.
- To the Eco-Charging algorithm: provide the relevant information from all the charging points to the Eco-Charging algorithm. This is basically a concatenation of the inputs in item 1 above (e.g., charging point status, charging point maximum power, etc).
- To the Eco-Charging algorithm: provide relevant grid information, such as electricity price schedule.

An overview of the inputs and outputs of the C2C interface is shown in Figure 24.

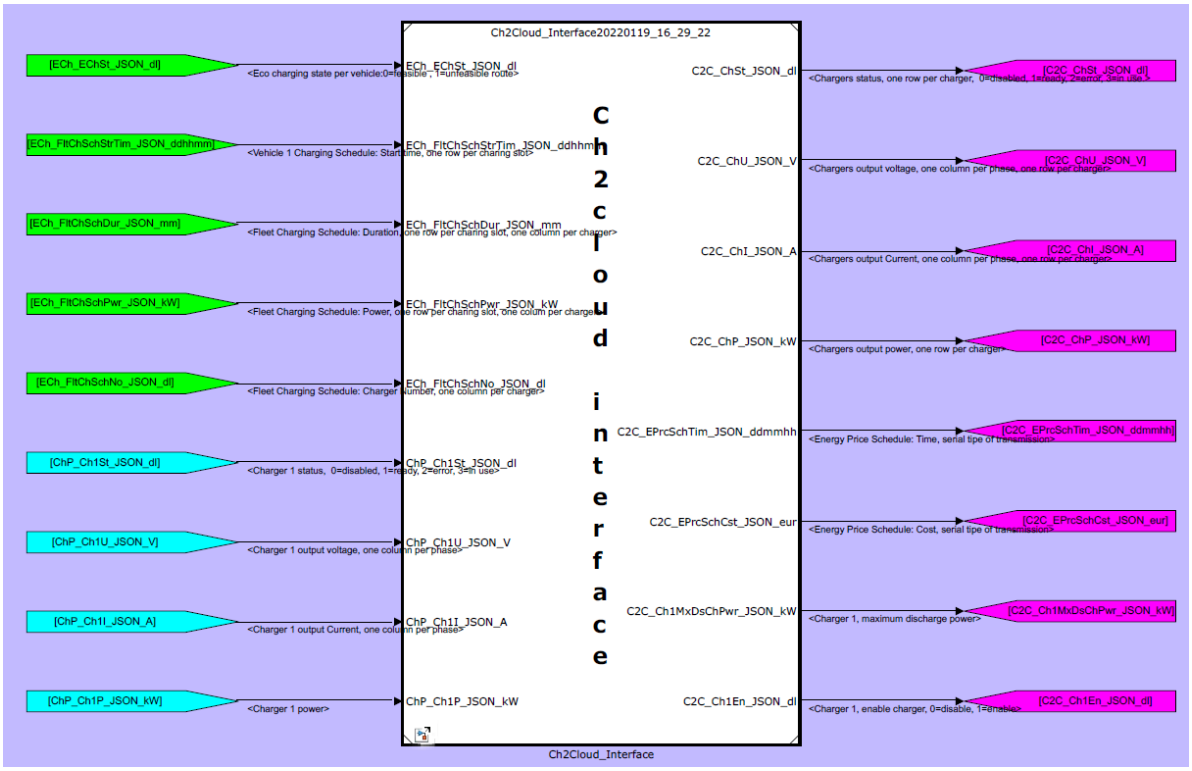


Figure 24: Charger 2 cloud interface overview.

### 4.1.5. Logistics Information

The core objective of this software component is to receive, process and format the requests for the routing algorithm. The main set of outputs define the requests to the routing algorithms

(e.g., places to be visited and time frames for such visits). This block takes inputs from the traffic prediction module, to extract features for the calculation of the time matrix, duration matrix, and the consumption matrix. Those matrices are returned as an output along with the requests and the time window constraints. An overview of the inputs and outputs of the Logistics Information can be seen in Figure 25.

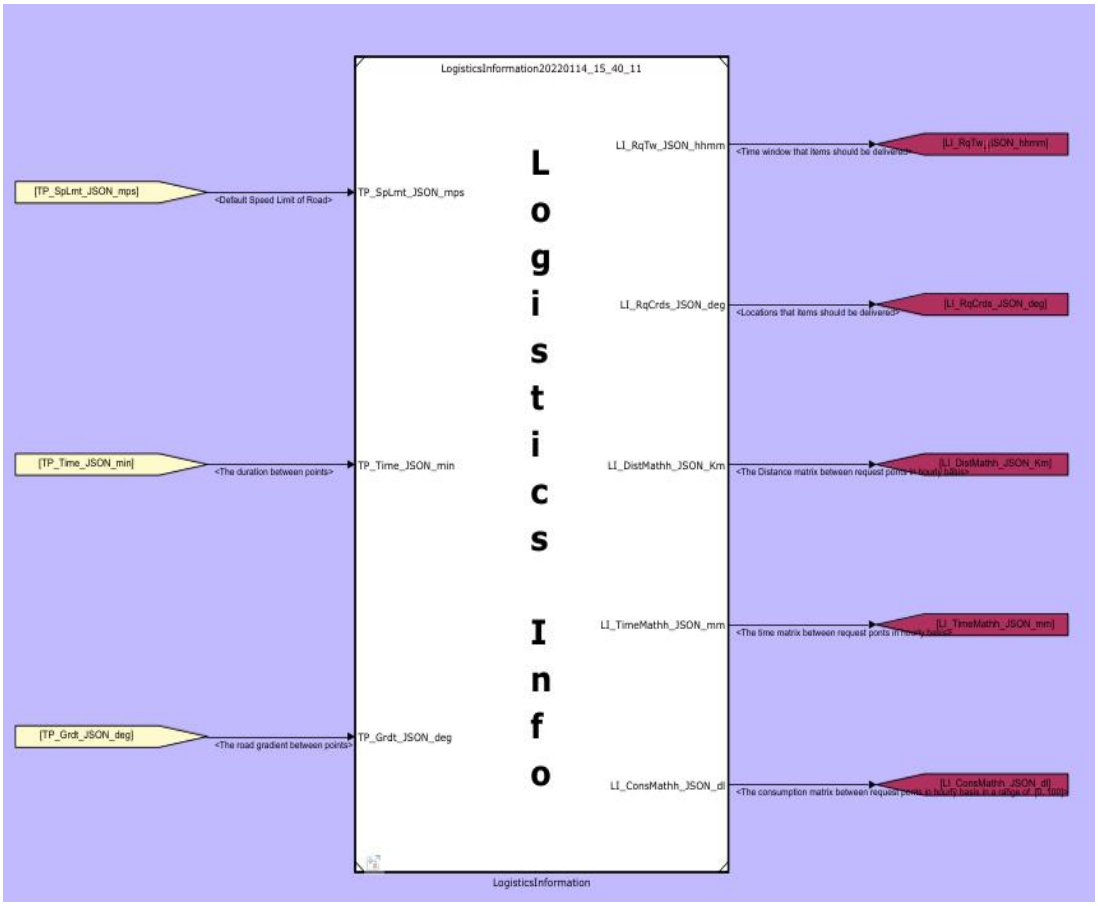


Figure 25: Logistic Information interfacing

### 4.1.6. Weather Forecast Information

The Weather Forecast information module receives online data from web services, therefore, the only inputs are related to where (which roads) and when (in which time frames) are these web services are required. The two main output groups are the following:

- Temperature and humidity forecast for Eco-Comfort
- Wind information, precipitation and visibility distance forecast for Eco-Driving.

The Weather Forecast information module is located in the URBANIZED cloud, while the EMSs mentioned above are onboard of the vehicle. Therefore, all signals are transferred to the vehicle through the Offboard Vehicle to Cloud Interface. An overview of the outputs signals is shown in Figure 26.

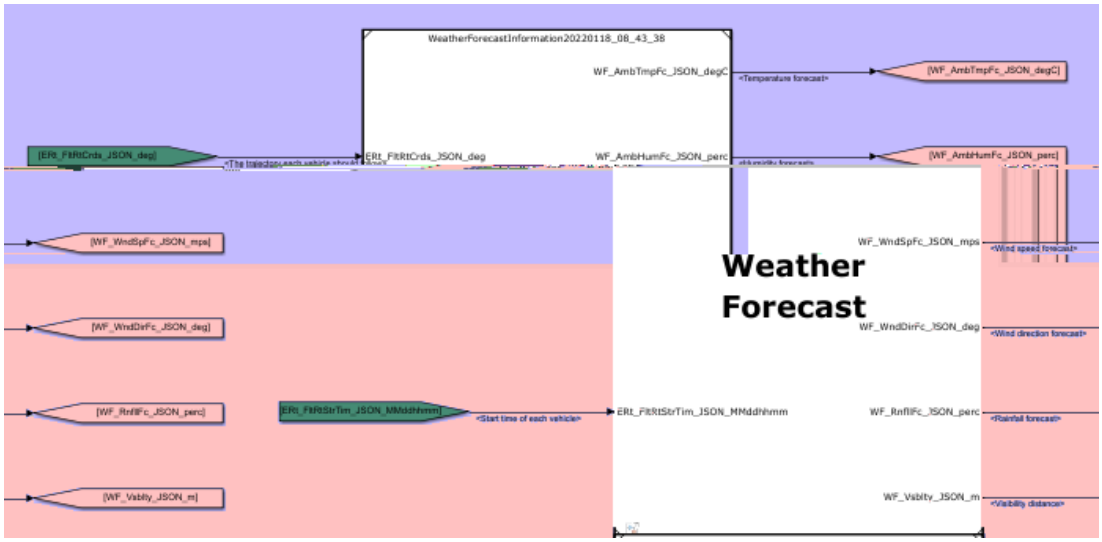


Figure 26. Weather forecast interface overview

### 4.1.7. Traffic Prediction Information

The traffic prediction module has the specific task of extracting information about the traffic and road status throughout the day. Therefore, it receives as an input the information about the request points and collects data online to compute the trip duration, the impact of congestion, and features related to the energy consumption calculation (e.g., elevation). An overview of the inputs and outputs of the Traffic Prediction component is shown in Figure 27.

