

Title:	Document Version:
D4.2 WiseGRID Interoperable Platform Process (WG IOP)	1.0

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

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:
Giuseppa Caruso and Leandro Lombardo	ENG	WP4

Authors (organisation):

Giuseppa Caruso(ENG), Leandro Lombardo (ENG), Giampaolo Fiorentino (ENG), Antonello Corsi (ENG), Alberto Zambrano (ETRA), Álvaro Nofuentes (ETRA), Mihai Sanduleac (CRE), Paul Lacatus(CRE), Catalin Chimirel (CRE), Foivos Palaogiannis(ICCS), Dimitris Siagkas (ICCS), Antonis Papanikolaou (HYP), Alexandros Nikolakakis (HYP), Benjamin Kraft(VS), Amparo Mocholi (ITE)

Abstract:

This document report presents the results of the activities carry out in the T4.3 about the definition and implementation of the WiseGRID interoperable platform (WG IOP), designed on standards and interoperable technologies based on web services and high performance middleware specifications. In addition this document provide also an overview of the activities done in the T4.2 about the implementation of a tool which allows the simulation of different scenarios of interest to the DSO, based on the results of the D4.1.

Keywords:

Interoperable Platform, RabbitMQ, standards, algorithmic, simulator.



Revision History

Revision	Date	Description	Author (Organisation)
V0.1	02.03.2018	ToC	Giuseppa Caruso, Antonello Corsi, Giampaolo Fiorentino, Leandro Lombardo (ENG)
V0.2	15.03.2018	New ToC with a new chapter	Alberto Zambrano (ETRA), Foivos Palaogiannis (ICCS).
V0.3	26.03.2018	Added contribution inside the chapters 2, 3	Leandro Lombardo(ENG), Alexandros Nikolakakis (HYP)
V0.4	05.04.2018	New version	Leandro Lombardo (ENG), Alberto Zambrano (ETRA), Mihai Sanduleac (CRE), Foivos Palaogiannis (ICCS),
V0.5	17.04.2018	Added last contribution in the different chapters. Version ready for internal review	Giuseppa Caruso , Leandro Lombardo (ENG), Alberto Zambrano (ETRA), Paul Lacatus (CRE), Foivos Palaogiannis (ICCS)
V0.6	20.04.2018	Version after peer review	Giuseppa Caruso (ENG), Alberto Zambrano (ETRA), Álvaro Nofuentes (ETRA), Catalin Chimirel (CRE), Foivos Palaogiannis (ICCS)
V1.0	24.04.2018	Addressed comments from peer review. Version ready for submission	Giuseppa Caruso, Leandro lombardo (ENG)

INDEX

EXECUTIVE SUMMARY	10
1 INTRODUCTION	12
1.1 Purpose of the Document.....	12
1.2 Scope of the Document	12
1.3 Structure of the Document	13
2 OVERVIEW ARCHITECTURE.....	14
3 WG IOP FRAMEWORK.....	18
3.1 BROKER	18
3.2 MQTT	23
3.3 AMQP.....	23
4 STANDARDS.....	24
4.1 USEF	24
4.2 CIM.....	27
4.3 openAdR.....	28
5 WG IOP INTERACTIONS WITH OTHER WG TOOLS.....	29
5.1 WG Cockpit.....	29
5.1.1 Messages exchanged by means of WG IOP & standards	29
5.1.2 Messages structure	31
5.2 WG RESCO	31
5.2.1 Messages exchanged by means of WG IOP & standards	31
5.2.2 Messages structure	32
5.3 WG StaaS/VPP	32
5.3.1 Messages exchanged by means of WG IOP & standards	32
5.3.2 Messages structure	33
5.4 WISEEVP	33
5.4.1 Messages exchanged by means of WG IOP & standards	33
5.4.2 Messages structure	34
5.5 WiseCOOP	34

5.5.1	Messages exchanged by means of WG IOP & standards	34
5.5.2	Messages structure	35
5.6	WG CORP	35
5.6.1	Messages exchanged by means of WG IOP & standards	35
5.6.2	Messages structure	36
5.7	Demand Response & WiseHOME	36
5.7.1	Messages exchanged by means of WG IOP & standards	36
5.7.2	Messages structure	37
6	WG IOP MICRO-SERVICES.....	37
6.1	Real Time (RT) Monitoring.....	37
6.1.1	EVSE wrapper	37
6.1.2	EV Wrapper.....	38
6.1.3	Wrappers for energy and grid monitoring	39
6.1.3.1	Unbundled Smart Meters (SMX) inside WiseGRID project	39
6.1.3.2	Integration of AMI systems	43
6.1.3.2.1	SITEL STGTedis AMI (Crevillent)	43
6.1.3.2.2	LoRa Gateway (Flanders)	45
6.1.3.2.3	Terni	45
6.1.3.2.4	Mesogia.....	46
6.1.3.3	SCADA.....	47
6.1.4	Battery wrapper.....	48
6.1.5	Domestic/building Assets wrapper.....	48
6.2	Ancillary Services Market	49
6.2.1	Flexibility requests generation	50
6.2.2	Offer selection	53
6.2.3	Offer revocation	55
6.2.4	DSO Flex. Market hub.....	55
6.2.5	Aggr. Flex. Market hub	56
6.3	Weather Forecasting	56
6.3.1	Weather forecast.....	56
6.3.2	Current weather	57
6.4	Energy Forecasting	58
6.5	Tariff provider.....	60
6.5.1	Communication mechanism.....	60
6.5.2	Tariff model	60
6.5.2.1	Rule.....	60

6.5.2.1.1	Definition	60
6.5.2.1.2	Examples	62
6.5.2.2	Plan	63
6.5.2.2.1	Definition	64
6.5.3	Implemented services	66
6.5.3.1	Access to tariff list	66
6.5.3.2	Access to tariff prices	67
6.5.3.3	Access to tariff model.....	68
6.6	Wholesale market prices provider	71
6.6.1	Spot prices	71
6.6.2	Implementation specific per pilot site.....	71
6.6.2.1	Crevillent.....	71
6.6.2.1.1	PVPC Prices (Spain)	72
6.6.2.2	Flanders	73
6.6.2.3	Terni.....	73
6.6.2.4	Mesogia and Kythnos	73
7	ALGORITHMIC FRAMEWORK FOR OPTIMIZING MULTI-OBJECTIVE ENERGY SYSTEMS .	74
7.1	Description	74
7.2	Algorithm.....	75
7.2.1	Objective Function.....	75
7.2.2	Distribution Network Constraints.....	75
7.2.3	Flexible Resources	75
7.2.4	Distributed Generation Units	75
7.2.5	Storage Units	76
7.2.6	CHP Units	76
7.3	GAMS model.....	76
7.4	Simulator interface.....	76
7.5	Practical demonstration	82
7.5.1	Model.....	82
7.5.2	Results	86
8	CONCLUSIONS	87
9	REFERENCES AND ACRONYMS.....	89
9.1	References	89
9.2	Acronyms.....	91

10 ANNEX A.....	92
10.1 RESCO Tool Messages Structure	92
10.2 STAAS/VPP Tool Messages Structure.....	95
10.3 Demand Response & WiseHOME messages structure	97
10.4 EVSE Wrapper messages.....	106
10.4.1 Messages initiated by EVSE Wrapper	106
10.4.2 Messages initiated by EVSE Scheduler	119
10.5 EV Wrapper messages	124
10.6 DLMS/COSEM publications of SMX to WG IOP example	126
10.7 Translation to CIM MeterReading Objects message.....	128
10.8 Battery Wrapper messages.....	130
10.9 Ancillary Services Market message specification	133
10.9.1 FlexRequest.xsd	133
10.9.2 FlexRequestResponse.xsd	140
10.9.3 FlexOffer.xsd	144
10.9.4 FlexOfferResponse.xsd	151
10.9.5 FlexOrder.xsd	155
10.9.6 FlexOrderResponse.xsd	162
10.9.7 FlexOfferRevocation.xsd	166
10.9.8 FlexOfferRevocationResponse.xsd	171

LIST OF FIGURES

Figure 1 – WG IOP architecture	15
Figure 2 – Authorization control security role example	16
Figure 3 – Authorization Access log.....	17
Figure 4 – Login Page.....	19
Figure 5 – Overview Page	19
Figure 6 – Connections section.....	20
Figure 7 – Queue Manage Page.....	21
Figure 8 – Exchange Manage Page	22
Figure 9 – Main difference between MQTT and AMQP protocol	24
Figure 10 – USEF Protocol Business Framework [7]	26
Figure 11 – Example of the CIM Data Model [10].	27
Figure 12 – OpenADR communication architecture schema [14]	29
Figure 13 – Overview of applications involved in the Ancillary Services Market.....	30
Figure 14 – Overview of communication flows from EVSEs to WiseEVP	37
Figure 15 – Detail of protocols used for communication with EVSEs	38
Figure 16 – Overview of the communication flows for EV monitoring	39
Figure 17 – Unbundled Smart Meter architecture	40
Figure 18 – Each USM is subscribing to the broker of the Smart Meter Wrapper (SMW)	41
Figure 19 – Smart Meter wrapper subscribing to each USM	41
Figure 20 – SMX Wrappers communication schema	42
Figure 21 – DLMS to CIM translator	43
Figure 22 – AMI Wrappers communication schema	43
Figure 23 – Integration scheme of AMI system for Crevillent pilot site.....	45
Figure 24 – Initial connection of the meters in Terni to the AMR system	46
Figure 25 – Functional connection of SMX to the meters in Terni and to the different actors	46
Figure 26 – SCADA Wrappers communication schema.....	47
Figure 27 – Overview of communication between batteries and WG StaaS/VPP via WG IOP	48
Figure 28 – Building assets integration architecture.....	49
Figure 29 – BMS wrapper architecture	49
Figure 30 – Flexibility requests generation	50
Figure 31 – Reception of flexibility offers.....	51
Figure 32 – Offer selection	53
Figure 33 – Offer revocation.....	55

Figure 34 – Forecast server structure integration schema	59
Figure 35 – Implementation of the Wholesale Market Prices provider module for Spain	72
Figure 36 – Implementation of the Wholesale Market Prices provider module for Greece	74
Figure 37 – Architecture of the simulator implementation	77
Figure 38 – Scenario creation	78
Figure 39 – Example of wind units model representation	80
Figure 40 – Patterns.....	81
Figure 41 – Summary of issues identified in the results of the simulated scenario.....	81
Figure 42 – Example of voltage magnitude representation with identified overvoltage	82
Figure 43 – Example of power flow representation with identified capacity threshold overpassed	82
Figure 44 – Topology of the example scenario	82
Figure 45 – Partial graph representation of the network generated automatically by the simulator.....	83
Figure 46 – Load curves for bus 2.....	84
Figure 47 – Overview of the characteristics modelled for the Wind units connected to bus 4.....	85
Figure 48 – Overview of the characteristics modelled for the Flexible Loads connected to bus 3.....	86
Figure 49 – Overview of voltage magnitude for bus 1	86
Figure 50 – Overview of power flows involving bus 2.....	87

LIST OF TABLES

Table 1 – WG RESCO main Standards.....	31
Table 2 – STaaS/VPP main Standards	32
Table 3 – Flexibility request properties	50
Table 4 – Flexibility offer properties.....	52
Table 5 – Flexibility order properties.....	53
Table 6 – Flexibility offer revocation properties	55
Table 7 – Rule objects.....	60
Table 8 – Plan object	64
Table 9 – DSO data	78
Table 10 – Line operational limits	83
Table 11 – Buses connected	84
Table 12 – List of Acronyms.....	91

EXECUTIVE SUMMARY

In the new smart energy environment different business actors and models operate using different technology tools. In same case those tools need to exchange data each other. To achieve this exchange of data, a specific technology solution is necessary. Inside WiseGRID, the WG IOP is the tool that provides a scalable, secure and open ICT platform, with interoperable interfaces, for real-time monitoring and decentralized control to support effective operation of the energy network.

