

Title:	Document Version:
D9.1 WiseEVP Design	1.0

Project Number:	Project Acronym:	Project Title:
H2020-731205	WiseGRID	Wide scale demonstration of Integrated Solutions for European Smart Grid

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type*-Security*:
M18 (April 2018)	M18 (April 2018)	R-PU

\*Type: P: Prototype; R: Report; D: Demonstrator; O: Other.

\*\*Security Class: PU: Public; PP: Restricted to other programme participants (including the Commission); RE: Restricted to a group defined by the consortium (including the Commission); CO: Confidential, only for members of the consortium (including the Commission).

Responsible:	Organisation:	Contributing WP:
Alberto Zambrano	ETRA	WP9
Álvaro Nofuentes	ETRA	WP9

Authors (organization):
Álvaro Nofuentes (ETRA), Alberto Zambrano (ETRA), Joachim Jacob (PARTA), Rik Bellens (PARTAGO), Francesco Bellesini (EMOT), Julio César Díaz (ITE), Javier Monreal (ITE), Antonis Papanikolaou (HYP), Catalin Chimirel (CRE)

Abstract:
This document reports the work performed within Task 9.1 “WiseEVP Design” in terms of functionalities offered and design of the solution for EV fleet managers and EVSE Operators. This framework takes as starting point the work done in the requirement definition, use case identification, and preliminary architectural design.

Keywords:
Electromobility, EV, EVSE, Fleet manager, Smart charging

## Revision History

Revision	Date	Description	Author (Organisation)
V0.0	01.03.2018	New document	Álvaro Nofuentes (ETRA)
V0.1	14.03.2018	First round of contributions	ETRA, ITE, EMOT
V0.2	29.03.2018	Second round of contributions	ETRA, EMOT,
V0.3	13.04.2018	Last contributions	ETRA, ITE, PARTA, CRE
V0.4	16.04.2018	Version ready for peer review	Alberto Zambrano (ETRA) and Álvaro Nofuentes (ETRA)
V0.5	23.07.2018	Integration of feedback received	ETRA, ITE, HYP
V1.0	25.07.2018	Final refinement and ready for submission	Álvaro Nofuentes (ETRA)

## INDEX

<b>EXECUTIVE SUMMARY .....</b>	<b>7</b>
<b>1 INTRODUCTION .....</b>	<b>9</b>
1.1 Purpose of the document .....	9
1.2 Scope of the document.....	9
1.3 Structure of the document .....	9
<b>2 BACKGROUND .....</b>	<b>10</b>
2.1 Overview .....	10
2.2 Relevant actors .....	10
2.3 Requirements .....	11
2.4 High Level use cases .....	14
2.5 Primary use cases.....	14
2.6 Architecture.....	14
<b>3 WISEEVP MODULES .....</b>	<b>17</b>
<b>3.1 Internal service bus .....</b>	<b>17</b>
3.1.1 Context .....	17
3.1.2 Messaging mechanism .....	18
3.1.3 Internal Enterprise Service Bus.....	18
3.1.4 RabbitMQ .....	19
3.1.5 Common communication patterns enabled by the ESB.....	20
3.1.6 Integration with modules .....	20
<b>3.2 Field devices integration.....</b>	<b>21</b>
3.2.1 EVSE Wrapper.....	21
3.2.2 EV Wrappers.....	25
<b>3.3 Charging session optimization and scheduling.....</b>	<b>30</b>
3.3.1 Demand forecast .....	30
3.3.2 Tariff provider.....	35
3.3.3 Energy mix provider.....	37
3.3.4 Booking service.....	37
3.3.5 EVSE scheduler .....	40
<b>3.4 Ancillary services.....</b>	<b>44</b>
3.4.1 Flexibility forecast.....	44

3.4.2 Ancillary Services Market Hub .....	46
<b>3.5 User Interface design.....</b>	<b>47</b>
3.5.1 Dashboard .....	47
3.5.2 Map.....	48
3.5.3 EVSE detail .....	51
3.5.4 EV detail.....	53
3.5.5 Cost indicators for EVs.....	54
3.5.6 Cost indicators for EVSEs .....	54
3.5.7 Optimization .....	55
3.5.8 EVSE groups .....	57
3.5.9 Demand and flexibility forecasts .....	59
3.5.10 Ancillary Services Market .....	60
<b>4 LINK WITH OTHER WISEGRID APPLICATIONS .....</b>	<b>61</b>
<b>5 DPIA CONSIDERATIONS.....</b>	<b>62</b>
<b>6 CONCLUSIONS .....</b>	<b>63</b>
<b>7 REFERENCES AND ACRONYMS.....</b>	<b>64</b>
7.1 References.....	64
7.2 Acronyms.....	65
<b>8 ANNEXES .....</b>	<b>67</b>
8.1 Specification of messages for exchange of flexibility.....	67
8.1.1 FlexRequest.xsd .....	67
8.1.2 FlexRequestResponse.xsd .....	74
8.1.3 FlexOffer.xsd.....	78
8.1.4 FlexOfferResponse.xsd .....	85
8.1.5 FlexOrder.xsd.....	89
8.1.6 FlexOrderResponse.xsd .....	97
8.1.7 FlexOfferRevocation.xsd.....	100
8.1.8 FlexOfferRevocationResponse.xsd .....	105

## List of Figures

Figure 1 – WiseEVP .....	10
Figure 2 – Overview of interactions among the modules of the WiseEVP application .....	16
Figure 3 – Monolithic and micro-service architecture comparison [1] .....	17
Figure 4 – Example of different communication patterns [2] .....	19
Figure 5 – RabbitMQ [3] .....	20
Figure 6 – EVSE Wrapper and its main interactions .....	21
Figure 7 – EV Wrapper and its main interactions .....	26
Figure 8 – OBD2 reader [8] .....	29
Figure 9 – Overview of the approach taken to monitor EV fleet at Crevillent pilot site .....	30
Figure 10 – Forecast server structure integration schema .....	31
Figure 11 – CSVMA network structure .....	32
Figure 12 – Training workflow schema .....	33
Figure 13 – Forecast workflow schema .....	33
Figure 14 – Overview of the inputs of the EVSE scheduler .....	36
Figure 15 – Overview of the workflow of the EVSE scheduler, and the role of the tariff provider ....	37
Figure 16 – Overview of the energy mix provider module .....	37
Figure 17 – Overview of the messages exchanged for the composition of a flexibility offer .....	46
Figure 18 – Overview of the messages exchanged for ordering and delivering flexibility .....	47
Figure 19 – Dashboard .....	48
Figure 20 – EV and EVSE Map .....	49
Figure 21 – EVSE Map .....	50
Figure 22 – EV Map .....	51
Figure 23 – EVSE details .....	52
Figure 24 – EV details .....	53
Figure 25 – EV Cost Indicators .....	54
Figure 26 – EVSE Cost indicators .....	55
Figure 27 – Charging session profiles .....	56
Figure 28 – Optimization .....	57
Figure 29 – EVSE groups .....	58
Figure 30 – Demand forecast .....	59
Figure 31 – Flexibility forecast .....	60
Figure 32 – Flexibility - Demand Response .....	61
Figure 33 – WiseEVP interactions .....	62

## List of Tables

Table 1 – WiseEVP actor's inventory .....	10
Table 2 – WiseEVP requirements .....	12
Table 3 – Terni EVSE's characteristics.....	24
Table 4 – Connection types .....	27
Table 5 – Demand forecast interface, error codes.....	34
Table 6 – Booking data model fields .....	39
Table 7 – Booking data model fields .....	39
Table 8 – Threat and feared events identification for WiseEVP.....	62
Table 9 – List of Acronyms.....	65

## EXECUTIVE SUMMARY

Although electromobility impacts on current power infrastructure and electricity demand is still manageable; a large scale integration of EVs on the European roads and the evolution of charging behaviours will have implications for infrastructure investments in the short and medium term. Proactive integration of the EV charging processes in the new smart grids can result in future savings for utilities and grid operators. Load shifting and supplying power to the grid (V2G) will be part of the future role of EVs in the new grid.

Some issues have to be addressed in order to allow a large-scale deployment of electric vehicle (EV) technology. Besides the very relevant challenges for electric utilities in terms of business models and infrastructure investments, EV technology raises a series of technical issues such as the impact that a large EV market penetration may have on the distribution network or the typical increasing uncertainty and intermittency of profiles associated with any distributed energy resource. As a result, grid management and network operation will become more complex, in terms of load balancing, survivability of network elements and overall power quality.

For that purposes, WiseGRID will deliver WiseEVP, a tool to be used by e-fleet managers and EVSE operators in order to optimize the activities related with the smart charging and discharging of the EVs, making it possible to use EVs as dynamic distributed storage devices, feeding electricity stored in their batteries back into the local distribution network when needed (V2G) and responding to the flexibility requests of the grid.

For this purpose, the platform will provide a reference load profile taking into consideration:

- The renewable generation profile.
- The tariffs
- The requirements from the EV drivers

That will allow to use the EV to better answer demand variations (e.g. use EV as supporting storage unit to cover peaks of demand, or even use the EV storage capability to flatten load curves). It is noteworthy that the aforementioned functionalities will respect EV user preferences, meaning that user constraints (e.g. charge required to be completed within a certain time period) will be prioritised in the charging sessions scheduling process. In order to maximise the flexibility provided by the EVSEs, WiseEVP will include a set of user functionalities, such as user's authentication, EVSE booking and charging session start; considering different types of charging (on demand, smart charging, smart charging with V2G) and therefore will enable to optimally allocate any excess of energy in the distribution lines helping to stabilize the grid when needed.

The design of WiseEVP has been done taking into account the specifications, the requirements and the expected test scenarios envisaged in previous stages of the project. The architecture and the modules reflected in this deliverable assure the performance of the expected characteristics of the tool and their synergic work with the other tools of the WiseGRID framework. In addition, the design of the communication of the internal modules (based on RabbitMQ) has been done in order to assure the proper interaction between the different functionalities of the tool.

In order to better explain the modules of the tools, they have been clustered into different categories: Field devices integration, Charging session optimization and scheduling, ancillary services and User interface design.

### Field devices integration

WiseEVP needs to collect data from the Electric Vehicles and the charging stations in order to work. The modules here explained will make this function.

### Charging session optimization and scheduling

The modules clustered in this section are the ones which has to deal with optimal charging schedule of the EVSEs. These modules can be seen as the core part of the tool.

### **Ancillary services**

The flexibility forecast and Ancillary services market hub modules will provide to WiseEVP the required functionalities for the integration of the tool in a DR framework.

### **User interface design**

The WiseEVP user interface is designed to give a useful insight to the fleet manager and EVSE manager on the different KPIs that are processed by the platform, and which will provide a clear overview of the status, history and trends of the different aspects related to the energy demand of the monitored elements.





## 1 INTRODUCTION

### 1.1 PURPOSE OF THE DOCUMENT

The present document aims to describe and explain the results acquired from the Task 9.1 “WiseEVP Design”. This task mainly deals with the design of the modules and functionalities of the WiseEVP tool which is going to be used by EV fleet managers and EVSE operators. Those specifications are explained with technical and accurate language but being comprehensive to several types of readers.

The functionalities and capabilities of this tool have been developed following the requirements and the use cases that have been set in the D2.1 “WiseGRID requirements, use cases and Pilot Sites analysis” as it is possible to check within the document.

Moreover, the authors of this document have provided a univocal view of the tool but considering the relationships between it and the other WiseGRID tools having in mind that they share common modules and have internal dependencies. To reach this objective, the authors have shared and collected information from other deliverables in order to avoid readers to be aware of the work done in other WPs to totally understand this product.

However, the information included in this deliverable could be enhanced later during the development phase and those improvements will be reported in the deliverable D9.2 “WiseEVP platform implementation and lab-testing”.

### 1.2 SCOPE OF THE DOCUMENT

The extent of this deliverable is focused in the needs that the WiseEVP has to address in order to provide valuable functionalities to its users. The description of the different modules and the design activities have been addressed within the document leaving the deployment and implementation issues for upcoming deliverables which have to deal with those tasks.

The design provided in this document could be seen as the starting point for the further implementation of the WiseEVP tool, by taking into account all the foundation documents provided previously in WiseGRID.

### 1.3 STRUCTURE OF THE DOCUMENT

Apart from this introductory section, the current document is structured as follows:

Firstly, it is explained the background needed to understand the previous work done in other WPs related to WiseEVP. For that purpose, the authors have included the required information from other deliverables. This section provides the basis for the future sections of the document.

Following that section, the document describes the different modules of the WiseEVP platform and how these modules address the required functionalities of the product. In order to give to the reader a high-level view of the tool, the modules have been clustered into different categories and thus a better understanding of the product is achieved. This section could be seen as the core of the document and the most technical part of it.

Once the deliverable has explained WiseEVP as a single product, the authors give to the reader a high-level view of the position of the tool inside the WiseGRID framework and some considerations about data protection.

Finally, the document ends with a brief summary and the exposition of the main conclusions that can be extracted from the whole document.

## 2 BACKGROUND

### 2.1 OVERVIEW

WiseEVP is the WiseGRID technological solution for:

- Vehicle-sharing companies or EV fleet managers
- EVSE infrastructure operators

In order to optimize the activities related with smart charging and discharging of the EVs including V2G (vehicle to grid, energy injection in the distribution network) and V2B (vehicle to building). The management of the EVs and the EVSEs charging and discharging processes will meet the following objectives:

- Reduce the EV charging energy bill.
- Follow flexibility requests from DSO to help the electric distribution network operation in exchange for an economic consideration.
- Follow high shares of RES production.
- Manage and control the whole EV fleet.

All the aforementioned objectives will be subordinated to the EV user preferences: desired state of charge (SoC) at the time of unplugging the EV.



Figure 1 – WiseEVP

### 2.2 RELEVANT ACTORS

During the first year of the project, the consortium has identified who are the main actors that participate in the WiseGRID framework. In this subsection will be shown and explained who are the actors (whatever the type: organization, system, person or device) that are included in the WiseEVP framework. The consortium has identified 21 actors that can be directly linked with the WiseEVP.

Table 1 – WiseEVP actor's inventory

Actor name	Description	Actor type
Aggregator	Accumulates flexibility from Prosumers and Consumers and sells it to the Supplier, the DSO or the TSO.	Organization
Data Provider	Independent entity responsible for undertaking and coordinating the information exchange and translation of the data of various sources into a common data model.	Organization
Distributed Energy Resource	Any type of generation units, storage units and load flexibility resources connected to the distribution network.	System

Actor name	Description	Actor type
DSO	<i>Distribution System Operator</i> . The entity responsible for: the distribution network planning and development; the safe and secure operation and management of the distribution system; for data management associated with the use of the distribution system; for procurement of flexibility services.	Organization
Electronic Meter	A physical device containing one or more registers.	Device
ESCO	<i>Energy Service COmpany</i> . Offers auxiliary energy-related services to Prosumers.	Organization
EV	<i>Electric Vehicle</i> . A vehicle that uses stored electricity as a source of energy.	Device
EV Fleet Manager	<i>Electric Vehicle Fleet Manager</i> . An organization that operates and controls an EV fleet.	Organization
EV User	<i>Electric Vehicle User</i> . The user of an EV.	Person
EVSE	<i>Electric Vehicle Supply Equipment</i> . The infrastructure external to the EV that provides connection to a power source for charging the EV.	Device
EVSE Operator	<i>Electric Vehicle Supply Equipment Operator</i> . The entity responsible for managing and operating the EV charging infrastructure.	Organization
Forecast Provider	The organization that provides, upon demand, forecasts regarding certain variables (e.g. electricity demand, RES production, weather conditions, etc.)	Organization
GIS	<i>Geographical Information System</i> .	System
Market Operator	The unique power exchange of trades for the actual delivery of energy that receives the bids from the Balance Responsible Parties that have a contract to bid. Determines the market energy price taking into account the technical constraints from the Transmission System Operator.	Organization
Producer	An entity connected to the grid that injects electricity to the grid.	Person
Prosumer	An entity that consumes and produces energy. There is no distinction between residential end-users, small and medium-sized enterprises or industrial users.	Person
RES Unit	<i>Renewable Energy Source Unit</i> . A type of Producer that transforms energy from renewable energy sources (e.g. sun, wind, etc.) to electricity and injects it to the grid.	Device
Sensor	A device that monitors and processes specific input from the physical environment (e.g. light, heat, motion, etc.).	Device
Smart Meter	An Electronic Meter with two-way communication capabilities.	Device
Storage Unit	A device that stores energy.	Device
Supplier	Supplies and invoices energy to its customers.	Organization

## 2.3 REQUIREMENTS

In order to be sure that the features and characteristics of the WiseGRID technological solutions meet the project goals, the consortium created a list of requirements that each partner agreed. The complete list of requirements can be found in Deliverable 2.1, however, in this subsection will be shown the ones related with the WiseEVP.

**Table 2 – WiseEVP requirements**

Requirement ID	Description	Type
EVP_001	Different models of EV's should easily be integrated in WiseEVP.	Functional and data requirements
EVP_002	Charging speed of charging station can be manually set.	Usability and humanity requirements
EVP_003	WiseEVP must provide API's to allow integration in other fleet management tools.	Functional and data requirements
EVP_004	WiseEVP collects data from vehicles and charging stations on one platform.	The scope of the product
EVP_005	Different models of charging stations should easily be integrated in WiseEVP.	The scope of the product
EVP_006	WiseEVP should provide basic fleet management functionalities: maintenance, documents administration, damage reporting, etc.	Usability and humanity requirements
EVP_007	WiseEVP should have a simple mechanism to link the availability / calendar of each car to the charging control engine.	Usability and humanity requirements
EVP_008	WiseEVP should perhaps also include access to a buffer of energy, such as a collection of standalone fixed batteries.	Operational requirements
EVP_010	WiseEVP will receive the state of the distribution network from the WiseGRID Cockpit.	Functional and data requirements
EVP_014	WiseEVP will receive the DSO flexibility needs through the WiseGRID Cockpit.	The scope of the product
EVP_015	WiseEVP will calculate the aggregated flexibility capabilities of the EVSEs under its management.	The scope of the product
EVP_017	WiseEVP will reschedule the EVSE charging sessions to meet DSO needs.	The scope of the product
EVP_018	WiseEVP will allow EVSE booking.	The scope of the product
EVP_019	WiseEVP will establish a two way communication with DSOs.	The scope of the product
EVP_020	WiseEVP should have a link to energy markets (energy exchange, balancing and ancillary services markets).	Functional and data requirements
EVP_021	The platform will share data in order to provide information to the demand forecasting module.	Functional and data requirements
EVP_022	WiseEVP could merge data from owned EVSE and from publicly available EVSE.	The scope of the product
EVP_024	EVSE unique identifier	Functional and data requirements
EVP_025	Smart card or unique digital token per user	Functional and data requirements
EVP_026	The users have to provide their personal data, EV data and preferences in the context of EV.	Usability and humanity requirements
EVP_027	Clear identification of the EVSE on its casing	Usability and humanity requirements
EVP_028	The end-user should be registered and authorised in order to charge and/or book an EV in the system.	Usability and humanity requirements

Requirement ID	Description	Type
EVP_029	List of available charging stations with booking possibilities	Operational requirements
EVP_031	EV status available (SoC)	Functional and data requirements
EVP_032	Notification of near charging stations	Usability and humanity requirements
EVP_033	WiseEVP should perform and make use of grouping of EVSE in regulation areas.	Operational requirements
EVP_034	It must facilitate the management of EV fleet of industries/companies, etc.	The scope of the product
EVP_035	WiseEVP should support the end-user's ability to choose the preferable type of EV charging (among the ones described in HL-UC 3).	Functional and data requirements
EVP_036	WiseEVP must provide real or near real-time monitoring of the EVSE equipment (and the EV fleet) to the EVSE manager (and EV fleet manager).	Functional and data requirements
EVP_037	Anonymising charging data to assure data privacy in case of private EVs.	Legal requirements
EVP_038	In case of EV fleets, WiseEVP shall provide to the EV fleet manager (owner) information on the EVs usage and position.	Functional and data requirements
EVP_040	WiseEVP shall provide historical consumption and charging session data from each EVSE.	Functional and data requirements
EVP_041	WiseEVP should contain a pricing mechanism for rewarding the performed balancing.	Functional and data requirements
EVP_042	WiseEVP should provide information about the energy price curves to the users.	Functional and data requirements
EVP_043	WiseEVP communicates with charging stations through open smart charging protocol of the Open Charge Alliance (OSCP).	Functional and data requirements
EVP_044	The status of EVSE can be changed easily from public, private, out-of-order, etc. by the manager, or by the EVSE itself.	Functional and data requirements
EVP_045	WiseEVP gathers data from EVs (location, SoC, status of the car, etc.) via API.	Functional and data requirements
EVP_046	The EVSE operator or the fleet manager has an updated list of known users (for authentication process).	The scope of the product
EVP_047	The EV user has a valid account in the WiseEVP for the authentication process.	The scope of the product
EVP_048	The EV user only will provide flexibility to the system allowing the system to module his charging session if he wants (charging types 2 and 3). If not, his car will be charged asap (charging type 1).	Users of the product
EVP_049	The EVSE Operator or the EV Fleet manager will be able to configure the EVSE network through the WiseEVP (EVSE location, number of sockets, maximum and minimum power, etc.).	The scope of the product
EVP_050	WiseEVP will calculate the EV load forecasting per regulation area and/or for the whole system.	The scope of the product
EVP_051	WiseEVP will calculate the reference load profile per regulation area.	The scope of the product
EVP_052	WiseEVP will schedule the type 2-3 charging sessions of the EVSE network following an economic criterion if no other grid or RES requests are triggered.	The scope of the product

Requirement ID	Description	Type
EVP_053	The EVSE Operator needs to know the regulation areas division based on coordinates or based on network topology (this information may be retrieved from the DSO outside the system).	Functional and data requirements
EVP_054	Before starting with the charging session schedule processes, the data collection from EVSE should be performed to have historical data and to know the status of the EVSEs.	Functional and data requirements
EVP_055	WiseEVP will modulate the power output of his EVSE network to help the DSO with the grid operation.	The scope of the work
EVP_056	WiseEVP will modulate the power output of his EVSE network to maximize the RES integration answering the flexibility requests performed by the RESCOs or Aggregators through the WG StaaS/VPP.	The scope of the product
EVP_057	WiseEVP allows setting a pricing model for booking an EVSE: reservation cost, cancellation cost, extending booking cost, etc.	Functional and data requirements
EVP_058	WiseEVP has a console at an EVSE or a hub of EVSEs.	Functional and data requirements
EVP_060	WiseEVP should allow grouping EVSEs in one hub, sharing the same location.	Functional and data requirements

## 2.4 HIGH LEVEL USE CASES

In order to facilitate the development and the assessment of the WiseGRID solutions, the demonstrations will be conducted following 7 High Level Use Cases. The WiseEVP is directly linked with one HL-UC and its description is shown hereafter.

- HL-UC 3: E-MOBILITY INTEGRATION IN THE GRID WITH V2G

Integration of e-mobility and electric transport systems into the network with the implementation of the V2G technology in order to provide services to the grid, such as storage capacity.

## 2.5 PRIMARY USE CASES

Simultaneously to the requirements' definition, a set of Primary Use Cases were defined in order to envisage which will be the main actions and scenarios to be tested in the project demonstration. The ones in which the WiseEVP is the main tool are:

- HL-UC 3\_PUC\_1\_EVSE and EV fleet monitoring
- HL-UC 3\_PUC\_2\_Interaction of the user with EVSE
- HL-UC 3\_PUC\_3\_EV charging management
- HL-UC 3\_PUC\_4\_Interaction with the energy infrastructure

In Deliverable 2.1 it is possible to find further information about these Primary Use Cases.

## 2.6 ARCHITECTURE

Based on the requirements and list of Use Cases to be realized by the WiseEVP application, the following modules have been defined:

### Data ingestion

The first step considered in the design of the application is the data ingestion. The procedure followed is common to other applications in the project, and implies the following steps:



1. Publication of data from Wrappers to the WiseGRID IOP Message Broker. Following the principle taken in the overall project, data sources publish data to the Interoperable Platform, allowing different application with the corresponding permissions to access to those data flows
2. Subscription to data flows of interest. In the case of the WiseEVP, the main data flows considered are those used to monitor the status of the Electric Vehicles (EVs) and Electric Vehicle Supply Equipment (EVSEs). Necessary software adaptations will be develop to allow those elements to publish data about their status that can be used by any application, and in particular by the WiseEVP, to track and monitor those units in real-time. In addition, the architecture also contemplates the possibility of EVSE managers controlling RES and storage units as well, whose management will be also considered. This subscription is performed by the *RT monitor* module
3. Store data for further analysis. The *RT monitor* module is in charge of populating both the *Operation* and the *Long-term DB* for further analysis

### Data analysis

Under this group, different modules have been defined in order to process the raw data coming from the different data sources in order to get the relevant information out of those. These modules include:

- *KPI engine* module, in charge of extracting different indicators and patterns from the raw data, focusing in energy demand requirements of the fleet of Electric Vehicles, usage of the EVSEs, energy cost and environmental impact analysis, etc.
- *Demand forecast* module, providing forecasts for the energy demand of the fleet of vehicles and the portfolio of EVSEs
- *Flexibility forecast* module, analysing the amount of flexibility that may be offered to the DSO when support for ensuring the quality of the distribution supply is requested

### Operation and control

Under this group, different modules have been defined implementing specific tasks in order to fulfil the different functional requirements of the application. The main functionality of WiseEVP with regards to control of the EVSEs is the implementation of Smart Charging strategies taking into account the complete set of EVSEs under control, with the objective of regulating the charging sessions in order to achieve an optimum schedule that either minimises the economic cost of the energy, or maximises the usage of green energy. The module implementing the algorithm that takes all constraints into account – reservations, flexibility sold to the DSO, EV fleet requirements, etc. - to calculate this optimum schedule is the *EVSE scheduler*.