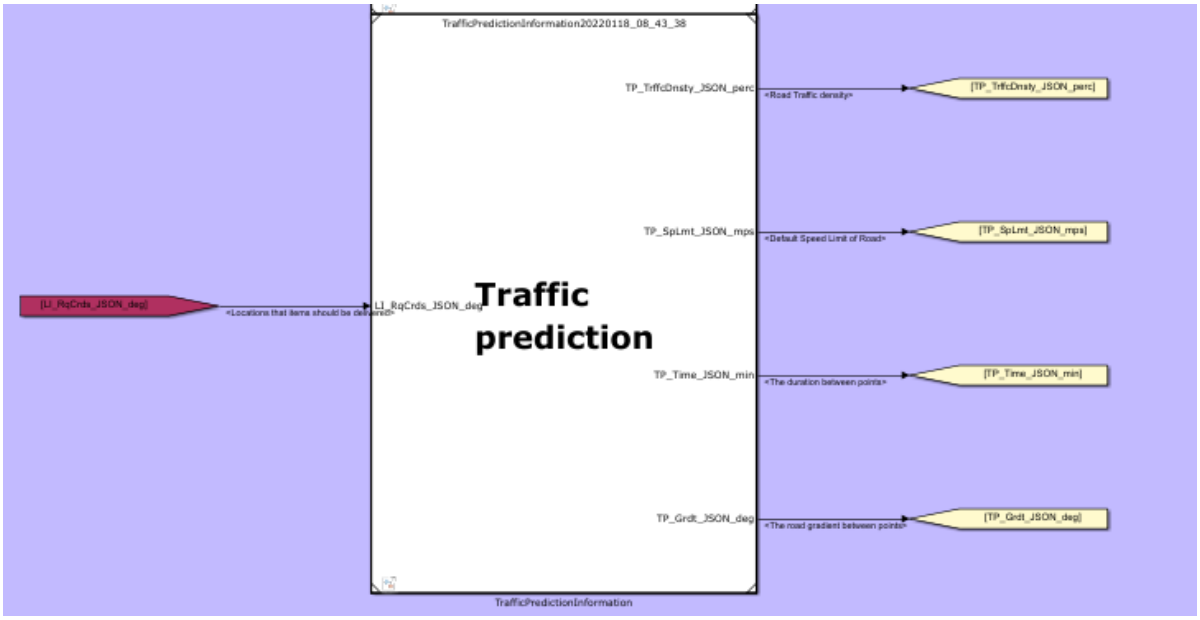


Figure 27: Traffic Prediction interfacing

## 4.2 On-board vehicle modules

The on-board vehicle components collect all the functions that need to be run in the vehicle during normal operation. These components connect with the cloud components via a live internet connection, enabled by the VBC. In the next subsections, the interfacing of each one of these components is described.

### 4.2.1. Eco-Driving

The Eco-Driving (EDr) block gets its inputs from a wide variety of functional blocks, some of them located onboard the vehicle, such as the Base Vehicle (BV) and the Human Machine Interface (HMI), while others are located in the cloud, including the Traffic Prediction (TP) block, the Weather Forecast (WF) block, and the ECO-routing (ERt) block. However, the EDr only has direct connection to the VBC: any signal coming to the EDr must be routed through the VBC. Similarly, any signals originating in the cloud must be routed through the Vehicle to Cloud (V2C) block first, before reaching the EDr via the VBC. On the other hand, the outputs of the EDr are meant for the HMI, the BV, and the Ecocomfort (ECf) block. While the signals meant for the BV must be routed through the VBC, the signals meant for the ECf are sent directly to it, since both the ECf and EDr are located in the same Electronic Control Unit (ECU) chip.

The following inputs are utilized by the EDr block:

- Inputs originating in the BV block includes: the vehicle configuration, the current time in the vehicle, the current vehicle coordinates (latitude, longitude, altitude), the current vehicle direction and speed, the current battery information (SoC, current, voltage, temperature), and the ambient temperature.
- Inputs originating in the HMI block includes: the Eco-Driving enable/disable signal initiated by the driver.
- Inputs originating in the Eco-routing block includes: the waypoint coordinates (list of latitudes and longitudes) that the vehicle must travel to, and the time window within which the vehicle must arrive at its respective waypoints.
- Inputs originating in the WF block includes: windspeed and direction, the probability of precipitation, and the surrounding visibility.
- Inputs originating in the TP block includes: real-time traffic density, and speed limits per route.

The following outputs are generated by the EDr block:

- Outputs meant for the ECf block include: the vehicle predicted power utilization and the forced enable signal of the Eco-Comfort. The Eco-Comfort and Eco-Driving become active irrespective of the driver's wish when the SoC of the battery falls below a certain threshold.
- Outputs meant for the HMI block include: the velocity setpoint that the driver must try to follow, according to the suggested acceleration and braking suggestions (i.e., pedal positions), to lower the energy consumption and maximize the energy recovery.

An overview of the Eco-Driving interface is shown in Figure 28. The full list of the inputs and outputs of this interface, as well as a brief description of each one is shown in Appendix B, Table 10 for the inputs and Table 11 for the outputs.

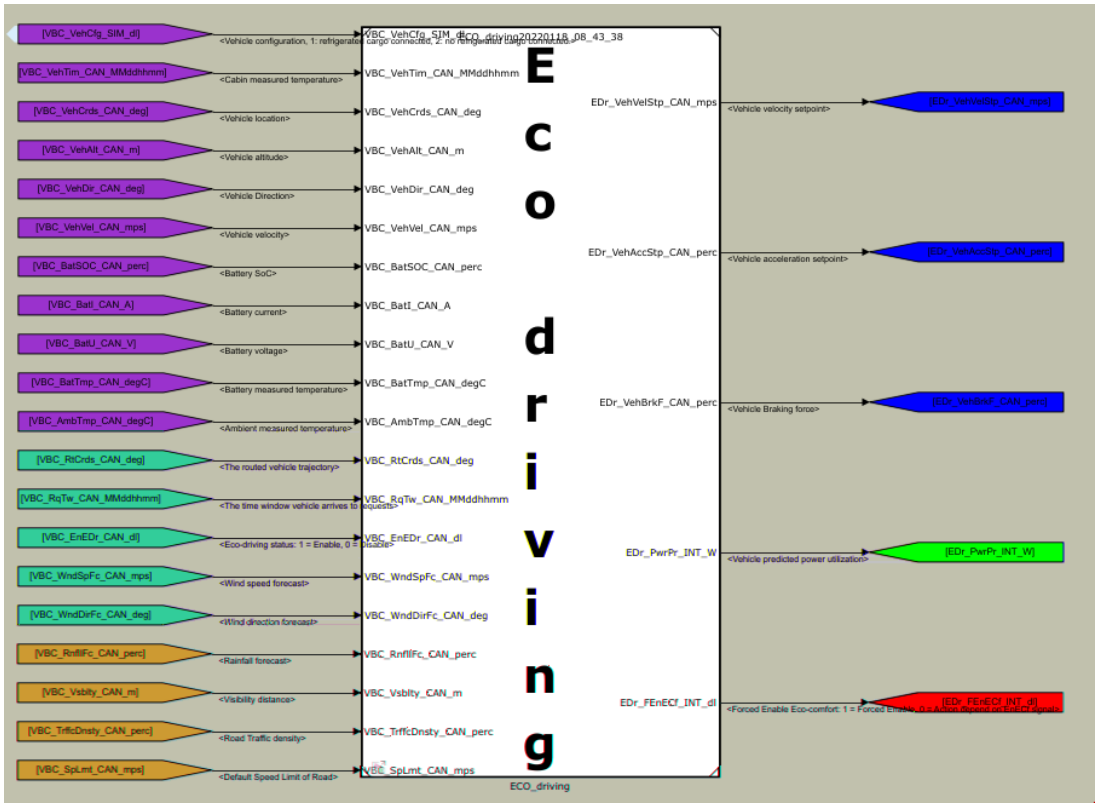


Figure 28. Eco-Driving interface overview

### 4.2.2. Eco-Comfort

Since the Eco-Comfort block has only direct interaction with the Vehicle Body Vomputer (VBC), almost all inputs and outputs are connected to that block. The only exception is a signal coming from the Eco-Driving block, which is implemented in the same ECU. However, the VBC is a middle point and, therefore, the inputs have a more elaborate path, which could be grouped as follows:

- Inputs from the Base Vehicle: these concern information such as measured temperatures from the ambient and other parts of the vehicle, vehicle's cargo and mode (i.e., driving or charging).
- Inputs from the cloud: information that is received from online resources, such as weather forecast, which are relevant for the Eco-Comfort prediction, but also information from the Eco-routing related to the starting time of the route, so that the preconditioning can be scheduled.
- Inputs from the HMI dashboard: the driver can enable or disable the Eo-Comfort function manually from the dashboard, and also set a temperature setpoint for the cabin.