The objective of the platform is to manage and process the heterogeneous and massive data stream coming from the distributed energy infrastructure deployed. This platform should enable new services and reduce ICT costs for prosumers and smaller players, whilst it will facilitate cross-network and cross-entity interoperability. Inside WiseGRID project, WG IOP will enable the cooperation and synergies among the different actors targeted by the different WiseGRID technological solutions. It facilitates complex coordination among devices (installations) connected to the distribution grid, offering the key enabling tool that allows application of advanced methods for distribution grid management while exploiting to the fullest highest degree the capabilities offered by the various types of resources (generating capacities) connected to the distribution grid.

In this perspective, the WG IOP architecture, as it is presented in the Figure 1, has been designed to allow the interaction of all the functionalities offered by the platform, from the message exchange among WG tools and services (so called micro-services) developed specifically inside the WiseGRID project through specific data wrappers able to translate complex messages into common standard messages.

The core of the architecture is the RabbitMQ message broker by which all messages are flowing. The broker is agnostic about the content of the message and its structure, representing a sort of “mailman”.

For the above mentioned reason, identifying and analysing the main standards used by the tools, devices and the different assets involved in the ecosystem to be managed by the IOP, has been the starting point for the design and implementation of this interoperable platform. Each message that travels through the WG IOP among WiseGRID tools and microservices in order to guaranty a trusted and reliable interaction, has a structure based on standards or otherwise based on paths shared within the WiseGRID project. Those standards and message structures are described in more detail in this report in the dedicated sections, where also specific example are available.

In particular, the messages exchanged have been analysed and defined for the follow WiseGRID tools: WG Cockpit, WG RESCO, WG STaaS/VPP, WiseEVP, WiseCOOP, WiseCORP and WiseHOME.

Consequently the WG IOP has to handle several communications among those tools and at the same time has to manage several vertical communication flows with different type of devices like batteries, PVs, EVSEs and toward horizontal modules of the WiseGRID project, like Energy forecast provider module, ancillary market, tariff provision, whether forecasting and so on. For this reason, the WG IOP includes also some modules able to wrap messages coming from different field devices from their internal data model into a common standard data model, as for example CIM format.

These components typically offer their services according to the following schema:

- They have a wrapper that translates from their internal data model to the agreed data model (mainly CIM).
- They send requests to the WG IOP, without knowing the endpoint details of the component which will answer its requests. WG IOP Message Broker consistently forwards messages to the proper modules.
- They are listening to a queue where they will receive asynchronously the answers. These answers will go through the same wrapper to be converted to the internal data model.

- Publish/subscribe pattern is also possible: components will receive data asynchronously without the need of a request.

As part of the implementation of the Interoperable Platform, also methods for assuring the authorization and the authentication control as well as some privacy policies have been developed.

This report provides also an overview about the work carried out in the context of Task 4.2 “Simulator for the resource portfolio and for the domains to use the services”, which starting from the results of the Task 4.1 “Development of an algorithmic framework for optimizing Multi-Objective Energy Systems”, aims to implement a tool allowing the simulation of different scenarios of interest to the DSO. Consequently, the targeted user is the DSO operator, who is the actor responsible of maintaining the distribution grid operating under acceptable levels and ensuring the quality of the energy.

The implemented simulator therefore allows the simulation of scenarios considering:

- Demand and production data
- Flexibility offered by other actors of the grid, and the price associated to the activation of this flexibility

The main elements composing the simulation environment are:

- GAMS [1]: GAMS is a high-level modelling system for mathematical optimization. The algorithm developed in T4.1 has been implemented using this system.
- Message broker: element used to communicate GAMS with the simulator server. Messages exchanged between both are Excel files filled with all data necessary to trigger the simulation of a scenario and with the results of the simulations.
- Simulator server: hosts the application that allows the DSO operator to define scenarios to be simulated, as well as new demand/production/flexibility patterns.
- Pattern and simulation database: holds all the information required to create new simulation scenarios, as well as the results of already completed simulations.
- User interface: interaction point of the simulator with the DSO operator.

The workflow for defining and simulating a scenario is as follows:

1. The DSO operator creates a new *scenario*. A *scenario* is a set of data (load curves per bus, production curves per bus, flexibility available...) that models a certain situation of interest on the grid that will be simulated.
2. Once the scenario is created, the DSO operator can populate the necessary data in order to complete the model.
3. Once the scenario model is completed, the DSO operator can trigger the simulation. At that point, the model is transferred to GAMS and a new item under *Simulation results* menu is created.
4. Once the simulation is completed, results are reported by GAMS and can be visualized under the *Simulation results* menu.

1 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

Within WP4 “WiseGRID Interoperable Platform” under SP2, the WiseGRID Interoperable Platform (WG IOP) is developed based on state of the art technologies for grid and ICT operation. The WG IOP application will facilitate complex coordination among devices (installations) connected to the distribution grid, offering the key enabling tool that allows application of advanced methods for distribution grid management while fully exploiting the capabilities offered by the various types of resources (generating capacities) connected to the distribution grid.

All these developments would enable the Distribution System Operator (DSO), as the entity responsible for the smooth and efficient operation of the distribution grid, to efficiently use Distributed Energy Resources (DERs) installations in the distribution grid with the main purpose of avoiding congestions within the grid. Also, the diverse technical characteristics of various DERs, such as electric vehicles, demand response, distributed generation units (DGs), storage assets, allows their deployment in multiple ways, from which the DSO, and by extension all grid users, can benefit. Thus, the decisions making process on the DSO part is increasingly complex, determining the DSO to require an appropriate tool.

Additionally, other actors are emerging in the smart grid ecosystems like aggregators, RESCOs, fleet managers, facility managers, EVSE managers, Virtual Power Plant (VPP) that aim at operating in different way into the smart grid providing services for enabling an efficient use of the DERs. In that ecosystem also those actors can benefit of tools like the WG IOP where to build energy applications.

Currently, in the smart grid it is possible to gather several assets producing valuable information for several actors. Tools like WG IOP can enable those actors to interact with each other to achieve common benefit, facilitating the management of all these different data sources and the interaction of all actors.

Consequently, this document describes such a complex and very useful tool and the managed interactions.

It is obvious that WG IOP shall handle several vertical communication flows with different type of devices such as batteries, PVs, and EVSEs towards horizontal modules of the WiseGRID project, like forecast provider module, ancillary services and also between other WiseGRID tools. The WG IOP includes also some modules able to wrap messages coming from different field devices from their internal data model into a common standard data model as for example CIM format. The aim of this document is to describe the WG IOP ecosystem and give a quickly guide on how to use the functionalities provided by the system.

Finally, this document also presents micro-services (for real time monitoring) and the proposed algorithm (model, simulator and demonstration) for the optimization of operation of DSO. Objective Function to be optimized can be different as the needs of the DSO are for the specific operating time. Mainly four targets (functions) are presented as main criteria: the cost of generation and general transactions, the distribution grid losses, the state changes of switchgear and the voltage deviation from nominal value.

1.2 SCOPE OF THE DOCUMENT

This document report presents the hypothesis, rationale and development logic behind WG IOP, as designed on standards and interoperable technologies based on web services and high performance middleware specifications. WiseGRID IOP would operate as the binder interfacing all other WiseGRID tools and facilitating involved actors activity, including the distributed resources. On the other hand, the design focuses on efficiency constraints that stem from the fact that very many (thousands to millions) devices will be linked through the WiseGRID IOP. All these connections, standards, and messages among devices also tool by tool WiseGRID versus the IOP are presented.

1.3 STRUCTURE OF THE DOCUMENT

The document is structured starting from clearly defined Executive Summary and Introduction (Chapter 1 INTRODUCTION), presenting the Overview architecture (Chapter 2 OVERVIEW ARCHITECTURE) and going through various Frameworks (Chapter 3 WG IOP FRAMEWORK). Then the general Standards are described (Chapter 4 STANDARDS) and tool by tool the standards by which sets of information are exchanged by the WiseGRID tools and WG IOP and also the message structures considered (Chapter 5 WG IOP INTERACTIONS WITH OTHER WG TOOLS). As an example, CIM model, as one of the most extended standard within the smart grids, is going to be used for energy metering. Then (Chapter 6 WG IOP MICRO-SERVICES) are presented the micro-services within IOP like: Real time monitoring, Weather forecasting and Energy forecasting, Tariff provider and Wholesale market prices provider.

Last chapter (not considering the Conclusions), is an extensive presentation of the Algorithmic framework for optimizing Multi-Objective Energy Systems (modelling with equations of distributed generation and storage assets), focusing on the changes performed with regards to the description included in D4.1 and the new development of the Simulator tool, including a Practical demonstrator (Chapter 7 ALGORITHMIC FRAMEWORK FOR OPTIMIZING MULTI-OBJECTIVE ENERGY SYSTEMS). In the end are placed the conclusions (Chapter 8 CONCLUSIONS) of presented approach for the IOP and all other WiseGRID tools overall development.

2 OVERVIEW ARCHITECTURE

The WiseGRID Interoperable Platform (WG IOP) is a scalable, secure and open ICT platform, with interoperable interfaces, for real-time monitoring and decentralized control to support effective operation of the energy network. The objective of the platform is to manage and process the heterogeneous and massive data streams coming from the distributed energy infrastructure deployed. This platform will enable new services and reduce ICT costs for prosumers and smaller players, whilst it will facilitate cross-network and cross-entity interoperability.

In this perspective, the WG IOP architecture has been designed to allow the interaction of all the functionalities offered by the platform, from the Authentication and Authorization control to the message exchange between tools and services of the WiseGRID project through specific data wrappers able to translate complex messages into common standard messages. The core of the architecture is the RabbitMQ message broker by which all message are flowing. The broker is agnostic about the content of the message and its structure, it represents a sort of “mailman”. The messages that travel through the WG IOP with which the WiseGrid tools and services interact have a structure based on standards or otherwise based on paths shared within the WiseGRID project. The standard and message structure are better described in each dedicated section inside the document.