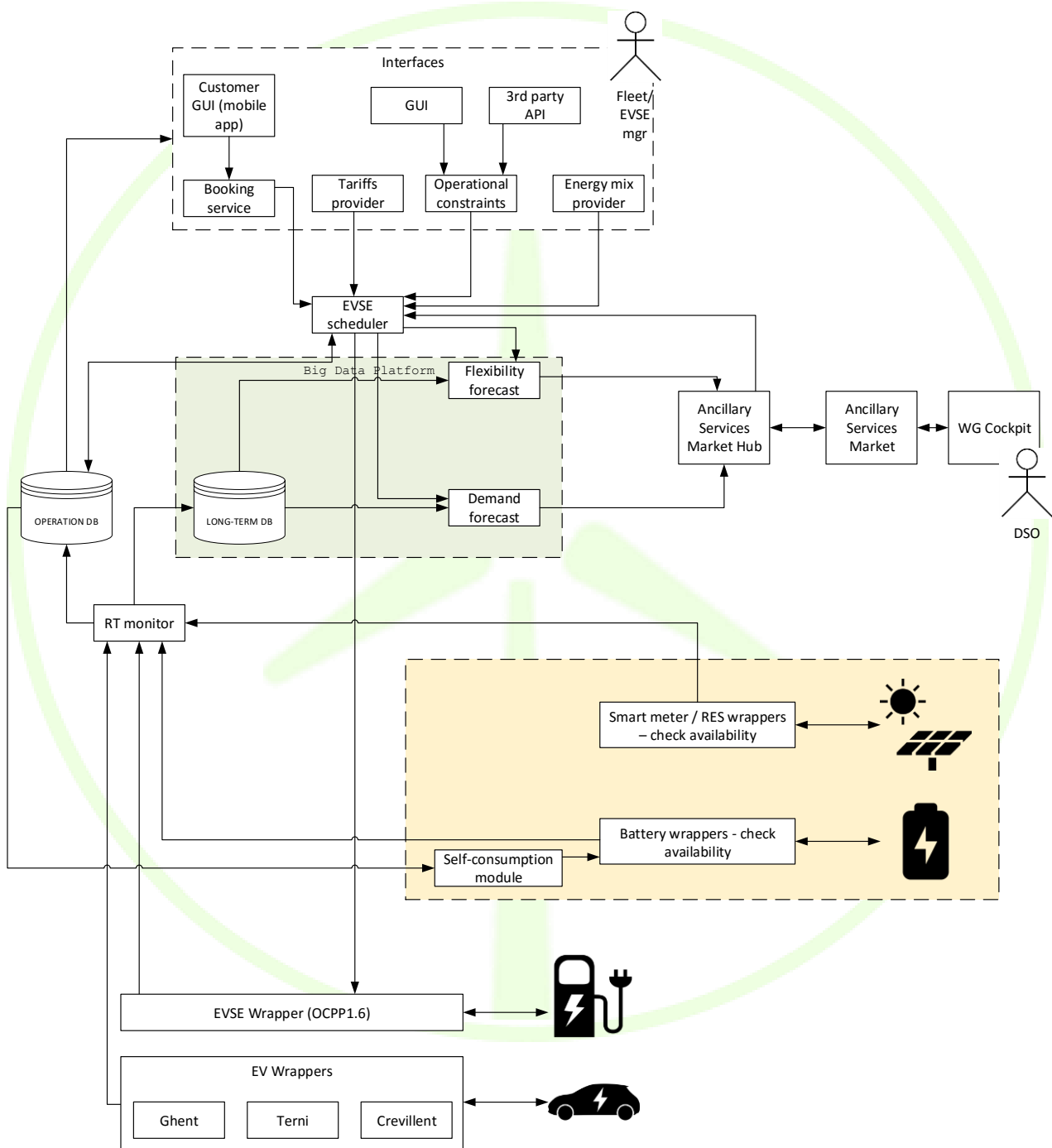
### Interaction with other applications

WiseEVP will be used to demonstrate how EVSE management systems implementing Smart Charging strategies can be beneficial to support DSO ensure the quality of the supply and meet with punctual peaks of demand. With this objective, WiseGRID Cockpit, the tool targeting DSO operators, will interact with WiseEVP through the *Ancillary Services Market*, in order to request support for assisting the correct operation of the distribution grid when required. WiseEVP participation in these market will be realized by modifying the Charging Sessions accordingly to meet both the WiseEVP operator and the DSO requirements simultaneously to the possible extent.

### Horizontal and support functionalities

Different modules will be used indirectly by the WiseEVP application. Summarizing, these modules are data providers that offer information needed by other modules of the application to fulfil their duties, which are reused among different applications developed within the project. The list includes the *Weather Forecast* – whose information will assist the forecast modules - , *Energy mix provider* – whose data is used to assess the

environmental impact of the used energy – and *Tariff Provider* modules – giving access to the actual price of the energy used. In addition, the *Big Data platform* that will support the long-term storage and analysis. Finally, the *WiseEVP User Interface* is included in this category, providing web-based access to the information and functionalities provided by the other modules, as well as allowing the operator to introduce the business constraints that need to be taken into account (e.g. required availability of the fleet or reservation of EVSEs).



**Figure 2 – Overview of interactions among the modules of the WiseEVP application**



### 3 WISEEVP MODULES

#### 3.1 INTERNAL SERVICE BUS

The architectural approach taken in the design of the applications of the project can benefit from the integration within the applications of the corresponding private internal service buses to handle the communication flows between the modules. This section presents the context leading to such decision and describes the adopted technology.

##### 3.1.1 Context

###### 3.1.1.1 The micro-services approach

As presented in the architectural view of WiseEVP, the application itself is actually composed by a set of specialized software modules, each of those dealing with its own particular task. This design approach has the following main principles:

- A service-oriented architecture is composed of loosely coupled elements that have bounded contexts.
- Services can be updated independently without affecting the others.
- Database coupling is avoided as well, thus preventing side effects produced by parallel access (and possible modification) of different modules to the same data source.
- Services communicate with each other strictly through APIs. They don't share data structures, database schemas, or other internal representations of objects.
- Components are stateless: they don't store any information related to previous requests. An incoming request can be sent to any instance of the service.

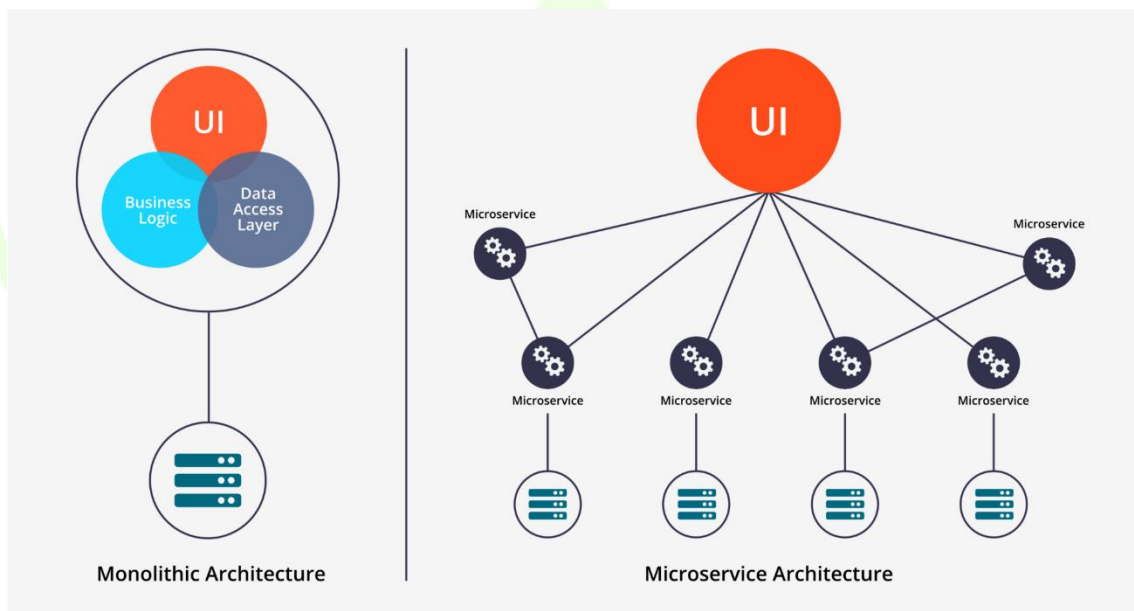


Figure 3 – Monolithic and micro-service architecture comparison [1]

Dividing applications in a set of micro-services presents several advantages when compared to the monolithic approach, the most important ones being:

- Scalability: each one of the modules can be replicated in different machines in order to increase the throughput as required
- Graceful degradation: if one of the modules fails to perform its work, functions of the system not affected by the failure will continue working. Single points of failure are avoided as far as possible

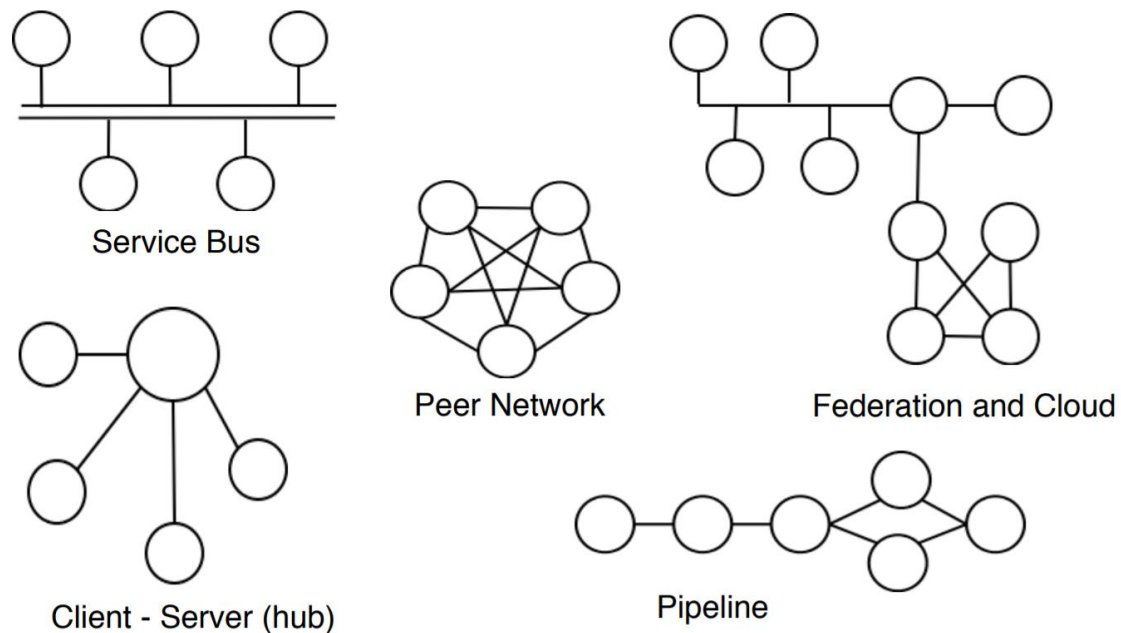
### 3.1.2 Messaging mechanism

In order to handle communications between the different micro-services, a reliable messaging mechanism is needed to appropriately deliver the data flows established between the different modules. The following properties have been taken into account when selecting the proper platform to handle messaging:

- Easy setup: approaches allowing simple configuration of the micro-services are preferred
- Monitoring of communications: systems allowing to monitor communications are desirable, since those can significantly help identifying runtime problems, bottlenecks, etc.
- Asynchronous communication: overall system performance, scalability and maintainability can benefit from an asynchronous communication pattern between the different modules
- Durability of the messages: messaging approaches are seek that offer persistence of the messages in those cases when receiver is not able to process those (e.g. the process is down), and leverage the data source from handling destination unavailability. This feature simplifies the design this task and making it in order to successfully implement an asynchronous system, the messaging platform needs to offer the ability to persist undelivered messages until the receiver of those is ready to process them
- Policies for access control and filtering
- Scalability: one of the advantages of using a micro-service oriented approach is its possibilities to easily scale when workload limit is reached. The communication mechanism employed shall facilitate this task, and be able to scale together with the system as well
- Flexible delivery patterns: in order to optimize communications, the messaging mechanism shall allow different patterns, such as publish/subscribe, one-to-many delivery, etc.
- Support of different protocols: by supporting different communication protocols, flexibility is given to the system and to the design of the micro-services, even allowing that software modules using different protocols can communicate with each other

### 3.1.3 Internal Enterprise Service Bus

Taking into account the requirements just presented, an internal service bus has been selected as the most appropriate approach to handle the communications between the different modules within WiseEVP.



**Figure 4 – Example of different communication patterns [2]**

The use of an internal service bus presents the following advantages:

- Simplifies configuration of the modules: in a system with a relatively high number of modules/micro-services that interact among each other, direct communication may lead to complexity for maintaining overall configuration. By centralizing communications through a central system, configuration is simplified
- Enables monitoring of communications between the different modules, making it easier to detect failures or bottlenecks in the system
- Decouples data sources from data sinks, adding versatility to the design of the data flows

### 3.1.4 RabbitMQ

RabbitMQ has been selected as the platform to implement an enterprise service bus within the application, since it features the set of desired characteristics exposed in the previous point. It also supports integration mechanisms (such as shovels – automatic delivery of messages to remote messaging platforms - and federation – automatic aggregation of messages from different sources into a single stream -) with different instances of RabbitMQ, thus enabling further integration possibilities with systems also using RabbitMQ, as the WiseGRID IOP message broker.

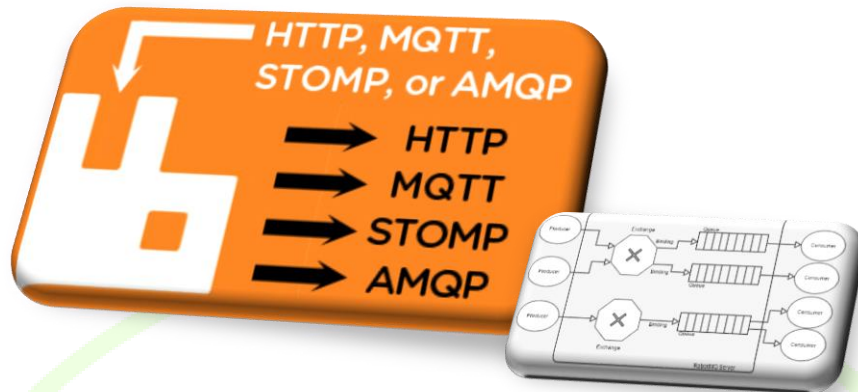


Figure 5 – RabbitMQ [3]

### 3.1.5 Common communication patterns enabled by the ESB

#### 3.1.5.1 Publish-subscribe

Publish-subscribe is a pattern for asynchronous communication particularly suited for one-to-many, many-to-one and many-to-many communication flows scenarios, very typical in IoT applications.

This pattern defines three core elements:

- **Publisher:** is an element producing data that needs to be processed by a third party module
- **Consumer:** is an element that is interested in processing data from different publishers
- **Broker:** the central element to the communication, which matches the information published by publishers with the information requirements of the consumers, and delivers the messages appropriately

#### 3.1.5.2 Asynchronous RPC

Asynchronous RPC is a pattern enabling Remote Procedure Calls (one-to-one flows) while offering the advantages of asynchronous architectures. It defines three core elements:

- **Client:** needs to consume a service offered by a server, without blocking until the response is received
- **Server:** offers a service, processing the incoming requests in order to produce a response
- **Broker:** the central element that upon reception of a RPC requests, determines the server that is capable of fulfilling it, and stores it in work queue for that server. Server is continuously checking if there exist pending requests in that queue, and processing them. If server is stopped or under heavy load, the broker maintains the elements in the queue to be processed as soon as the server is available.

### 3.1.6 Integration with modules

In order to successfully integrate the modules with the internal service bus, those need to specify the following properties for each one of the communication flows they implement:

- **Protocol to be used:** the module needs to select between HTTP, MQTT or AMQP. RabbitMQ supports all three and provides interoperability among them. The main characteristics are:
  - **MQTT:** MQTT (MQ Telemetry Transport or Message Queuing Telemetry Transport) is an ISO standard (ISO/IEC PRF 20922) publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol. It is designed for connections with remote locations where a "small

code footprint" is required or the network bandwidth is limited, therefore very popular in the field of IoT sensors. It uses publish-subscribe messaging pattern.

- AMQP: The Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security.
- HTTP: The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, and therefore also supported due to its availability in all kind of technologies.
- Details depending on the selected protocol:
  - MQTT topic where data will be published (publish/subscribe pattern). RabbitMQ provides seamless interaction between MQTT and AMQP clients, by mapping the concepts of MQTT topics to AMQP Routing Keys
  - AMQP queue where the module listens for requests (RPC pattern). If this is the case, further properties may be declared, such as "reply\_to" (sender specifies the name of the queue where the response is expected) and "correlation\_id" (sender specifies a unique id to the message that must be also included in the response)
  - Message payload: message payload supports any kind of content. Most modules will actually use JSON messages.

## 3.2 FIELD DEVICES INTEGRATION

### 3.2.1 EVSE Wrapper

The EVSE wrapper is the module within WiseEVP that will implement the communication with the different charging stations to be integrated with the application.

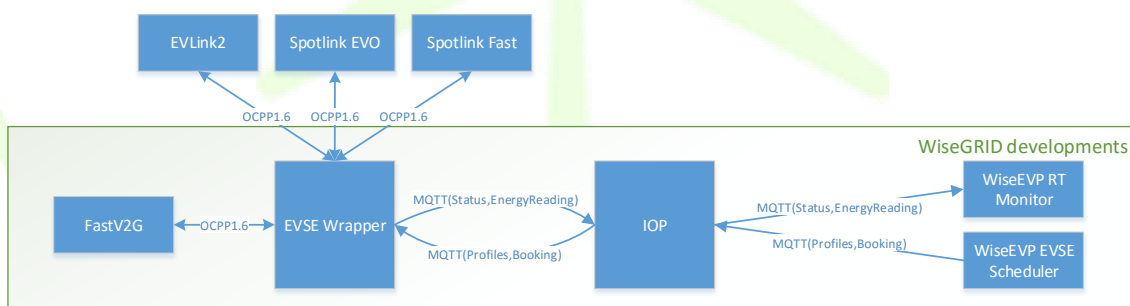


Figure 6 – EVSE Wrapper and its main interactions

#### 3.2.1.1 Communication between EVSEs and EVSE Wrapper

The purpose of the EVSE Wrapper is to establish communication with the EVSEs, handle the whole operative and offer the high level information required by the WiseEVP platform for the data analysis.

After an analysis of the available EVSEs to be integrated with WiseEVP, the decision has been taken to support OCPPv1.6 SOAP and JSON versions [4].

The following list is the minimum required set of messages (as defined by OCPPv1.6) needed to integrate an EVSE with the EVSE Wrapper:

- Mandatory messages
  - Heartbeat: needed to track the comm. status of the EVSEs.

- StatusNotification: needed to track status of the EVSE and its plugs.
- Authorize: WiseEVP will handle the list of authorized users.
- StartTransaction: needed to track occurrence and characteristics of charging sessions.
- StopTransaction: needed to track occurrence and characteristics of charging sessions.
- SetChargingProfile: needed to modulate the energy demand of the system
- Encouraged messages (allowing development of further functionalities on the EVSEs)
  - BootNotification: enables the automatic registration of new EVSEs in the WiseEVP
  - DataTransfer: opens the possibility to receive extra information from the WiseEVP
  - DiagnosticStatusNotification: enables failure detection
  - ReserveNow: enables booking capabilities of the WiseEVP

### 3.2.1.2 Communication between EVSE Wrapper and WiseEVP

MQTT is foreseen as the protocols to be used to implement communication via IOP, between the EVSE Wrapper and the modules of the WiseEVP.

For this communication, a subset of the OCPPv1.6 specification has been selected. Messages exchanged via IOP will follow the OCPPv1.6 JSON specification (as defined by [5]) and will be encapsulated in MQTT payloads.

Base topic: **MQTT/EV/[UID of the EVSE]**

Base topic uniquely identifies the EVSE in the whole system

#### 3.2.1.2.1 Messages initiated by EVSE Wrapper:

Topic: **bootnotification**

After start-up, a Charge Point SHALL send a request to the Central System with information about its configuration (e.g. version, vendor, etc.). This mechanism facilitates the automatic registration of new EVSEs into the platform.

QoS-Level: 1

E.g. MQTT/EVSE/EVSE00001/bootnotification

Payload JSON Schema: BootNotificationRequest

Topic: **metervalues**

A Charge Point MAY sample the energy meter or other sensor/transducer hardware to provide extra information about its meter values.

QoS-Level: 1

E.g. MQTT/EVSE/EVSE00001/metervalues

Payload JSON Schema: MeterValuesRequest

Topic: **status**

A Charge Point sends a notification to the Central System to inform the Central System about a status change or an error within the Charge Point.

QoS-Level: 1

E.g. MQTT/EVSE/EVSE00001/status

Payload JSON Schema: StatusNotificationRequest

**Topic: starttransaction**

The Charge Point SHALL send a StartTransaction.req PDU to the Central System to inform about a transaction that has been started.

QoS-Level: 1

E.g. MQTT/EVSE/EVSE00001/starttransaction

Payload JSON Schema: StartTransactionRequest

**Topic: stoptransaction**

When a transaction is stopped, the Charge Point SHALL send a StopTransaction.req PDU, notifying to the Central System that the transaction has stopped.

QoS-Level: 1

E.g. MQTT/EVSE/EVSE00001/stoptransaction

Payload JSON Schema: StopTransactionRequest

### **3.2.1.2.2 Messages initiated by EVSE Scheduler**

**Topic: reservenow**

A Central System can issue a ReserveNow.req to a Charge Point to reserve a connector for use by a specific idTag.

QoS-Level: 1

E.g. MQTT/EVSE/EVSE00001/reservenow

Payload JSON Schema: ReserveNowRequest

**Topic: cancelreservation**

To cancel a reservation the Central System SHALL send a CancelReservation.req PDU to the Charge Point.

QoS-Level: 1

E.g. MQTT/EVSE/EVSE00001/cancelreservation

Payload JSON Schema: CancelReservationRequest

**Topic: setchargingprofile**

Central System can send a SetChargingProfile.req to a Charge Point, to set a charging profile. At this particular point, an extension is proposed by the project to enable V2G profiles as well by using negative values.

QoS-Level: 1

E.g. MQTT/EVSE/EVSE00001/setchargingprofile

Payload JSON Schema: SetChargingProfileRequest

### **3.2.1.3 EVSEs available per pilot site**

#### **3.2.1.3.1 Terni**



Emotion will deploy three electric vehicles (EVs) and three electric vehicle supply equipments (EVSEs) in Terni pilot site for demonstrating WiseGRID integrated ecosystem.

Regarding EVSEs, in Terni will be used two Spotlink EVO and one Spotlink FAST; the EVSEs just mentioned are produced by Emotion and in the table below their characteristics are shown.

**Table 3 – Terni EVSE's characteristics**

 <p><b>Spotlink EVO</b></p>	<b>Plug Numbers</b>	1/2
	<b>Plug Type</b>	Type 2
	<b>Power MAX</b>	25 kW
	<b>Voltage</b>	400 V
	<b>Frequency</b>	50-60 Hz
	<b>Controllable Power</b>	Yes
	<b>Monophase/Triphase</b>	Both
	<b>Degree of Protection</b>	IP 54
	<b>Interlocking</b>	Yes
	<b>Display</b>	Touch screen 7"
	<b>RFID</b>	Yes
	<b>Communication Protocol</b>	OCPP
 <p><b>Spotlink FAST</b></p>	<b>Plug Numbers</b>	2
	<b>Plug Type</b>	CCS/CHADEMO
	<b>Power MAX</b>	50 kW
	<b>Voltage Input</b>	300-500 V (AC)
	<b>Voltage Output</b>	50-900 V (DC)
	<b>Frequency</b>	50-60 Hz
	<b>Controllable Power</b>	Yes
	<b>Monophase/Triphase</b>	Triphase
	<b>Degree of Protection</b>	IP 54
	<b>Interlocking</b>	Yes
	<b>Display</b>	Touch screen 7"
	<b>RFID</b>	Yes
	<b>Communication Protocol</b>	OCPP
	<b>Reporting</b>	Yes

The EVSE data will be pushed to the WiseEVP through EV wrappers using MQTT protocol and the EVSEs will be accessible by third parties through OCPP v1.6 JSON protocol.



### 3.2.1.3.2 Ghent

At the Belgian pilot site, 2 charging points will be available for testing purposes. Those charging points are currently connected with a commercial platform that manages authentication, charging sessions, monitoring and public station management including access control and public pricing. However, these charging stations are not public and only in use by Partago cars. The charging points are of the type EVLink 2 (manufacturer Schneider), one with max 22kW power output and the other one with max 11 kW. Both are equipped with an RFID reader for authentication, and are communicating by the OCPP1.6 protocol via wireless internet connection. Every charging point has a digital meter to measure the total amount of electricity consumed, and are currently set up as “plug ‘n’ charge”, since this is most preferred by the users of Partago. Nevertheless, every Partago electric car has its own charging pass that can activate a charging session at public charging stations. The charging points are installed behind the EAN connection point of homes of Partago members. Other assets in these homes are SMX smart meters and PV panels. One home is also equipped with a heat pump. While not relevant for WiseEVP, these assets are available too for integrating by other WiseGRID tools.

At the premises of Partago (co-working space) there is a Schneider EVLink 2 Parking to be installed in May 2018. This EVSE is a pole with 2 charging plugs, each of mennekes mode 3, 22kW charging. It is also equipped with RFID and mobile internet connection. It is also located behind the EAN connection of the premises, so not directly and independently attached to the local distribution grid. One of the plugs will be used publicly, the other only for the shared electric cars of Partago. This charging station will also be managed by a commercial platform (see paragraph above), but can be temporarily disconnected from this platform for testing purposes of WiseEVP. There are no other controllable assets and RES production at the site.

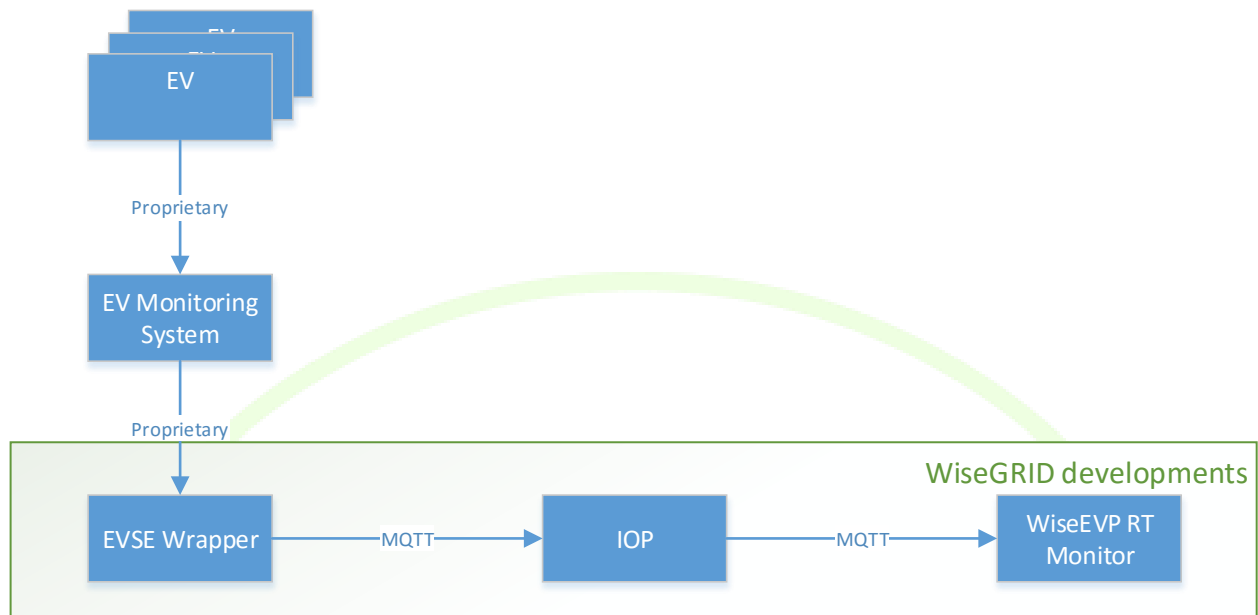
The pilot site of Flanders is in continuous development, with new EVs and charging stations being expected to be installed on a regular basis. All charging stations will be installed behind the EAN connection point.

### 3.2.1.3.3 Crevillent

The Crevillent pilot site will demonstrate the FastV2G, the EVSE with V2G capabilities being developed under WP8. More details will be found in D8.2 [6].

## 3.2.2 EV Wrappers

The EV wrappers are the modules within WiseEVP that will implement the communication with the different EV monitoring systems available at the pilot sites. The main purpose is to take advantage of the existing monitoring systems and push the information retrieved by those to the IOP in order to be used by the other WiseEVP modules. Specification of this module is documented in D4.2, and also included here for completeness of the deliverable.



**Figure 7 – EV Wrapper and its main interactions**

As depicted in Figure 7, EV Wrappers main objective is to take advantage of the existing EV monitoring systems and integrate the information to the WiseGRID ecosystem. Currently, the pilot sites of Terni and Ghent have EV Monitoring Systems in place. For the Crevillent pilot site, a simple EV Monitoring System for the fleet will be developed within the project, as described below.

*NOTE: the proposed approach, which will be followed during the course of the project, considers that information from field assets is delivered to the WiseGRID IOP platform, which properly distributes it to the corresponding actors and applications interested in processing it. This may arise some issues when the data published by the field assets is considered private or sensible, as it may be the case with the information about the EVs. Within this context, some remarks need to be done. First of all, the Message Broker module inside the WiseGRID IOP implements the necessary authentication and authorization mechanisms in order to provide access to each actor only to the required data flows. Second of all, the technologies and protocols considered for the WiseGRID IOP are very versatile, making it possible to consider different architectural approaches. For instance, a single Wrapper could send data to different Message Brokers under control of different actors only with minor changes on the developed modules.*

### 3.2.2.1 Communication specifications

For the communication between the EV Wrappers and the IOP, the MQTT protocol has been selected. The motivation is that EVs basically producing information that will be consumed by any application requiring it, and the publish/subscribe mechanism of MQTT clearly applies to this kind of scenario.