Similarly, the Eco-Comfort outputs can be summarized in the following two categories:

- Temperature outputs: The main goal of the Eco-Comfort block is to calculate optimal temperature setpoints for the cabin, battery and cargo (if the refrigerated cargo is connected). These are sent to the Base Vehicle through the VBC and are then handled by the low-level controllers.
- Preconditioning information: For the preconditioning function, the algorithm will send a power and time schedule to the Eco-Charging component.

An overview of the inputs and outputs of the Eco-Comfort algorithm is shown in Figure 29. The full list of the inputs and outputs of this interface, as well as a brief description of each one is shown in Appendix B, Table 12 for the inputs and Table 13 for the outputs.

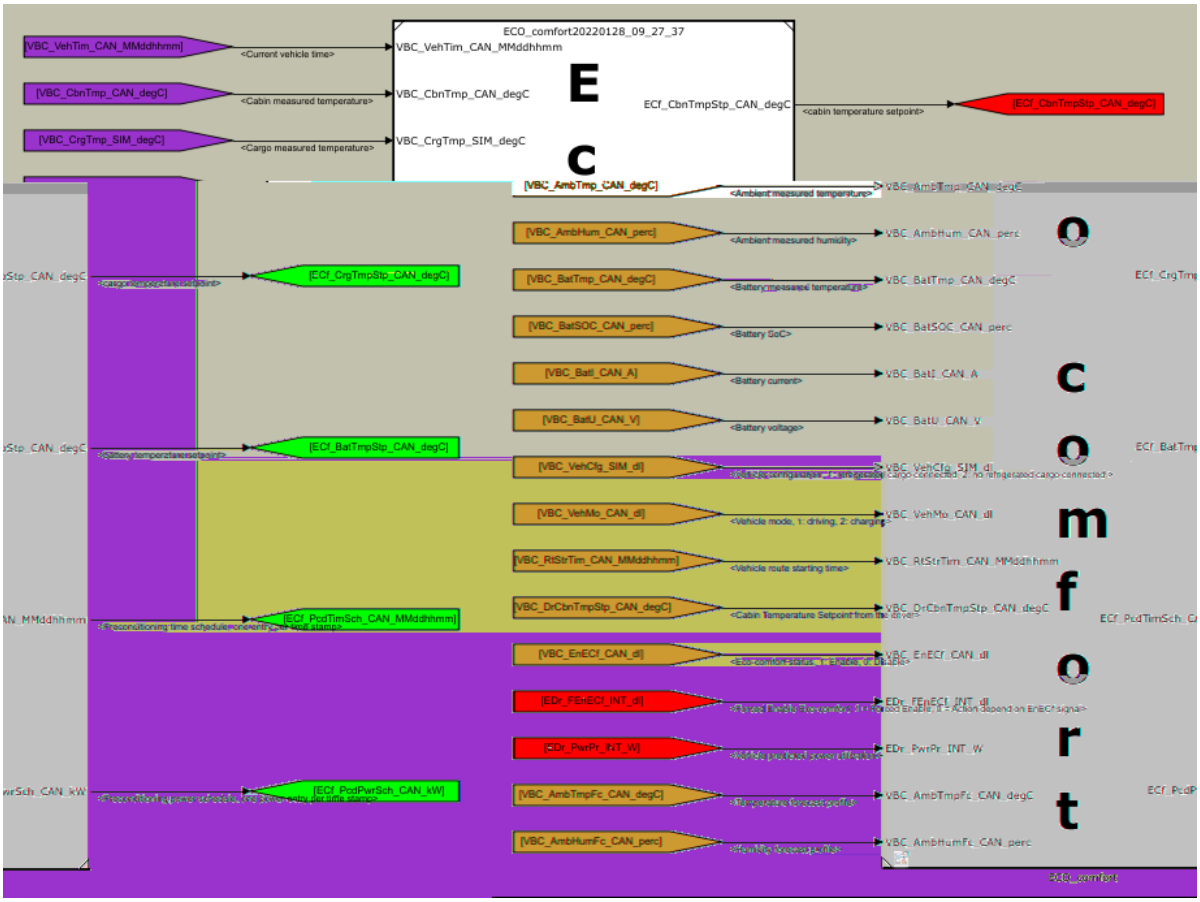


Figure 29: Eco-Comfort interfacing

### 4.2.3. Vehicle Body Computer (VBC)

The Vehicle Body Computer receives its inputs from these main sources:

- The Base Vehicle: the Base Vehicle generates most of vehicle status and driving data shared with the cloud platform and other eco-oriented blocks. In this context this input collects cabin and cargo measured temperature, ambient measured temperature and humidity, battery measured temperature, battery SoC, vehicle configuration, vehicle mode, battery capacity left, current time, vehicle location, altitude, direction, battery current, battery voltage and vehicle velocity.

- The Vehicle to Cloud Interface: this group of inputs provides data such as the time window for arrival to requests, vehicle route starting time, cabin temperature setpoint, Eco-Comfort status, temperature forecast profile, humidity forecast profile, the routed vehicle trajectory, Eco-Driving status, wind speed forecast, wind direction forecast, rainfall forecast, visibility distance, road Traffic density and default speed limit of road.
- The Eco-Comfort: this group of inputs provide data such as preconditioning power schedule, cargo temperature setpoint, battery temperature setpoint, preconditioning time schedule, cabin temperature setpoint.
- The Eco-Driving shares input data such as vehicle velocity setpoint, vehicle acceleration setpoint and vehicle braking force.

The Vehicle Body Computer provides as outputs:

- The Eco-Comfort gets from VBC information about cabin measured temperature, cargo measured temperature, Eco-Comfort status, vehicle route starting time, vehicle mode, ambient measured temperature, battery current, battery SoC, battery measured temperature, battery voltage and vehicle configuration.
- The Eco-Driving gets from VBC info about ambient measured temperature, battery current, battery SoC, battery measured temperature, battery voltage, vehicle configuration, Eco-Driving status, rainfall forecast, the time window that vehicle arrives to requests, the routed vehicle trajectory, default speed limit of road, road traffic density, vehicle altitude, vehicle location, vehicle direction, cabin measured temperature, vehicle velocity, visibility distance, wind direction forecast, wind speed forecast.
- The Vehicle to Cloud Interface gets from VBC info about the vehicle configuration, battery capacity left, battery current, battery SoC, battery measured temperature, battery voltage, cabin measured, temperature, preconditioning power schedule, preconditioning time schedule, vehicle acceleration setpoint, vehicle braking force, vehicle mode, vehicle velocity setpoint

A graphical overview of all the inputs and outputs of the VBC is shown in Figure 30.