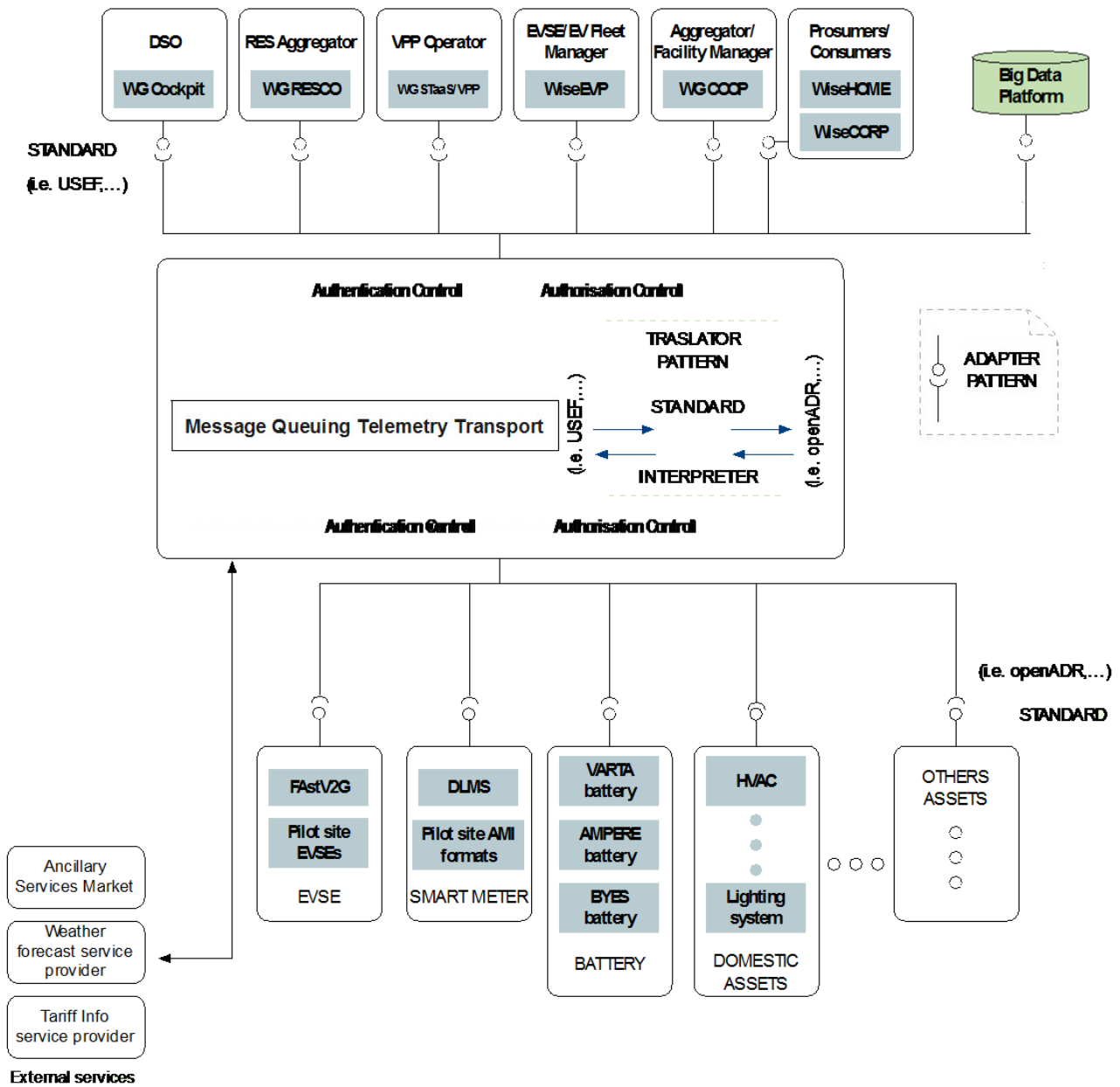


Figure 1 – WG IOP architecture

As depicted in the WG IOP Architecture image, several components (tools and services) are deployed around the WG IOP broker exchange core, each of these components provides individual features in a micro services fashion and make the WG IOP a complex and well-structured ecosystem

These components typically offer their services according the following schema:

- They have a wrapper that translates from their internal data model to the agreed common data model.

- They send requests to the WG IOP, without knowing the endpoint details of the component which will answer its requests. WG IOP Message Broker consistently forwards messages to the proper modules.
- They are listening to a queue where they will receive asynchronously the answers. These answers will go through the same wrapper to be converted to the internal data model.
- Publish/subscribe patterns is also possible: components will receive data asynchronously without the needing of a request

Currently the message exchange among those components and the WiseGRID tools via WG IOP is secured by two different security levels, the first one by an Authorization Control and the second one by Authentication Control.

- **Low level Authorization Control:**

One of the most important aspects of the WG IOP is to ensure an adequate level of security for the data exchanged through it. To do this, the system must not only verify access to the WG IOP through an authentication level, but allows only authorized resources to reach the system and so login to it.

If you were to ask a networking engineer how you would add a security layer to protect certain resources on a sensitive network, the answer would probably be with a dedicated firewall, or with an ACL (Access Control List). This is a simple rule on the network equipment that says that point A is allowed to communicate with point B only when it complies with specific criteria, such as:

- Source / Destination
- Protocol
- Port

The WG IOP authorization control does it by specific OpenStack security rule that are created for each of the partner that need to implement the dialog between their tools/micro services. The next image show one of the security role created for the ETRA partner that allow the source named ETRA_195.77.187.234 to connect to the destination IOP server for the TCP protocol at the 5672 and 1883 (the default ports for the AMQP and MQTT protocol).

645	2K	ETRA_195.77.187.234	PSMDC_109.232.32.221_WISEGRID	TCP TCP_5672 TCP ssh TCP TCP_27017 TCP tcp_1883	accept	Log
-----	----	---------------------	-------------------------------	--	--------	-----

Figure 2 – Authorization control security role example

The next image shows the log of the attempts to access the system by different IPs. The green ones are the authorized IPs, the red ones show the not authorized.

Time	Origin	Service	Source	Destination	Rule
12:04:41	PSMDCVSW2-1_PSMDCVSW2	TCP	60543	PSMDC_109.232.32.221_WISEGRID	1017
12:04:50	PSMDCVSW2-1_PSMDCVSW2	TCP	TCP_2323	PSMDC_109.232.32.221_WISEGRID	1017
12:05:19	PSMDCVSW2-1_PSMDCVSW2	gprs-data	5.188.10.2	PSMDC_109.232.32.221_WISEGRID	1017
12:05:23	PSMDCVSW2-1_PSMDCVSW2	TCP	7760	PSMDC_109.232.32.221_WISEGRID	1017
12:05:52	PSMDCVSW2-1_PSMDCVSW2	CP_SSL_Network_E...	134.119.182.6	PSMDC_109.232.32.221_WISEGRID	1017
12:06:21	PSMDCVSW2-1_PSMDCVSW2	TCP	45022	PSMDC_109.232.32.221_WISEGRID	1017
12:06:35	PSMDCVSW2-1_PSMDCVSW2	TCP	TCP_3335	PSMDC_109.232.32.221_WISEGRID	1017
12:06:44	PSMDCVSW2-1_PSMDCVSW2	TCP	TCP_27017	Eng-psm_151.78.73.132_Ad...	645
12:07:05	PSMDCVSW2-1_PSMDCVSW2	ssh	221.181.192.52	PSMDC_109.232.32.221_WISEGRID	1017
12:07:56	PSMDCVSW2-1_PSMDCVSW2	http-93	185.130.212.19	PSMDC_109.232.32.221_WISEGRID	1017
12:08:24	PSMDCVSW2-1_PSMDCVSW2	Sql_Sema_1433	103.228.152.46	PSMDC_109.232.32.221_WISEGRID	1017
12:09:36	PSMDCVSW2-1_PSMDCVSW2	TCP	4545	PSMDC_109.232.32.221_WISEGRID	1017
12:09:58	PSMDCVSW2-1_PSMDCVSW2	TCP	TCP_27017	PSMDC_109.232.32.221_WISEGRID	645
12:10:10	PSMDCVSW2-1_PSMDCVSW2	tcp_1883	EII_89.97.237.254_Palermo	PSMDC_109.232.32.221_WISEGRID	645
12:10:58	PSMDCVSW2-1_PSMDCVSW2	ssh	EII_89.97.237.254_Palermo	PSMDC_109.232.32.221_WISEGRID	645
12:11:08	PSMDCVSW2-1_PSMDCVSW2	TCP	TCP_5672	PSMDC_109.232.32.221_WISEGRID	645
12:11:18	PSMDCVSW2-1_PSMDCVSW2	TCP	45022	PSMDC_109.232.32.221_WISEGRID	1017
12:13:37	PSMDCVSW2-1_PSMDCVSW2	Sql_Sema_1433	39.83.234.76	PSMDC_109.232.32.221_WISEGRID	1017
12:13:55	PSMDCVSW2-1_PSMDCVSW2	mpsysrsvr	185.143.223.254	PSMDC_109.232.32.221_WISEGRID	1017
12:14:21	PSMDCVSW2-1_PSMDCVSW2	newton-dock	164.132.136.165	PSMDC_109.232.32.221_WISEGRID	1017
12:14:37	PSMDCVSW2-1_PSMDCVSW2	TCP	60543	PSMDC_109.232.32.221_WISEGRID	1017
12:14:48	PSMDCVSW2-1_PSMDCVSW2	TCP	3303	PSMDC_109.232.32.221_WISEGRID	1017
12:15:09	PSMDCVSW2-1_PSMDCVSW2	TCP	3839	PSMDC_109.232.32.221_WISEGRID	1017
12:15:18	PSMDCVSW2-1_PSMDCVSW2	iad1	185.232.29.199	PSMDC_109.232.32.221_WISEGRID	1017
12:15:54	PSMDCVSW2-1_PSMDCVSW2	TCP	6023	PSMDC_109.232.32.221_WISEGRID	1017
12:15:54	PSMDCVSW2-1_PSMDCVSW2	TCP	TCP_2022	PSMDC_109.232.32.221_WISEGRID	1017

Figure 3 – Authorization Access log

The authorization Control is performed not only at network level to authorize specific resource to reach the system, it is also granted by specific configuration of each single account as written into the next Authentication Control section.

• Authentication Control

Each tool and micro service, represented by the boxes in the upper and lower side of the architecture figure, in order to connect to the WG IOP and interact with its services needs an authentication account. The Authentication system chosen for the WG IOP is the SASL plain mechanism of RabbitMQ [2], this mechanism is enabled by default in the RabbitMQ server and clients, and is the default for most other clients. The WG IOP authentication control permit to create different typology of accounts and give it different right to perform specific action inside the system. Two main user profiles have been identified at this stage of the project. The first user profile concerns all those common services offered by WG IOP such as wrappers, translators, etc. for which no specific configurations are required. The second profile, the most complex, is represented by all those accounts, such as that of the various tools, for which it is essential to create real groups and roles and give specific permissions to interact with specific services and read and write on specific queues that other users are not allowed to read or write. One of the way to create groups by the WG IOP Authentication Control is to work with the virtual host concept, when an user connects to the WG IOP it can specifies a virtual host (vhost) name to connect to, this user must be authorized to connect to it and to perform some operation like configuration that enable the user to create or destroy resources, read to retrieve a message from resources or write to injects a message to a resource.

• High Level Authorization Control

The chosen platform for implementing the WG IOP Message Broker – RabbitMQ – provides also flexibility to define the resources that each individual user (identified by a credential consisting in a username and a password) can access [3]. These mechanisms are described hereunder.

User credentials and Virtual Hosts

When a RabbitMQ client establishes a connection to a server with a given set of credentials (user and password), it specifies a virtual host within which it intends to operate. A virtual host is a logical separation of the resources managed by the message broker. These can be used to group resources (exchanges and queues) accordingly to their ultimate purpose (e.g. one virtual host can handle communications with com-

mon modules, and different virtual hosts can be devoted to specific WiseGRID applications or field device types). A first level of access control is enforced at this point, with the server checking whether the user has any permissions to access the virtual hosts, and rejecting the connection attempt otherwise.

Resources, i.e. exchanges and queues, are named entities inside a particular virtual host; the same name denotes a different resource in each virtual host. A second level of access control is enforced when certain operations are performed on resources.

RabbitMQ distinguishes between configure, write and read operations on a resource. The configure operations create or destroy resources, or alter their behaviour. The write operations inject messages into a resource. And the read operations retrieve messages from a resource. In order to perform an operation on a resource the user must have been granted the appropriate permissions for it.

Topic Authorisation

RabbitMQ supports topic authorisation for topic exchanges. The routing key of a message published to a topic exchange is taken into account when publishing authorisation is enforced. Topic authorisation is particularly relevant to the MQTT protocol, which is structured around topics and use topic exchanges under the hood.