The different EV Wrappers will produce messages under topics that ultimately identify each one of the monitored EVs.

**Base topic: MQTT/EV/[UID of the EV]**

Base topic uniquely identifies the electric vehicle in the whole system.

ID of the vehicle will follow this pattern in order to be unique: “[ORGANISATION ID]-[ID WITHIN ORGANISATION]”. E.g. “PARTAGO-00001”.

E.g. MQTT/EV/EV00001

**Subtopic for EV characteristics: characteristics**

EV wrappers publish the characteristics of the monitored EVs upon connection to the MQTT broker. This

mechanism facilitates the automatic registration of new vehicles into the platform, which will happen automatically upon reception of the first message received from a vehicle.

This publication may be flagged as “retained” (i.e. broker will send the characteristics to any new subscriber automatically)

QoS-Level: 1







E.g. MQTT/EV/EV00001/characteristics

Exemplary payload:

```
{
  "id" : "EV00001",
  "name" : "Electric Vehicle X", //Free text description
  "connector" : {
    "type" : "chademo",
    "power" : "22" //kW
  },
  "capacity" : 41, //kWh
}
```

Accepted values for connector types are:

**Table 4 – Connection types**

Connector	Shape
chademo	
combo	
tesla	
mennekes	
j1772	
ccs	

Subtopic for EV status: **status**

EV wrappers periodically publish their status under this topic.

This publication may be flagged as “retained” (i.e. broker will send the last status to any new subscriber automatically)

The EV wrapper must configure a *MQTT LWT message* updating the status to “nocomm” when the EV wrapper disconnects from the MQTT broker.

QoS-Level: 1

E.g. MQTT/EV/EV00001/status

Exemplary payload:

```
{
  "id" : "EV00001",
  "timestamp" : "2018-01-01T00:00:00.000Z", //ISO8601
  "status" : "charging", //disconnected, connected, charging, nocomm
  "soc" : 81.1, //%
  "autonomy" : 100, //autonomy range in km
  "coordinates" : [0.9, 47.0], //longitude, latitude
  "travelledDistance" : 10000 //accumulated travelled distance
}
```

### 3.2.2.2 Integration of EV monitoring systems per pilot site

#### 3.2.2.2.1 Terni

Emotion will deploy three electric vehicles (EVs) and three electric vehicle supply equipments (EVSEs) in Terni pilot site for demonstrating WiseGRID integrated ecosystem.

Regarding EV monitoring, Emotion will use an OBD device to retrieve data from the EV; OBD is a IoT component that utilize a TCP/IP communication to a TCP/IP server. The network connectivity of the OBD device is via data SIM and the server is a python software, which queries the EV each 5 seconds.

From the EV will be retrieved the following data: Battery State-of-Charge (%), residual Autonomy (Km), minutes to Full Charge (m), Geolocalization, Doors Car State (Open/Close), Engine Car State (On/Off).

The EV data will be pushed to the WiseEVP through EV wrappers using MQTT protocol.

#### 3.2.2.2.2 Ghent

Partago operates twenty-three cars in Ghent (Belgium) and two in Brasschaat (Belgium). Of these 25 cars, 24 are Renault ZOE's and 1 is a Nissan eNV200. Eight Renault ZOE's have a battery capacity of 22kWh and 17 have a battery capacity of 41kWh. Three Renault Zoé's and one Nissan eNV200 are ordered and expected to arrive at the end of Q3 2018. The Renault Zoé with Q engines (12 in the fleet) have maximum charge power of 43 kW AC. Renault Zoé with R engines (13 in the fleet) have a maximum charge power of 22 kW AC. The Renault Zoé has one Mennekes type 2 charging connector. The Nissan eNV200 has two charging connectors, a Mennekes type (J1772) connector with maximum power of 7.8kW and a Chademo connector with maximum power of 50kW.

To retrieve status information from the cars such as position, state of charge (SOC) and so on, Partago installs a device in each car (Cloudboxx by INVERS gmbh [7]). This device determines the geographical position and collects basic car status information retrieved via a CAN bus connection. The Cloudboxx has a continuous connection to the INVERS servers over 3g mobile data connection. When 3g is not available in the area or signal quality is bad, there is fallback mechanism that uses sms.

On particular events, like change of ignition status (engine switched on/off) or charging status (charging on/off), the Cloudboxx automatically pushes an update of the car status to the Partago server. Actively requesting an update of the status from the server is possible when needed, for example during charging to

update the SOC.

On each status update, as well received as requested, we store the latest status in our database.

The WiseEVP EV Wrapper that was implemented by Partago runs as a separate service on the Partago server. The WiseEVP wrapper module is thus a module that needs to be installed on the partners' server. The EV wrapper connects to the Partago database and listens to updates. When a new car object is added to the database, the WiseEVP EV Wrapper publishes a 'characteristics' message. It is read by WiseEVP EV RT monitor. When the status of a car changes, it will publish a 'status' message. WiseEVP will publish the 'public' message to the IOP, with limited not privacy sensitive data for use by other modules in WiseGRID.

### 3.2.2.2.3 Crevillent

Currently, Crevillent is missing a monitoring system for their fleet of vehicles, composed by 3 Renault Zoe and Kangoo EVs. Within the WP9, an EV wrapper for monitoring those vehicles will be developed. The system will consist in the following:

- An OBD2 reader with Bluetooth capabilities will be installed in each of the vehicles



Figure 8 – OBD2 reader [8]

- The EV Wrapper software will consist in an Android app which will
  - Connect via Bluetooth with the OBD2 reader in order to trigger queries to the ECU in order to obtain the necessary parameters (SoC, travelled distance, battery capacity...)
  - Publish the information in real-time to the WiseGRID IOP using the agreed communication mechanisms and common data model for EVs (as described in previous section 3.2.2.1)

Drivers will therefore install the app on their phones to allow it to monitor and send the required data to WiseEVP.

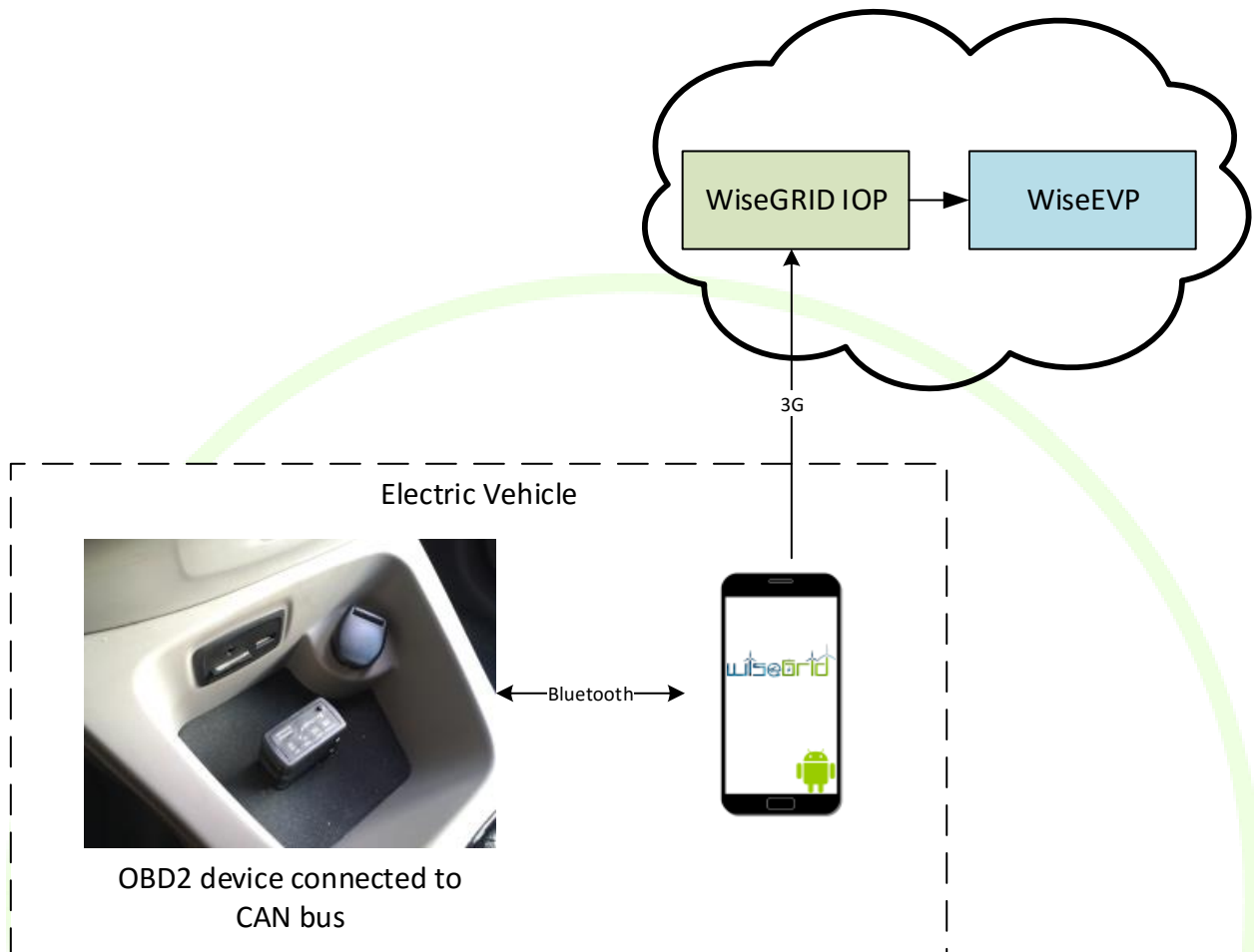


Figure 9 – Overview of the approach taken to monitor EV fleet at Crevillent pilot site

### 3.3 CHARGING SESSION OPTIMIZATION AND SCHEDULING

#### 3.3.1 Demand forecast

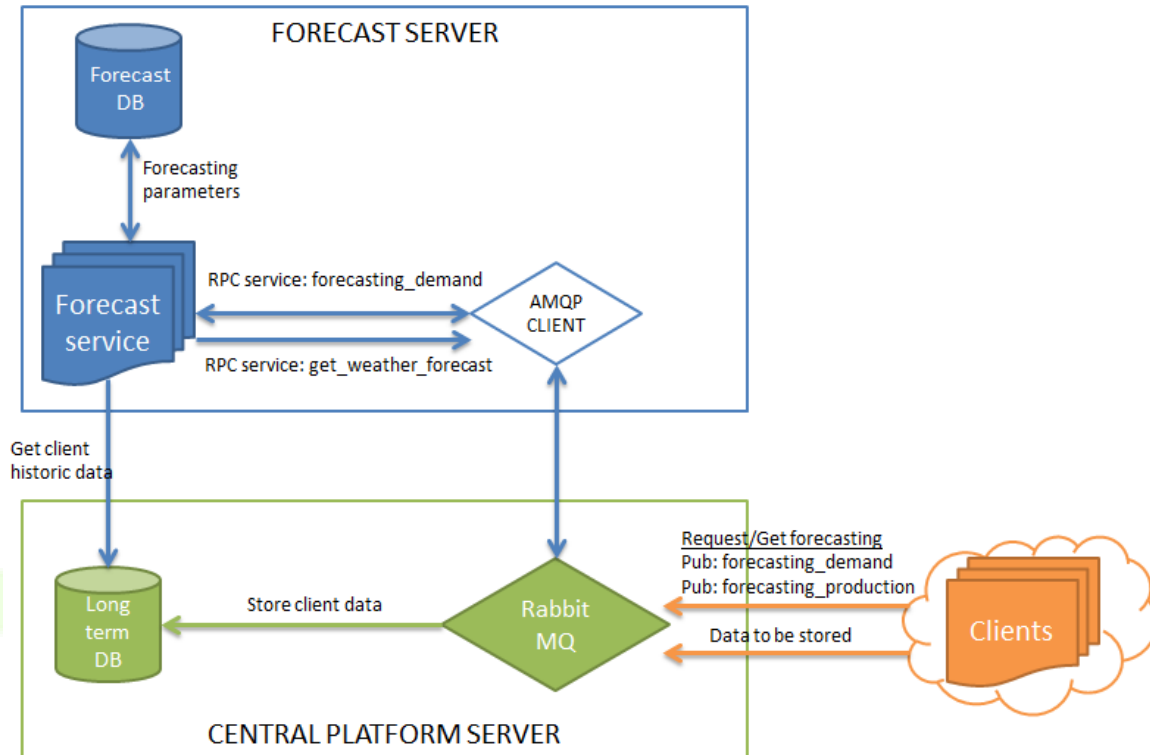
The daily loads of EV charging station are complicated and non-linear, and the traditional load forecasting algorithms such as linear time series and regression analysis hardly have the ability to simulate complex power load. Instead of the traditional load forecasting algorithms, the modern load forecasting algorithms are able to self-learn and perform non-linear modelling and adaptation

The EV demand forecast provider is a wrapper of the forecast algorithms developed by ITE for an easy integration with the WiseEVP tool.

The forecasting module developed in the WiseGRID project could be seen as a function that forecasts the next values taking into consideration historical data, the future booked charging session and the charging session in progress. Besides, further data such as a calendar (with working days, holidays and weekends), and maybe exogenous variables (weather) could be also considered.

Figure 10 contains the structure defined to integrate EV demand forecast service into the WiseEVP developments. The forecast server will communicate with clients through the central platform server, specifically, by means of the RabbitMQ services in the central control platform. Thus, the forecast server will include an AMQP client with two main objectives: route client request to the RPC forecasting function and call for the RPC functions which obtain charging sessions as well as other estimated variables affecting

directly EV load demand such as weather.



**Figure 10 – Forecast server structure integration schema**

Based on this architecture, the implemented developments will support the next general procedure.

1. The WiseEVP modules (called client or tenant) can request the forecasting profile calling a Remote Procedure Call function with some required parameters (more details in following sections).
2. The forecast service will receive the forecasting request, by means the AMQP client, and will perform the forecasting based on historical data, booked and in process charging session (long term data base), as well as additional data if proceed.
3. Once the forecasting service has completed the forecast request, the forecast service will return as a response the results to that specific client.

The forecast database is in charge of storing every model and parameter directly needed for the training of the algorithm model and for the forecast. The Long Term database is needed for storing all the historical data about EVSE load demand and charging sessions to be taken into account for the forecasts. Furthermore in the Long Term database should be available information about calendars. The characteristics of the historical data will affect the forecast output requirements. As an example, the forecasting algorithm will be not able to obtain a forecast with fifteen minutes detail if the historical data does not have at least the same time granularity.

Concrete to the input arguments, for each forecast to be generated, this module will take into consideration next inputs from the client:

- The client identifier. One client identifier is associated to an aggregation thus the same client application can request a forecast under different client identifiers.
- The period of time between two consecutive forecasting values.



- The time frame horizon for the forecast output.

The algorithm has also a high number of internal parameters that will be automatically optimized to obtain the best prediction, such as: number of past days for training, number of past hours for prediction and variables related with internal supervised learning and optimization algorithms.

Concerning the forecasting algorithm integrated in the EV load forecasting service, it has been implemented under CSVMA (Cascaded Support-Vector Machine Algorithm) forecasting model premises that are based on SVM algorithms. The CSVMA forecasting model consists on some SVM (Support Vector Machine) in cascade with an optimal chose of input parameters. The iterative under algorithm uses a SVM to forecast the EV power load demand of the first hour ( $h+1$ ) this result is used as an input of the next SVM to forecast the power load demand for the next hour ( $h+2$ ) and the algorithm repeat this process until the last forecasting slot is achieved. Thus, it is required 24 SVM executed in cascade for doing a forecasting of 24 values [9].

The proper way to obtain a good forecasting result in a short term power forecasting is to use past days with similar power profiles. Therefore, if the historical information available in the long term data base is not enough the error of the forecasting result will increase. To reduce this error, in the first step of the CSVMA model, it is chosen the more influencer variables from the data base in an optimal way.

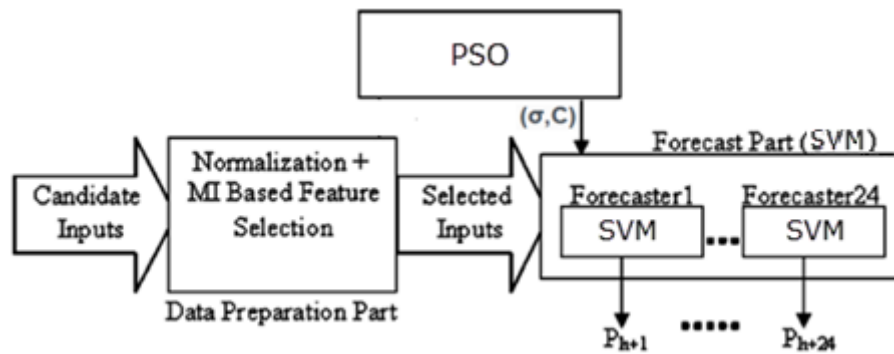


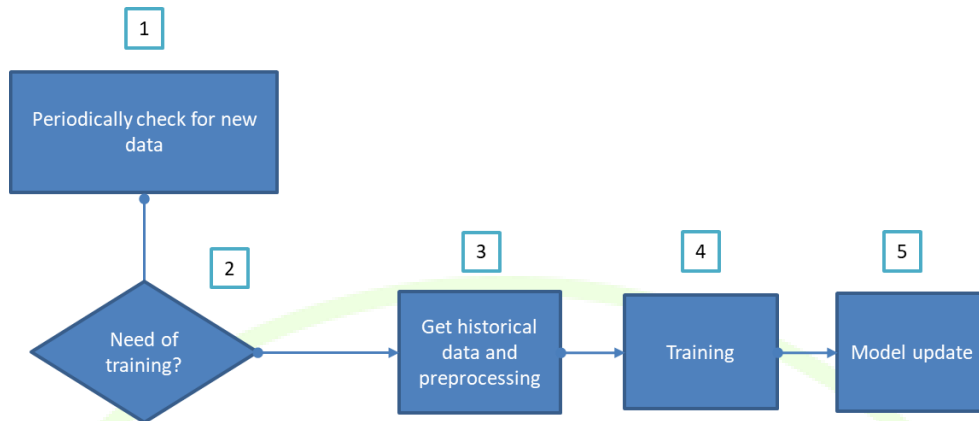
Figure 11 – CSVMA network structure

The forecasting algorithms will have two different workflows: one for training the models and the other one to attend the forecasting requests.

The workflow for training the forecasting model must run periodically and follows next steps:

1. This service checks for new historical data
2. Not always is needed to train forecasting models. Depending on the conditions and current performance of the forecasting service could be not necessary re-train the model.
3. Historical EV load demand and updated charging session data will be obtained through a Long Term DB call.
4. With all collected data the training of the model is performed.
5. Finally, old model is deleted and the new one remains to perform new EV load demand forecasts.

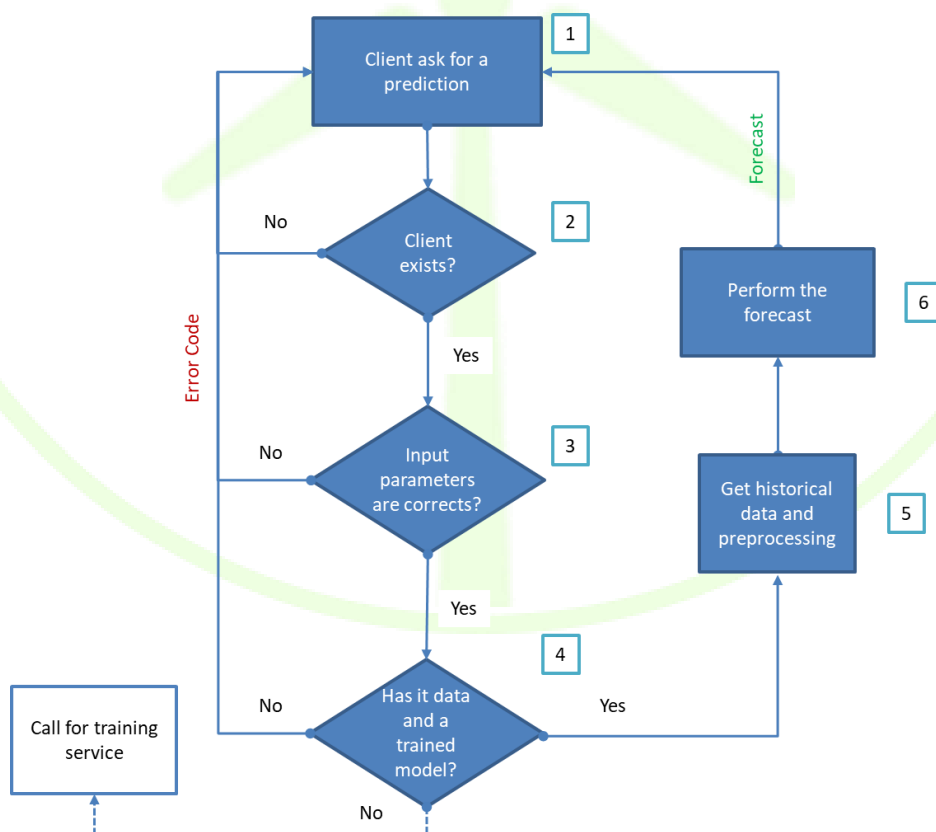




**Figure 12 – Training workflow schema**

Finally, the workflow to attend the EV load demand forecast requests follows next steps:

1. A client asks for a forecast by invoking a RPC function and adding some input parameters.
2. The identifier of that client is checked to know if exists.
3. The input parameters are checked to validate coherence.
4. It is checked if a model is trained and historical data is available.
5. Historical demand data will be obtained through a Long Term DB call.
6. Forecast is performed and returned to the client.



**Figure 13 – Forecast workflow schema**

### 3.3.1.1 Demand Forecast Interface

Provides load demand forecast for a client/tenant who ask for a prediction.

AMQP queue: “forecasting\_production”

#### Message properties

- reply\_to: name of the queue where response will be delivered
- correlation\_id: free text for query/response correlation (RPC pattern <https://www.rabbitmq.com/tutorials/tutorial-six-python.html>)
- Payload<sup>1</sup>:
  - client\_id: client identifier in the WiseGRID database.
  - Horizon: time frame the client wants to predict.
  - Period: Time period between forecast samples. 15 min / 60 min. Default 60 minutes.

#### Response properties

- errCode: Error code regarding possible exceptions.
- Forecast: Desired prediction formed by key value pair (“Timestamp” : Value)
  - Timestamp: UNIX time seconds (UTC).
  - Value: predicted value for the specified timestamp.
- Units: Value units.

**Table 5 – Demand forecast interface, error codes**

Error Code	Meaning
0	No error
1	Invalid client identifier
2	No enough historical data to perform forecasts
3	Untrained forecast model
4	Unfound forecast model
5	Invalid parameter horizon
6	Invalid parameter period
7	Unauthorized access to forecast model
255	Unhandled error

#### Message payload examples

```
{  "Client_id": 1,
   "Horizon": 1,
   "Period": 60,
```

<sup>1</sup> Note that for a production forecast, a weather forecast for a specific location is needed. Clients do not need to specify its location due that this information should be available into the database.

```
}
```

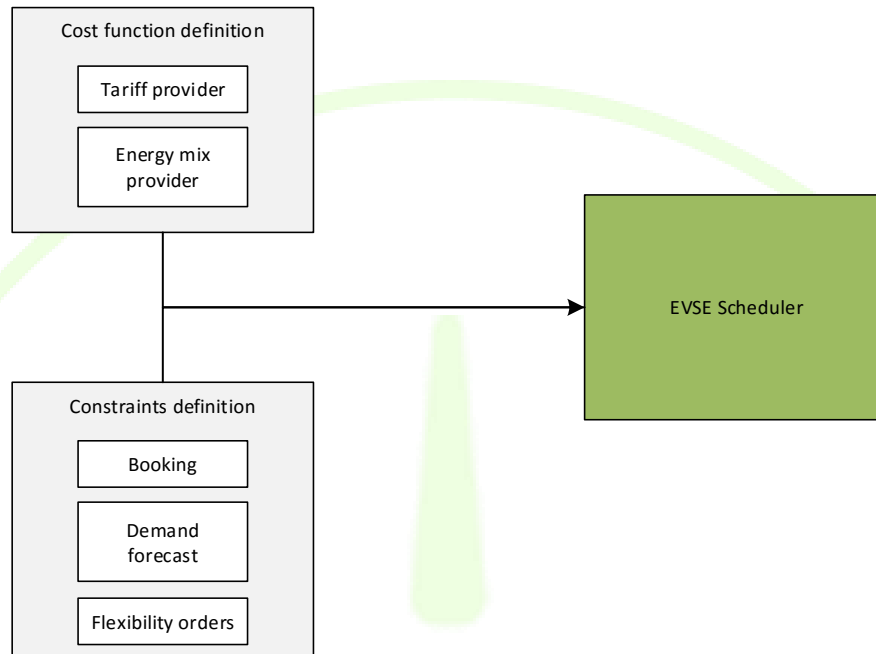
#### Response example:

```
{
  "errCode":0,
  "forecast":
  {
    "1507154400":22.544,
    "1507158000":21.438,
    "1507161600":12.242,
    "1507165200":12.116,
    "1507168800":10.985,
    "1507172400":12.235,
    "1507176000":9.152,
    "1507179600":58.837,
    "1507183200":65.365,
    "1507186800":22.05,
    "1507190400":38.03,
    "1507194000":8.861,
    "1507197600":1.071,
    "1507201200":15.919,
    "1507204800":20.187,
    "1507208400":16.721,
    "1507212000":9.775,
    "1507215600":2.027,
    "1507219200":4.288,
    "1507222800":3.249,
    "1507226400":6.186,
    "1507230000":2.068,
    "1507233600":3.909,
    "1507237200":2.478
  },
  "units":"kW"
}
```

### 3.3.2 Tariff provider

The tariff provider module is one of the horizontal modules developed within the project (as documented in

D4.2 “WiseGRID interoperable Integrated Process (WG IOP)”, with the purpose of encapsulating the task of analyzing the structure of the tariff and provide a simple interface to the applications to access the energy price curves defined by the tariff. Within WiseEVP, its main purpose is to provide the price of the energy to the optimizer, in order to define the cost function that enables economic optimization of the charging sessions.



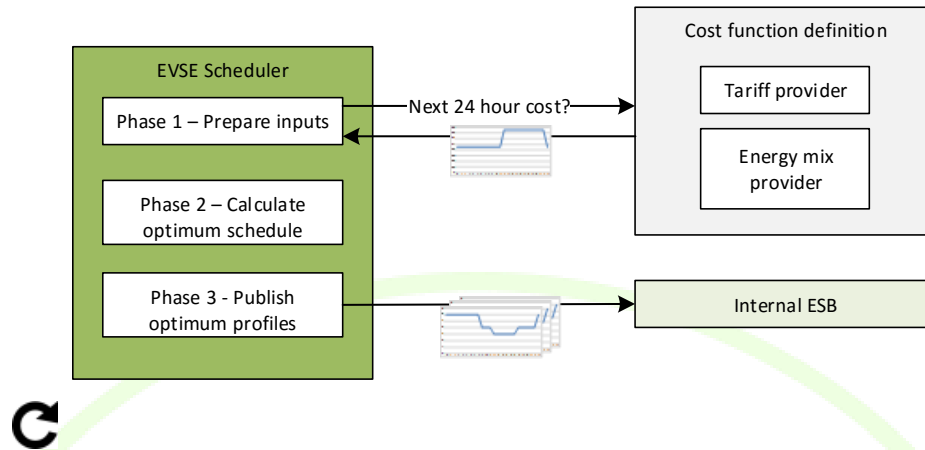
**Figure 14 – Overview of the inputs of the EVSE scheduler**

### 3.3.2.1 Workflow

As described in section 3.3.5, the EVSE scheduler processes all information using fixed-length timeslots of 15 minutes. In order to accommodate changes in the constraints of the optimization, particularly when new bookings or flexibility orders need to be considered in the schedule, this algorithm is periodically triggered every 15 minutes.