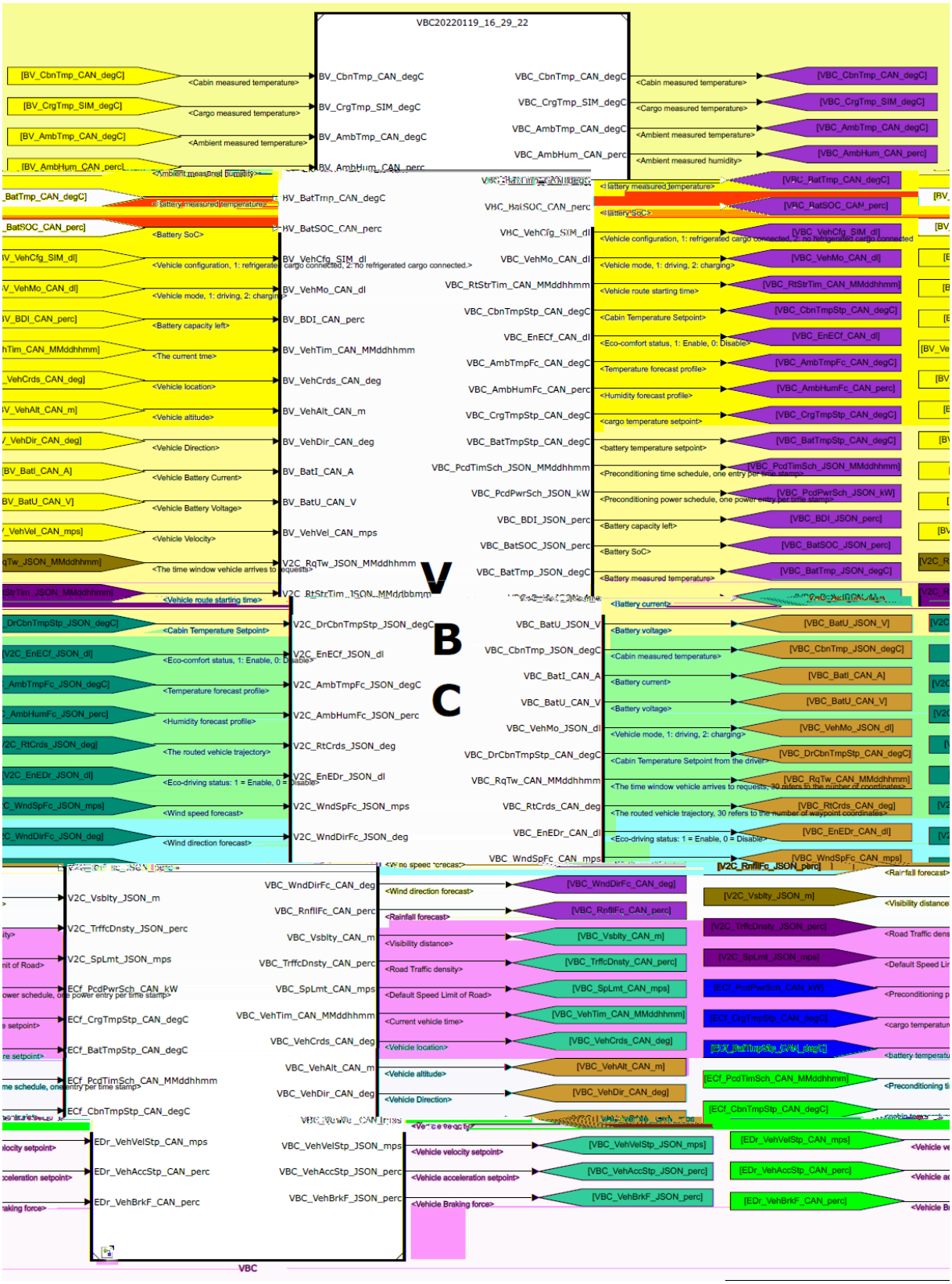


Figure 30: VBC interfacing

### 4.2.4. Vehicle Display Dashboard

The vehicle dashboard display receives its inputs from this source:

- The Base Vehicle: this module generates most of vehicle status and driving data shared with the cloud platform and other eco-oriented blocks. In this context, t

- Vehicle to Cloud Interface: it shares data such the maximum desired charging power for a specific ID linked charging point.

The charge infrastructure provides as output:

- Vehicle to Cloud Interface: it sends data such as output current, power, status and output voltage for a specific ID linked charging point.

A graphical overview of all the inputs and outputs of the vehicle display interface is shown in Figure 33.

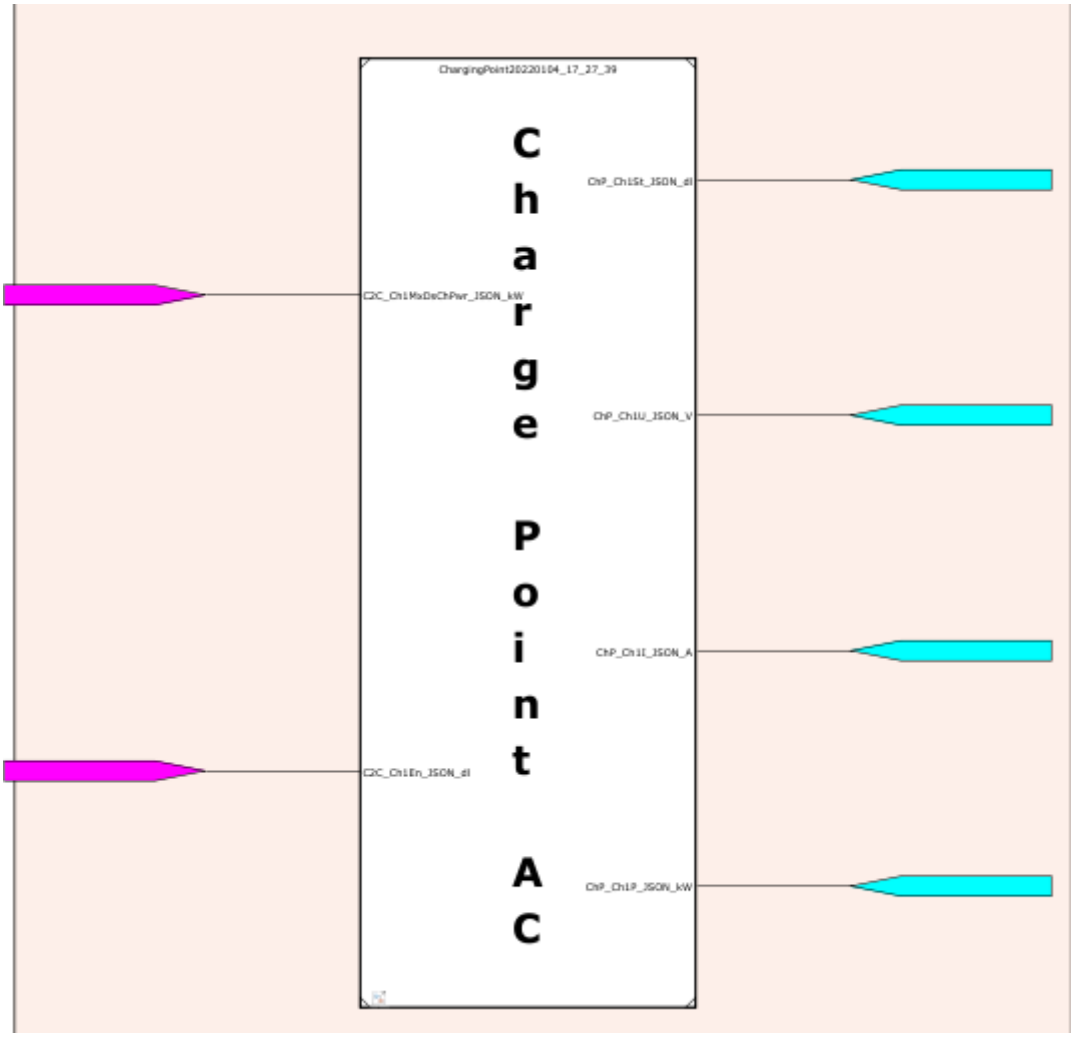


Figure 33: Charging Point interfacing

# 5. EMSs functions interfacing with virtual simulation environments

The EMSs developed in Urbanized are to be tested in future work packages in (among others) virtual simulation environments. In the following subsections, such simulation environments are described as well as their connections with the EMSs.

## 5.1 Interfacing with vehicle simulation tool

The vehicle simulation tool developed in MATLAB/Simulink under T3.1 can be re-used in the Base Vehicle block shown in Figure 6. The vehicle model is based on forward-facing approach and low-fidelity models (i.e., efficiency map look up tables) to evaluate the vehicle's performances such as acceleration, and energy consumption. An optimization algorithm has been integrated into this vehicle model to find the right sizing of drivetrain components under different driving cycles or transport assignments. The characteristics of Alke's vehicle (e.g., behaviour of e-motor, battery, gear) can be updated in this simulation tool, which reflects the realistic driving conditions. The general input/output interfacing with this low-fidelity vehicle model can be found in Figure 6.

The input data could be the weather and traffic situation, as well as list of waypoints and their respective arrival time windows. The simulation tool will simulate the vehicle going through this list of waypoints, within the specified timings, while ensuring traffic discipline and safety.

The input data for the simulations, such as the weather, traffic, and waypoints information, will be communicated from the cloud. The other data required for the energy management functionality, such as the current vehicle location and velocity information will be generated from within the simulation framework. To ensure realism and accuracy of the vehicle's energy requirements, and reduce the number of assumptions, it is necessary to get real traffic data. To such end, the virtual simulation tool will interface with a cloud system capable of accessing public databases updated at run time, based on actual number of vehicles in roads. Furthermore, the cloud system should also be able to simulate the effect of the vehicle itself on the existing traffic in the road.

## 5.2 Interfacing with Thessaloniki Smart Mobility Living Lab

Thessaloniki Smart Mobility Living Lab (Hellenic Institute of Transport, Center for Research & Technology Hellas, 2022) is to be used as a simulation framework for the cloud-based functions of Urbanized. The Thessaloniki Smart Mobility Living Lab includes, among others:

- Real time traffic data in Thessaloniki (cars and trains).
- Short-term predictions of traffic conditions from multiple sources.
- Exporting and formulating mobility and activity patterns.

- Extended IoT equipment.

For testing, evaluation, and optimal calibration purposes of fleet management algorithms of the URBANIZED project, a discrete event simulation will be developed with the use of ANYLOGIC simulation software. The EMS modules that take place in this simulation are: the Eco-Routing, Logistics Information, and Eco-Charging. The THESSM@LL will host the simulation that will be developed by CERTH/HIT with TNO support. The simulation will contain all the important features and processes take place in last-mile logistics operations. The algorithms will be integrated via API requests for each corresponding module. The results will be available in a dashboard service provided by the THESSM@LL.

# 6. Conclusions

This report summarizes the steps taken to develop the architecture of the Multi-layer EMS in work package 3 of the Urbanized project. The introduced architecture for the multi-layer EMS consists of the EMS modules (Eco-Driving, Eco-Comfort, Eco-Routing and Eco-Charging), as well as supporting software modules, which are required for the correct interfacing of the EMSs. Examples of such software modules are the Vehicle to Cloud Interface, Charger to Cloud Interface, etc. Lastly, functional modules are described, which represent the physical components installed on the vehicle. Each EMSs is described in detail in terms of functionality, data requirements and intended implementation. Likewise, a brief description of the functionality of the supporting software modules and functional modules is provided.