Topic authorisation is an additional layer on top of existing checks for publishers. Publishing a message to a topic-typed exchange will go through both the basic.publish and the routing key checks. The latter is never called if the former refuses access.

Topic authorisation can also be enforced for topic consumers. Consumers consume from queues and thus the standard resource permissions apply. In addition, binding routing keys between a topic exchange and a queue/exchange are checked against the topic permissions configured, if any.

3 WG IOP FRAMEWORK

3.1 BROKER

RabbitMQ [4] is an open source multiprotocol messaging broker. The RabbitMQ server is written in the Erlang programming language and is built on the Open Telecom Platform framework for clustering and failover. RabbitMQ is lightweight and easy to deploy on premises and in the cloud, it can be deployed in distributed and federated configurations to meet high-scale, high-availability requirements. RabbitMQ runs on many operating systems and cloud environments and provides a wide range of developer tools for most popular languages such as: Java, .NET, PHP, Python, JavaScript, Ruby, Go, and many others.

The RabbitMQ broker provide HTTP-API, command line tool and a simple to use UI for managing and monitoring its functionalities. Thanks to all its useful features, RabbitMQ has been chosen as message exchanging core through the Interoperable Platform (WG IOP).

The next images show the WISEGRID RabbitMQ web UI by which it is possible to configure and manage the available services.

The UI of the broker set up for lab-testing can be reached by authorized source at the following URL: <http://161.27.206.92:15672/#/>

Username: wisegrid

Password: wisegrid

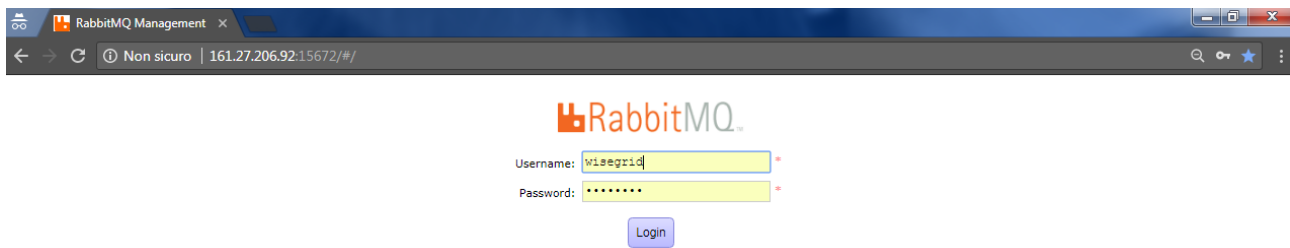


Figure 4 – Login Page

A clear and intuitive dashboard is offered by the RabbitMQ UI, by which it is possible to have immediately a clear overview of the Broker status and fast configure most of its parameters.

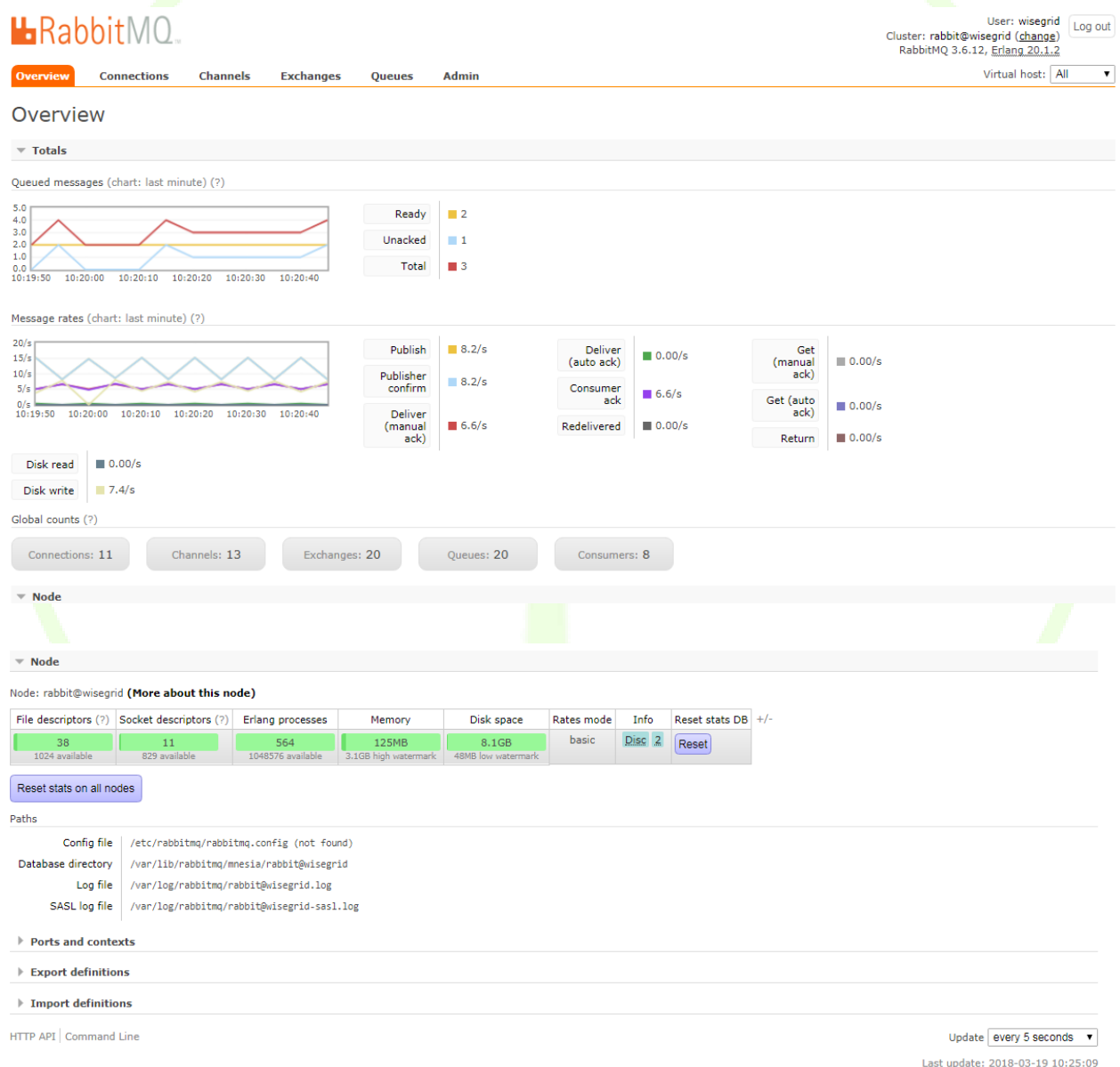
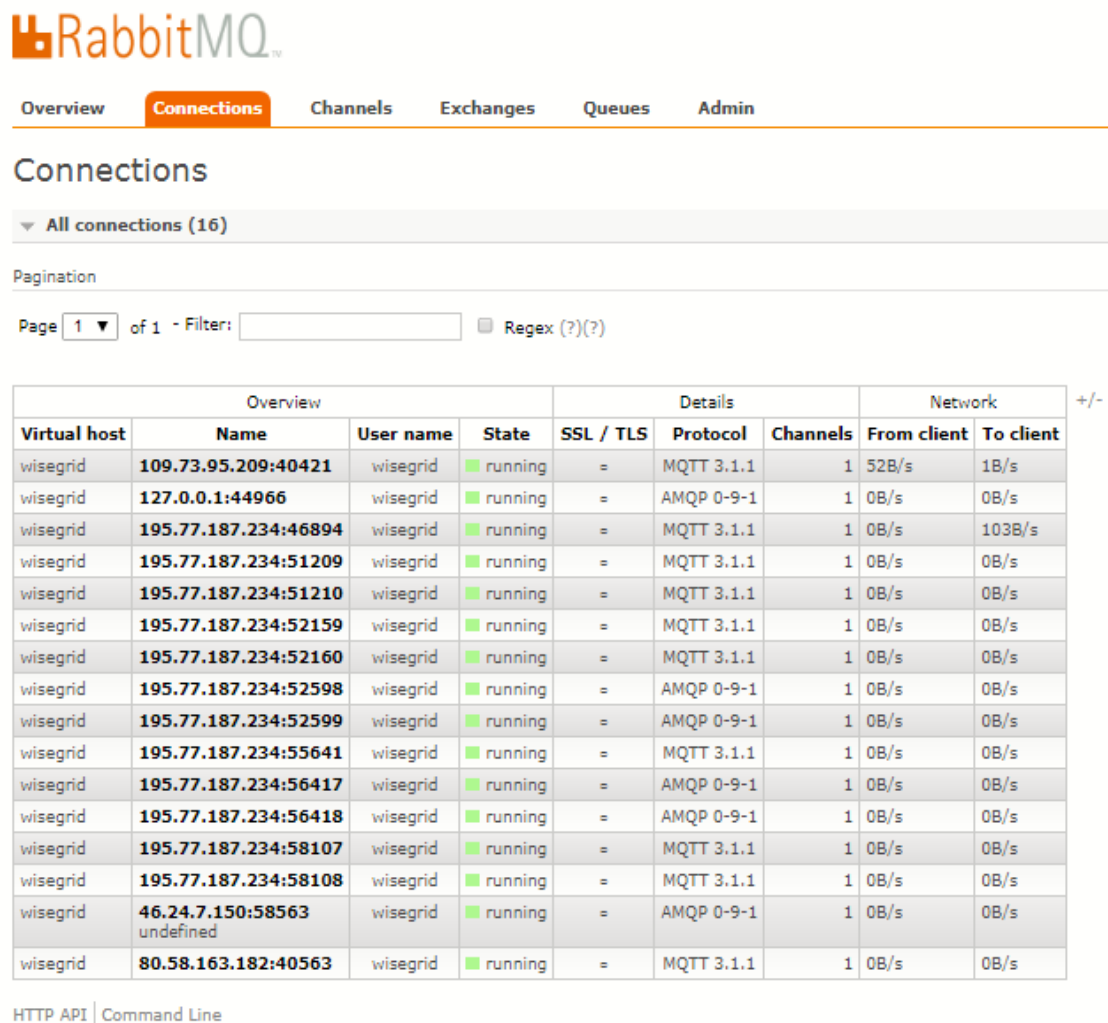


Figure 5 – Overview Page

It is also possible to monitor all connections for individual virtual hosts and know their status as the next image shows.



The screenshot shows the RabbitMQ web interface with the 'Connections' tab selected. It displays a table of 16 connections for the 'wisegrid' virtual host. The table is divided into three sections: Overview, Details, and Network. The Overview section shows the Virtual host, Name, User name, and State. The Details section shows SSL / TLS, Protocol, and Channels. The Network section shows From client and To client data rates. The table is paginated to show 16 connections on page 1 of 1. A filter and a checkbox for 'Regex (?)(?)' are also visible.