The EVSE scheduler main purpose is to propose the charging profile of every individual EVSE controlled by WiseEVP that will minimise an objective cost. WiseEVP will consider two kind of impacts towards this optimization, which can be combined by using a combination of weighted costs: economic and environmental.

As a first step of the execution of the EVSE scheduler, during the preparation of all inputs required to calculate the schedule, a request to the Tariff Provider module will be set to obtain the curve of energy prices for the next 24 hours, accordingly to the tariff contracted by the EVSE manager.

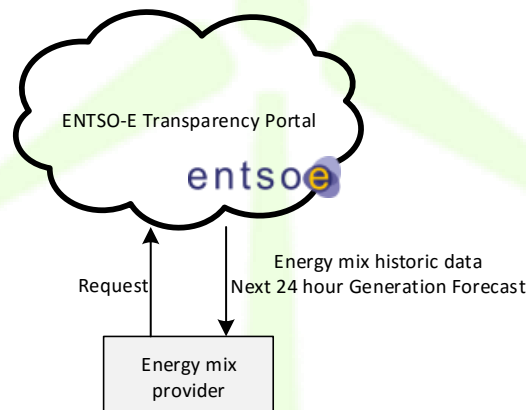


**Figure 15 – Overview of the workflow of the EVSE scheduler, and the role of the tariff provider**

In order to access to the 24 hour curve of energy prices, the *prices* method of the tariff provider module (as defined in D4.2 [10]) will be used.

### 3.3.3 Energy mix provider

The energy mix provider module is a module with the purpose of providing a simple interface to the application to access the detail of the generation sources of the energy of the country, as reported by the ENTSO-E transparency portal. Within WiseEVP, its main purpose is to provide information about the forecast of generation of renewable energy, one of the main inputs to assess the cost of the energy in terms of environmental impact to the EVSE Scheduler, in order to enable optimization for promoting usage of green energy in the charging sessions, as depicted in Figure 14.



**Figure 16 – Overview of the energy mix provider module**

#### 3.3.3.1 Workflow

The workflow follows exactly the same structure as defined in Section 3.3.2.1. As part of the preparation phase of the EVSE scheduler, the forecast of renewable production for the next 24 hours is retrieved and introduced in the cost function of the algorithm with the objective of promoting a shift of the consumption of energy to those periods when expected renewable production is higher.

### 3.3.4 Booking service

One of the services that an EVSE operator is interested in offering to its customer is the capability to reserve the Charging Stations in advance.

In order to cover this functionality, a booking service module has been included in the architecture of the WiseEVP. This module basically will implement a friendly user interface that will be accessible to the customers of the EVSE operator in the form of a website. In this website, registered customers will be able to visualize the current status of the EVSEs and the foreseen availability of these in the future. By selecting an EVSE and a period in time, WiseEVP will check the availability of the EVSE and will reserve it for this user during the requested period.

The management of reservations in an EVSE management system imposes new issues that are being considered in the design of the WiseEVP. First of all, it provides information that can be considered by the *demand forecast* module to fine-tune the estimation of the short-term future demand. In addition, the EVSE Scheduler needs to take upcoming reservations into account in the calculations of the optimum charging profiles for the whole set of EVSEs and the estimation of the flexibility that will be available for participation in the Ancillary Services Market. Finally, the practical implementation of the reservation mechanism also imposes some requirements to the business logic. The OCPP protocol used to manage the EVSEs contemplates the possibility to book the EVSE by locking it until the corresponding user gets to the Charging Station. This *short-term EVSE booking* mechanism is usually implemented as follows:

- Customer initiates the booking period only short in advance to the required period of time (e.g. 15 minutes in advance). Customer accesses the booking platform, checks current availability of EVSEs and requests the reservation of the selected one. Business logic may set a price to be paid by the customer for the rights to reserve an EVSE in advance
- Upon reservation of an EVSE, the control system immediately locks the EVSE during a short time of period (e.g. 20 minutes). During this period, the EVSE will only accept a new charging session from the user that reserved the Charging Station, rejecting any other user trying to make use of it
- During this locked period, two events may happen:
  - The customer that reserved the EVSE arrives and initiates a charging session
  - The period times out and the EVSE gets unlocked, accepting again charging session from any customer. In this case, the business logic of the reservation may impose a penalty to the customer that performed the reservation for blocking a station which was no longer used

*Short-term EVSE booking* mechanism has the advantage of simplicity in the implementation, and is fully supported by the OCPP protocol *Reservation profile* and the EVSEs implementing this profile.

In addition, *Long-term EVSE booking* feature will also be implemented by WiseEVP. The core difference with the mechanism explained above is that, in this case, the reservation is performed for a period in the future that makes it inefficient to immediately lock the EVSE (e.g. more than 1 hour in the future). In this case, the implementation that will be performed takes the following considerations:

- Whenever time of reservation is close (e.g. less than 1 hour) and the EVSE is available, the same logic described for the *short-term EVSE booking* mechanism will be triggered
- Whenever a new charging session is to be initiated at an EVSE of the system, WiseEVP needs to extend the basic white/black list-based authorization mechanisms to consider the following points
  - If the EVSE has any reservation in the future, an estimation of the duration of the session being established needs to be calculated. This duration will depend on the requirements of the user (current and desired SoC and time) and the Charging Session Profiles calculated by the scheduler for this EVSE
  - If the estimation shows that the charging session will be extended over an existing reservation, the customer will be informed and asked to modify the requirements of the charging session or move to a different EVSE

- WiseEVP will track the estimation of the duration of the charging sessions, considering those in the availability to be presented to the customers when these perform the reservation

This second approach does not only present bigger challenges in the implementation, but also requires from additional information to the data defined by the OCPP1.6 protocol from the EVSEs to work properly. In particular, upon start of a new charging session, WiseEVP requires to know not only the user, but the requirements of the charging session – in terms of energy supply and time availability – in order to properly estimate the required times and manage the availability of the EVSEs properly. For this reason, the second approach will be likely only tested with the FastV2G station.

### 3.3.4.1 Booking data model

The booking service module will interact with the operational database to retrieve and write the required data. This data includes:

#### Availability (collection *evseavailability*)

This collection of the operational database contains one document per active charging session in the system, with information about the estimated duration of the sessions. The corresponding document gets deleted once the charging session finishes.

**Table 6 – Booking data model fields**

Field	Description
<b>Evse</b>	Id of the EVSE to which the reservation is referred
<b>Card</b>	Id of the Customer (card) that performs the charging session
<b>Start</b>	Timestamp when charging session started
<b>EstimatedEnd</b>	Timestamp of the estimated end time of the charging session

Exemplary document:

```
{
  "_id" : ObjectId("5aa957563d00a446f342e958"),
  "evse" : "1001",
  "card" : "0002542",
  "start" : ISODate("2018-03-15T12:00:00.000Z"),
  "estimatedEnd" : ISODate("2018-03-15T18:00:00.000Z")
}
```

#### Reservations (collection *evsebooking*)

This collection will contain one document per reservation, including the following information:

**Table 7 – Booking data model fields**

Field	Description
<b>Evse</b>	Id of the EVSE to which the reservation is referred
<b>Card</b>	Id of the Customer (card) that performs the reservation
<b>Start</b>	Timestamp of the start time of the reservation
<b>End</b>	Timestamp of the end time of the reservation

Exemplary document:

```
{
  "_id" : ObjectId("5a02b8a23d00a446f338a627"),
  "evse" : "1001",
  "card" : "0002542",
  "start" : ISODate("2018-03-15T12:00:00.000Z"),
  "end" : ISODate("2018-03-15T18:00:00.000Z")
}
```

### 3.3.5 EVSE scheduler

A proper management and control of EVs at fleet level by means of an EVSE scheduler can provide ancillary services, contributing to stabilizing the network and integrating DER and balancing intermittent renewable energy sources (RES). Such results can be achieved by combining management of fleet charging processes.

#### 3.3.5.1 Objective

The main objective of the EVSE scheduler presented here is to find a proper trade-off between two different needs:

- To minimize the cost – either measured in economic or environmental impact terms – for users, by exploiting the time varying nature of the energy tariffs in order to provide the charging service at a competitive price, and the composition of the energy mix;
- To perform the regulation of the aggregated power, reacting to grid and RES integration support requests of each regulation area.

These two objectives would define an aggregated target function that should be minimized in order to achieve an optimal solution.

As a result an optimized load charging profile will be obtained for each EVSE in the regulation area.

#### 3.3.5.2 Problem description

In the following section it is described generically the discrete-time open loop optimal control problem that the EVSE optimal scheduler solves at each iteration time.

The inputs to the scheduler should include, among others, the following:

- Information of bookings and ongoing charging sessions.
- EV parameters and driver's charging preferences
- Cost signals (Price signals – ToU tariff – and energy mix)
- EV Demand forecasts for the day
- Flexibility orders established according to grid operator and RESCOs/aggregators requests

The output will be an optimized schedule of charging, that must be split among the different chargers, following rules that must be defined. The optimization will execute periodically, and could even be triggered by events, such as flexibility request. This period of execution will be decided in an initial task to define the system's specifications.

Each EV is characterized by three fundamental variables:

- current state (charge level);
- final desired charge level (set by the driver upon arrival);



- departure time (also set by the driver).

Hence, assuming that no other EV will join the fleet, by solving an open loop optimal control problem it is possible to find, for each EV, an optimal charging strategy that satisfies the driver's preferences (of course, only if they can be satisfied) while minimizing the aggregated cost function.

The resulting solution is intrinsically open-loop, every time an event changes the initial data set (arrival of new EVs, change of the electricity tariff, flexibility requests, etc.) the solution must be recalculated. Thus, in order to maintain the optimal solution along the entire scheduling period, the optimization process must be performed every time a "relevant" event happens or every time sample with updated information according to a MPC framework to close the loop. So, after each iteration, the new calculated control sequence replaces the portion of the previous control that has not been actuated yet.

### 3.3.5.3 Assumptions

The mathematical problem discussed in the following section has been achieved based on the following assumptions:

- The power withdrawal/injection from the grid into the EV and vice-versa (V2G) is a semi-continuous variable. It means that beyond the standby mode (no power flow), the power absorbed/injected by the EV is within a range that can differ from EV to EV, and from EVSE to EVSE, being further limited by EV and EVSE manufacturers depending on proprietary additional requirements.
- We consider a single electrical tariff for all the EVs.
- The cost of the energy absorbed by the EVs and the price of V2G energy are equal. However, the present strategy allows taking into account different prices depending on the direction of the power flow.
- Input and output battery efficiency coefficients are assumed to be coincident for each EV. These assumptions are made in order to obtain a simple, yet meaningful problem formulation.

### 3.3.5.4 Mathematical model

#### 3.3.5.4.1 Indices and Sets

$M$	Set of EVs currently connected to the charging stations
$M_k$	Subset of $M$ denoting the set of EVs connected to the charging stations during the $k$ -th time
$I$	First time interval of problem definition
$E_m$	Last allowed time interval for charging operations on the $m$ -th EV
$E$	Last time interval of problem definition
$\Delta P_m$	Charging power of the $m$ -th EV
$T$	Discretization time step
$C_m$	Electricity tariff of the $m$ -th EV
$U_m$	Control variable related to the charging operations on the $m$ -th EV
$x_m$	State of the battery (charge level) of the $m$ -th EV
$x_m^{max}$	Capacity of the battery of the $m$ -th EV
$x_m^{ref}$	Desired final charge level of the $m$ -th EV
$\xi_m$	$m$ -th EV's battery performance coefficient

$C_m^0$	Estimated cost for charging operation on the m-th EV
$P_{ref}$	Aggregated power reference
$P$	Aggregated power
$P^*$	Maximum power available for the EV fleet
$P_{ref}$	Aggregated power reference
$\mu$	Trade-off parameter
$\Lambda$	Weights matrix of tracking error along the control horizon

### 3.3.5.5 Mathematical formulation

In each iteration, the open loop optimal control problem can be modelled as the following mathematical optimization problem [11]:

Given a set  $M$  of EVs at a given time  $I$  associated with user preferences  $\{X_m^{ref}, E_m\}$ , technical parameters  $\{\Delta P_m, X_m^{min}, X_m^{max}, \xi_m\}$ , state measures  $X_m^0$  and common market and grid data  $\{C_m, P_{ref}, P^*\}$ , solve with decision variables  $U_m$ :

$$\begin{aligned} \min_{U_m, m \in M} J \\ \text{subject to a set of constraints defined in the sequel.} \end{aligned}$$

#### 3.3.5.5.1 Target function

In order to take into account both the problem objectives at each time of computation  $I$  we define a multi-objective target function as

$$J = J_{cost} + \mu J_{reg} \quad (1.1)$$

where  $\mu$  is a proper weight. The former term represents the cumulative cost that the EV fleet have to sustain for charging. If we consider a set  $M$  of EVs under charging in the same area, each of them characterized by a departure time  $E_m T$ , a maximum power flow  $\Delta P_m$ , a control  $U_m[\cdot]$  and subject to the electricity tariff  $C[\cdot]$ , the cost is given by:

$$J_{cost} = \sum_{m \in M} \sum_{k=I}^{E_m-1} \Delta P_m T C[k] U_m[k] \quad (1.2)$$

The control variable  $U_m[k]$  represents the controlled power flow normalized with respect to  $\Delta P_m$ , and then  $\Delta P_m U_m[k]$  is the power actually flowing in the cable connecting the EVSE and the EV. The latter term appearing in (1.1) is introduced in order to minimize the error related to the tracking of the power reference according to the flexibility orders from DSO and RESCO/Aggregators. We denote such a reference by the vector  $P_{ref}$  and model this "regulation" term as

$$J_{reg} = \|\Lambda(P - P_{ref})\|_{\infty} \quad (1.3)$$

where  $\|\cdot\|_{\infty}$  is the  $l_{\infty}$  - norm defined in real space.

The vector  $P$  represents the controlled aggregated power, which is given at the generic time instant  $k$  by:

$$P[k] = P^s[k] + \sum_{m \in M_k} \Delta P_m U_m[k] \quad (1.4)$$

The set  $M_k$  is included in  $M$  and is defined as:

$$M_k = \{m \in M : I \leq k \leq E_m\} \quad (1.5)$$

This set represents the family of flexible EVs possibly involved in the charging/discharging operation at time  $k$ , while  $P^s[k]$  is the aggregated power consumption related to non-flexible EVs. The diagonal matrix  $\Lambda$  in (1.3) allows weighting differently the tracking error along the control horizon.

By properly choosing the weights  $\mu$  and  $\Lambda$  it is possible to tune the behaviour of the controller and to find a proper trade-off between the needs of minimizing the cost for drivers and the tracking error in a grid status and RES integration perspective.

#### 3.3.5.2 Prediction model

In MPC problems, the prediction model is used to evaluate the future state of the system, allowing optimization and constraints satisfaction over the entire defined control horizon. In this case, the prediction model is given by the dynamics of the EV batteries. In particular, the physical variables of interest are here the batteries levels of charge, assumed as state  $x$  of the control problem. We take the following first-order model:

$$\begin{cases} x_m[k+1] = x_m[k] + \Delta P_m T (U_m[k] - \xi_m |U_m[k]|) \\ x_m[I] = X_m^0 \end{cases} \quad (1.6)$$

where  $m \in M$ ,  $k \in [I, E_m]$ , and  $\xi_m$  is the charge/discharge efficiency.

#### 3.3.5.3 Control constraints

By control constraints we assure control variables to remain within tolerable ranges so that the system well behaves. The first set of control constraints is related to the nature of the control variables. For the generic  $m$ -th EV we have

$$U_m[k] \in [-1, -\alpha_m] \cup \{0\} \cup [\alpha_m, 1], \quad 0 < \alpha_m < 1 \quad (1.7)$$

The second set of control constraints aims to avoid that the aggregated power could exceed a given threshold sequence  $P^*[k]$ . The difference between the threshold and the reference can be seen as the maximum displacement which is allowed. It is straightforward to model such a set of constraints as

$$P^s[k] + \sum_{m \in M_k} \Delta P_m U_m[k] \leq P^*[k] \quad \forall k \in [I, E] \quad (1.8)$$

where the current time  $I$  and  $E$ , defined as:

$$E = \max_{m \in M} E_m \quad (1.9)$$

are the boundaries of the control horizon.

The last set of control constraints guarantees that the cost of the charging service for each single driver remains bounded iteration after iteration, and then it cannot grow unpredictably. If we denote by  $c_m^*$  the cost “budgeted” to the user upon the arrival at the EVSE (after the first iteration of the algorithm) and by  $c_m[I]$  the cost already counted for billing at the time  $I$ , we can bound the control action as follows:

$$c_m[I] + \sum_{k=I}^{E_m-1} \Delta P_m T C_m[k] U_m[k] \leq (1 + \varepsilon) c_m^* \quad \forall m \in M \quad (1.10)$$

where  $\varepsilon$  is a small tolerance parameter.

#### 3.3.5.5.4 State and termination constraints

State constraints are related to the capacity of the batteries and some related technical limitations. In principle, we could say that the level of charge must be non-negative and upper bounded by the battery capacity. In practice, for reasons related to efficiency and life cycle, the battery pack is never allowed to fully charge or deplete. Then it is straightforward to establish that

$$X_m^{min} \leq x_m[k+1] \leq X_m^{max} \quad \forall m \in M \quad \forall k \in [I, E_m - 2] \quad (1.11)$$

where  $X_m^{max}$  is the maximum allowed level of charge and  $X_m^{min}$  represents the allowed depth of discharge of the  $m$ -th EV.

Finally we consider a termination constraint for each EV, aimed at guaranteeing the desired state of charge  $X_m^{ref}$  at the end of the stop at the EVSE:

$$X_m^{ref} \leq x_m[E_m] \leq X_m^{max} \quad \forall m \in M \quad (1.12)$$

### 3.4 ANCILLARY SERVICES

#### 3.4.1 Flexibility forecast

The objective of the EV Flexibility Forecaster is the development of a forecaster to produce aggregated flexibility offers according to EV energy availability that may be used by the Wise EVP manager for trading in a

ancillary services market. One issue that must be taken into account is that, whenever a flexibility offer is accepted, an order will be sent back to the Wise EVP to provide the accepted flexibility, therefore the EVSE scheduler module will have to react as a consequence, dynamically modifying the charging schedule.

In the following section we describe generically the calculation of flexibility for each regulation area or for the whole system.

At every moment, an EVSE Operator has a certain amount of flexibility while still meeting the charging demands. This flexibility allows him/her to respond to the demand of the grid operator, for absorption of excess energy, or for lowering energy demand. The flexibility depends on:

- Number of sockets in use.
- Charging types being requested by the users.
- The capabilities of the EVSE.
- The capabilities of the connected cars.

WiseEVP calculates the flexibility of its EVSE network periodically (every number of hours) for each regulation area system and will cover different time frames (the following day, the rest of the day, the following hours, etc.) indicating the uncertainty on the predictions. The EVSE manager can consult the flexibility estimation to manage the charging costs. The WiseEVP can send the flexibility capabilities per regulation area to the WG IOP to be available for other WG IOP (and also to match these flexibility offers with flexibility request for grid agents).

Type 1 charging (charging on demand) does not offer flexibility because the user just wants his EV charged ASAP. The other types 2-3 offer different levels of flexibility as the user allows modulate the absorbed power and even also to inject electricity to the network. Cars with V2G capabilities can deliver energy back to the grid. In this case, an EV Fleet Manager, operating V2G-capable cars and compatible EVSE (FastV2G), can deliver energy back.

As a result flexibility capabilities with different time horizons (for the whole system and/or per regulation area) and a variable bandwidth of power modulation will be updated in the WG IOP.

This information should be sent to the Ancillary Services Market Hub. In the case that any of these orders would match with flexibility requests sent by DSO and RESCO/Aggregators and were accepted, the order will be sent to the Wise EVP scheduler, which should modify the charging schedule, or even interrupt charging sessions in the toughest scenario.

The flexibility forecaster will provide different flexibility offers, in power level and time frames, based on the charging schedule and the characteristics of the charging stations in the network. This component will use as inputs the current schedule, the demand forecasts, and any other information provided regarding the characteristics of the chargers: charging mode, power level, availability to interrupt the load or to modulate it, etc.

The flexibility estimation will be calculated based on the following assumptions:

- The charging session bookings should be available and corresponding EV information should be linked (and therefore dependencies to make this happen should be fulfilled).
- The data collection from EVSE should be performed periodically to have nearly-real-time data and to know the charging sessions that the EVSEs are performing.
- The EVSE network should have been configured.

### 3.4.2 Ancillary Services Market Hub

This interface defines the interaction between the DSO (WG Cockpit) and the rest of the applications which are in position of providing ancillary services (congestion and voltage support, related to explicit demand response campaigns as discussed in D10.2 [12]).

This interface is based on USEF (<https://www.usef.energy/>). The core concepts of that framework will be used to implement an implementation of ancillary services to the DSO following a market-based approach including several aggregators in the loop.

#### 3.4.2.1 Workflow for multi-actor interaction

The workflow that will be implemented in the project to demonstrate how the applications (including WiseEVP) can support the DSO operation by offering their flexibility is defined in detail in D4.2 [10].

In essence, the WiseGRID project will take advantage of the WiseGRID IOP platform as a communication mechanism between the different actors involved, with different exchanges/queues allowing seamless and extensible interaction. This implementation is open to the addition of new actors capable of providing flexibility. The main steps of the workflow are:

1. Flexibility requests generation: upon forecast of congestion, the DSO will publish a request for flexibility to all the interested parties
2. Reception of flexibility offers: DSO will give some time to parties to process the request and post the corresponding offers, detailing the amount of flexibility they are able to provide and the price
3. Offer selection: once the period for the reception of offers is closed, the DSO will process all offers received in order to make the most suitable selection, notifying back to the different aggregators the decision result
4. Offer revocation: at any time, aggregators may revoke a posted offer if their situation changes and makes it impossible to fulfil the delivery of the flexibility

#### 3.4.2.2 Workflow within WiseEVP

Within WiseEVP, the Ancillary Services Market Hub will have the following duties:

- Will process incoming requests coming from the DSO
- Upon reception of a request, will cross-check the feasibility to deliver the required flexibility. This will be done by accessing the information calculated by the flexibility forecaster, which takes into account the results of the EVSE scheduler to calculate the extent to which the charging profiles can be changed while still fulfilling the requirements
- The corresponding *offer* will be composed and send back to the DSO. Price calculation of the offered flexibility can respond to different business decisions. One of the basic methods to calculate the retribution for the delivery of the flexibility will be based on using the price of the energy - as paid by the EVSE manager to the retailer – and a surplus

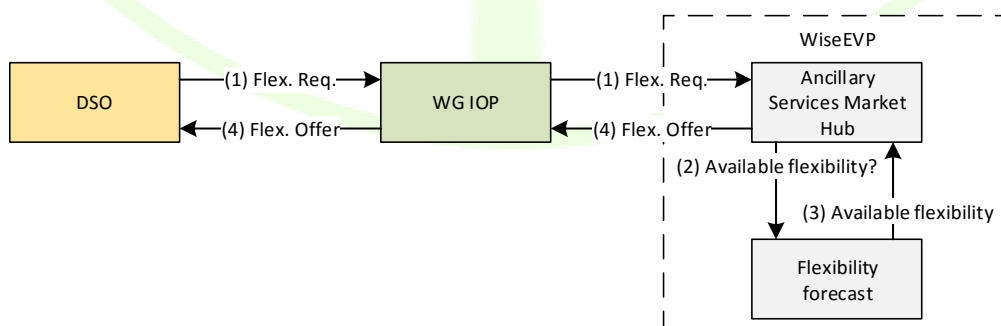


Figure 17 – Overview of the messages exchanged for the composition of a flexibility offer

- Whenever an offer is accepted by the DSO (confirmed by the reception of an *Order* message), information is fed to the EVSE Optimal Scheduler to be taken into account in the generation of the charging profiles for the EVSEs.

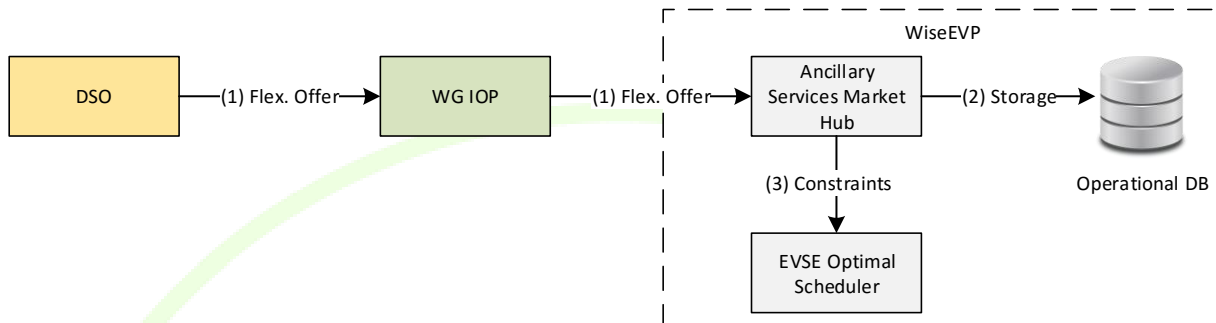


Figure 18 – Overview of the messages exchanged for ordering and delivering flexibility

### 3.5 USER INTERFACE DESIGN

The WiseEVP user interface is designed to give a useful insight to the fleet manager and EVSE manager on the different KPIs that are processed by the platform, and which will provide a clear overview of the status, history and trends of the different aspects related to the energy demand of the monitored elements.

WiseEVP has two main target users: fleet managers and EVSE managers. Depending on the business of the organization, it may comply either with one of the roles, or with both roles at once. The design of the UI takes this fact into account and proposes different screen with focused data. Access to different sections of the UI will be managed by the profile of the operator (Fleet manager, EVSE manager, or both).