The envisioned architecture is realized with a tool that was developed for that purpose in MATLAB/Simulink and demonstrates all the interfaces and connections among the EMS modules. The tool can be accessed online and used by multiple users concurrently through different models that come together in a single interface. The tool verifies that all connections are complete by automatically linking inputs and outputs with identical names. In total, there are 158 input and 140 output signals. Strict naming rules are introduced to avoid mistakes between different modules/communication protocols etc. Additionally, an automatically generated report lists all the interfaces in detail, which is used for debugging purposes, e.g., for floating inputs/outputs, data type mismatch etc. The tool ensures that relevant aspects of the interfacing are defined, such as sampling frequency and data size, which will be also useful in the implementation of the EMSs in further work packages.

The interconnection of this tool with other software solution is also described in the report. A vehicle simulation framework is developed in parallel in this work package, which focuses on the high fidelity modelling of the powertrain components of the vehicle. This report describes the interfacing of the two on-board EMS functions (Eco-Comfort and Eco-Driving) with it. Similarly, it is described how the Thessaloniki Smart Mobility Living lab will be used as a testing tool for the Eco-Charging and Eco-Routing modules.

The results of this task are to be used in further WPs to develop and implement the EMSs and the supporting software modules. Since the functionality and interfaces of the EMSs is detailed in this report, this will ease the integration between the developments made by different partners.

# 7. Acronyms

| Acronym | Meaning |
| --- | --- |
| BEe4(n)4(yms)] T | |

# 8. References

Hellenic Institute of Transport, Center for Research & Technology Hellas. (2022, 02 08). *Thessaloniki Smart Mobility Living Lab.* Retrieved from https://smartmlab.imet.gr/

Kallehauge, B. J. (2005). Vehicle routing problem with time windows. *Column generation*, 67-98.

Medina , R., Avramis , N., & Purnot, T. (2021). *Using the interfacing tool definition.* TNO.

Ralphs, T. K. (2003). On the capacitated vehicle routing problem. *Mathematical programming*, 343-359.

TNO, VuB, ALKE, CERTH. (2021, 11 26). Official Interface Report in Excel. TNO.

TNO, VuB, ALKE, CERTH. (2021, 11 26). Simulink Interface Rerpot. TNO.

# 9. Appendix A: naming convention tables

## 9.1 Modules abbreviations

Table 2. Component keyword abbreviations

| Keyword | Component |
|---|---|
| **BV** | Base Vehicle |
| **C2C** | Charger-to-Cloud interface |
| **ChP** | Charging Point |
| **ECf** | Eco-Comfort |
| **ECh** | Eco-Charging |
| **EDr** | Eco-Driving |
| **ERt** | Eco-Routing |
| **HMI** | HMI Display |
| **LI** | Logistics Information |
| **V2C** | Vehicle-to-Cloud interface |
| **VBC** | Vehicle Body Computer |
| **VDD** | Vehicle Display Dashboard |
| **WF** | Weather Forecast |

## 9.2 Description abbreviations

Table 3: Description keyword abbreviations

| Keyword | Description |
|---|---|
| **Ahd** | Ahead |
| **Alt** | Altitude |
| **Amb** | Ambient |
| **Bat** | Battery |
| **Brk** | Brake |

| | |
|---|---|
| **C** | Current |
| **Cbn** | Cabin |
| **Cch** | Charge capacity |
| **Cel** | Cell |
| **Cfg** | Configuration |
| **Ch** | Charging |
| **Cl** | Close |
| **Coec** | Magnetic Field / Coectivity |
| **Cov** | Covariance |
| **Cp** | Specific heat capacity |
| **Cr** | Core |
| **Crds** | Cordinates of Requests |
| **Crg** | Cargo |
| **Cst** | Cost |
| **Ct** | Contactor |
| **Cth** | Thermal/heat capacity |
| **dd** | Day |
| **Dh** | Discharging |
| **Diam** | Diameter |
| **Dir** | Direction |
| **DistMat** | Distance Matrix |
| **Dmp** | Damping coefficient |
| **Dp** | Pressure drop |
| **Dr** | Driver |
| **Ds** | Desired |
| **Dt** | Temperature Difference |
| **Dur** | Duration |
| **E** | energy |
| **ECh** | Eco charging |
| **EDr** | Eco driving |
| **En** | Enable |

| | |
|---|---|
| **Er** | Error |
| **Et** | Estimation |
| **Eta** | Dynamic viscosity |
| **Exp** | Expected |
| **F** | Force |
| **fac** | Factor |
| **Fc** | Forecast |
| **Fea** | Feasible |
| **flg** | Flag |
| **Flt** | Fleet |
| **Fq** | frequency |
| **Gen** | Generation |
| **Grdt** | Gradient |
| **H** | Height |
| **Hr** | Horizon |
| **Ht** | Heat |
| **Hum** | Humidity |
| **Hv** | High Voltage |
| **I** | Current |
| **Init** | Initial |
| **Irr** | Solar Irradiance |
| **Is** | Isolation |
| **L** | Electric inductivity |
| **Lat** | Latitude |
| **Len** | length |
| **Lmt** | Limit |
| **Lon** | Longitude |
| **M** | Mass |
| **Md** | Module |
| **Me** | Measurement |
| **Mn** | Minimum |

| | |
|---|---|
| **Mo** | Mode |
| **MoI** | Moment of inertia |
| **Mtx** | Matrix |
| **Mu** | Friction coefficient |
| **Mx** | Maximum |
| **N** | Rotational Speed |
| **No** | Number |
| **OC** | Open Circuit |
| **Op** | Open |
| **Ov** | Over |
| **P** | Pressure |
| **Par** | Paralle |
| **Pc** | Pack |
| **Pcd** | Preconditioning |
| **perc** | Percentage |
| **Phi** | Magnetic Flux |
| **Posn** | Position |
| **Pr** | Prediction |
| **Prc** | Price |
| **Pwr** | Power |
| **Q** | Heat |
| **Qt** | Heat Flow |
| **R** | Resistance |
| **Rd** | Radius |
| **Rem** | Remanence |
| **Rho** | Density |
| **Rmd** | Recommended |
| **Rnfll** | Rainfall |
| **Rq** | Request |
| **Rt** | Route |
| **Rth** | Thermal resistance |

| | |
|---|---|
| **Sch** | Schedule |
| **Sf** | Surface |
| **SOC** | State of Charge |
| **SOE** | State of Energy |
| **SOH** | State of Health |
| **Sp** | Speed |
| **St** | Status |
| **Stfn** | Stiffness |
| **Stp** | Set point |
| **StrTim** | Start Time |
| **Tim** | Time |
| **TimMat** | Time Matrix |
| **Tmp** | Temperature |
| **Tq** | Torque |
| **Tra** | Trajector |
| **TrffcDnsty** | Traffic Density in the road |
| **Tw** | Time Window |
| **Typ** | Type |
| **U** | Voltage |
| **Unf** | Unfeasible |
| **Var** | Variance |
| **Vct** | Vector |
| **Veh** | Vehicle |
| **Vel** | Velocity |
| **Vol** | Volume |
| **Vsblty** | Visibility in front of vehicle |
| **Vt** | Volume flow |
| **Wd** | Width |
| **Wnd** | Wind |
| **z** | Ratio |
| **zeta** | Resistance Coefficient (Darcy Weissback Equation) |

## 9.3 Physical interface abbreviations

Table 4: Physical interface keyword abbreviations

| Keyword | Description |
|---|---|
| **CAN** | Can signal |
| **EAN** | Electrical Analog |
| **EHV** | Electrical  High Voltage |
| **ELV** | Electrical Low voltage |
| **Hy** | Hydraulic |
| **Me** | Mechanical |
| **TCP** | TCP/IP (internet) |
| **UDP** | UDP/IP (internet) |
| **JSON** | JSON |
| **SIM** | Simulation |
| **INT** | Internal communication |

## 9.4 Unit abbreviations

Table 5: Unit keyword abbreviations

| Unit | Description |
|---|---|
| **A** | Ampere |
| **Ah** | ampere hours |
| **Apm** | ampere per meter |
| **Arms** | Ampere rms |
| **bl** | Boolean |
| **dd** | days |
| **deg** | degrees |
| **degC** | degrees Celsius |
| **dl** | dimensionless |
| **eur** | Euro |

| | |
|---|---|
| **F** | farad |
| **H** | Henry |
| **hh** | hour |
| **Hz** | Hz |
| **J** | Joule |
| **JpK** | Joule per kelvin |
| **Jpkg** | Joule per Kilograms |
| **JpKpkg** | Joule per Kelvin per Kilograms |
| **K** | Kelvin |
| **kg** | Kilograms |
| **kgm2** | kilogram square meter |
| **kgpm3** | kilogram per cubic meter |
| **Km** | kilometers |
| **KpW** | Kelvin per Watt |
| **m** | meters |
| **m2** | square meters |
| **m3** | cubic meters |
| **m3ps** | cubic meters per seconds |
| **mm** | minutes |
| **MM** | month |
| **mps** | meters per seconds |
| **mps2** | meters per square seconds |
| **N** | Newton |
| **Nm** | Newton-meter |
| **Npm** | newton per meter |
| **Nspm** | newton seconds per meter |
| **Nspm2** | Newton seconds per square meters |
| **Ohm** | Ohm |
| **Pa** | Pascal |
| **perc** | percentage |
| **rad** | radiant |

| | |
|---|---|
| **radps** | radiant per seconds |
| **s** | seconds |
| **str** | structure |
| **T** | Tesla |
| **V** | Volt |
| **Vrms** | Volt rms |
| **W** | Watt |
| **Wb** | Weber = 1 tesla square meters |
| **Wpm2** | Watt per square meters |
| **YYYY** | year |