Overview				Details			Network	
Virtual host	Name	User name	State	SSL / TLS	Protocol	Channels	From client	To client
wisegrid	109.73.95.209:40421	wisegrid	running	=	MQTT 3.1.1	1	52B/s	1B/s
wisegrid	127.0.0.1:44966	wisegrid	running	=	AMQP 0-9-1	1	0B/s	0B/s
wisegrid	195.77.187.234:46894	wisegrid	running	=	MQTT 3.1.1	1	0B/s	103B/s
wisegrid	195.77.187.234:51209	wisegrid	running	=	MQTT 3.1.1	1	0B/s	0B/s
wisegrid	195.77.187.234:51210	wisegrid	running	=	MQTT 3.1.1	1	0B/s	0B/s
wisegrid	195.77.187.234:52159	wisegrid	running	=	MQTT 3.1.1	1	0B/s	0B/s
wisegrid	195.77.187.234:52160	wisegrid	running	=	MQTT 3.1.1	1	0B/s	0B/s
wisegrid	195.77.187.234:52598	wisegrid	running	=	AMQP 0-9-1	1	0B/s	0B/s
wisegrid	195.77.187.234:52599	wisegrid	running	=	AMQP 0-9-1	1	0B/s	0B/s
wisegrid	195.77.187.234:55641	wisegrid	running	=	MQTT 3.1.1	1	0B/s	0B/s
wisegrid	195.77.187.234:56417	wisegrid	running	=	AMQP 0-9-1	1	0B/s	0B/s
wisegrid	195.77.187.234:56418	wisegrid	running	=	AMQP 0-9-1	1	0B/s	0B/s
wisegrid	195.77.187.234:58107	wisegrid	running	=	MQTT 3.1.1	1	0B/s	0B/s
wisegrid	195.77.187.234:58108	wisegrid	running	=	MQTT 3.1.1	1	0B/s	0B/s
wisegrid	46.24.7.150:58563 undefined	wisegrid	running	=	AMQP 0-9-1	1	0B/s	0B/s
wisegrid	80.58.163.182:40563	wisegrid	running	=	MQTT 3.1.1	1	0B/s	0B/s

HTTP API | Command Line

Figure 6 – Connections section

It is possible to know the status of each queue, monitor and manage it.



Overview Connections Channels Exchanges **Queues** Admin

Queues

▼ All queues (25)

Pagination

Page 1 ▼ of 1 - Filter: ☐ Regex (?)

Overview				Messages			Message rates			+/-
Virtual host	Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
/	awd	D	idle	0	0	0				
/	awdaw	D	idle	0	0	0				
/	esiosprovider		idle	0	0	0				
/	simulate_queue	D	idle	0	0	0				
/	task_queue	D	idle	0	0	0				
/	weatherforecastprovider		idle	0	0	0				
wisegrid	esiosprovider		idle	0	0	0	0.00/s	0.00/s	0.00/s	
wisegrid	esiosprovider_test	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	
wisegrid	forecastingQueue	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	
wisegrid	forecasting_demand		idle	0	0	0	0.00/s	0.00/s	0.00/s	
wisegrid	forecasting_demands		idle	0	0	0	0.00/s	0.00/s	0.00/s	
wisegrid	mqtt-subscription-LabVIEW_MQTT_Clientqos1	D Exp	idle	0	0	0				
wisegrid	mqtt-subscription-PartagoEvWrapperqos1	D Exp	idle	0	0	0				
wisegrid	mqtt-subscription-mqttjs_009fa355qos0	AD	idle	0	0	0	0.00/s	0.00/s	0.00/s	
wisegrid	mqtt-subscription-mqttjs_247aabf9qos0	AD	idle	0	0	0	0.00/s	0.00/s	0.00/s	
wisegrid	mqtt-subscription-mqttjs_2df26941qos0	AD	idle	0	0	0	0.00/s	0.00/s	0.00/s	
wisegrid	mqtt-subscription-mqttjs_55b10852qos0	AD	running	0	0	0	0.20/s	0.40/s	0.00/s	
wisegrid	mqtt-subscription-mqttjs_a2f8291bqos0	AD	idle	0	0	0	0.00/s	0.00/s	0.00/s	
wisegrid	mqtt-subscription-obis2cimSubqos1	D AD	idle	0	0	0				
wisegrid	queuetest	D	idle	1	0	1	0.00/s			
wisegrid	simulate_queue	D	idle	3	0	3	0.00/s	0.00/s	0.00/s	
wisegrid	task_queue	D	idle	0	0	0				
wisegrid	testnewqueue	D	idle	1	0	1	0.00/s			
wisegrid	weatherforecastprovider		idle	0	0	0	0.00/s	0.00/s	0.00/s	
wisegrid	weatherforecastprovider_test	D	idle	0	0	0	0.00/s	0.00/s	0.00/s	

Figure 7 – Queue Manage Page

Exchanges

▼ All exchanges (20)

Pagination

Page 1 ▼ of 1 - Filter: ☐ Regex (?)(?)

Virtual host	Name	Type	Features	Message rate in	Message rate out	+/-
/	(AMQP default)	direct				
/	8942122	topic				
/	WiseExTopic	topic				
/	WiseExTopics	topic				
/	amq.direct	direct				
/	amq.fanout	fanout				
/	amq.headers	headers				
/	amq.match	headers				
/	amq.rabbitmq.log	topic				
/	amq.rabbitmq.trace	topic				
/	amq.topic	topic				
wisegrid	(AMQP default)	direct		0.00/s	0.00/s	
wisegrid	WiseExTopic	topic		0.00/s	0.00/s	
wisegrid	WiseGridTopic	topic		0.00/s		
wisegrid	amq.direct	direct				
wisegrid	amq.fanout	fanout				
wisegrid	amq.headers	headers				
wisegrid	amq.match	headers				
wisegrid	amq.rabbitmq.trace	topic				
wisegrid	amq.topic	topic		0.00/s	0.00/s	

Figure 8 – Exchange Manage Page

RabbitMQ supports several messaging protocols like AMQP, MQTT, HTTP and STOMP. These protocols are available directly by means of the RabbitMQ implementation and through the use of plugins. Two of the most used protocols by RabbitMQ are the original AMQP and MQTT protocols. Those protocols are currently used by the WISEGRID project tools and services to exchange messages through the WG IOP. The choice of one or the other protocol depends on the specific needs of each tool/service and it is left to the owners of each tool.

In the context of the WiseGRID project both protocols has been used, MQTT mainly for the Smart meter data transmission and AMQP for the interaction between tools and services like the Energy Forecasting, the Tariff provider and the Wholesale market.

3.2 MQTT

MQTT [5] (MQ Telemetry Transport or Message Queue Telemetry Transport) is a standard ISO protocol (ISO/IEC PRF 20922) of light messaging of the publish-subscribe type placed at the top of TCP/IP. It was originally developed out of IBM's pervasive computing team and their work with partners in the industrial sector. Over the past couple of years the protocol has been moved into the open source community, seen significant growth in popularity as mobile applications have taken off, and it is in the process of moving into the hands of a standards body.

It is designed for situations where low impact is required and where the band is limited. One of the advantages MQTT has over more full-featured "enterprise messaging" brokers like RabbitMQ, is that its intentionally low footprint makes it ideal for today's mobile and developing "Internet of Things" - IOT applications. Several companies, like "Facebook", are using MQTT protocol in some of their mobile applications mainly for its light on network bandwidth.

The design principles and objectives of MQTT are much simpler and more targeted than those of AMQP, as it provides publish-and-subscribe messaging (there is no virtual host concept no queues, despite the name), MQTT's strengths are simplicity (just few API methods), a compact binary packet payload (no message properties, compressed headers, much less verbose than something text-based like HTTP), and it makes a good fit for simple push messaging scenarios such as smart meter measurements.

3.3 AMQP

AMQP [6] (Advanced Message Queuing Protocol) is a binary, application layer protocol, designed to efficiently support a wide variety of messaging applications and communication patterns. The main characteristics of AMQP are reliability and interoperability. As the name suggests, it provides a wide range of messaging functionalities, including trusted queues, topic-based messaging, publish-and-subscribe messaging, flexible routing, transactions and security. AMQP exchanges route messages directly in fan-out format, by topic and also on the basis of headers. This protocol was designed for reliability at the many large companies who depend on messaging to integrate applications and move data around their organisation. In the case of RabbitMQ, there are many different language implementations and great number of samples available, making it a good choice for building large scale, reliable, resilient, or clustered messaging infrastructures. Like for MQTT several companies use it, only to give some examples company like JP Morgan use it to process 1 billion messages a day. NASA uses it for Nebula Cloud Computing and Google uses it for complex event processing.

Below an image that shows the main differences between the two protocols and features.

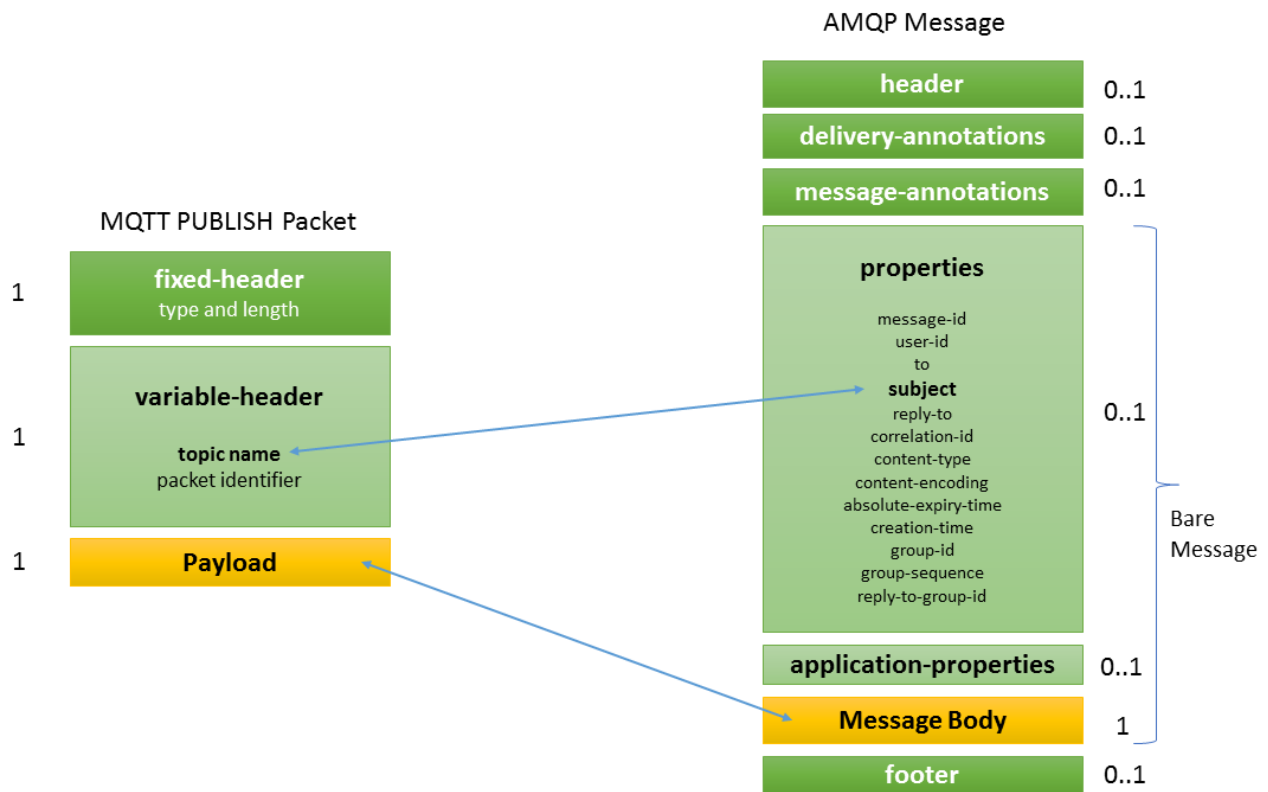


Figure 9 – Main difference between MQTT and AMQP protocol

4 STANDARDS

This part aims to provide an overview of the standards to be considered for the management of the data flows inside the interoperable platform. Considering that the analysis of the specific reference standards has already been done in the Task T3.3 “Standards and interoperable data models” and it is available in the deliverable D3.2 “WiseGRID architecture, data models, standards and data protection (V2)”, in this part only the USEF framework and the relationship with some main standards within WiseGRID project are considered.