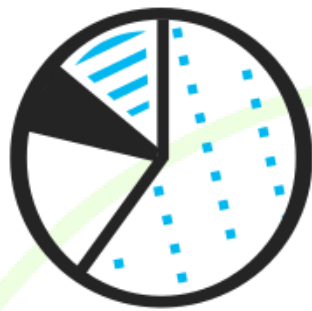
#### 3.5.1 Dashboard

Dashboard will be the first site displayed to the user of WiseEVP. It will contain summarized information of the current status of the EVSEs and EV fleet, as well as a selection of the most relevant KPIs:

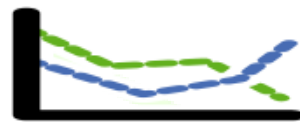
- Summary of source of the energy demand of the last 24 hours (self-consumption, or source according to the energy mix – renewable or non-renewable)
- Daily economic cost of the energy consumed the last 30 days
- Daily environmental impact of the energy consumed the last 30 days
- Percentage of the EVSEs and EVs that currently communicate with the WiseEVP

## Dashboard

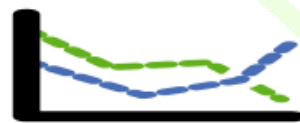
### Energy usage



Energy source over last 24 hours  
(self-consumed /  
external - renewable sources /  
external - exhaustible sources)



Daily cost over last 30 days (€ & CO2)



Daily demand over last 30 days (kWh)

### Field devices comm. status



EVs



EVSEs

Figure 19 – Dashboard

### 3.5.2 Map

The map will provide an overview of the monitored EVSEs and EVs. Different status will be displayed with different colours:

- EVSEs
  - Grey: unknown status
  - Green: EVSE is free
  - Red: all plugs on EVSE are occupied
  - Yellow: EVSE is occupied, but there are still free plugs
- EVs
  - Color scale red/yellow/green: State of Charge
  - Grey: unknown status

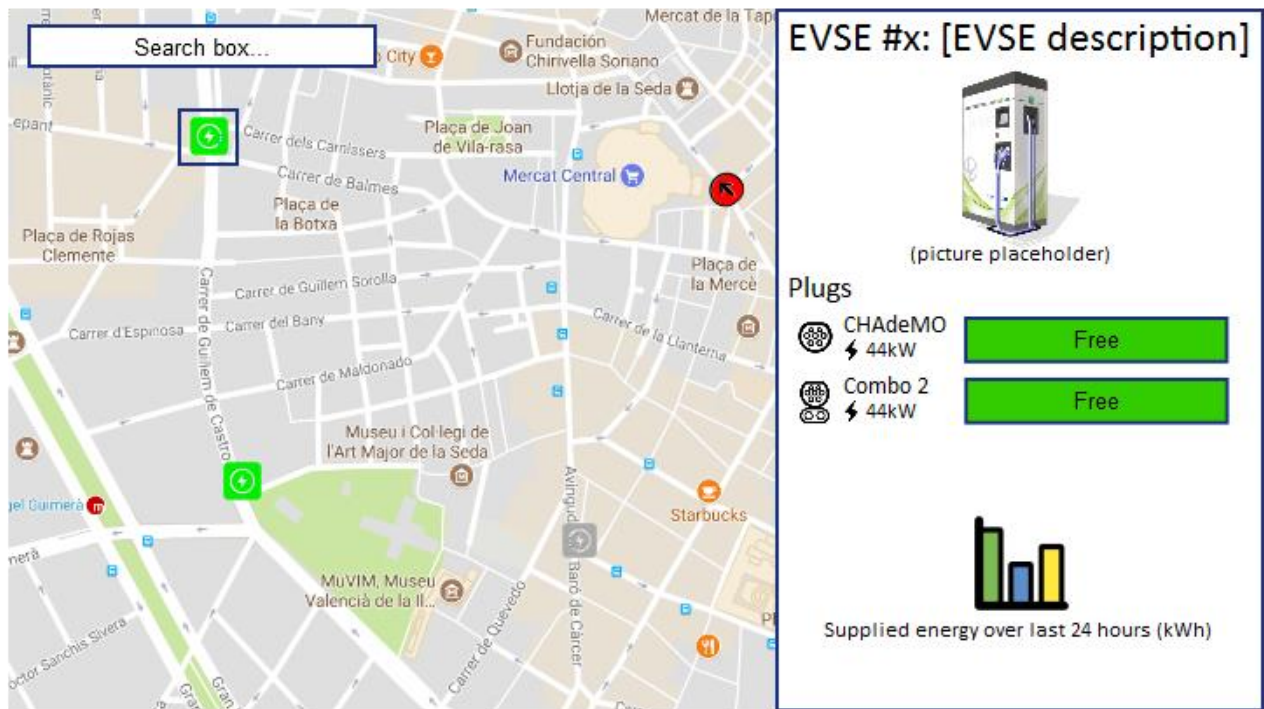




**Figure 20 – EV and EVSE Map**

By clicking in an EVSE, a summary view will be shown on the right-hand side. Summary includes static information, occupancy information and a summary of the supplied energy over the last 24 hours.

## Map



**Figure 21 – EVSE Map**

By clicking in an EV, a summary view is shown on the right-hand side. Summary includes static information, state of communication, Current SoC and current autonomy range.

## Map

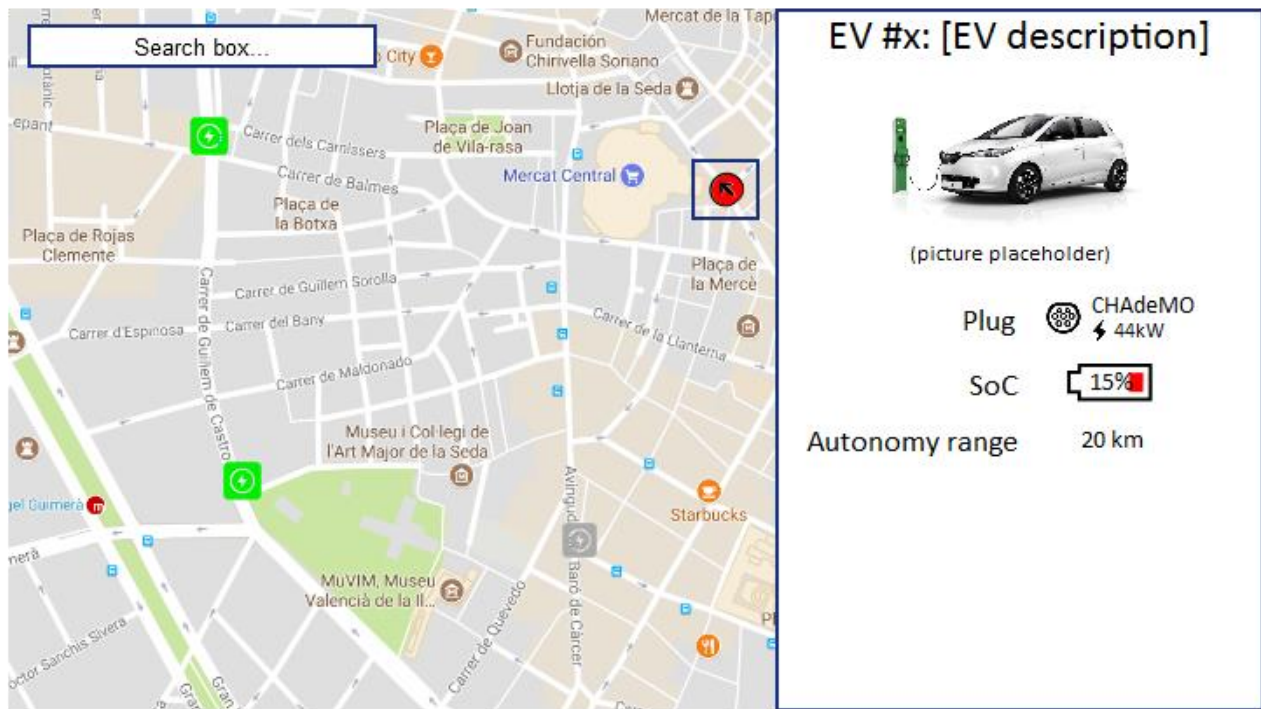


Figure 22 – EV Map

### 3.5.3 EVSE detail

This section provides details of the selected EVSE, including:

- Static information (picture, connectors, technical characteristics)
- Current occupancy status
- Graphs with supplied energy and currently applied profiles as calculated by the EVSE Optimal Scheduler (supply limits)
- Overview of the reservations over a calendar

## EVSE details

Search EVSE...

EVSE #x: [EVSE description]

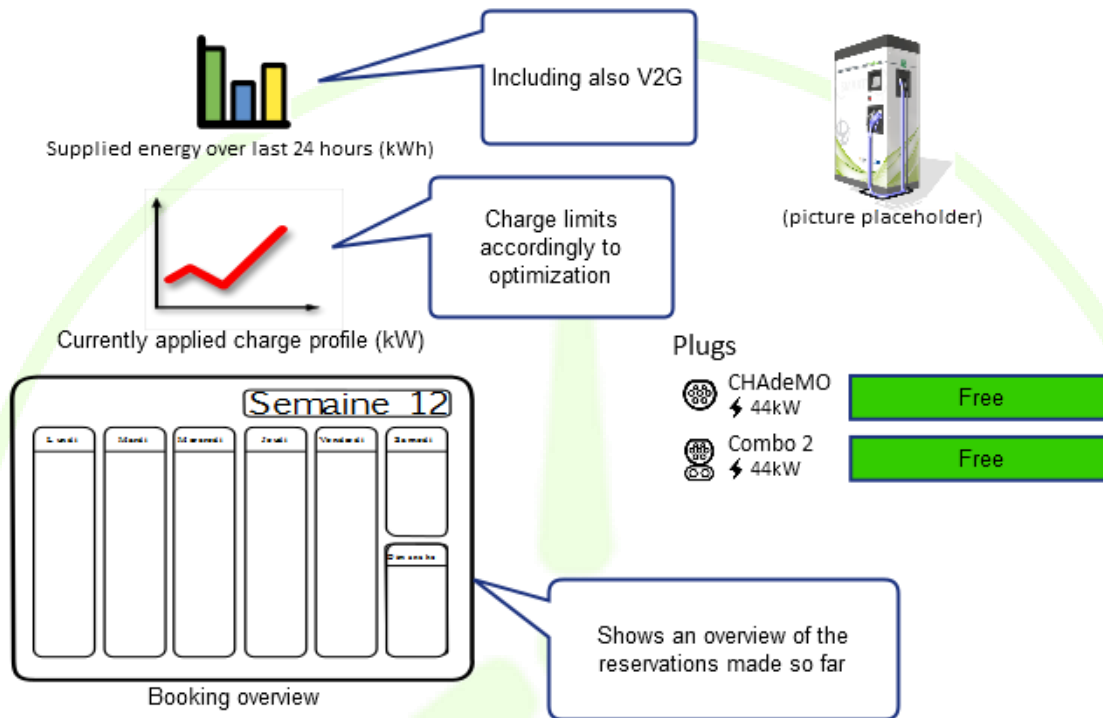


Figure 23 – EVSE details

### 3.5.4 EV detail

This section provides details of the selected EV, including the following information:

- Static information of the vehicle (plug, model, capacity)
- Current status of the vehicle (State of Charge, autonomy range, communication status)
- Driving efficiency indicator, representing energy spent per kilometre over the last 24 hours
- Graphs indicating usage parameters of the vehicle, such as evolution of travelled distance and State of Charge over time
- Requirements for the charging sessions of this vehicle: the purpose of this section is to allow the fleet manager to introduce the usual charge requirements for this vehicle, mainly the desired State of Charge within a schedule. This allows the EVSE optimal scheduler to account for the regular use of each of the vehicles of the fleet
- Minimum required State of Charge at any time: this parameter will be used by the EVSE Optimal Scheduler to ensure that, upon connection of the vehicle to a charging station under control of the WiseEVP, this vehicle gets charged as fast as possible to reach this minimum State of Charge. This parameter is particularly relevant when the fleet manager wants to ensure a minimum SoC on the vehicles of the fleet, accounting for instance for unforeseen trips. Flexibility will therefore be limited to the smart scheduling of the charge of the remaining SoC until 100%

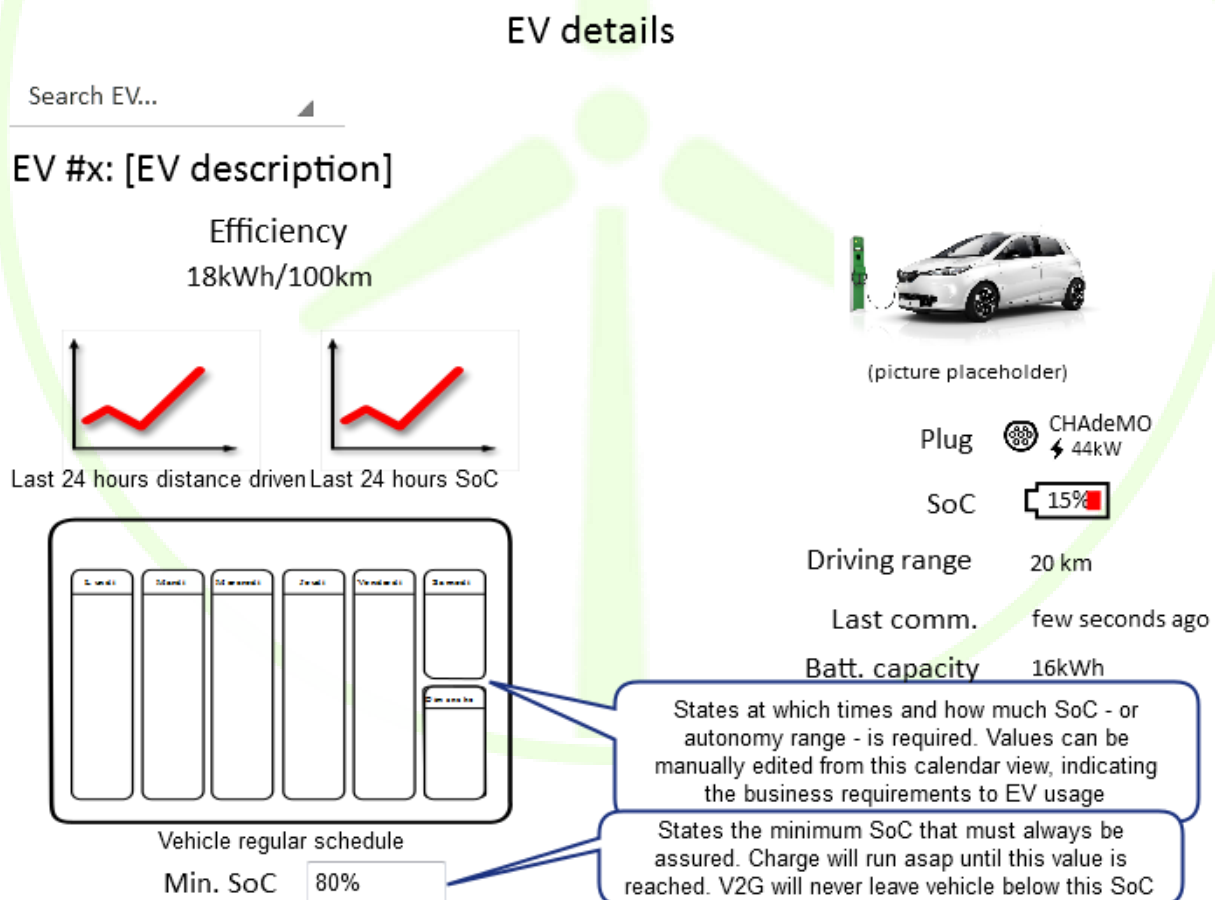
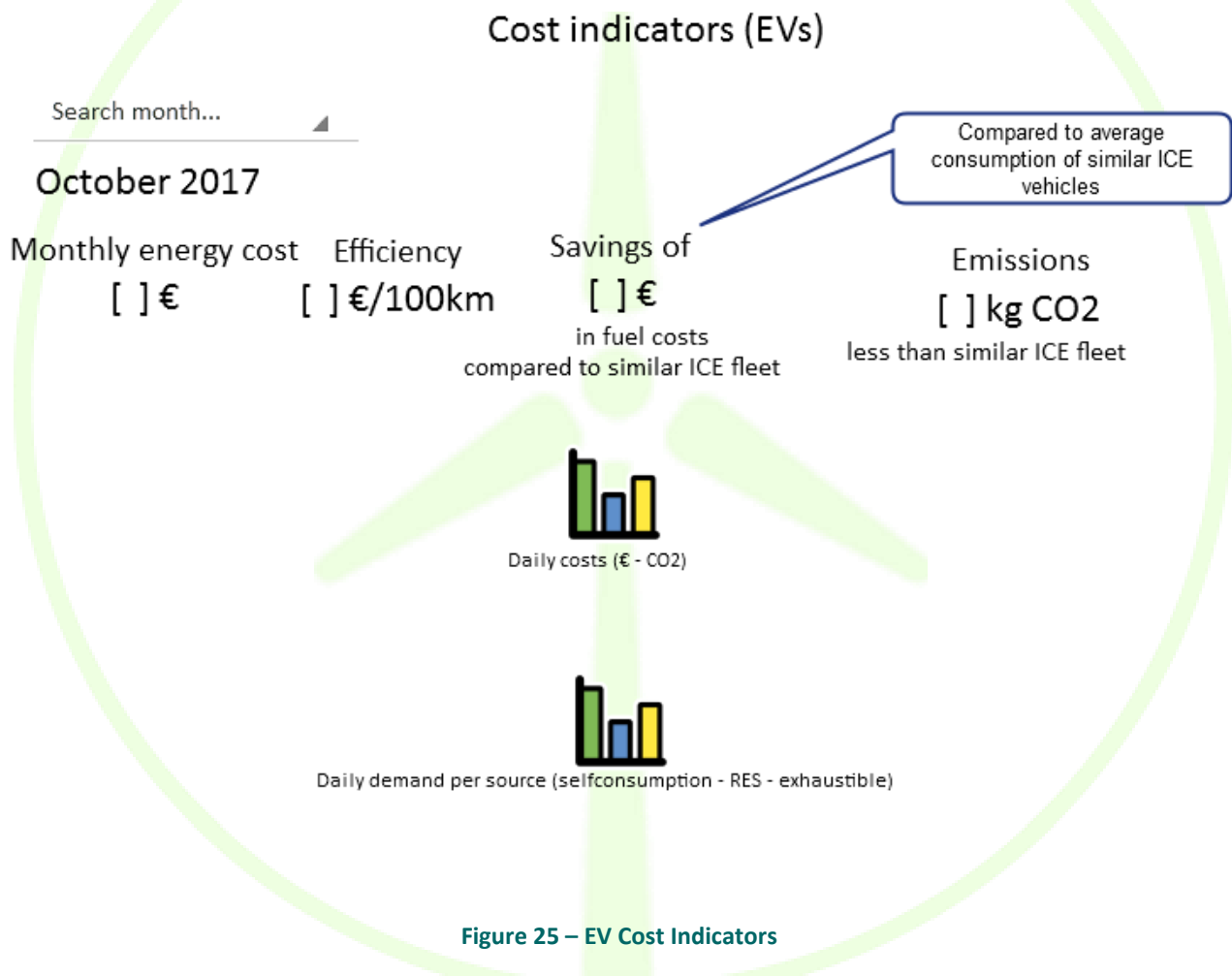


Figure 24 – EV details

### 3.5.5 Cost indicators for EVs

This section provides monthly cost indicators over the whole EV fleet. KPIs displayed in this section include:

- Monthly cost of the energy supplied to the fleet
- Overall efficiency of the faced cost (cost per km)
- Comparison of economic costs with those calculated for a similar ICE-based fleet. For this, a model of the consumption of equivalent ICE vehicles, as well as average fuel prices, will be used as baseline
- Comparison of environmental impact with a similar ICE-based fleet, also based on a model of equivalent vehicles as a baseline
- Chart with daily economic costs and environmental impact (CO2 emissions)
- Chart with daily demand, disaggregated accordingly to the source (self-consumed, RES, non-RES)



### 3.5.6 Cost indicators for EVSEs

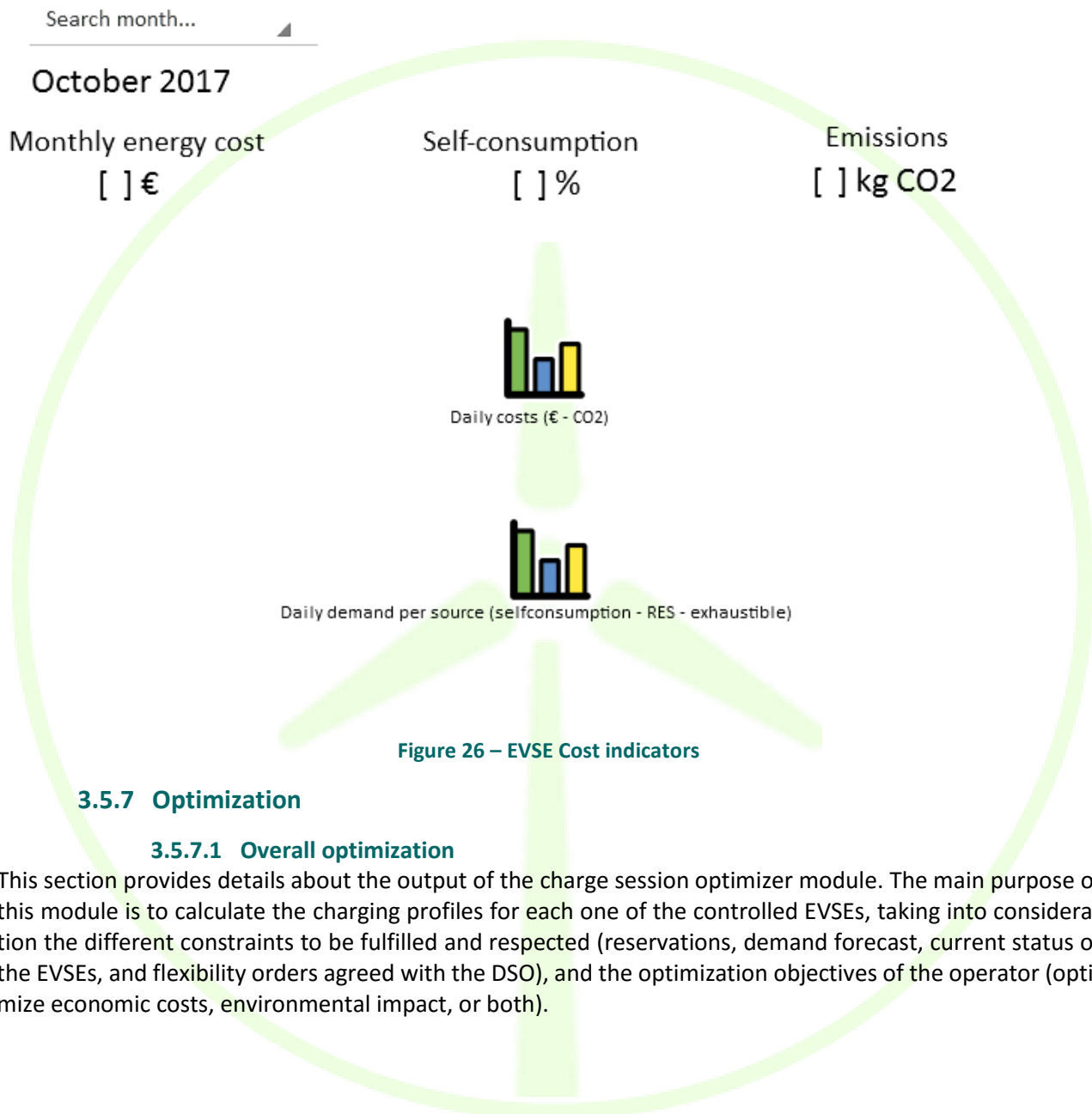
This section provides monthly cost indicators over the whole EVSE infrastructure. KPIs displayed in this section include:

- Monthly cost of the energy supplied by the EVSEs.
- Percentage of energy supplied that is produced by RES owned by the EVSE operator.
- Environmental impact (equivalent CO2 emissions) of the energy supplied by the EVSEs.



- Chart with daily economic costs and environmental impact (CO2 emissions).
- Chart with daily supply, disaggregated accordingly to the source (self-consumed, RES, non-RES).

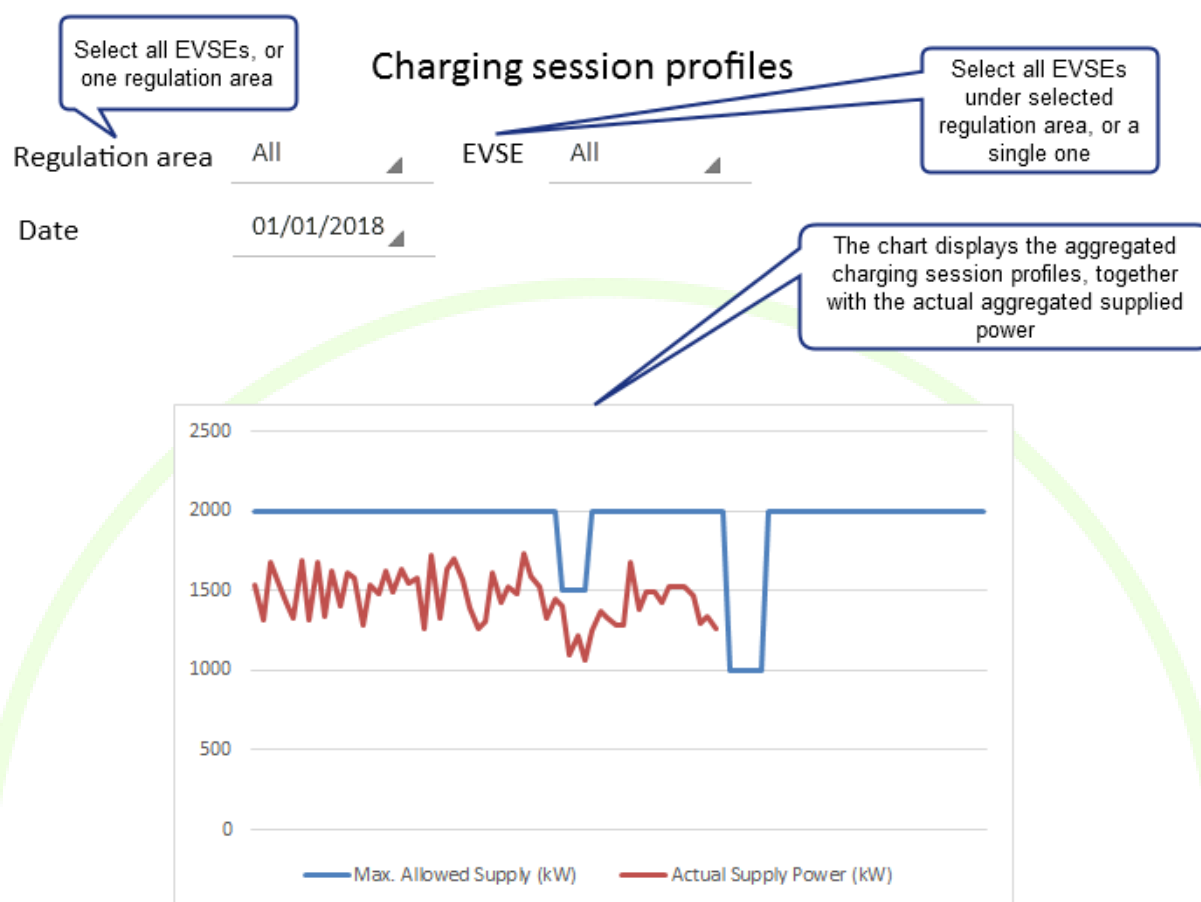
## Cost indicators (EVSE infrastructure)



### 3.5.7 Optimization

#### 3.5.7.1 Overall optimization

This section provides details about the output of the charge session optimizer module. The main purpose of this module is to calculate the charging profiles for each one of the controlled EVSEs, taking into consideration the different constraints to be fulfilled and respected (reservations, demand forecast, current status of the EVSEs, and flexibility orders agreed with the DSO), and the optimization objectives of the operator (optimize economic costs, environmental impact, or both).



**Figure 27 – Charging session profiles**

### 3.5.7.2 Individual charging sessions

For each active charging session, this section shows the requirements, the scheduled parameters, and a comparison of the evolution of the charging session with dumb charge, in terms of economic costs, environmental impact and expected delay.



## Optimization

EVSE EVSE #1 [...]