# 10. Appendix B: summary of interfacing tables per module

## 10.1 Eco-Charging

Table 6: Eco-Charging inputs overview

| Input name | Description | Source output block |
|---|---|---|
| **V2C_FltBatSOC_JSON_perc** | Fleet Vehicle SoC | OffboardVehicle2CloudInterface |
| **V2C_FltBDI_JSON_perc** | Fleet Battery Discharge indicator | OffboardVehicle2CloudInterface |
| **V2C_FltBatTmp_JSON_degC** | Fleet battery temperature | OffboardVehicle2CloudInterface |
| **V2C_FltBatI_JSON_A** | Fleet battery current | OffboardVehicle2CloudInterface |
| **V2C_FltBatU_JSON_V** | Fleet battery voltage | OffboardVehicle2CloudInterface |
| **V2C_FltCbnTmp_JSON_degC** | Fleet Cabin temperature | OffboardVehicle2CloudInterface |
| **V2C_FltCfg_JSON_dl** | Fleet cargo configuration,0 = NA, 1 = cooled cargo, 2 = non cooled cargo | OffboardVehicle2CloudInterface |
| **V2C_FltMo_JSON_dl** | Fleet status: 1 driving, 2 charging | OffboardVehicle2CloudInterface |
| **V2C_FltPcdTimSch_JSON_MMddhhmm** | Preconditioning time schedule for the fleet of vehicles, one entry per time stamp | OffboardVehicle2CloudInterface |

| | | |
|---|---|---|
| **V2C_FltPcdPwrSch_JSON_kW** | Preconditioning power schedule for the fleet of vehicles, one power entry per time stamp | OffboardVehicle2CloudInterface |
| **ERt_FltDesSOC_JSON_perc** | The desired state of charge for each vehicle | ECO_routing |
| **ERt_FltRtCrds_JSON_deg** | The trajectory each vehicle should follow | ECO_routing |
| **ERt_FltRtTim_JSON_hhmmss** | The most likely timestamp each request will be served | ECO_routing |
| **ERt_FltRqTw_JSON_MMddhhmm** | The most likely timestamp each vehicle will arrive to depot | ECO_routing |
| **ERt_FltChSOC_JSON_hhmmss** | The estimated SoC when the vehicle return to depot | ECO_routing |
| **C2C_ChSt_JSON_dl** | Chargers status, one row per charger, 0=disabled, 1=ready, 2=error, 3=in use. | Ch2Cloud_Interface |
| **C2C_ChU_JSON_V** | Chargers output voltage, one column per phase, one row per charger | Ch2Cloud_Interface |
| **C2C_ChI_JSON_A** | Chargers output Current, one column per phase, one row per charger | Ch2Cloud_Interface |

| | | |
|---|---|---|
| **C2C_ChP_JSON_kW** | Chargers output power, one row per charger | Ch2Cloud_Interface |
| **C2C_EPrcSchTim_JSON_ddmmhh** | Energy Price Schedule: Time, serial tipe of transmission | Ch2Cloud_Interface |
| **C2C_EPrcSchCst_JSON_eur** | ergy Price hedule: Cost, ial tipe of nsmission | Ch2Cloud_Interface |

Table 7: Eco-Charging outputs overview

| Output name | Description | Destination block |
|---|---|---|
| **ECh_EChSt_JSON_dl** | Eco charging state per vehicle:0=feasible , 1=unfeasible route | Ch2Cloud_Interface, ECO_routing |
| **ECh_FltChSchDur_JSON_mm** | Fleet Charging Schedule: Duration, one row per charing slot, one column per charger | Ch2Cloud_Interface |
| **ECh_FltChSchNo_JSON_dl** | Fleet Charging Schedule: Charger Number, one column per charger | Ch2Cloud_Interface |
| **ECh_FltChSchPwr_JSON_kW** | Fleet Charging Schedule: Power, one row per charing slot, one colum per charger | Ch2Cloud_Interface |
| **ECh_FltChSchStrTim_JSON_ddhhmm** | Vehicle 1 Charging Schedule: Start time, one row per charing slot | Ch2Cloud_Interface |
| **ECh_FltSOC_JSON_perc** | Expected SoC at the begining of the day, one row per vehicle | ECO_routing |
| **ECh_FltTim_JSON_hhmm** | The timestamp the SoC will be reached for each vehicle | ECO_routing |

## 10.2 Eco-Routing

Table 8. Eco-Routing inputs overview

| Input name | Description | Source output block |
|---|---|---|
| **LI_RqTw_JSON_hhmm** | Time window requested that items should be delivered | LogisticsInformation |
| **LI_RqCrds_JSON_deg** | Locations that items should be delivered | LogisticsInformation |
| **LI_DistMathh_JSON_Km** | The Distance matrix between request ponts in hourly basis | LogisticsInformation |
| **LI_TimeMathh_JSON_mm** | The time matrix between request ponts in hourly basis | LogisticsInformation |
| **LI_ConsMathh_JSON_dl** | The consumption matrix between request ponts in hourly basis in a range of [0, 100] | LogisticsInformation |
| **ECh_EChSt_JSON_dl** | Eco charging state per vehicle:0=feasible , 1=unfeasible route | ECO_charging |
| **ECh_FltTim_JSON_hhmm** | The timestamp the SoC will be reached for each vehicle | ECO_charging |
| **ECh_FltSOC_JSON_perc** | The SoC for each vehicle | ECO_charging |

Table 9. Eco-Routing outputs overview

| Output name | Description | Destination block |
|---|---|---|
| **ERt_FltChSOC_JSON_hhmmss** | The estimated SoC when the vehicle return to depot | ECO_charging |
| **ERt_FltDesSOC_JSON_perc** | The desired state of charge for each vehicle | ECO_charging |
| **ERt_FltRqTw_JSON_MMddhhmm** | the tw vehs arrive to requests | ECO_charging, OffboardVehicle2CloudInterface |
| **ERt_FltRtCrds_JSON_deg** | The trajectory each vehicle should follow | ECO_charging, OffboardVehicle2CloudInterface, WeatherForecastInformation |
| **ERt_FltRtStrTim_JSON_MMddhhmm** | Start time of each vehicle | OffboardVehicle2CloudInterface, WeatherForecastInformation |
| **ERt_FltRtTim_JSON_hhmmss** | The most likely timestamp each vehicle will arrive to depot | ECO_charging |

## 10.3 Eco-Driving

Table 10. Eco-Driving inputs overview

| Input name | Description | Source output block |
|---|---|---|
| **VBC_VehTim_CAN_MMddhhmm** | Current time inside vehicle, (given as month, day, hour, and minute) | BV, via VBC |
| **VBC_VehCrds_CAN_deg** | The coordinates of the vehicle (latitude and longitude) | BV, via VBC |
| **VBC_VehAlt_CAN_m** | The altitude of the vehicle (given at meters above sea level) | BV, via VBC |
| **VBC_VehDir_CAN_deg** | The direction that the vehicle is facing (given as degree from North) | BV, via VBC |
| **VBC_VehVel_CAN_mps** | The velocity of the vehicle (given as kilometres per hour) | BV, via VBC |
| **VBC_BatSOC_CAN_perc** | The battery state of charge (given as % capacity remaining) | BV, via VBC |
| **VBC_BatI_CAN_A** | The current experienced by the battery (given as amperes) | BV, via VBC |
| **VBC_BatU_CAN_V** | The battery voltage level (given as volts) | BV, via VBC |
| **VBC_VehCfg_SIM_dl** | Vehicle configuration, 1: refrigerated cargo connected, 2: no refrigerated cargo connected | BV, via VBC |
| **VBC_BatTmp_CAN_degC** | The battery temperature (given as degrees Celsius) | BV, via VBC |
| **VBC_AmbTmp_CAN_degC** | The ambient temperature (given as degrees Celsius) | BV, via VBC |
| **VBC_RtCrds_CAN_deg** | The list of waypoint coordinates that the vehicle must follow (given as arrays of longitudes and latitudes) | ERt, via V2C and VBC |
| **VBC_RqTw_CAN_MMddhhmm** | The list of time windows within which the vehicle must arrive at its | ERt, via V2C and VBC |

| | | |
|---|---|---|
| | respective waypoints (given as array of time in month, day, hour, and minutes) | |
| **VBC_EnEDr_CAN_dl** | The user preference for ECO-driving: 0 = driving in normal mode, 1 = driving in ECO mode | HMI, via V2c and VBC |
| **VBC_WndSpFc_CAN_mps** | The wind speed forecast for the day based on real time weather information (given in meters per second) | WF, via V2C and VBC |
| **VBC_WndDirFc_CAN_deg** | The wind direction forecast for the day based on real time weather information (given in degrees from North) | WF, via V2C and VBC |
| **VBC_RnfllFc_CAN_perc** | The rainfall forecast for the day based on real time weather information (given in % probability of rain) | WF, via V2C and VBC |
| **VBC_Vsblty_CAN_m** | The surrounding visibility forecast for the day based on real time weather information (given in meters around the vehicle that is visible to the driver) | WF, via V2C and VBC |
| **VBC_TrffcDnsty_CAN_perc** | The predicted traffic density of the road based on real time traffic information (given as a percentage capacity of the road filled with vehicles) | TP, via V2C and VBC |
| **VBC_SpLmt_CAN_mps** | The upper and lower speed limits of the road taken from municipal databases (given as a pair of values in kilometres per hour) | TP, via V2C and VBC |