4.1 USEF

USEF (Universal Smart Energy Framework) [7] provides a modular design for smart energy systems that can be customized to the needs of smart grid project implementations. It guarantees the interoperability of products and services and it ensures that solutions become repeatable and replicable, safeguarding investments in the smart energy future [8].

USEF provides a complete set of specifications to secure the essential interoperability between all the components in a smart energy system. **The main focus is on the business perspective** and not on the technical implementation of a DSM (Demand Side Management) framework, where OPENADR standard (which will be further explained in this document) is widely covering this aspect. To optimize the value of flexibility across all roles in the system, USEF introduces a new market-based coordination mechanism (MCM) along with new processes. The USEF Market-based Coordination Mechanism (MCM) favours the delivery of value

propositions (i.e. marketable services) to various market parties without imposing limitations on the diversity and customization of the propositions. The USEF MCM is designed for all energy commodities and enables the market to optimize in time, capacity and power. MCM provides access, under equal conditions, for all stakeholders to a single integrated market. This unique approach aims for a future-proof design of the targeted European unique energy market. The USEF MCM operations scheme distinguishes 5 phases and the corresponding processes are summarized hereafter [7]:

- **Contract:** It is required to establish in advance some types of contractual relationships. One example can be the contract between prosumers and aggregators regarding how to activate prosumer's flexibility capacity.
- **Plan:** In the plan phase the demand and supply of energy are planned for the upcoming period, usually a calendar day. Both the BRP and Aggregator carry out an initial portfolio optimization. During this phase, the BRP can procure flexibility from its Aggregators. The Planning phase results in an agreed-upon Aggregator plan (A-plan) between the Aggregator and the BRP.
- **Validate:** In the Validate phase, the DSO validates whether the demand and supply of energy can be distributed safely without any limitations, with the use of D-prognoses. If congestion happens, the DSO can procure flexibility from Aggregators to solve the grid capacity issues. It is important to note that iterations exist between the Plan and Validate phases. This means that after validation, it is possible to go back to the Plan phase. These iterations take place until the foreseen energy flows can be distributed safely in an economically optimized way.
- **Operate:** In the Operate phase, the actual assets and appliances are dispatched and the Aggregator adheres to its D-prognoses and A-plan. When needed, DSOs and BRPs can invoke additional flexibility from the Aggregators to resolve unexpected congestions (for the DSO) or to provide balancing services or re-optimize their portfolio (for the BRP).
- **Settle:** in the Settle phase the flexibility that the Aggregator has sold to the BRPs or DSOs is settled. For this purpose the actual consumed and produced volumes are first allocated to the responsible parties; unsolved or disputed volumes are reconciled shortly thereafter.

The next schema presents the different business layers of USEF framework, once the contracts between actors are signed, highlighting also the different services to be supported by each layer:

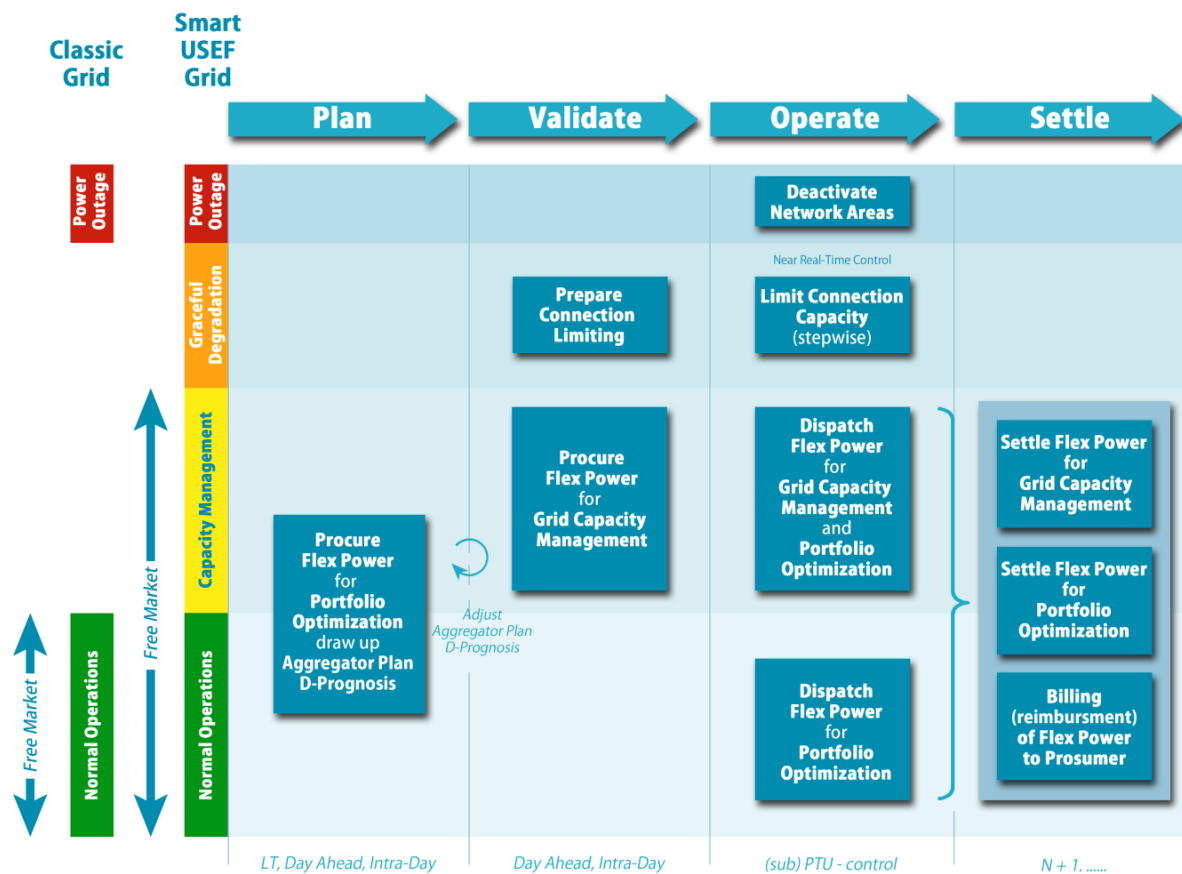


Figure 10 – USEF Protocol Business Framework [7]

The USEF specification is technology and implementation agnostic. It standardizes the logical interfaces and defines the minimum functionality of the components in the form of use cases and message transport specifications and message descriptions. While the USEF specification is technology and implementation agnostic, there are some basic principles to which any USEF implementation architecture must adhere.

USEF in WiseGRID

As WiseGRID is a project mostly focused in the Distribution Grid, the USEF framework can be specially interesting because identifies different flexibility services for the DSO which provide value by helping the DSO increase its performance and efficiency in managing the distribution grid [8].

- **Congestion management** refers to avoiding the thermal overload of system components by reducing peak loads. In contrast with grid capacity management, this is a situation where failure due to overloading may occur. It is a short-term problem (with respect to the duration of a grid reinforcement project) for the DSO that requires a relatively swift response. The conventional solution is grid reinforcement (e.g., cables, transformers). The alternative (load flexibility) may defer or even avoid the necessity of grid investments.
- **Voltage problems** typically occur when solar PV systems generate significant amounts of electricity. This will “push up” the voltage level in the grid. Using load flexibility by increasing the load or decreasing generation is an option to avoid exceeding the voltage limits. This mechanism can reduce the need for grid investments (such as automatic tap changers) or prevent generation curtailment.

- **Grid capacity management** aims to use load flexibility primarily to optimize operational performance and asset dispatch by reducing peak loads, extending component lifetimes, distributing loads evenly, and so forth. An added benefit may be the reduction of grid losses.
- **Controlled islanding** aims to prevent supply interruption in a given grid section when a fault occurs in a section of the grid feeding into it.
- **Redundancy (n-1) support** refers to actions that help to reduce the frequency and duration of outages. An example is supplying emergency power (or shedding loads) in the event of a severe power shortage, or supplying backup power during grid maintenance actions.

4.2 CIM

One of the data model used in WiseGRID is the Common Information Model (CIM). The CIM, originally designed as an EMS-API [9], has been developed as an internal database model (relational schema) for EMS and SCADA systems, but rapidly turned into a much more useful object-oriented approach to model all the relevant objects and their links in the electric distribution, transmission and generation frameworks. The domain ontology is quite large, covering most aspects being focused on subparts of the model like generation, outage, documentation, transmission, wires, measurements, and so on. The main focus of this standard, is in the system integration. The IEC (the International Electrotechnical Commission) started its efforts to establish the CIM as an international standard and chartered the TC 57: Power System Management and Associated Information Exchange.

The CIM Data Model Overview

The Common Information Model CIM defines objects and relationships

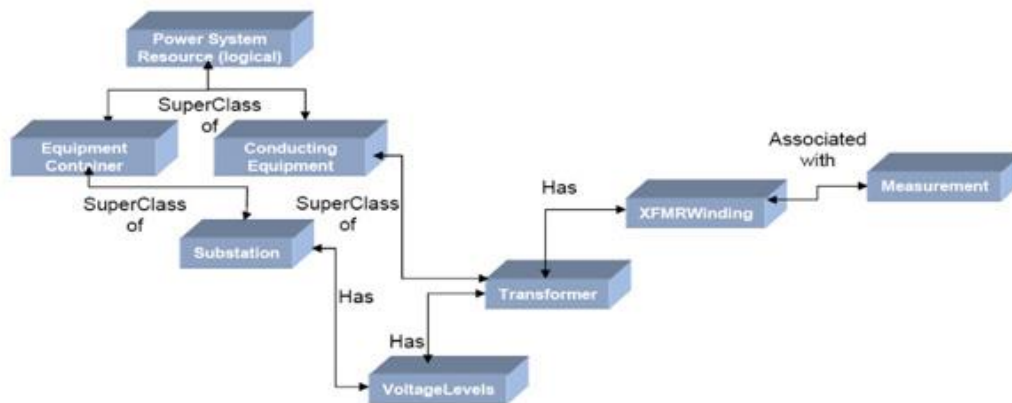


Figure 11 – Example of the CIM Data Model [10].

The TC 57 generated among other things the extensive CIM data model that has become widely used in power systems. The main use cases for CIM are:

- To be used as a large domain ontology, providing a language to communicate business messages which are exchanged between various systems intra-utility.
- To transmit topology data.

- To serve as a basis for defining more tightened CIM models and structures than using the common data model and pre-defined serialization types. For instance, as part of the IEC 61968 [9], pre-defined processes and payloads exist that can be used as blueprints to start on using standards-compliant processes and XML schemata. This practically means that different information messages (with headers, contents, payload, etc.) based on the base CIM model are defined for specific communication purposes. The following processes are standardized:
 - Monitoring and control of equipment for power delivery
 - Management processes to ensure system reliability
 - Voltage management
 - Demand side management
 - Outage management
 - Work management
 - Automated mapping
 - Facilities management
- To foster electricity-market communication and data exchange, considering European and US-style markets.

The CIM standards are continuously evolving to meet the changing requirements for data exchange, which are increasing in both frequency and type, with higher RES integration and the introduction of smart grids [11]. Having this into account, the future trends and challenges of the CIM standard are [12]:

- Extension of the model to collect all the needs of the future distribution (management of distributed generation, active consumption, etc.)
- Application of the model in new types of facilities such as railway installations on high-speed lines.
- Harmonization of the model with other information models of the electrical system, mainly the model defined in the IEC 61850 standard for automation systems of electrical installations.