Ongoing charging session details

Vehicle	EV #1 [...]	Max. batt. capacity	16kWh
		Input power	3.6 kW
Initial charge	51%	Required charge	100%
Charging start time	01/01/2017 12:00	Required end time	01/01/2017 21:00

Optimized schedule

All information related to the scheduled session, including comparisons with dumb charge



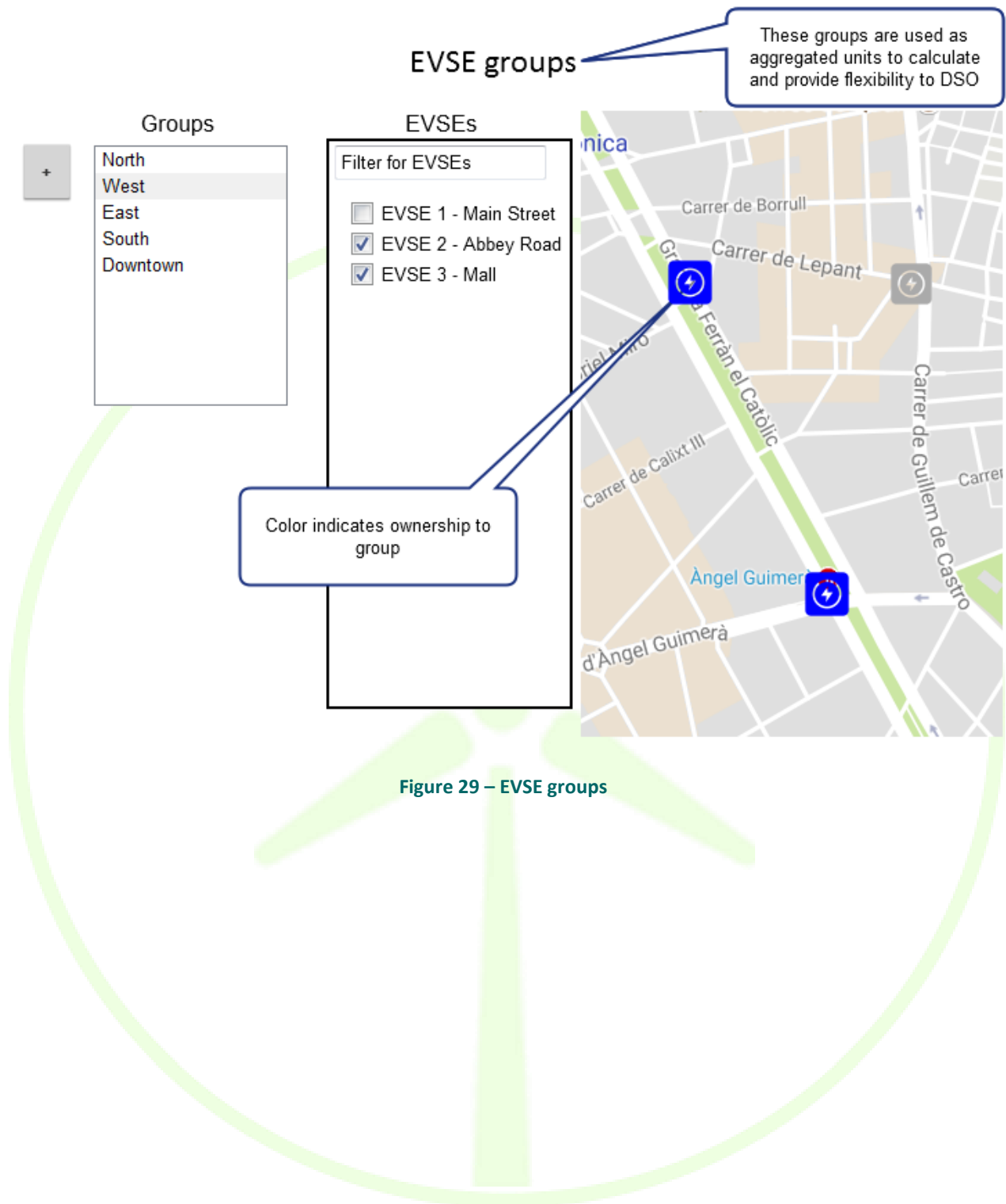
	Dumb charge	Scheduled	
Charging end time	01/01/2017 17:00	01/01/2017 20:00	
Energy consumption	16.4 kWh	16.4 kWh	
Final SoC	100%	100%	
Energy cost	3.21 €	1.91 €	-1.3 €
Emissions	4.92 kgCO2	3 kgCO2	-1.92 kgCO2
Charging time	5h	8h	+3h

Figure 28 – Optimization

### 3.5.8 EVSE groups

Participation in the Ancillary Services Market in order to provide support to the DSO requires the EVSE operator to aggregate the demand of several EVSEs into groups (regulation areas), accordingly to the area of the grid those EVSEs are connected. These groups will be referred by the DSO whenever flexibility from aggregators is requested.

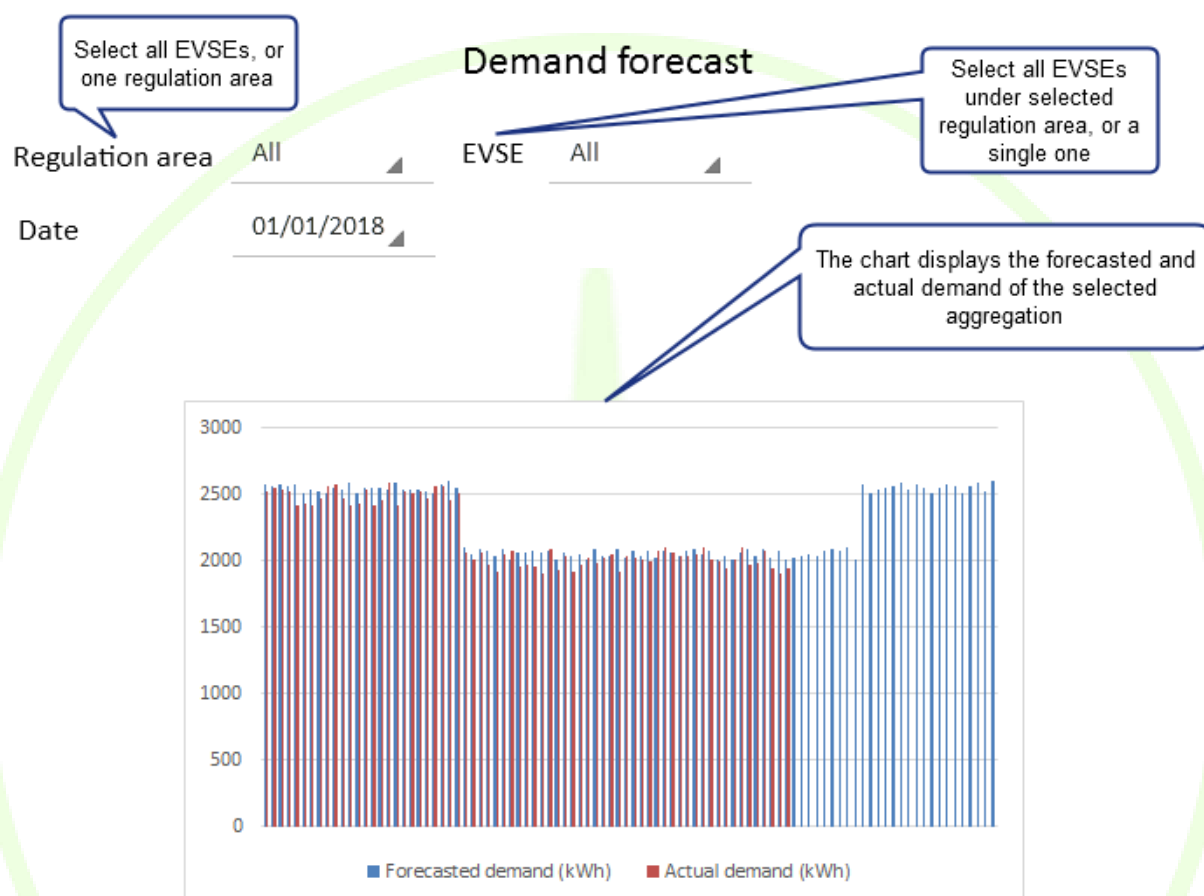
This section simply allows the operator to define which EVSEs fall under each one of those regulation areas. Flexibility offered to the DSO will be calculated and operated on a per regulation area basis.



### 3.5.9 Demand and flexibility forecasts

This section will provide an insight on the output of the forecasting modules of WiseEVP. Two forecasts are provided by the tool: demand and flexibility.

Curves of demand forecast will be displayed, together with actual demand measurements as far as those are available. This indicator will give an overview of the expected usage of the EVSE network in the short-term future.



**Figure 30 – Demand forecast**

Curves of flexibility forecast will give the operator an overview of the capabilities of the network of EVSEs to provide Ancillary Services to the DSO by rescheduling the charging sessions appropriately.

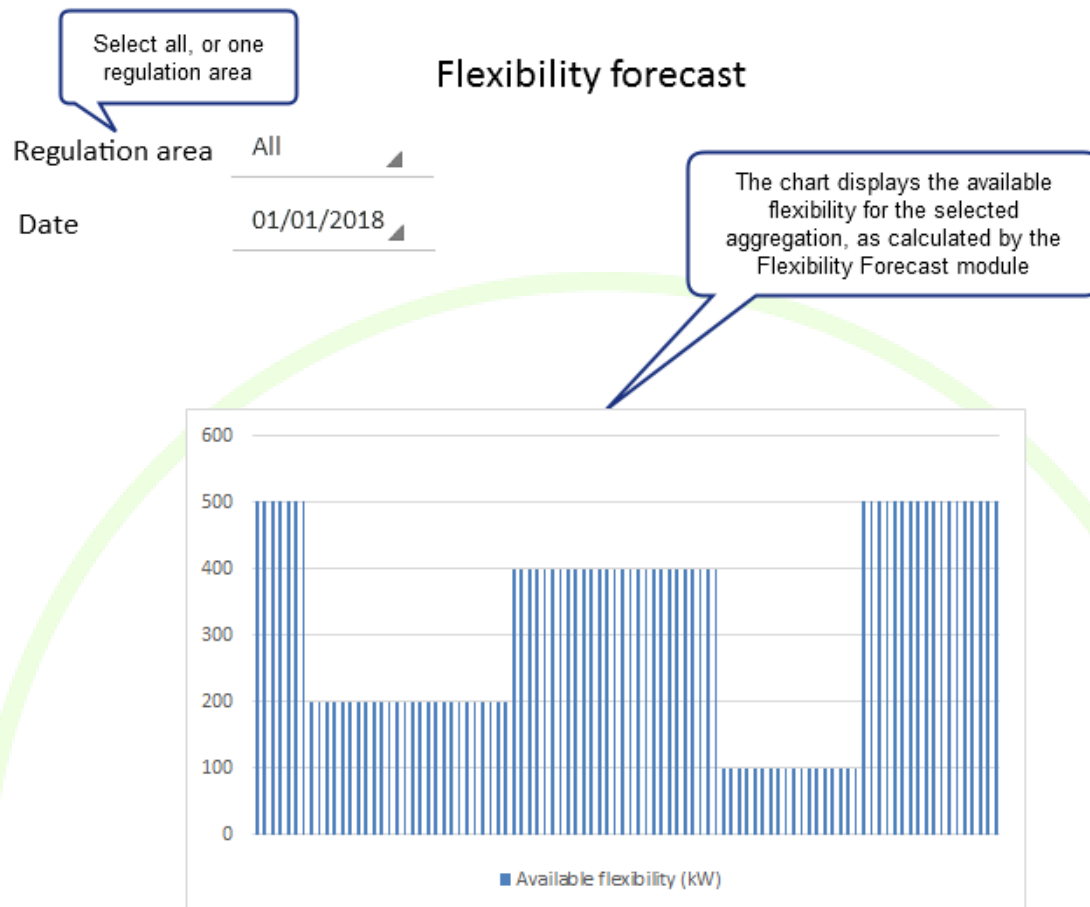


Figure 31 – Flexibility forecast

### 3.5.10 Ancillary Services Market

For each of the groups, this section shows the history and currently active demand-response campaigns, triggered by DSO via the ancillary services market. Each campaign is composed by the following data:

- Timestamp (PTU) when the requested flexibility shall be provided.
- Requested power reduction for that period.
- Price requested by operator of the EVSE network to DSO in exchange for the flexibility delivery.
- Status of the operation: request received from market, offer sent to market, offer accepted/rejected from market, on-going or finalized.

## Flexibility - Demand response

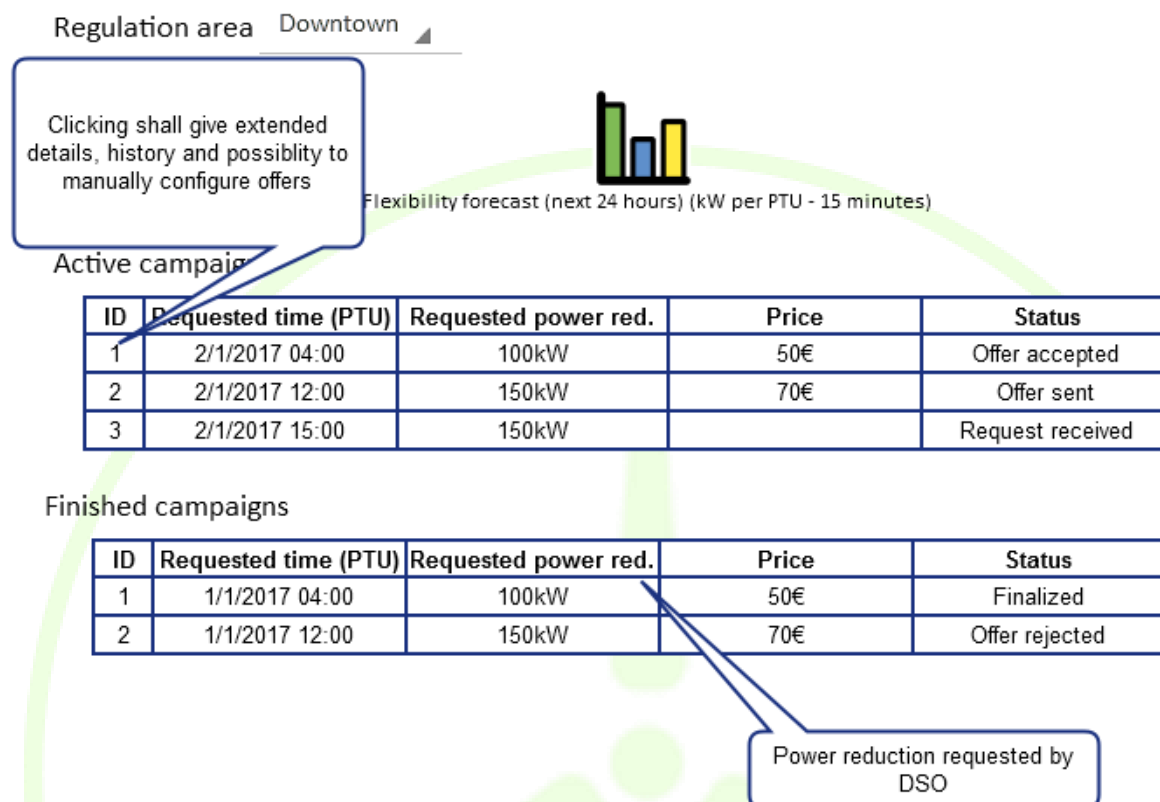
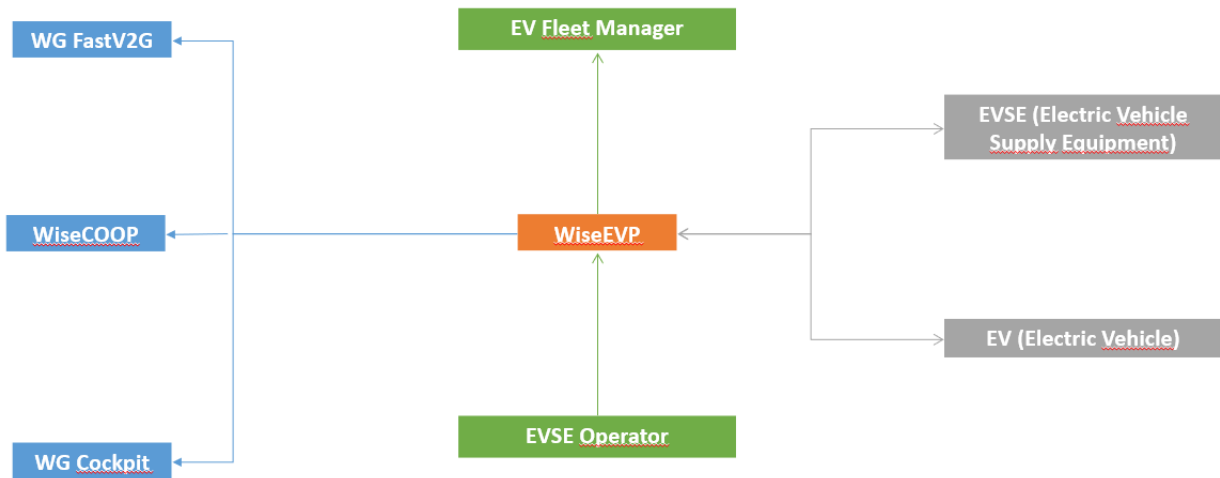


Figure 32 – Flexibility - Demand Response

## 4 LINK WITH OTHER WISEGRID APPLICATIONS

WiseEVP has connections with another three WiseGRID tools: WG FastV2G, WiseCOOP and WG Cockpit. Meanwhile links with WG FastV2G are directly related with the main activities of the tool, links with WiseCOOP and WG Cockpit are related with its DR capabilities.



**Figure 33 – WiseEVP interactions**

As explained in D10.2 “WiseGRID Flexibility based DR Optimization Framework Specifications” in an implicit DR context, WiseEVP sends to WiseCOOP the expected information of energy demand in a day ahead basis. Then, WiseEVP receives from WiseCOOP price signals that will modulate WiseEVP consumption patterns.

When the DSO triggers an explicit DR campaign, WiseEVP will send the data of the available flexibility of its EVs (according to the forecasted schedules) to WG Cockpit. In case WG Cockpit finally needs WiseEVP’s flexibility, the WiseEVP tool will send the required commands to the corresponding charging stations.

Finally, one of the most valuable capacities of WiseEVP, is its connection with WG FastV2G. WiseEVP could be seen as “the brain” behind the smart charging station which is being developed in WiseGRID. WiseEVP collects the information that WG FastV2G sends to it (user’s preferences, sockets on use...) and perform all the required calculations and commands that the charging station will need in order to smart charge the EV. WiseEVP’s interactions with WiseCOOP, WG Cockpit and WG FastV2G are of course performed through WG IOP. Also the interactions between the EVs and the EVSEs with WiseEVP are performed by means of the WG IOP.

## 5 DPIA CONSIDERATIONS

WiseEVP is a specific tool within the Project and therefore it becomes subject of an assessment to evaluate risks level in relation to data privacy conformity. The evaluation (assessment was performed under centralised procedure within D3.1 and was recently updated within deliverable D3.2. Relevant threats and events were assessed and conclusions are that no significant risks could affect WiseEVP tool from the perspective of personal data protection. However, there are general and tool specific recommendations resulting from the updated DPIA that are considered within the design phase of the Project.

The synthesis of threats and events identified within the assessment for WiseEVP is in the following table:

**Table 8 – Threat and feared events identification for WiseEVP**

Feared events	Threat ID	Threat name	Brief explanation why relevant
Disappearance of personal data: they are not or no longer available	ED	Eavesdropping of computer channels	Interception of Ethernet traffic; acquisition of data sent over a Wi-Fi network, etc.

Feared events	Threat ID	Threat name	Brief explanation why relevant
Breach of use of personal data distributed to people	LT	Lack of transparency	Data processing is not made transparent, or information is not provided in a timely manner...

The results of the Risk treatment (Risk modification) based on applied controls for WG EVP, have kept the risk level within the “Limited” level or below. Nevertheless, is considering as control action the reduction of the vulnerabilities of computer communication networks.

However, due to complete integration of WiseEVP within WiseGRID, the general controls should be applied for safer approach:

1. Take written consent from all customers involved in the usage of designated tools (the form of the consent will be developed in due time, before launching the implementation on pilot sites).
2. Data will be anonymized as soon as affordable within the process, with no influence in the functionality.

## 6 CONCLUSIONS

In this deliverable, the work performed in the framework of task T9.1 has been presented.

Task 9.1 *WiseEVP Design* focuses in the analysis of the different requirements and use cases defined by the WP2 that need to be addressed by the WiseEVP application, the identification of the necessary modules to fulfil those and the specification of the functions to be implemented within each one of these modules. In addition, interfaces with the target users of the applications have been designed following several iterations while keeping potential end users within the consortium involved in the process.

This deliverable additionally presents or references the formal specification of the APIs to be implemented by each one of the modules. This will allow the parallel development of the different modules that are responsibility of different teams within the consortium during the implementation phase, and will serve as a basis for future module-independent unitary testing and first assessment of the integration task to be performed as part of the development phase.

Upcoming work in task T9.2 includes the development of the different modules described in this deliverable. These modules will be developed independently and will implement the documented APIs described in this report. Task T9.2 will finalise with the integration of the different modules that compose the complete WiseEVP application.

Finally, under task T9.3 different functional tests over the applications – once all independent modules have been successfully integrated together – will be carried out in a lab-testing environment, thus ensuring the correct operation of the WiseEVP application under controlled environment before proceeding with the deployment of the application in the different pilot sites. Implementation and lab-testing will be finalised in month M21 and reported accordingly in the follow-up deliverable D9.2 *WiseEVP implementation and lab-testing*.

## 7 REFERENCES AND ACRONYMS

### 7.1 REFERENCES

- [1] "<https://hackernoon.com/microservices-are-hard-an-invaluable-guide-to-microservices-2d06bd7bcf5d>," [Online].
- [2] "<http://www.rabbitmq.com/resources/google-tech-talk-final/alexis-google-rabbitmq-talk.pdf>," [Online].
- [3] "<https://www.rabbitmq.com/>," [Online].
- [4] "<http://www.openchargealliance.org/protocols/ocpp/ocpp-16/>," [Online].
- [5] "[http://www.openchargealliance.org/uploads/files/protected/OCPP\\_1.6\\_Full\\_4\\_2016.zip](http://www.openchargealliance.org/uploads/files/protected/OCPP_1.6_Full_4_2016.zip)," [Online].
- [6] "D8.2 WiseGRID FastV2G and other innovative optimized storage solutions".
- [7] "<https://invers.com/>," [Online].
- [8] "<http://canze.fisch.lu/>," [Online].
- [9] X. Guo and J. Su, "Improved Support Vector Machine Short-term Power Load Forecast Model Based on Particle Swarm Optimization Parameters," *Journal of Applied Sciences*, vol. 13, no. 9, pp. 1467-1472, 2013.
- [10] "D4.2 WiseGRID interoperable Integrated Process (WG IOP)".
- [11] F. L. S. C. A. Di Giorgio, "IEC 61851 compliant electric vehicle charging control in Smartgrids," in *21st Mediterranean Conference on Control & Automation (MED)*, Crete, Greece, 2013.
- [12] "D10.2 WiseGRID Flexibility-based DR Optimization Framework Specifications".



## 7.2 ACRONYMS

**Table 9 – List of Acronyms**

Acronyms List	
API	Application Programming Interface
DER	Distributed Energy Resources
DPIA	Data Protection Impact Assessment
DR	Demand Response
DSO	Distribution System Operator
EAN	European Article Numbering
ESCO	Energy Services Company
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
GIS	Geographical Information System
HL-UC	High-Level Use Case
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
OSCP	Open Smart Charging Protocol
PDC	Phasor Data Concentrator
PLC	Programmable Logic Controller
PMU	Phasor Measurement Unit
PoD	Point of Delivery
PS	Primary Substation
PUC	Primary Use Case
PV	Photovoltaic
RES	Renewable Energy Sources
RESCO	Renewable Energy Services Company
RFID	Radio Frequency Identification
SCADA	Supervisory Control And Data Acquisition
SMX	Smart Meter eXtension
SoC	State of Charge
SoH	State of Health
SP	Sub-project
SS	Secondary Substation

Acronyms List	
SUC	Secondary Use Case
SVM	Support Vector Machine
TSO	Transmission System Operator
UC	Use Case
USM	Unbundled Smart Meter
UTC	Coordinated Universal Time
V2B	Vehicle to Building
V2G	Vehicle to Grid
VPP	Virtual Power Plant
WG	WiseGRID
WP	Work Package

## 8 ANNEXES

### 8.1 SPECIFICATION OF MESSAGES FOR EXCHANGE OF FLEXIBILITY

#### 8.1.1 FlexRequest.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="UUID">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-
Fa-f]{12}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="CurrencyAmount">
    <xs:restriction base="xs:decimal">
      <xs:fractionDigits value="4" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Disposition-AvailableRequested">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Available" />
      <xs:enumeration value="Requested" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="MessagePrecedence">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Critical" />
      <xs:enumeration value="Routine" />
      <xs:enumeration value="Transactional" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="InternetDomain">
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z0-9]+(-[a-z0-9]+)*\.)+[a-z]{2,}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="USEF-Role">
    <xs:restriction base="xs:string">
```

```

        <xs:enumeration value="AGR" />
        <xs:enumeration value="BRP" />
        <xs:enumeration value="CRO" />
        <xs:enumeration value="DSO" />
        <xs:enumeration value="MDC" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="EntityAddress">
    <xs:restriction base="xs:string">
        <xs:pattern value="(ea1\[0-9]{4}-[0-9]{2}\. {1,244} | ean\[0-9]{12,34})" />
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Period">
    <xs:restriction base="xs:date" />
</xs:simpleType>
<xs:simpleType name="TimeZoneName">
    <xs:restriction base="xs:string">
        <xs:pattern value="(Africa|America|Australia|Europe|Pacific)/[a-zA-Z0-9_]{3,}" />
    </xs:restriction>
</xs:simpleType>

<xs:element name="MessageMetadata">
    <xs:annotation>
        <xs:documentation>The MessageMetadata element contains mandatory metadata
which is common for each non-wrapper USEF message. This element is always included as part of a message
and never transmitted by itself.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:attribute name="SenderDomain" type="InternetDomain" use="required">
            <xs:annotation>
                <xs:documentation>The Internet domain of the USEF participant
sending this message. When receiving a message, its value should match the value specified in the
SignedMessage wrapper: otherwise, the message must be rejected as invalid. When replying to this message,
this attribute, combined with the SenderRole, is used to look up the USEF endpoint the reply message should
be delivered to.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="SenderRole" type="USEF-Role" use="required">

```

```

<xs:annotation>
    <xs:documentation>USEF role of the participant sending this
message: AGR, BRP, CRO, DSO or MDC. When receiving a message, its value should match the value specified
in the SignedMessage wrapper: otherwise, the message must be rejected as invalid.</xs:documentation>
</xs:annotation>
</xs:attribute>
<xs:attribute name="RecipientDomain" type="InternetDomain" use="required">
    <xs:annotation>
        <xs:documentation>Internet domain of the participant this message
is intended for. When sending a message, this attribute, combined with the RecipientRole, is used to look up
the USEF endpoint the message should be delivered to.</xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute name="RecipientRole" type="USEF-Role" use="required">
    <xs:annotation>
        <xs:documentation>USEF role of the participant this message is
intended for: AGR, BRP, CRO, DSO or MDC.</xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute name="TimeStamp" type="xs:dateTime" use="required">
    <xs:annotation>
        <xs:documentation>Date and time this message was created,
including the time zone (ISO 8601 formatted as per http://www.w3.org/TR/NOTE-datetime).</xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute name="MessageID" type="UUID" use="required">
    <xs:annotation>
        <xs:documentation>Unique identifier (UUID/GUID as per IETF RFC
4122) for this message, to be generated when composing each message.</xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute name="ConversationID" type="UUID" use="required">
    <xs:annotation>
        <xs:documentation>Unique identifier (UUID/GUID as per IETF RFC
4122) used to correlate responses with requests, to be generated when composing the first message in a
conversation and subsequently copied from the original message to each reply
message.</xs:documentation>
    </xs:annotation>
</xs:attribute>

```

```
</xs:attribute>
```

```
<xs:attribute name="Precedence" type="MessagePrecedence" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Indication of the importance and impact of the message: Routine, Transactional or Critical. Used to determine time-out values during message exchange and the level of error notification used in the sending implementation.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="ValidUntil" type="xs:dateTime">
```

```
<xs:annotation>
```

<xs:documentation>Optional absolute date and time (ISO 8601 formatted, including time zone) this message expires. Used by implementations to determine if a pending message should still be processed.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
</xs:complexType>
```

```
</xs:element>
```

```
<!-- -->
```

```
<!-->
```

```
<xs:element name="PTU">
```

```
<xs:annotation>
```

<xs:documentation>The PTU element represents one or more Program Time Units and is used by Prognosis and Flex-related messages. This element is always included as part of a message and never transmitted by itself. |