Table 11. Eco-Driving outputs overview

| Output name | Description | Destination block |
|---|---|---|
| **EDr_VehVelStp_CAN_mps** | The velocity setpoint for the driver to follow, displayed in the HMI (given as kilometers per hour) | HMI, via VBC and V2C |
| **EDr_VehAccStp_CAN_perc** | The acceleration suggestion for the driver to try his best to follow (given as accelerator pedal position) | HMI, via the VBC and V2C |
| **EDr_VehBrkF_CAN_perc** | The braking suggestion for the driver to try his best to follow (given as brake pedal position) | HMI, via the VBC and V2C |
| **EDr_PwrPr_INT_W** | The predicted traction power utilization profile (given in kilowatts) | ECf |
| **EDr_FEnECf_INT_dl** | A forced Eco-Comfort enable flag. The flag is set when the battery falls below a given threshold. 1 = Force Enable Eco-Comfort even if the driver did not set it in the HMI), 0 = follow the flag set by the driver in the HMI | ECf |

## 10.4 Eco-Comfort

Table 12.Eco-Comfort inputs overview

| Input name | Description | Source output block |
|---|---|---|
| **VBC_VehTim_CAN_MMddhhmm** | Current vehicle time | VBC |
| **VBC_CbnTmp_CAN_degC** | Cabin measured temperature | VBC |

| | | |
|---|---|---|
| **VBC_CrgTmp_SIM_degC** | Cargo measured temperature | VBC |
| **VBC_AmbTmp_CAN_degC** | Ambient measured temperature | VBC |
| **VBC_AmbHum_CAN_perc** | Ambient measured humidity | VBC |
| **VBC_BatTmp_CAN_degC** | Battery measured temperature | VBC |
| **VBC_BatSOC_CAN_perc** | Battery SoC | VBC |
| **VBC_BatI_CAN_A** | Battery current | VBC |
| **VBC_BatU_CAN_V** | Battery voltage | VBC |
| **VBC_VehCfg_SIM_dl** | Vehicle configuration, 1: refrigerated cargo connected, 2: no refrigerated cargo connected | VBC |
| **VBC_VehMo_CAN_dl** | Vehicle mode, 1: driving, 2: charging | VBC |
| **VBC_RtStrTim_CAN_MMddhhmm** | Vehicle route starting time | VBC |
| **VBC_DrCbnTmpStp_CAN_degC** | Cabin Temperature manually set by driver | VBC |
| **VBC_EnECf_CAN_dl** | Eco-Comfort status, 1: Enable, 0: Disable | VBC |
| **EDr_PwrPr_INT_W** | Traction power demand profile | ECO_driving |

| | | |
|---|---|---|
| **VBC_AmbTmpFc_CAN_degC** | Temperature forecast profile | VBC |
| **VBC_AmbHumFc_CAN_perc** | Humidity forecast profile | VBC |

Table 13. Eco-Comfort outputs overview

| Output name | Description | Destination block |
|---|---|---|
| **ECf_BatTmpStp_CAN_degC** | battery temperature setpoint | VBC |
| **ECf_CbnTmpStp_CAN_degC** | cabin temperature setpoint | VBC |
| **ECf_CrgTmpStp_CAN_degC** | cargo temperature setpoint | VBC |
| **ECf_PcdPwrSch_CAN_kW** | Preconditioning power schedule, one power entry per time stamp | VBC |
| **ECf_PcdTimSch_CAN_MMddhhmm** | Preconditioning time schedule, one entry per time stamp | VBC |

## 10.5 Vehicle to Cloud Interface

Table 14. V2C inputs overview

| Input name | Description | Source output block |
|---|---|---|
| **ERt_FltRqTw_JSON_MMddhhmm** | the tw vehs arrive to requests | ECO_routing |
| **ERt_FltRtStrTim_JSON_MMddhhmm** | Vehicle route starting time | ECO_routing |

| | | |
|---|---|---|
| **ERt_FltRtCrds_JSON_deg** | The routed vehicle trajectory | ECO_routing |
| **HMI_DrCbnTmpStp_JSON_degC** | Cabin Temperature Setpoint | HMI |
| **HMI_EnECf_JSON_dl** | Eco-Comfort status, 1: Enable, 0: Disable | HMI |
| **HMI_EnEDr_JSON_dl** | Eco-Driving status: 1 = Enable, 0 = Disable | HMI |
| **WF_AmbTmpFc_JSON_degC** | Temperature forecast profile | WeatherForecastInformation |
| **WF_AmbHumFc_JSON_perc** | Humidity forecast profile | WeatherForecastInformation |
| **WF_WndSpFc_JSON_mps** | Wind speed forecast | WeatherForecastInformation |
| **WF_WndDirFc_JSON_deg** | Wind direction forecast | WeatherForecastInformation |
| **WF_RnfllFc_JSON_perc** | Rainfall forecast | WeatherForecastInformation |
| **WF_Vsblty_JSON_m** | Visibility distance | WeatherForecastInformation |
| **VBC_PcdTimSch_JSON_MMddhhmm** | Preconditioning time schedule, one entry per time stamp | VBC |
| **VBC_PcdPwrSch_JSON_kW** | Preconditioning power schedule, one power entry per time stamp | VBC |
| **VBC_BatSOC_JSON_perc** | Vehicle 1 SoC | VBC |
| **VBC_BatTmp_JSON_degC** | Vehice 1 battery temperature | VBC |
| **VBC_BatI_JSON_A** | Vehice 1 battery current | VBC |
| **VBC_BatU_JSON_V** | Vehice 1 battery voltage | VBC |

| | | |
|---|---|---|
| **VBC_CbnTmp_JSON_degC** | Vehicle 1 Cabin temperature | VBC |
| **VBC_VehCfg_SIM_dl** | Vehicle 1 configuration, 0 = NA, 1 = cooled cargo, 2 = non cooled cargo | VBC |
| **VBC_BDl_JSON_perc** | Battery capacity left | VBC |
| **VBC_VehMo_JSON_dl** | Vehicle mode, 1: driving, 2: charging | VBC |
| **VBC_VehVelStp_JSON_mps** | Vehicle velocity setpoint | VBC |
| **VBC_VehAccStp_JSON_perc** | Vehicle acceleration setpoint | VBC |
| **VBC_VehBrkF_JSON_perc** | Vehicle Braking force | VBC |
| **TP_TrffcDnsty_JSON_perc** | Road Traffic density | TrafficPredictionInformation |
| **TP_SpLmt_JSON_mps** | Default Speed Limit of Road | TrafficPredictionInformation |

Table 15. V2C outputs overview

| Output name | Description | Destination block |
|---|---|---|
| **V2C_AmbHumFc_JSON_perc** | Humidity forecast profile | VBC |
| **V2C_AmbTmpFc_JSON_degC** | Temperature forecast profile | VBC |
| **V2C_DrCbnTmpStp_JSON_degC** | Cabin Temperature Setpoint | VBC |
| **V2C_EnECf_JSON_dl** | Eco-Comfort status, 1: Enable, 0: Disable | VBC |
| **V2C_EnEDr_JSON_dl** | Eco-Driving status: 1 = Enable, 0 = Disable | VBC |
| **V2C_FltBDl_JSON_perc** | fleet Battery capacity left | ECO_charging |
| **V2C_FltBatI_JSON_A** | fleet battery current | ECO_charging |
| **V2C_FltBatSOC_JSON_perc** | Fleet battery SoC | ECO_charging |

| | | |
|---|---|---|
| **V2C_FltBatTmp_JSON_degC** | Fleet battery temperature | ECO_charging |
| **V2C_FltBatU_JSON_V** | fleet battery voltage | ECO_charging |
| **V2C_FltCbnTmp_JSON_degC** | fleet Cabin temperature | ECO_charging |
| **V2C_FltCfg_JSON_dl** | fleet configuration, 0 = NA, 1 = cooled cargo, 2 = non cooled cargo | ECO_charging |
| **V2C_FltMo_JSON_dl** | fleet mode, 1: driving, 2: charging | ECO_charging |
| **V2C_FltPcdPwrSch_JSON_kW** | Preconditioning power schedule, one power entry per time stamp | ECO_charging |
| **V2C_FltPcdTimSch_JSON_MMddhhmm** | Preconditioning time schedule, one entry per time stamp | ECO_charging |
| **V2C_RnfllFc_JSON_perc** | Rainfall forecast | VBC |
| **V2C_RqTw_JSON_MMddhhmm** | The time window vehicle arrives to requests | VBC |
| **V2C_RtCrds_JSON_deg** | The routed vehicle trajectory | VBC |
| **V2C_RtStrTim_JSON_MMddhhmm** | Vehicle route starting time | VBC |
| **V2C_SpLmt_JSON_mps** | Default Speed Limit of Road | VBC |
| **V2C_TrffcDnsty_JSON_perc** | Road Traffic density | VBC |
| **V2C_VehAccStp_JSON_perc** | Vehicle acceleration setpoint | HMI |
| **V2C_VehBrkF_JSON_perc** | Vehicle Braking force | HMI |
| **V2C_VehVelStp_JSON_mps** | Vehicle velocity setpoint | HMI |

| | | |
|---|---|---|
| **V2C_Vsblty_JSON_m** | Visibility distance | VBC |
| **V2C_WndDirFc_JSON_deg** | Wind direction forecast | VBC |
| **V2C_WndSpFc_JSON_mps** | Wind speed forecast | VBC |