CIM in WiseGRID

As later described in this deliverable, the overall WiseGRID architecture decouples information producers (field devices such as smart meters, or monitoring systems such as SCADAs or AMIs) and information consumers (the different applications). Moreover, application architectures are based on the micro-service approach, each one of them being actually composed of smaller modules with very specific duties. While these architectures provide significant advantages, such as scalability and resilience, the definition of clear data models for the communication of information among the different elements becomes very important. Within this context, CIM plays a role in the definition of the data provided by elements typically managed by the DSO (smart meters, SCADAs and AMIs), and also in the exchange of information within modules of certain applications (as, for instance, the WiseGRID Cockpit). The project benefits from selecting an already well-established and standard definition of information to be exchanged, such as energy readings.

4.3 OPENADR

One of the standard used in WiseGRID is OpenADR [13] (Automated Demand Response). This initiative lead by North American research labs and companies was crated, as they claim, “to accelerate the development, adoption and compliance of OpenADR standards throughout the energy industry” and “provide common language”. With this aim, OpenADR tries to provide non-proprietary standardized interfaces for electric services companies to communicate demand response and distributed energy resources to their users, making use of existing information and communication technologies, such as internet. The aim of this open model is to make electric grids smarter, enabling a two-ways communication among the different agents of the electric market and its users, guaranteeing interoperability in the electric grids. The next figure shows a schema of this standard communication architecture.

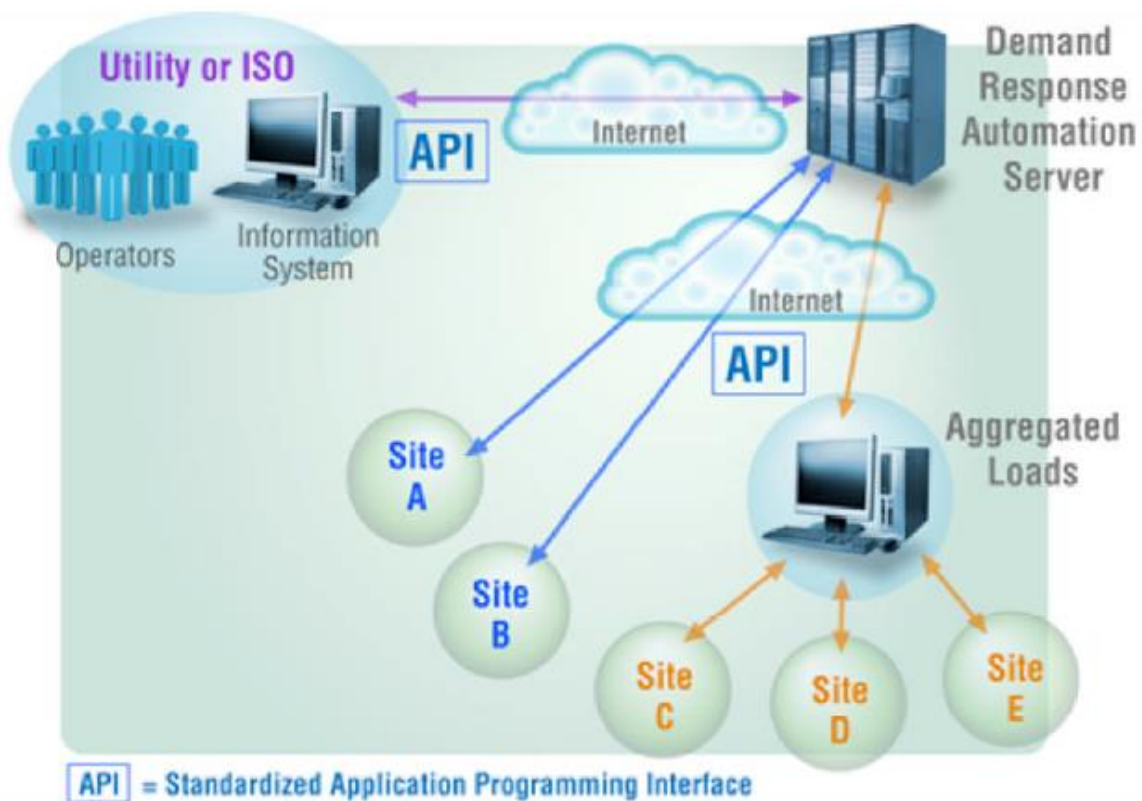


Figure 12 – OpenADR communication architecture schema [14]

The OpenADR standard specifies the application programming interfaces (APIs) to, and the functions of, the Demand Response Automation Server (DRAS), which serves as the common platform between all providers and consumers of electricity.

OpenADR, making use of Web Service schemas, is able to transmit information in push and pull data exchange. To accomplish this, a server that allocates all the data transmission mechanisms and clients that select the most suitable mechanism is required. After evaluating the available data transmission technologies, such as Simple Object Access Protocol (SOAP), Representational State Transfer (REST), Hyper Text Transfer Protocol (HTTP), eXtensible Messaging and Presence Protocol (XMPP); REST-styled simple HTTP has been the chosen protocol to implement OpenADR. OpenADR adapts to standard the achievement of research, providing insights for interoperability systems.

5 WG IOP INTERACTIONS WITH OTHER WG TOOLS

5.1 WG COCKPIT

5.1.1 Messages exchanged by means of WG IOP & standards

The WiseGRID Cockpit application will interact with the required field data sources using the WG IOP. These data sources include:

- **Unbundled smart meters:** these smart meters equipped with WiFi, 3G and/or Ethernet, are capable of communicating data with low-period frequency using regular internet connections. These devices will directly communicate with the WG IOP, thus making this measurements to any application of the project requiring it. This information will be published using the CIM format (IEC61968), encapsulated in MQTT payloads.

- **Advanced Monitoring Infrastructure:** a specific wrapper will be implemented for each one of the pilot sites that already has an AMI system for the deployed smart meters. The wrapper will be in charge of retrieving measurements collected by the AMI system and publish those to the WG IOP, thus making them available to the different applications of the WiseGRID project. This information will be published using the CIM format (IEC61968), encapsulated in MQTT payloads.
- **SCADA:** a specific wrapper will be implemented for each one of the pilot sites that already has a SCADA controlling the distribution grid, in order to take advantage of the equipment already existing on site to retrieve data from distribution grid substations. This wrapper will also read the status of different safety elements of the grid, as required by the WiseGRID Cockpit application. All this information will be published to the WiseGRID IOP to make it available to the WiseGRID Cockpit. This information will be published using the CIM format (IEC61968), encapsulated in MQTT payloads.

In addition, different aggregators targeted by the applications of the WiseGRID ecosystem can collaborate with the DSO in order to alleviate certain issues that the distribution grid may face, such as the foreseen congestion at certain areas of the grid in the short-term future. With this objective, the WiseGRID Cockpit will communicate through the WG IOP with the WiseCOOP, WG STaaS/VPP and the WiseEVP, the three applications aggregating demand that may be modulated for this purposes. The interaction among the four applications is coordinated in the form of an Ancillary Services Market based on the USEF framework [15]. WiseGRID Cockpit will be the application implementing the core of the Market, thus triggering the requests for flexibility upon detection of a problem in the grid, processing the different offers received and posting the flexibility orders.

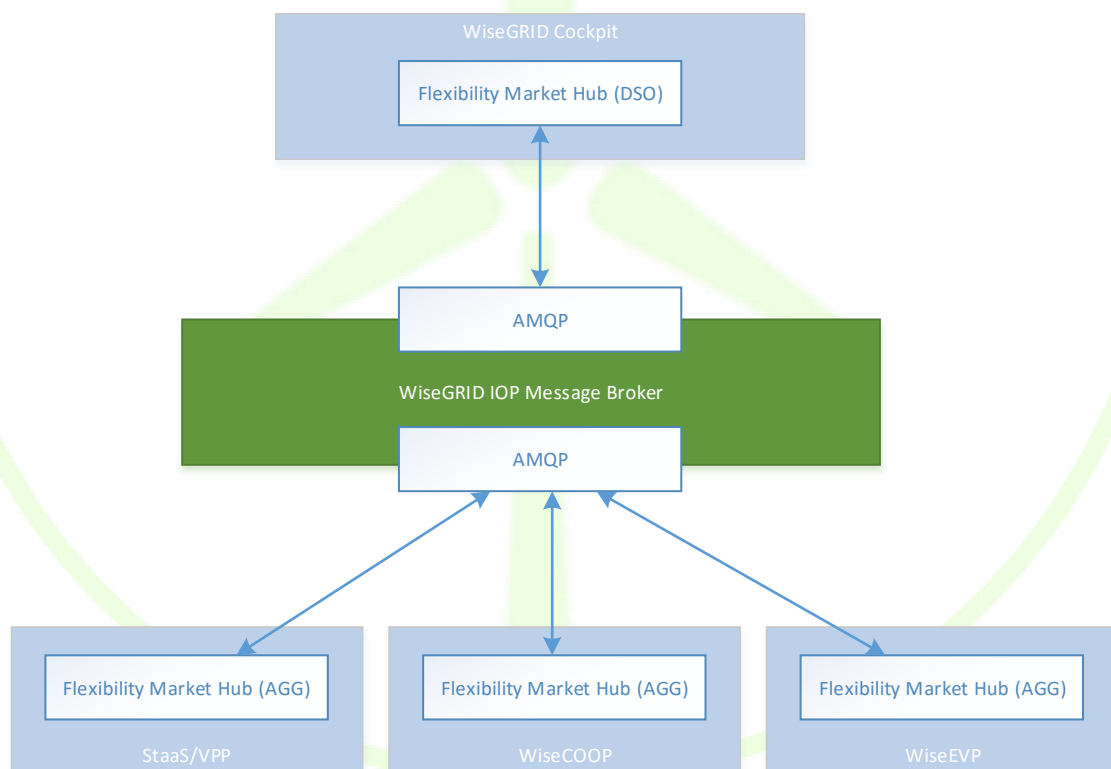


Figure 13 – Overview of applications involved in the Ancillary Services Market

The procedures and formal definition of the messages exchanged between the applications via the WiseGRID IOP for the implementation of this Ancillary Services Market are further documented in in section 6.2 and in deliverable D13.1 WiseGRID Cockpit design [16]. These information exchanges will use the AMQP protocol enabled by the WiseGRID IOP Message Broker.

5.1.2 Messages structure

Messages structures exchanged by the WG Cockpit application with other modules via WG IOP are defined in the corresponding subsections under section 6 and some examples are available in the ANNEX A .

5.2 WG RESCO

5.2.1 Messages exchanged by means of WG IOP & standards

The following table from the deliverable document D3.2 “WiseGRID architecture, data models, standards and data protection (V2)”, summarizes some of the relevant standards, data models and open protocols identified for the data to be handled by WG RESCO.

Table 1 – WG RESCO main Standards

Data item	Related data model or standard
Energy metering	CIM - DLMS/COSEM
Weather forecast	Custom data model
Flexibility offer	USEF
Retail electricity price	OpenADR

The WG RESCO tool will interact with the following elements of the WG IOP:

- Weather forecast provider: this information is used as an input to the demand and production forecast algorithms. The data format used by the module is described in section 6.3.
- Energy tariff provider: this information is used to calculate the bill costs for each prosumer identified by a specific contract and a supply point with custom and third party tariffs. The tariff is used by the billing algorithm to perform the calculation. The model used for tariff modelling is described in section 6.5.
- Wholesale market prices: prices from wholesale market are retrieved in order to allow the retailer or aggregator of prosumers to optimize the demand of the portfolio against the cost of the energy. The data format used by the module is described in section 6.6.
- Demand/production forecast service: this information is central to estimate energy to be bought from the wholesale market, estimate future energy costs and estimate flexibility that may be marketed to offer ancillary services. The data format used by the module is described in section 6.4.
- Unbundled Smart Meter wrapper: one of the main data sources of the RESCO tool corresponds to energy metering information provided by Unbundled Smart Meters that publish information to the WG IOP. The data format of this information is based on CIM MeterReadings (IEC61968)
- AMI Wrapper: the second main data source of the RESCO tool corresponds to energy metering information extracted from the AMI system deployed in the pilot sites (where this system is available). In order to homogenize formats, this data is provided using a format based on CIM MeterReadings (IEC61968), the same used by Unbundled Smart Meters

5.2.2 Messages structure

Some examples of the messages structures exchanged by the WG RESCO application with other modules via WG IOP are available in the ANNEX A paragraph “ RESCO Tool Messages Structure” while under section 6 are available some examples about the exchanged messages with the other services like the meter forecast of the RT monitoring.