Each PTU includes a Power value, the sign of which is used to distinguish between production and consumption, from the perspective of the Prosumer. A positive value indicates that power flows towards the Prosumer (consumption), a negative value indicates flow towards the grid (production). |

For FlexRequests, the single Power value is insufficient, since it would be impossible to make a distinction between a request for the reduction of the amount of energy produced (two minus signs, thus a positive value) and the indication of room for more consumption (a positive value as well). Hence, each PTU also includes an indicator, Disposition, to distinguish between these situations. |

Note that a request is always relative to an earlier prognosis, and the intent of the party sending the request can only be determined by taking into account whether the net outcome of the PTU is expected to be production or consumption. |

Also note that all scenarios have valid alternative realizations: a reduction in production by 100 can also be accomplished by an increase in consumption by that amount. The resulting flexibility offer for the PTU will have the same Power value in both cases.

```

    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="Disposition" type="Disposition-AvailableRequested"
use="optional">
      <xs:annotation>
        <xs:documentation>Optional, used only for FlexRequest messages:
indication whether the Power specified for this PTU represents available capacity or a request for
reduction/increase.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="Power" type="xs:integer" use="required">
      <xs:annotation>
        <xs:documentation>Power specified for this PTU in Watts. Also see
the important notes about the sign of this attribute in the main documentation entry for the PTU
element.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="Start" type="xs:integer" use="required">
      <xs:annotation>
        <xs:documentation>Number of the first PTU this element refers to.
The first PTU of a day has number 1.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="Duration" type="xs:integer" use="optional" default="1">
      <xs:annotation>
        <xs:documentation>The number of the PTUs this element
represents. Optional, default value is 1.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="Price" type="CurrencyAmount" use="optional">
      <xs:annotation>
        <xs:documentation>The price offered or accepted for supplying the
indicated amount of flexibility in this PTU. Only valid for FlexOffer and FlexOrder messages; the currency
associated with this amount is included in the main part of those messages.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>

```

```

        </xs:attribute>
    </xs:complexType>
</xs:element>
<!-->

```

```

<xs:complexType name="FlexBase" abstract="true">

```

```

    <xs:annotation>

```

`<xs:documentation>`FlexBase elements are the basis for the various Flex\* messages, such as FlexRequest, FlexOffer and FlexOrder, and contain all attributes common to those messages. This is an abstract element which is always used to instantiate another message type and never used or transmitted by itself.`</xs:documentation>`

```

    </xs:annotation>

```

```

    <xs:sequence>

```

```

        <xs:element ref="MessageMetadata" />

```

```

        <xs:element ref="PTU" minOccurs="0" maxOccurs="unbounded" />

```

```

        <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded" />

```

```

    </xs:sequence>

```

```

    <xs:attribute name="PTU-Duration" type="xs:duration" use="required">

```

```

        <xs:annotation>

```

`<xs:documentation>`ISO 8601 time interval (minutes only, for example PT15M) indicating the duration of the PTUs referenced in this Flex\* message. Although the PTU length is a market-wide fixed value, making this assumption explicit in each message is important for validation purposes, allowing implementations to reject messages with an errant PTU duration.`</xs:documentation>`

```

        </xs:annotation>

```

```

    </xs:attribute>

```

```

    <xs:attribute name="Period" type="Period" use="required">

```

```

        <xs:annotation>

```

`<xs:documentation>`Day (in yyyy-mm-dd format) the PTUs referenced in this Flex\* message belong to.`</xs:documentation>`

```

        </xs:annotation>

```

```

    </xs:attribute>

```

```

    <xs:attribute name="TimeZone" type="TimeZoneName" use="required">

```

```

        <xs:annotation>

```

`<xs:documentation>`Time zone ID (as per the IANA time zone database, <http://www.iana.org/time-zones>, for example: Europe/Amsterdam) indicating the UTC offset that applies to the Period referenced in this message. Although the time zone is a market-wide fixed value, making this assumption explicit in each message is important for validation purposes, allowing implementations to reject messages with an errant UTC offset.`</xs:documentation>`



```

        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="CongestionPoint" type="EntityAddress" use="optional">
        <xs:annotation>
            <xs:documentation>Entity Address of the Congestion Point this Flex*
message applies to. Optional: if left out (which is only legal for BRP FlexOffers), it applies to all BRP
Connections.</xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="Sequence" type="xs:long" use="required">
        <xs:annotation>
            <xs:documentation>Sequence number of this message, which should be
incremented each time a new revision of a Flex* message is sent. To ensure unique incrementing sequence
numbers, use of the format yyyyymmddHHMMSSsss (year, month, day, hour, minutes, seconds and
milliseconds, respectively) is highly recommended.</xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="ExpirationDateTime" type="xs:dateTime" use="required">
        <xs:annotation>
            <xs:documentation>Date and time, including the time zone (ISO 8601
formatted as per http://www.w3.org/TR/NOTE-datetime) until which the Flex* message is
valid.</xs:documentation>
        </xs:annotation>
    </xs:attribute>
</xs:complexType>

<xs:element name="FlexRequest">
    <xs:annotation>
        <xs:documentation>FlexRequest messages are used by BRPs and DSOs to request
flexibility from Aggregators. In addition to one or more PTU elements with Disposition=Requested, indicating
the actual need to reduce consumption or production, the message should also include the remaining PTUs
for the current Period where Disposition=Available, so the receiving Aggregator can decide whether time-
shifting load is an option to meet the needs of the requesting party.</xs:documentation>
    </xs:annotation>
<xs:complexType>
    <xs:complexContent>
        <xs:extension base="FlexBase">
            <xs:attribute name="PrognosisOrigin" type="InternetDomain"

```

use="required">

<xs:annotation>

<xs:documentation>The Internet domain of the USEF participant that sent the Prognosis message (more specifically: the D-Prognosis) this request is based on.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="PrognosisSequence" type="xs:long"

use="required">

<xs:annotation>

<xs:documentation>Sequence number of the D-Prognosis this request is based on. The combination of PrognosisOrigin and PrognosisSequence should be unique.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:extension>

</xs:complexContent>

</xs:complexType>

</xs:element>

</xs:schema>

### 8.1.2 FlexRequestResponse.xsd

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:simpleType name="InternetDomain">

<xs:restriction base="xs:string">

<xs:pattern value="([a-z0-9]+(-[a-z0-9]+)\*\.)+[a-z]{2,}" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="USEF-Role">

<xs:restriction base="xs:string">

<xs:enumeration value="AGR" />

<xs:enumeration value="BRP" />

<xs:enumeration value="CRO" />

<xs:enumeration value="DSO" />

<xs:enumeration value="MDC" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="UUID">

```

<xs:restriction base="xs:string">
  <xs:pattern value="[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}" />
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="MessagePrecedence">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Critical" />
    <xs:enumeration value="Routine" />
    <xs:enumeration value="Transactional" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Disposition-AvailableRequested">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Available" />
    <xs:enumeration value="Requested" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Disposition-AcceptedRejected">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Accepted" />
    <xs:enumeration value="Rejected" />
  </xs:restriction>
</xs:simpleType>
<xs:element name="MessageMetadata">
  <xs:annotation>
    <xs:documentation>The MessageMetadata element contains mandatory metadata which is common
for each non-wrapper USEF message. This element is always included as part of a message and never
transmitted by itself.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="SenderDomain" type="InternetDomain" use="required">
      <xs:annotation>
        <xs:documentation>The Internet domain of the USEF participant sending this message. When
receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise,
the message must be rejected as invalid. When replying to this message, this attribute, combined with the
SenderRole, is used to look up the USEF endpoint the reply message should be delivered
to.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
</xs:element>

```

```
</xs:attribute>
```

```
<xs:attribute name="SenderRole" type="USEF-Role" use="required">
```

```
<xs:annotation>
```

<xs:documentation>USEF role of the participant sending this message: AGR, BRP, CRO, DSO or MDC. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="RecipientDomain" type="InternetDomain" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Internet domain of the participant this message is intended for. When sending a message, this attribute, combined with the RecipientRole, is used to look up the USEF endpoint the message should be delivered to.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="RecipientRole" type="USEF-Role" use="required">
```

```
<xs:annotation>
```

<xs:documentation>USEF role of the participant this message is intended for: AGR, BRP, CRO, DSO or MDC.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="TimeStamp" type="xs:dateTime" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Date and time this message was created, including the time zone (ISO 8601 formatted as per <http://www.w3.org/TR/NOTE-datetime>).</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="MessageID" type="UUID" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) for this message, to be generated when composing each message.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="ConversationID" type="UUID" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) used to correlate responses with requests, to be generated when composing the first message in a conversation and subsequently copied from the original message to each reply message.</xs:documentation>

```

    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="Precedence" type="MessagePrecedence" use="required">
    <xs:annotation>
      <xs:documentation>Indication of the importance and impact of the message: Routine,
      Transactional or Critical. Used to determine time-out values during message exchange and the level of error
      notification used in the sending implementation.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="ValidUntil" type="xs:dateTime">
    <xs:annotation>
      <xs:documentation>Optional absolute date and time (ISO 8601 formatted, including time zone)
      this message expires. Used by implementations to determine if a pending message should still be
      processed.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="FlexRequestResponse">
  <xs:annotation>
    <xs:documentation>Upon receiving and processing a FlexRequest message, the receiving
    implementation must reply with a FlexRequestResponse, indicating whether the flex request was processed
    successfully.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="MessageMetadata" />
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="Sequence" type="xs:long" use="required">
      <xs:annotation>
        <xs:documentation>Sequence number of the FlexRequest that has been
        received.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="Result" type="Disposition-AcceptedRejected" use="required">
      <xs:annotation>
        <xs:documentation>Indication whether the flex request was accepted or rejected. Rejection is

```

allowed in case the FlexRequest is not based on our latest Prognosis, a FlexRequest from the same participant for the indicated period with a higher sequence number was already accepted previously, the FlexRequest does not contain any PTUs with Disposition=Requested, or in case those PTUs do not cover the entire Period, no PTUs with Disposition=Available are included.</xs:documentation>

```
</xs:annotation>
</xs:attribute>
<xs:attribute name="Message" type="xs:string" use="optional">
  <xs:annotation>
    <xs:documentation>In case the request was rejected, this attribute must contain a human-
readable description of the reason.</xs:documentation>
  </xs:annotation>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:schema>
```

### 8.1.3 FlexOffer.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="UUID">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ISO4217Currency">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{3}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="CurrencyAmount">
    <xs:restriction base="xs:decimal">
      <xs:fractionDigits value="4" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Disposition-AvailableRequested">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Available" />
      <xs:enumeration value="Requested" />
    </xs:restriction>
```

```

</xs:simpleType>
<xs:simpleType name="MessagePrecedence">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Critical" />
    <xs:enumeration value="Routine" />
    <xs:enumeration value="Transactional" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="InternetDomain">
  <xs:restriction base="xs:string">
    <xs:pattern value="([a-z0-9]+(-[a-z0-9]+)*\.)+[a-z]{2,}" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="USEF-Role">
  <xs:restriction base="xs:string">
    <xs:enumeration value="AGR" />
    <xs:enumeration value="BRP" />
    <xs:enumeration value="CRO" />
    <xs:enumeration value="DSO" />
    <xs:enumeration value="MDC" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="EntityAddress">
  <xs:restriction base="xs:string">
    <xs:pattern value="(ea1\.[0-9]{4}-[0-9]{2}\.?.{1,244}?:{1,244}|ean\.[0-9]{12,34})" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Period">
  <xs:restriction base="xs:date" />
</xs:simpleType>
<xs:simpleType name="TimeZoneName">
  <xs:restriction base="xs:string">
    <xs:pattern value="(Africa|America|Australia|Europe|Pacific)/[a-zA-Z0-9_]{3,}" />
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="FlexBase" abstract="true">

```

```
<xs:annotation>
```

`<xs:documentation>`FlexBase elements are the basis for the various Flex\* messages, such as FlexRequest, FlexOffer and FlexOrder, and contain all attributes common to those messages. This is an abstract element which is always used to instantiate another message type and never used or transmitted by itself.`</xs:documentation>`

```
</xs:annotation>
```

```
<xs:sequence>
```

```
  <xs:element ref="MessageMetadata" />
```

```
  <xs:element ref="PTU" minOccurs="0" maxOccurs="unbounded" />
```

```
  <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
```

```
</xs:sequence>
```

```
<xs:attribute name="PTU-Duration" type="xs:duration" use="required">
```

```
  <xs:annotation>
```

`<xs:documentation>`ISO 8601 time interval (minutes only, for example PT15M) indicating the duration of the PTUs referenced in this Flex\* message. Although the PTU length is a market-wide fixed value, making this assumption explicit in each message is important for validation purposes, allowing implementations to reject messages with an errant PTU duration.`</xs:documentation>`

```
  </xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="Period" type="Period" use="required">
```

```
  <xs:annotation>
```

`<xs:documentation>`Day (in yyyy-mm-dd format) the PTUs referenced in this Flex\* message belong to.`</xs:documentation>`

```
  </xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="TimeZone" type="TimeZoneName" use="required">
```

```
  <xs:annotation>
```

`<xs:documentation>`Time zone ID (as per the IANA time zone database, <http://www.iana.org/time-zones>, for example: Europe/Amsterdam) indicating the UTC offset that applies to the Period referenced in this message. Although the time zone is a market-wide fixed value, making this assumption explicit in each message is important for validation purposes, allowing implementations to reject messages with an errant UTC offset.`</xs:documentation>`

```
  </xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="CongestionPoint" type="EntityAddress" use="optional">
```

```
  <xs:annotation>
```

`<xs:documentation>`Entity Address of the Congestion Point this Flex\* message applies to. Optional: if left out (which is only legal for BRP FlexOffers), it applies to all BRP Connections.`</xs:documentation>`

```
  </xs:annotation>
```



```
</xs:attribute>
```

```
<xs:attribute name="Sequence" type="xs:long" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Sequence number of this message, which should be incremented each time a new revision of a Flex\* message is sent. To ensure unique incrementing sequence numbers, use of the format yyyymmddHHMMSSsss (year, month, day, hour, minutes, seconds and milliseconds, respectively) is highly recommended.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="ExpirationDateTime" type="xs:dateTime" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Date and time, including the time zone (ISO 8601 formatted as per <http://www.w3.org/TR/NOTE-datetime>) until which the Flex\* message is valid.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
</xs:complexType>
```

```
<xs:element name="MessageMetadata">
```

```
<xs:annotation>
```

<xs:documentation>The MessageMetadata element contains mandatory metadata which is common for each non-wrapper USEF message. This element is always included as part of a message and never transmitted by itself.</xs:documentation>

```
</xs:annotation>
```

```
<xs:complexType>
```

```
<xs:attribute name="SenderDomain" type="InternetDomain" use="required">
```

```
<xs:annotation>
```

<xs:documentation>The Internet domain of the USEF participant sending this message. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid. When replying to this message, this attribute, combined with the SenderRole, is used to look up the USEF endpoint the reply message should be delivered to.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="SenderRole" type="USEF-Role" use="required">
```

```
<xs:annotation>
```

<xs:documentation>USEF role of the participant sending this message: AGR, BRP, CRO, DSO or MDC. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="RecipientDomain" type="InternetDomain" use="required">
```

```
<xs:annotation>
```

```
<xs:documentation>Internet domain of the participant this message is intended for. When sending a message, this attribute, combined with the RecipientRole, is used to look up the USEF endpoint the message should be delivered to.</xs:documentation>
```

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="RecipientRole" type="USEF-Role" use="required">
```

```
<xs:annotation>
```

```
<xs:documentation>USEF role of the participant this message is intended for: AGR, BRP, CRO, DSO or MDC.</xs:documentation>
```

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="TimeStamp" type="xs:dateTime" use="required">
```

```
<xs:annotation>
```

```
<xs:documentation>Date and time this message was created, including the time zone (ISO 8601 formatted as per http://www.w3.org/TR/NOTE-datetime).</xs:documentation>
```

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="MessageID" type="UUID" use="required">
```

```
<xs:annotation>
```

```
<xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) for this message, to be generated when composing each message.</xs:documentation>
```

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="ConversationID" type="UUID" use="required">
```

```
<xs:annotation>
```

```
<xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) used to correlate responses with requests, to be generated when composing the first message in a conversation and subsequently copied from the original message to each reply message.</xs:documentation>
```

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="Precedence" type="MessagePrecedence" use="required">
```

```
<xs:annotation>
```

```
<xs:documentation>Indication of the importance and impact of the message: Routine, Transactional or Critical. Used to determine time-out values during message exchange and the level of error notification used in the sending implementation.</xs:documentation>
```

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="ValidUntil" type="xs:dateTime">
```

```
<xs:annotation>
```

<xs:documentation>Optional absolute date and time (ISO 8601 formatted, including time zone) this message expires. Used by implementations to determine if a pending message should still be processed.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
</xs:complexType>
```

```
</xs:element>
```

```
<xs:element name="PTU">
```

```
<xs:annotation>
```

<xs:documentation>The PTU element represents one or more Program Time Units and is used by Prognosis and Flex-related messages. This element is always included as part of a message and never transmitted by itself. |

Each PTU includes a Power value, the sign of which is used to distinguish between production and consumption, from the perspective of the Prosumer. A positive value indicates that power flows towards the Prosumer (consumption), a negative value indicates flow towards the grid (production). |

For FlexRequests, the single Power value is insufficient, since it would be impossible to make a distinction between a request for the reduction of the amount of energy produced (two minus signs, thus a positive value) and the indication of room for more consumption (a positive value as well). Hence, each PTU also includes an indicator, Disposition, to distinguish between these situations. |

Note that a request is always relative to an earlier prognosis, and the intent of the party sending the request can only be determined by taking into account whether the net outcome of the PTU is expected to be production or consumption. |

Also note that all scenarios have valid alternative realizations: a reduction in production by 100 can also be accomplished by an increase in consumption by that amount. The resulting flexibility offer for the PTU will have the same Power value in both cases.

```
</xs:documentation>
```

```
</xs:annotation>
```

```
<xs:complexType>
```

```
<xs:attribute name="Disposition" type="Disposition-AvailableRequested" use="optional">
```

```
<xs:annotation>
```

<xs:documentation>Optional, used only for FlexRequest messages: indication whether the Power specified for this PTU represents available capacity or a request for reduction/increase.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="Power" type="xs:integer" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Power specified for this PTU in Watts. Also see the important notes about the sign of this attribute in the main documentation entry for the PTU element.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="Start" type="xs:integer" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Number of the first PTU this element refers to. The first PTU of a day has number 1.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="Duration" type="xs:integer" use="optional" default="1">
```

```
<xs:annotation>
```

<xs:documentation>The number of the PTUs this element represents. Optional, default value is 1.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="Price" type="CurrencyAmount" use="optional">
```

```
<xs:annotation>
```

<xs:documentation>The price offered or accepted for supplying the indicated amount of flexibility in this PTU. Only valid for FlexOffer and FlexOrder messages; the currency associated with this amount is included in the main part of those messages.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
</xs:complexType>
```

```
</xs:element>
```

```
<!-->
```

```
<xs:element name="FlexOffer">
```

```
<xs:annotation>
```

<xs:documentation>FlexOffer messages are used by Aggregators to make DSOs and BRPs an offer for providing flexibility. A FlexOffer message contains a list of PTUs, with for each PTU the change in consumption or production offered, plus the price for this amount of flexibility. FlexOffer messages should only be sent once a FlexRequest message has been received and must never be sent unsolicited. Note that multiple FlexOffer messages may be sent based on a single FlexRequest: for example, one offer that exactly matches the power reduction requested, plus one with a different amount of reduction, with more favorable pricing. When responding to a BRP-originated FlexRequest, an Aggregator may send an empty FlexOffer message (i.e. a message not containing any PTU elements) in order to indicate that no flexibility is available

and the submitted A-plan is expected to be approved as-is.</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:complexContent>

<xs:extension base="FlexBase">

<xs:attribute name="FlexRequestOrigin" type="InternetDomain" use="required">

<xs:annotation>

<xs:documentation>The Internet domain of the USEF participant that sent the FlexRequest this offer is based on.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="FlexRequestSequence" type="xs:long" use="required">

<xs:annotation>

<xs:documentation>Sequence number of the FlexRequest message this request is based on. The combination of FlexRequestOrigin and FlexRequestSequence should be unique.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="Currency" type="ISO4217Currency" use="required">

<xs:annotation>

<xs:documentation>ISO 4217 code indicating the currency that applies to the prices listed for each PTU.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:extension>

</xs:complexContent>

</xs:complexType>

</xs:element>

</xs:schema>

#### 8.1.4 FlexOfferResponse.xsd

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:simpleType name="InternetDomain">

<xs:restriction base="xs:string">

<xs:pattern value="([a-z0-9]+(-[a-z0-9]+)\*\.)+[a-z]{2,}" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="USEF-Role">

<xs:restriction base="xs:string">

```

    <xs:enumeration value="AGR" />
    <xs:enumeration value="BRP" />
    <xs:enumeration value="CRO" />
    <xs:enumeration value="DSO" />
    <xs:enumeration value="MDC" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="UUID">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="MessagePrecedence">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Critical" />
    <xs:enumeration value="Routine" />
    <xs:enumeration value="Transactional" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Disposition-AvailableRequested">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Available" />
    <xs:enumeration value="Requested" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Disposition-AcceptedRejected">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Accepted" />
    <xs:enumeration value="Rejected" />
  </xs:restriction>
</xs:simpleType>
<xs:element name="MessageMetadata">
  <xs:annotation>

```

<xs:documentation>The MessageMetadata element contains mandatory metadata which is common for each non-wrapper USEF message. This element is always included as part of a message and never transmitted by itself.</xs:documentation>

```
</xs:annotation>
```

```
<xs:complexType>
```

```
  <xs:attribute name="SenderDomain" type="InternetDomain" use="required">
```

```
    <xs:annotation>
```

<xs:documentation>The Internet domain of the USEF participant sending this message. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid. When replying to this message, this attribute, combined with the SenderRole, is used to look up the USEF endpoint the reply message should be delivered to.</xs:documentation>

```
    </xs:annotation>
```

```
  </xs:attribute>
```

```
  <xs:attribute name="SenderRole" type="USEF-Role" use="required">
```

```
    <xs:annotation>
```

<xs:documentation>USEF role of the participant sending this message: AGR, BRP, CRO, DSO or MDC. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid.</xs:documentation>

```
    </xs:annotation>
```

```
  </xs:attribute>
```

```
  <xs:attribute name="RecipientDomain" type="InternetDomain" use="required">
```

```
    <xs:annotation>
```

<xs:documentation>Internet domain of the participant this message is intended for. When sending a message, this attribute, combined with the RecipientRole, is used to look up the USEF endpoint the message should be delivered to.</xs:documentation>

```
    </xs:annotation>
```

```
  </xs:attribute>
```

```
  <xs:attribute name="RecipientRole" type="USEF-Role" use="required">
```

```
    <xs:annotation>
```

<xs:documentation>USEF role of the participant this message is intended for: AGR, BRP, CRO, DSO or MDC.</xs:documentation>

```
    </xs:annotation>
```

```
  </xs:attribute>
```

```
  <xs:attribute name="TimeStamp" type="xs:dateTime" use="required">
```

```
    <xs:annotation>
```

<xs:documentation>Date and time this message was created, including the time zone (ISO 8601 formatted as per <http://www.w3.org/TR/NOTE-datetime>).</xs:documentation>

```
    </xs:annotation>
```

```
  </xs:attribute>
```

```
  <xs:attribute name="MessageID" type="UUID" use="required">
```

```
    <xs:annotation>
```

<xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) for this message, to be generated when composing each message.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="ConversationID" type="UUID" use="required">

<xs:annotation>

<xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) used to correlate responses with requests, to be generated when composing the first message in a conversation and subsequently copied from the original message to each reply message.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="Precedence" type="MessagePrecedence" use="required">

<xs:annotation>

<xs:documentation>Indication of the importance and impact of the message: Routine, Transactional or Critical. Used to determine time-out values during message exchange and the level of error notification used in the sending implementation.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="ValidUntil" type="xs:dateTime">

<xs:annotation>

<xs:documentation>Optional absolute date and time (ISO 8601 formatted, including time zone) this message expires. Used by implementations to determine if a pending message should still be processed.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:complexType>

</xs:element>

<xs:element name="FlexOfferRevocationResponse">

<xs:annotation>

<xs:documentation>Upon receiving and processing a FlexOfferRevocation message, the receiving implementation must reply with a FlexOfferRevocationResponse, indicating whether the revocation was handled successfully.</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:sequence>

<xs:element ref="MessageMetadata" />

<xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />

</xs:sequence>



```

<xs:attribute name="Sequence" type="xs:long" use="required">
  <xs:annotation>
    <xs:documentation>Sequence number of the FlexOffer that a revocation message was received
for.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="Result" type="Disposition-AcceptedRejected" use="required">
  <xs:annotation>
    <xs:documentation>Indication whether the revocation was accepted or rejected. Rejection is only
allowed in case the FlexOffer is unknown (it is the responsibility of the sending party not to revoke FlexOffer
messages which have not yet been accepted) or if it applies to a period of which a PTU is already in the
Operate phase (at which time USEF explicitly forbids revocation).</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="Message" type="xs:string" use="optional">
  <xs:annotation>
    <xs:documentation>In case the revocation was rejected, this attribute must contain a human-
readable description of the reason.</xs:documentation>
  </xs:annotation>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:schema>

```

### 8.1.5 FlexOrder.xsd

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="UUID">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ISO4217Currency">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{3}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="CurrencyAmount">
    <xs:restriction base="xs:decimal">

```

```

    <xs:fractionDigits value="4" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Disposition-AvailableRequested">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Available" />
    <xs:enumeration value="Requested" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="MessagePrecedence">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Critical" />
    <xs:enumeration value="Routine" />
    <xs:enumeration value="Transactional" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="InternetDomain">
  <xs:restriction base="xs:string">
    <xs:pattern value="([a-z0-9]+(-[a-z0-9]+)*\.)+[a-z]{2,}" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="USEF-Role">
  <xs:restriction base="xs:string">
    <xs:enumeration value="AGR" />
    <xs:enumeration value="BRP" />
    <xs:enumeration value="CRO" />
    <xs:enumeration value="DSO" />
    <xs:enumeration value="MDC" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="EntityAddress">
  <xs:restriction base="xs:string">
    <xs:pattern value="(ea1\[0-9]{4}-[0-9]{2}\.{1,244}:\{1,244}\}|ean\[0-9]{12,34})" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Period">

```

```

<xs:restriction base="xs:date" />
</xs:simpleType>
<xs:simpleType name="TimeZoneName">
  <xs:restriction base="xs:string">
    <xs:pattern value="(Africa|America|Australia|Europe|Pacific)/[a-zA-Z0-9_]{3,}" />
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="FlexBase" abstract="true">

```

`<xs:annotation>`  
`<xs:documentation>`FlexBase elements are the basis for the various Flex\* messages, such as FlexRequest, FlexOffer and FlexOrder, and contain all attributes common to those messages. This is an abstract element which is always used to instantiate another message type and never used or transmitted by itself.`</xs:documentation>`  
`</xs:annotation>`

```

</xs:annotation>
<xs:sequence>
  <xs:element ref="MessageMetadata" />
  <xs:element ref="PTU" minOccurs="0" maxOccurs="unbounded" />
  <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>

```

```

<xs:attribute name="PTU-Duration" type="xs:duration" use="required">
  <xs:annotation>

```

`<xs:documentation>`ISO 8601 time interval (minutes only, for example PT15M) indicating the duration of the PTUs referenced in this Flex\* message. Although the PTU length is a market-wide fixed value, making this assumption explicit in each message is important for validation purposes, allowing implementations to reject messages with an errant PTU duration.`</xs:documentation>`  
`</xs:annotation>`

```

</xs:annotation>
</xs:attribute>
<xs:attribute name="Period" type="Period" use="required">
  <xs:annotation>

```

`<xs:documentation>`Day (in yyyy-mm-dd format) the PTUs referenced in this Flex\* message belong to.`</xs:documentation>`  
`</xs:annotation>`