## 10.6 Vehicle Body Computer (VBC)

Table 16. VBC inputs overview

| Input name | Description | Source output block |
|---|---|---|
| **BV_CbnTmp_CAN_degC** | Cabin measured temperature | BaseVehicle |
| **BV_CrgTmp_SIM_degC** | Cargo measured temperature | BaseVehicle |
| **BV_AmbTmp_CAN_degC** | Ambient measured temperature | BaseVehicle |
| **BV_AmbHum_CAN_perc** | Ambient measured humidity | BaseVehicle |
| **BV_BatTmp_CAN_degC** | Battery measured temperature | BaseVehicle |
| **BV_BatSOC_CAN_perc** | Battery SoC | BaseVehicle |
| **BV_VehCfg_SIM_dl** | Vehicle configuration, 1: refrigerated cargo connected, 2: no refrigerated cargo connected. | BaseVehicle |
| **BV_VehMo_CAN_dl** | Vehicle mode, 1: driving, 2: charging | BaseVehicle |
| **BV_BDl_CAN_perc** | Battery capacity left | BaseVehicle |
| **BV_VehTim_CAN_MMddhhmm** | The current tme | BaseVehicle |
| **BV_VehCrds_CAN_deg** | Vehicle location | BaseVehicle |
| **BV_VehAlt_CAN_m** | Vehicle altitude | BaseVehicle |
| **BV_VehDir_CAN_deg** | Vehicle Direction | BaseVehicle |
| **BV_BatI_CAN_A** | Battery current | BaseVehicle |
| **BV_BatU_CAN_V** | Battery voltage | BaseVehicle |
| **BV_VehVel_CAN_mps** | Vehicle Velocity | BaseVehicle |
| **V2C_RqTw_JSON_MMddhhmm** | The time window vehicle arrives to requests | OffboardVehicle2 CloudInterface |

| | | |
|---|---|---|
| **V2C_RtStrTim_JSON_MMddhhmm** | Vehicle route starting time | OffboardVehicle2CloudInterface |
| **V2C_DrCbnTmpStp_JSON_degC** | Cabin Temperature Setpoint | OffboardVehicle2CloudInterface |
| **V2C_EnECf_JSON_dl** | Eco-Comfort status, 1: Enable, 0: Disable | OffboardVehicle2CloudInterface |
| **V2C_AmbTmpFc_JSON_degC** | Temperature forecast profile | OffboardVehicle2CloudInterface |
| **V2C_AmbHumFc_JSON_perc** | Humidity forecast profile | OffboardVehicle2CloudInterface |
| **V2C_RtCrds_JSON_deg** | The routed vehicle trajectory | OffboardVehicle2CloudInterface |
| **V2C_EnEDr_JSON_dl** | Eco-Driving status: 1 = Enable, 0 = Disable | OffboardVehicle2CloudInterface |
| **V2C_WndSpFc_JSON_mps** | Wind speed forecast | OffboardVehicle2CloudInterface |
| **V2C_WndDirFc_JSON_deg** | Wind direction forecast | OffboardVehicle2CloudInterface |
| **V2C_RnfllFc_JSON_perc** | Rainfall forecast | OffboardVehicle2CloudInterface |
| **V2C_Vsblty_JSON_m** | Visibility distance | OffboardVehicle2CloudInterface |
| **V2C_TrffcDnsty_JSON_perc** | Road Traffic density | OffboardVehicle2CloudInterface |
| **V2C_SpLmt_JSON_mps** | Default Speed Limit of Road | OffboardVehicle2CloudInterface |
| **ECf_PcdPwrSch_CAN_kW** | Preconditioning power schedule, one power entry per time stamp | ECO_comfort |
| **ECf_CrgTmpStp_CAN_degC** | cargo temperature setpoint | ECO_comfort |
| **ECf_BatTmpStp_CAN_degC** | battery temperature setpoint | ECO_comfort |
| **ECf_PcdTimSch_CAN_MMddhhmm** | Preconditioning time schedule, one entry per time stamp | ECO_comfort |
| **ECf_CbnTmpStp_CAN_degC** | cabin temperature setpoint | ECO_comfort |

| | | |
|---|---|---|
| **EDr_VehVelStp_CAN_mps** | Vehicle velocity setpoint | ECO_driving |
| **EDr_VehAccStp_CAN_perc** | Vehicle acceleration setpoint | ECO_driving |
| **EDr_VehBrkF_CAN_perc** | Vehicle Braking force | ECO_driving |

Table 17. VBC outputs overview

| Output name | Description | Destination block |
|---|---|---|
| **VBC_AmbHumFc_CAN_perc** | Humidity forecast profile | ECO_comfort |
| **VBC_AmbHum_CAN_perc** | Ambient measured humidity | ECO_comfort |
| **VBC_AmbTmpFc_CAN_degC** | Temperature forecast profile | ECO_comfort |
| **VBC_AmbTmp_CAN_degC** | Ambient measured temperature | ECO_comfort, ECO_driving |
| **VBC_BDI_JSON_perc** | Battery capacity left | |

| | | |
|---|---|---|
| **VBC_BatTmp_JSON_degC** | Battery measured temperature | OffboardVehicle2CloudInterface |
| **VBC_BatU_CAN_V** | Battery voltage | ECO_comfort, ECO_driving |
| **VBC_BatU_JSON_V** | Battery voltage | OffboardVehicle2CloudInterface |
| **VBC_CbnTmpStp_CAN_degC** | Cabin Temperature Setpoint | BaseVehicle |
| **VBC_CbnTmp_CAN_degC** | Cabin measured temperature | ECO_comfort |
| **VBC_CbnTmp_JSON_degC** | Cabin measured temperature | OffboardVehicle2CloudInterface |
| **VBC_CrgTmpStp_CAN_degC** | cargo temperature setpoint | BaseVehicle |
| **VBC_CrgTmp_SIM_degC** | Cargo measured temperature | ECO_comfort |
| **VBC_DrCbnTmpStp_CAN_degC** | Cabin Temperature Setpoint from the driver | BaseVehicle, ECO_comfort |
| **VBC_EnECf_CAN_dl** | Eco-Comfort status, 1: Enable, 0: Disable | ECO_comfort |
| **VBC_EnEDr_CAN_dl** | Eco-Driving status: 1 = Enable, 0 = Disable | ECO_driving |
| **VBC_PcdPwrSch_JSON_kW** | Preconditioning power schedule, one power entry per time stamp | OffboardVehicle2CloudInterface |
| **VBC_PcdTimSch_JSON_MMddhhmm** | Preconditioning time schedule, one entry per time stamp | OffboardVehicle2CloudInterface |
| **VBC_RnfllFc_CAN_perc** | Rainfall forecast | ECO_driving |
| **VBC_RqTw_CAN_MMddhhmm** | The time window vehicle arrives to requests, 30 refers to the nunber of coordinates | ECO_driving |
| **VBC_RtCrds_CAN_deg** | The routed vehicle trajectory, 30 refers to the number of waypoint coordinates | ECO_driving |
| **VBC_RtStrTim_CAN_MMddhhmm** | Vehicle route starting time | ECO_comfort |
| **VBC_SpLmt_CAN_mps** | Default Speed Limit of Road | ECO_driving |

| | | |
|---|---|---|
| **VBC_TrffcDnsty_CAN_perc** | Road Traffic density | ECO_driving |
| **VBC_VehAccStp_JSON_perc** | Vehicle acceleration setpoint | OffboardVehicle2CloudInterface |
| **VBC_VehAlt_CAN_m** | Vehicle altitude | ECO_driving |
| **VBC_VehBrkF_JSON_perc** | Vehicle Braking force | OffboardVehicle2CloudInterface |
| **VBC_VehCfg_SIM_dl** | Vehicle configuration, 1: refrigerated cargo connected, 2: no refrigerated cargo connected. | ECO_comfort, ECO_driving, OffboardVehicle2CloudInterface |
| **VBC_VehCrds_CAN_deg** | Vehicle location | ECO_driving |
| **VBC_VehDir_CAN_deg** | Vehicle Direction | ECO_driving |
| **VBC_VehMo_CAN_dl** | Vehicle mode, 1: driving, 2: charging | ECO_comfort |
| **VBC_VehMo_JSON_dl** | Vehicle mode, 1: driving, 2: charging | OffboardVehicle2CloudInterface |
| **VBC_VehTim_CAN_MMddhhmm** | Current vehicle time | ECO_comfort, ECO_driving |
| **VBC_VehVelStp_JSON_mps** | Vehicle velocity setpoint | OffboardVehicle2CloudInterface |
| **VBC_VehVel_CAN_mps** | Vehicle velocity | ECO_driving |
| **VBC_Vsblty_CAN_m** | Visibility distance | ECO_driving |
| **VBC_WndDirFc_CAN_deg** | Wind direction forecast | ECO_driving |

| | | |
|---|---|---|
| **VBC_WndSpFc_CAN_mps** | Wind speed forecast | ECO_driving |