```

</xs:annotation>
</xs:attribute>
<xs:attribute name="TimeZone" type="TimeZoneName" use="required">
  <xs:annotation>

```

`<xs:documentation>`Time zone ID (as per the IANA time zone database, <http://www.iana.org/time-zones>, for example: Europe/Amsterdam) indicating the UTC offset that applies to the Period referenced in this message. Although the time zone is a market-wide fixed value, making this assumption explicit in each message is important for validation purposes, allowing implementations to reject messages with an errant

UTC offset.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="CongestionPoint" type="EntityAddress" use="optional">

<xs:annotation>

<xs:documentation>Entity Address of the Congestion Point this Flex\* message applies to. Optional: if left out (which is only legal for BRP FlexOffers), it applies to all BRP Connections.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="Sequence" type="xs:long" use="required">

<xs:annotation>

<xs:documentation>Sequence number of this message, which should be incremented each time a new revision of a Flex\* message is sent. To ensure unique incrementing sequence numbers, use of the format yyyyymmddHHMMSSsss (year, month, day, hour, minutes, seconds and milliseconds, respectively) is highly recommended.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="ExpirationDateTime" type="xs:dateTime" use="required">

<xs:annotation>

<xs:documentation>Date and time, including the time zone (ISO 8601 formatted as per <http://www.w3.org/TR/NOTE-datetime>) until which the Flex\* message is valid.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:complexType>

<xs:element name="MessageMetadata">

<xs:annotation>

<xs:documentation>The MessageMetadata element contains mandatory metadata which is common for each non-wrapper USEF message. This element is always included as part of a message and never transmitted by itself.</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:attribute name="SenderDomain" type="InternetDomain" use="required">

<xs:annotation>

<xs:documentation>The Internet domain of the USEF participant sending this message. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid. When replying to this message, this attribute, combined with the SenderRole, is used to look up the USEF endpoint the reply message should be delivered to.</xs:documentation>

</xs:annotation>

```
</xs:attribute>
```

```
<xs:attribute name="SenderRole" type="USEF-Role" use="required">
```

```
<xs:annotation>
```

<xs:documentation>USEF role of the participant sending this message: AGR, BRP, CRO, DSO or MDC. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="RecipientDomain" type="InternetDomain" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Internet domain of the participant this message is intended for. When sending a message, this attribute, combined with the RecipientRole, is used to look up the USEF endpoint the message should be delivered to.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="RecipientRole" type="USEF-Role" use="required">
```

```
<xs:annotation>
```

<xs:documentation>USEF role of the participant this message is intended for: AGR, BRP, CRO, DSO or MDC.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="TimeStamp" type="xs:dateTime" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Date and time this message was created, including the time zone (ISO 8601 formatted as per <http://www.w3.org/TR/NOTE-datetime>).</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="MessageID" type="UUID" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) for this message, to be generated when composing each message.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="ConversationID" type="UUID" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) used to correlate responses with requests, to be generated when composing the first message in a conversation and subsequently copied from the original message to each reply message.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="Precedence" type="MessagePrecedence" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Indication of the importance and impact of the message: Routine, Transactional or Critical. Used to determine time-out values during message exchange and the level of error notification used in the sending implementation.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="ValidUntil" type="xs:dateTime">
```

```
<xs:annotation>
```

<xs:documentation>Optional absolute date and time (ISO 8601 formatted, including time zone) this message expires. Used by implementations to determine if a pending message should still be processed.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
</xs:complexType>
```

```
</xs:element>
```

```
<xs:element name="PTU">
```

```
<xs:annotation>
```

<xs:documentation>The PTU element represents one or more Program Time Units and is used by Prognosis and Flex-related messages. This element is always included as part of a message and never transmitted by itself. |

Each PTU includes a Power value, the sign of which is used to distinguish between production and consumption, from the perspective of the Prosumer. A positive value indicates that power flows towards the Prosumer (consumption), a negative value indicates flow towards the grid (production). |

For FlexRequests, the single Power value is insufficient, since it would be impossible to make a distinction between a request for the reduction of the amount of energy produced (two minus signs, thus a positive value) and the indication of room for more consumption (a positive value as well). Hence, each PTU also includes an indicator, Disposition, to distinguish between these situations. |

Note that a request is always relative to an earlier prognosis, and the intent of the party sending the request can only be determined by taking into account whether the net outcome of the PTU is expected to be production or consumption. |

Also note that all scenarios have valid alternative realizations: a reduction in production by 100 can also be accomplished by an increase in consumption by that amount. The resulting flexibility offer for the PTU will have the same Power value in both cases.

```

</xs:documentation>
</xs:annotation>
<xs:complexType>
  <xs:attribute name="Disposition" type="Disposition-AvailableRequested" use="optional">
    <xs:annotation>
      <xs:documentation>Optional, used only for FlexRequest messages: indication whether the Power
specified for this PTU represents available capacity or a request for reduction/increase.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="Power" type="xs:integer" use="required">
    <xs:annotation>
      <xs:documentation>Power specified for this PTU in Watts. Also see the important notes about
the sign of this attribute in the main documentation entry for the PTU element.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="Start" type="xs:integer" use="required">
    <xs:annotation>
      <xs:documentation>Number of the first PTU this element refers to. The first PTU of a day has
number 1.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="Duration" type="xs:integer" use="optional" default="1">
    <xs:annotation>
      <xs:documentation>The number of the PTUs this element represents. Optional, default value is
1.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="Price" type="CurrencyAmount" use="optional">
    <xs:annotation>
      <xs:documentation>The price offered or accepted for supplying the indicated amount of flexibility
in this PTU. Only valid for FlexOffer and FlexOrder messages; the currency associated with this amount is
included in the main part of those messages.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>
</xs:element>
<!-->
<xs:element name="FlexOrder">

```

<xs:annotation>

<xs:documentation>FlexOrder messages are used by DSOs and BRPs to purchase flexibility from an Aggregator based on a previous FlexOffer. A FlexOrder message contains a list of PTUs, with, for each PTU, the change in consumption or production to be realized by the Aggregator, plus the accepted price to be paid by the DSO or BRP for this amount of flexibility. This PTU list should be copied from the FlexOffer message without modification: Aggregator implementations will (and must) reject FlexOrder messages where the PTU list is not exactly the same as offered.</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:complexContent>

<xs:extension base="FlexBase">

<xs:attribute name="FlexOfferOrigin" type="InternetDomain" use="required">

<xs:annotation>

<xs:documentation>The Internet domain of the USEF participant that sent the FlexOffer this order is based on.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="FlexOfferSequence" type="xs:long" use="required">

<xs:annotation>

<xs:documentation>Sequence number of the FlexOffer message this order is based on. The combination of FlexOfferOrigin and FlexOfferSequence should be unique.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="Currency" type="ISO4217Currency" use="required">

<xs:annotation>

<xs:documentation>ISO 4217 code indicating the currency that applies to the prices listed for each PTU.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="OrderReference" type="xs:string" use="required">

<xs:annotation>

<xs:documentation>Order number assigned by the BRP or DSO originating the FlexOrder. To be stored by the Aggregator and used in the settlement phase.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:extension>

</xs:complexContent>

</xs:complexType>



</xs:element>

</xs:schema>

### 8.1.6 FlexOrderResponse.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
  <xs:simpleType name="InternetDomain">
```

```
    <xs:restriction base="xs:string">
```

```
      <xs:pattern value="([a-z0-9]+(-[a-z0-9]+)*\.)+[a-z]{2,}" />
```

```
    </xs:restriction>
```

```
  </xs:simpleType>
```

```
  <xs:simpleType name="USEF-Role">
```

```
    <xs:restriction base="xs:string">
```

```
      <xs:enumeration value="AGR" />
```

```
      <xs:enumeration value="BRP" />
```

```
      <xs:enumeration value="CRO" />
```

```
      <xs:enumeration value="DSO" />
```

```
      <xs:enumeration value="MDC" />
```

```
    </xs:restriction>
```

```
  </xs:simpleType>
```

```
  <xs:simpleType name="UUID">
```

```
    <xs:restriction base="xs:string">
```

```
      <xs:pattern value="[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}" />
```

```
    </xs:restriction>
```

```
  </xs:simpleType>
```

```
  <xs:simpleType name="MessagePrecedence">
```

```
    <xs:restriction base="xs:string">
```

```
      <xs:enumeration value="Critical" />
```

```
      <xs:enumeration value="Routine" />
```

```
      <xs:enumeration value="Transactional" />
```

```
    </xs:restriction>
```

```
  </xs:simpleType>
```

```
  <xs:simpleType name="Disposition-AvailableRequested">
```

```
    <xs:restriction base="xs:string">
```

```
      <xs:enumeration value="Available" />
```

```
      <xs:enumeration value="Requested" />
```

```
    </xs:restriction>
```

```
  </xs:simpleType>
```

```
<xs:simpleType name="Disposition-AcceptedRejected">
```

```
  <xs:restriction base="xs:string">
```

```
    <xs:enumeration value="Accepted" />
```

```
    <xs:enumeration value="Rejected" />
```

```
  </xs:restriction>
```

```
</xs:simpleType>
```

```
<xs:element name="MessageMetadata">
```

```
  <xs:annotation>
```

**<xs:documentation>**The MessageMetadata element contains mandatory metadata which is common for each non-wrapper USEF message. This element is always included as part of a message and never transmitted by itself.**</xs:documentation>**

```
  </xs:annotation>
```

```
  <xs:complexType>
```

```
    <xs:attribute name="SenderDomain" type="InternetDomain" use="required">
```

```
      <xs:annotation>
```

**<xs:documentation>**The Internet domain of the USEF participant sending this message. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid. When replying to this message, this attribute, combined with the SenderRole, is used to look up the USEF endpoint the reply message should be delivered to.**</xs:documentation>**

```
      </xs:annotation>
```

```
    </xs:attribute>
```

```
    <xs:attribute name="SenderRole" type="USEF-Role" use="required">
```

```
      <xs:annotation>
```

**<xs:documentation>**USEF role of the participant sending this message: AGR, BRP, CRO, DSO or MDC. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid.**</xs:documentation>**

```
      </xs:annotation>
```

```
    </xs:attribute>
```

```
    <xs:attribute name="RecipientDomain" type="InternetDomain" use="required">
```

```
      <xs:annotation>
```

**<xs:documentation>**Internet domain of the participant this message is intended for. When sending a message, this attribute, combined with the RecipientRole, is used to look up the USEF endpoint the message should be delivered to.**</xs:documentation>**

```
      </xs:annotation>
```

```
    </xs:attribute>
```

```
    <xs:attribute name="RecipientRole" type="USEF-Role" use="required">
```

```
      <xs:annotation>
```

```

    <xs:documentation>USEF role of the participant this message is intended for: AGR, BRP, CRO, DSO
or MDC.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="TimeStamp" type="xs:dateTime" use="required">
  <xs:annotation>
    <xs:documentation>Date and time this message was created, including the time zone (ISO 8601
formatted as per http://www.w3.org/TR/NOTE-datetime).</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="MessageID" type="UUID" use="required">
  <xs:annotation>
    <xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) for this message, to be
generated when composing each message.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="ConversationID" type="UUID" use="required">
  <xs:annotation>
    <xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) used to correlate re-
sponses with requests, to be generated when composing the first message in a conversation and subse-
quently copied from the original message to each reply message.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="Precedence" type="MessagePrecedence" use="required">
  <xs:annotation>
    <xs:documentation>Indication of the importance and impact of the message: Routine, Transac-
tional or Critical. Used to determine time-out values during message exchange and the level of error notifi-
cation used in the sending implementation.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="ValidUntil" type="xs:dateTime">
  <xs:annotation>
    <xs:documentation>Optional absolute date and time (ISO 8601 formatted, including time zone)
this message expires. Used by implementations to determine if a pending message should still be pro-
cessed.</xs:documentation>
  </xs:annotation>
</xs:attribute>
</xs:complexType>

```

```
</xs:element>
```

```
<xs:element name="FlexOrderResponse">
```

```
  <xs:annotation>
```

```
    <xs:documentation>Upon receiving and processing a FlexOrder message, the receiving implementation must reply with a FlexOrderResponse, indicating whether the update was handled successfully. FlexOrderResponse messages must always be sent with Precedence=Critical.</xs:documentation>
```

```
  </xs:annotation>
```

```
  <xs:complexType>
```

```
    <xs:sequence>
```

```
      <xs:element ref="MessageMetadata" />
```

```
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
```

```
    </xs:sequence>
```

```
    <xs:attribute name="Sequence" type="xs:long" use="required">
```

```
      <xs:annotation>
```

```
        <xs:documentation>Sequence number of the FlexOrder that has just been received.</xs:documentation>
```

```
      </xs:annotation>
```

```
    </xs:attribute>
```

```
    <xs:attribute name="Result" type="Disposition-AcceptedRejected" use="required">
```

```
      <xs:annotation>
```

```
        <xs:documentation>Indication whether the order was accepted or rejected. Rejection is only allowed in case the FlexOrder was already accepted previously, can not be found, or does not exactly match the contents of the corresponding FlexOffer.</xs:documentation>
```

```
      </xs:annotation>
```

```
    </xs:attribute>
```

```
    <xs:attribute name="Message" type="xs:string" use="optional">
```

```
      <xs:annotation>
```

```
        <xs:documentation>In case the order was rejected, this attribute must contain a human-readable description of the reason.</xs:documentation>
```

```
      </xs:annotation>
```

```
    </xs:attribute>
```

```
  </xs:complexType>
```

```
</xs:element>
```

```
</xs:schema>
```

### 8.1.7 FlexOfferRevocation.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
  <xs:simpleType name="InternetDomain">
```

```

<xs:restriction base="xs:string">
  <xs:pattern value="([a-z0-9]+(-[a-z0-9]+)*\.)+[a-z]{2,}" />
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="USEF-Role">
  <xs:restriction base="xs:string">
    <xs:enumeration value="AGR" />
    <xs:enumeration value="BRP" />
    <xs:enumeration value="CRO" />
    <xs:enumeration value="DSO" />
    <xs:enumeration value="MDC" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="UUID">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="MessagePrecedence">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Critical" />
    <xs:enumeration value="Routine" />
    <xs:enumeration value="Transactional" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Disposition-AvailableRequested">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Available" />
    <xs:enumeration value="Requested" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="CurrencyAmount">
  <xs:restriction base="xs:decimal">
    <xs:fractionDigits value="4" />
  </xs:restriction>
</xs:simpleType>

```

```
<xs:element name="MessageMetadata">
```

```
  <xs:annotation>
```

    <xs:documentation>The MessageMetadata element contains mandatory metadata which is common for each non-wrapper USEF message. This element is always included as part of a message and never transmitted by itself.</xs:documentation>

```
  </xs:annotation>
```

```
  <xs:complexType>
```

```
    <xs:attribute name="SenderDomain" type="InternetDomain" use="required">
```

```
      <xs:annotation>
```

        <xs:documentation>The Internet domain of the USEF participant sending this message. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid. When replying to this message, this attribute, combined with the SenderRole, is used to look up the USEF endpoint the reply message should be delivered to.</xs:documentation>

```
      </xs:annotation>
```

```
    </xs:attribute>
```

```
    <xs:attribute name="SenderRole" type="USEF-Role" use="required">
```

```
      <xs:annotation>
```

        <xs:documentation>USEF role of the participant sending this message: AGR, BRP, CRO, DSO or MDC. When receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise, the message must be rejected as invalid.</xs:documentation>

```
      </xs:annotation>
```

```
    </xs:attribute>
```

```
    <xs:attribute name="RecipientDomain" type="InternetDomain" use="required">
```

```
      <xs:annotation>
```

        <xs:documentation>Internet domain of the participant this message is intended for. When sending a message, this attribute, combined with the RecipientRole, is used to look up the USEF endpoint the message should be delivered to.</xs:documentation>

```
      </xs:annotation>
```

```
    </xs:attribute>
```

```
    <xs:attribute name="RecipientRole" type="USEF-Role" use="required">
```

```
      <xs:annotation>
```

        <xs:documentation>USEF role of the participant this message is intended for: AGR, BRP, CRO, DSO or MDC.</xs:documentation>

```
      </xs:annotation>
```

```
    </xs:attribute>
```

```
    <xs:attribute name="TimeStamp" type="xs:dateTime" use="required">
```

```
      <xs:annotation>
```

        <xs:documentation>Date and time this message was created, including the time zone (ISO 8601

formatted as per <http://www.w3.org/TR/NOTE-datetime>).</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="MessageID" type="UUID" use="required">

<xs:annotation>

<xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) for this message, to be generated when composing each message.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="ConversationID" type="UUID" use="required">

<xs:annotation>

<xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) used to correlate responses with requests, to be generated when composing the first message in a conversation and subsequently copied from the original message to each reply message.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="Precedence" type="MessagePrecedence" use="required">

<xs:annotation>

<xs:documentation>Indication of the importance and impact of the message: Routine, Transactional or Critical. Used to determine time-out values during message exchange and the level of error notification used in the sending implementation.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="ValidUntil" type="xs:dateTime">

<xs:annotation>

<xs:documentation>Optional absolute date and time (ISO 8601 formatted, including time zone) this message expires. Used by implementations to determine if a pending message should still be processed.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:complexType>

</xs:element>

<xs:element name="PTU">

<xs:annotation>

<xs:documentation>The PTU element represents one or more Program Time Units and is used by Prognosis and Flex-related messages. This element is always included as part of a message and never transmitted by itself. |

Each PTU includes a Power value, the sign of which is used to distinguish between production and consumption, from the perspective of the Prosumer. A positive value indicates that power flows towards the Prosumer (consumption), a negative value indicates flow towards the grid (production). |

For FlexRequests, the single Power value is insufficient, since it would be impossible to make a distinction between a request for the reduction of the amount of energy produced (two minus signs, thus a positive value) and the indication of room for more consumption (a positive value as well). Hence, each PTU also includes an indicator, Disposition, to distinguish between these situations. |

Note that a request is always relative to an earlier prognosis, and the intent of the party sending the request can only be determined by taking into account whether the net outcome of the PTU is expected to be production or consumption. |

Also note that all scenarios have valid alternative realizations: a reduction in production by 100 can also be accomplished by an increase in consumption by that amount. The resulting flexibility offer for the PTU will have the same Power value in both cases.

```

</xs:documentation>
</xs:annotation>
<xs:complexType>
  <xs:attribute name="Disposition" type="Disposition-AvailableRequested" use="optional">
    <xs:annotation>
      <xs:documentation>Optional, used only for FlexRequest messages: indication whether the Power
specified for this PTU represents available capacity or a request for reduction/increase.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="Power" type="xs:integer" use="required">
    <xs:annotation>
      <xs:documentation>Power specified for this PTU in Watts. Also see the important notes about
the sign of this attribute in the main documentation entry for the PTU element.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="Start" type="xs:integer" use="required">
    <xs:annotation>
      <xs:documentation>Number of the first PTU this element refers to. The first PTU of a day has
number 1.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="Duration" type="xs:integer" use="optional" default="1">
    <xs:annotation>

```



`<xs:documentation>`The number of the PTUs this element represents. Optional, default value is 1.`</xs:documentation>`

`</xs:annotation>`

`</xs:attribute>`

`<xs:attribute name="Price" type="CurrencyAmount" use="optional">`

`<xs:annotation>`

`<xs:documentation>`The price offered or accepted for supplying the indicated amount of flexibility in this PTU. Only valid for FlexOffer and FlexOrder messages; the currency associated with this amount is included in the main part of those messages.`</xs:documentation>`

`</xs:annotation>`

`</xs:attribute>`

`</xs:complexType>`

`</xs:element>`

`<xs:element name="FlexOfferRevocation">`

`<xs:annotation>`

`<xs:documentation>`The FlexOfferRevocation message is used by the Aggregator to revoke a FlexOffer previously sent to a DSO or BRP. It voids the FlexOffer, even if its validity time has not yet expired, even if a FlexOrder has already been issued based on this offer. The FlexOffer should exist and have been previously acknowledged, though, and may NOT apply to a period of which one PTU is already in the operate phase.`</xs:documentation>`

`</xs:annotation>`

`<xs:complexType>`

`<xs:sequence>`

`<xs:element ref="MessageMetadata"/>`

`<xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>`

`</xs:sequence>`

`<xs:attribute name="Sequence" type="xs:long" use="required">`

`<xs:annotation>`

`<xs:documentation>`Sequence number of the FlexOffer message that is being revoked: this FlexOffer must have been accepted previously.`</xs:documentation>`

`</xs:annotation>`

`</xs:attribute>`

`</xs:complexType>`

`</xs:element>`

`</xs:schema>`

### 8.1.8 FlexOfferRevocationResponse.xsd

`<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">`

```

<xs:simpleType name="InternetDomain">
  <xs:restriction base="xs:string">
    <xs:pattern value="([a-z0-9]+(-[a-z0-9]+)*\.)+[a-z]{2,}" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="USEF-Role">
  <xs:restriction base="xs:string">
    <xs:enumeration value="AGR" />
    <xs:enumeration value="BRP" />
    <xs:enumeration value="CRO" />
    <xs:enumeration value="DSO" />
    <xs:enumeration value="MDC" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="UUID">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="MessagePrecedence">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Critical" />
    <xs:enumeration value="Routine" />
    <xs:enumeration value="Transactional" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Disposition-AvailableRequested">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Available" />
    <xs:enumeration value="Requested" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Disposition-AcceptedRejected">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Accepted" />
    <xs:enumeration value="Rejected" />
  </xs:restriction>

```

```

</xs:restriction>
</xs:simpleType>
<xs:simpleType name="CurrencyAmount">
  <xs:restriction base="xs:decimal">
    <xs:fractionDigits value="4" />
  </xs:restriction>
</xs:simpleType>
<xs:element name="MessageMetadata">
  <xs:annotation>
    <xs:documentation>The MessageMetadata element contains mandatory metadata which is common
for each non-wrapper USEF message. This element is always included as part of a message and never
transmitted by itself.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="SenderDomain" type="InternetDomain" use="required">
      <xs:annotation>
        <xs:documentation>The Internet domain of the USEF participant sending this message. When
receiving a message, its value should match the value specified in the SignedMessage wrapper: otherwise,
the message must be rejected as invalid. When replying to this message, this attribute, combined with the
SenderRole, is used to look up the USEF endpoint the reply message should be delivered
to.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="SenderRole" type="USEF-Role" use="required">
      <xs:annotation>
        <xs:documentation>USEF role of the participant sending this message: AGR, BRP, CRO, DSO or
MDC. When receiving a message, its value should match the value specified in the SignedMessage wrapper:
otherwise, the message must be rejected as invalid.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="RecipientDomain" type="InternetDomain" use="required">
      <xs:annotation>
        <xs:documentation>Internet domain of the participant this message is intended for. When
sending a message, this attribute, combined with the RecipientRole, is used to look up the USEF endpoint the
message should be delivered to.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="RecipientRole" type="USEF-Role" use="required">
      <xs:annotation>

```

```

    <xs:documentation>USEF role of the participant this message is intended for: AGR, BRP, CRO, DSO
or MDC.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="TimeStamp" type="xs:dateTime" use="required">
  <xs:annotation>
    <xs:documentation>Date and time this message was created, including the time zone (ISO 8601
formatted as per http://www.w3.org/TR/NOTE-datetime).</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="MessageID" type="UUID" use="required">
  <xs:annotation>
    <xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) for this message, to be
generated when composing each message.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="ConversationID" type="UUID" use="required">
  <xs:annotation>
    <xs:documentation>Unique identifier (UUID/GUID as per IETF RFC 4122) used to correlate
responses with requests, to be generated when composing the first message in a conversation and
subsequently copied from the original message to each reply message.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="Precedence" type="MessagePrecedence" use="required">
  <xs:annotation>
    <xs:documentation>Indication of the importance and impact of the message: Routine,
Transactional or Critical. Used to determine time-out values during message exchange and the level of error
notification used in the sending implementation.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="ValidUntil" type="xs:dateTime">
  <xs:annotation>
    <xs:documentation>Optional absolute date and time (ISO 8601 formatted, including time zone)
this message expires. Used by implementations to determine if a pending message should still be
processed.</xs:documentation>
  </xs:annotation>
</xs:attribute>
</xs:complexType>

```

```
</xs:element>
```

```
<xs:element name="PTU">
```

```
<xs:annotation>
```

`<xs:documentation>`The PTU element represents one or more Program Time Units and is used by Prognosis and Flex-related messages. This element is always included as part of a message and never transmitted by itself. |

Each PTU includes a Power value, the sign of which is used to distinguish between production and consumption, from the perspective of the Prosumer. A positive value indicates that power flows towards the Prosumer (consumption), a negative value indicates flow towards the grid (production). |

For FlexRequests, the single Power value is insufficient, since it would be impossible to make a distinction between a request for the reduction of the amount of energy produced (two minus signs, thus a positive value) and the indication of room for more consumption (a positive value as well). Hence, each PTU also includes an indicator, Disposition, to distinguish between these situations. |

Note that a request is always relative to an earlier prognosis, and the intent of the party sending the request can only be determined by taking into account whether the net outcome of the PTU is expected to be production or consumption. |

Also note that all scenarios have valid alternative realizations: a reduction in production by 100 can also be accomplished by an increase in consumption by that amount. The resulting flexibility offer for the PTU will have the same Power value in both cases.

```
</xs:documentation>
```

```
</xs:annotation>
```

```
<xs:complexType>
```

```
<xs:attribute name="Disposition" type="Disposition-AvailableRequested" use="optional">
```

```
<xs:annotation>
```

`<xs:documentation>`Optional, used only for FlexRequest messages: indication whether the Power specified for this PTU represents available capacity or a request for reduction/increase.`</xs:documentation>`

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="Power" type="xs:integer" use="required">
```

```
<xs:annotation>
```

`<xs:documentation>`Power specified for this PTU in Watts. Also see the important notes about the sign of this attribute in the main documentation entry for the PTU element.`</xs:documentation>`

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="Start" type="xs:integer" use="required">
```

```

<xs:annotation>
  <xs:documentation>Number of the first PTU this element refers to. The first PTU of a day has
number 1.</xs:documentation>
</xs:annotation>
</xs:attribute>
<xs:attribute name="Duration" type="xs:integer" use="optional" default="1">
  <xs:annotation>
    <xs:documentation>The number of the PTUs this element represents. Optional, default value is
1.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="Price" type="CurrencyAmount" use="optional">
  <xs:annotation>
    <xs:documentation>The price offered or accepted for supplying the indicated amount of flexibility
in this PTU. Only valid for FlexOffer and FlexOrder messages; the currency associated with this amount is
included in the main part of those messages.</xs:documentation>
  </xs:annotation>
</xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="FlexOfferRevocationResponse">
  <xs:annotation>
    <xs:documentation>Upon receiving and processing a FlexOfferRevocation message,
the receiving implementation must reply with a FlexOfferRevocationResponse, indicating whether the
revocation was handled successfully.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="MessageMetadata"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="Sequence" type="xs:long" use="required">
      <xs:annotation>
        <xs:documentation>Sequence number of the FlexOffer that a
revocation message was received for.</xs:documentation>
      </xs:annotation>
    </xs:attribute>

```

```
<xs:attribute name="Result" type="Disposition-AcceptedRejected" use="required">
```

```
<xs:annotation>
```

<xs:documentation>Indication whether the revocation was accepted or rejected. Rejection is only allowed in case the FlexOffer is unknown (it is the responsibility of the sending party not to revoke FlexOffer messages which have not yet been accepted) or if it applies to a period of which a PTU is already in the Operate phase (at which time USEF explicitly forbids revocation).</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
<xs:attribute name="Message" type="xs:string" use="optional">
```

```
<xs:annotation>
```

<xs:documentation>In case the revocation was rejected, this attribute must contain a human-readable description of the reason.</xs:documentation>

```
</xs:annotation>
```

```
</xs:attribute>
```

```
</xs:complexType>
```

```
</xs:element>
```

```
</xs:schema>
```