5.3 WG STAAS/VPP

5.3.1 Messages exchanged by means of WG IOP & standards

The following table from the deliverable document D3.2 “WiseGRID architecture, data models, standards and data protection (V2)”, summarizes some of the relevant standards, data models and open protocols identified for the data to be handled by WG StaasVPP.

Table 2 – STaaS/VPP main Standards

Data item	Related data model or standard
Energy metering	CIM - DLMS/COSEM
Weather forecast	Custom data model
Flexibility offer	USEF
Retail electricity price	OpenADR

The Staas/VPP tool will interact with the following elements of the WG IOP:

- Weather forecast provider: this information is used as an input to the demand and production forecast algorithms. The data format used by the module is described in section 6.3.
- Energy tariff provider: this information is used to calculate the bill costs for each prosumer identified by a specific contract and a supply point with custom and third party tariffs. The tariff is used by the billing algorithm to perform the calculation. The model used for tariff modelling is described in section 6.5.
- Wholesale market prices: prices from wholesale market are retrieved in order to allow the retailer or aggregator of prosumers to optimize the demand of the portfolio against the cost of the energy. The data format used by the module is described in section 6.6.
- Demand/production forecast service: this information is central to estimate energy to be bought from the wholesale market, estimate future energy costs and estimate flexibility that may be marketed to offer ancillary services. The data format used by the module is described in section 6.4.
- Unbundled Smart Meter wrapper: one of the main data sources of the Staas/VPP tool corresponds to energy metering information provided by Unbundled Smart Meters that publish information to the WG IOP. The data format of this information is based on CIM MeterReadings (IEC61968)
- AMI Wrapper: the second main data source of the Staas/VPP tool corresponds to energy metering information extracted from the AMI system deployed in the pilot sites (where this system is available). In order to homogenize formats, this data is provided using a format based on CIM MeterReadings (IEC61968), the same used by Unbundled Smart Meters

5.3.2 Messages structure

Some examples of the messages structures exchanged by the WG STaaS/VPP application with other modules via WG IOP are available in the ANNEX A paragraph “10.2 STAAS/VPP Tool Messages Structure” while under section 6 are available some examples about the exchanged messages with the other services like the meter forecast of the RT monitoring.

5.4 WISEEVP

5.4.1 Messages exchanged by means of WG IOP & standards

As it has been stated, WG EVP is a technological solution to optimize the activities related to EV smart charging and discharging, including V2G (vehicle to grid, energy injection in the distribution network) and V2B (vehicle to building, energy injection in the household electric installation)

This solution, in order to perform its role must interact with the WG IOP, and via this solution, interacts with the rest of solutions involved in the usual WiseEVP performance (WG FastV2G and WG Cockpit). Due to the role WG EVP must play, it is necessary to implement within this solution features such as:

- Management of EVSE portfolio.
- Management of EV portfolio.
- Operational management of charging sessions.
- Operational management of ancillary services and V2G.

The WiseEVP application will interact with the required field devices using the WG IOP. These devices include:

- Electric Vehicle Supply Equipment (EVSE): these infrastructure elements supply electric energy to the electric vehicles. They shall provide information about the charging sessions (identification of user charging the vehicle, energy supplied...) and allow the WiseEVP to control the energy flow they shall supply or extract from the vehicle under V2G operations.
- Electric Vehicles (EV): in order to enable the holistic management of the fleet of electric vehicles and the charging points, data from the vehicles (such as location, state of charge, travelled distance...) need to be retrieved.

The approach taken within the WiseGRID project implies the development of different wrappers to encapsulate the communication with the actual EVSEs and EVs, in order to provide a unified interface to retrieve data from and send commands to those elements, independently from the vendor. Those elements will communicate directly to the WiseGRID IOP, making it therefore easy to integrate the information coming from those elements in the applications of the project

In consequence of the defined features, there exists a set of information which must be exchanged by the WG EVP, making use of the standards and data models defined within definition of the WiseGRID solutions. The standards implemented within this solution, as it is stated in the deliverable document D3.2 “WiseGRID architecture, data models, standards and data protection (V2)”, are CIM model, USEF, OCPP and GeoJSON.

CIM model, as one of the most extended standard within the smart grids, it is going to be used for energy metering. This model is explained in this document in section 4.2.

At the same time, in order to provide and offer flexibility, it is going to be implemented USEF protocol. This protocol, developed by E-Laad foundation in the Netherlands, provides a flexibility service to the solution,

making possible exchange information of the demand forecast, flexibility forecast and flexibility request. Like the case of CIM model, this standard is defined in a previous section 4.1.

For exchanging information related to the EVSE, the OCPP protocol is going to be implemented. This protocol is going to be implemented to exchange EVSE metadata, charging profiles, EV information and EV user information related to the charging session. OCPP protocol is one of the most extended protocols to implement communications among EVSEs, being in Europe the preferred protocol to implement these communications. Since its first version, it has been issued four versions of this protocol, within this solution it is going to be implemented the OCPP v1.6. This version is the first version to include the usage of JSON, in addition to SOAP, which was the only protocol above previous versions were implemented.

And finally, within this WiseEVP, to define and exchange asset geolocation data, GeoJSON models will be used. This open format is widely used in cartographic applications, providing a simple and fast data exchange. This format is based on JSON format, and it defines several JSON objects and how to combine them to represent data related to geographic features, their properties and their spatial extents.

5.4.2 Messages structure

Communications to the WiseEVP are going to be implemented via WG IOP. This means that, due to communications with IOP must be implemented in MQTT, it is necessary to implement a wrapper to encapsulate in MQTT the different OCPP messages to be exchanged. This wrapper is the module within WiseEVP in charge of implementing communications with the different charging stations. The wrapper module it is detailed in the section 6.1.1.

Within WiseEVP development, its exchanged messages via IOP will follow the OCPP v1.6 JSON spec, being encapsulated for its exchange in MQTT payloads.

These message structures are defined in detail in the deliverable document D9.1 "WiseEVP Design".

5.5 WISECOOP

5.5.1 Messages exchanged by means of WG IOP & standards

The WiseCOOP application will interact with the following elements of the WG IOP:

- Weather forecast provider: this information is used as an input to the demand and production forecast algorithms. The data format used by the module is described in section 6.3.
- Energy tariff provider: this information is used to calculate the costs associated to the demand of the portfolio of aggregated prosumers with custom and third party tariffs, in order to get information for decision support. The model used for tariff modelling is described in section 6.5.
- Wholesale market prices: prices from wholesale market are retrieved in order to allow the retailer or aggregator of prosumers to optimize the demand of the portfolio against the cost of the energy. The data format used by the module is described in section 6.6.
- Demand/production forecast service: this information is central to estimate energy to be bought from the wholesale market, estimate future energy costs and estimate flexibility that may be marketed to offer ancillary services. The data format used by the module is described in section 6.4.
- Unbundled Smart Meter wrapper: one of the main data sources of the WiseCOOP application corresponds to energy metering information provided by Unbundled Smart Meters that publish information to the WG IOP. The data format of this information is based on CIM MeterReadings (IEC61968)

- AMI Wrapper: the second main data source of the WiseCOOP application corresponds to energy metering information extracted from the AMI system deployed in the pilot sites (where this system is available). In order to homogenize formats, this data is provided using a format based on CIM MeterReadings (IEC61968), the same used by Unbundled Smart Meters
- WiseCORP DR optimization framework: WiseCOOP will use flexibility from buildings with big demand to provide ancillary services to other actors, such as the DSO. In order to implement those mechanisms, openADR messages are exchanged between the applications via WG IOP.
- WiseHOME: WiseCOOP will be the main information source for the WiseHOME application, providing access to information of energy usage of individual prosumers and aggregated portfolio, as well as some KPIs and trends. This information exchange will happen via WG IOP, as described in section 5.7

5.5.2 Messages structure

Messages structures exchanged by the WiseCOOP application with other modules via WG IOP are defined in the corresponding subsections under section 6 and paragraph 5.7 as mentioned above.

5.6 WG CORP

5.6.1 Messages exchanged by means of WG IOP & standards

The WiseCORP application will interact with the following elements of the WG IOP:

- Weather forecast provider: this information is used as an input to the demand and production forecast algorithms. The data format used by the module is described in section 6.3.
- Energy tariff provider: this information is used to calculate the costs associated to the demand of the building under different tariff schemes, in order to get information for decision support. The model used for tariff modelling is described in section 6.5.
- Demand/production forecast service: this information is central to estimate future energy demand, energy costs and estimate flexibility that may be marketed to offer ancillary services. The data format used by the module is described in section 6.4.
- Unbundled Smart Meter wrapper: one of the data sources of the WiseCORP application corresponds to energy metering information provided by Unbundled Smart Meters that publish information to the WG IOP. The data format of this information is based on CIM MeterReadings (IEC61968).
- AMI Wrapper: energy demand of buildings without Unbundled Smart Meters still can be monitored with the information of their demand extracted from the AMI system deployed in the pilot sites (where this system is available). In order to homogenize formats, this data is provided using a format based on CIM MeterReadings (IEC61968), the same used by Unbundled Smart Meters.
- Battery Wrapper: this wrapper provides access to monitor status and send commands (setpoints) to batteries of the building. WG IOP handles the authorization, so monitoring and commanding can only be performed from the authorized applications (such as WiseCORP). Data format used by this wrapper is described in section 6.1.4.
- CHP Wrapper: this wrapper provides access to monitor status and send commands (setpoints) to CHP assets of the building. WG IOP handles the authorization, so monitoring and commanding can only be performed from the authorized applications (such as WiseCORP). Data format used by this wrapper is described in section 6.1.5.

- HVAC Wrapper: this wrapper provides access to monitor status and send commands (setpoints) to HVACs of the building. WG IOP handles the authorization, so monitoring and commanding can only be performed from the authorized applications (such as WiseCORP). Data format used by this wrapper is described in section 6.1.5.
- Sensor Wrapper: this wrapper provides indoor context data (such as temperature) from sensors located in a building. WG IOP handles the authorization, so monitoring can only be performed from the authorized applications (such as WiseCORP). Data format used by this wrapper is described in section 6.1.3.
- Gas Meter Wrapper: this wrapper provides gas usage reading from the facilities' gas meter. WG IOP handles the authorization, so monitoring can only be performed from the authorized applications (such as WiseCORP). Data format used by this wrapper is described in section 6.1.5.
- WiseCOOP DR optimization framework: WiseCORP and WiseCOOP applications will exchange information mainly to allow the participation in explicit demand response campaigns by modelling the flexibility that may be offered by the controllable assets of the building.

5.6.2 Messages structure

Messages structures exchanged by the WiseCOOP application with other modules via WG IOP are defined in the corresponding subsections under section 6.

5.7 DEMAND RESPONSE & WISEHOME

5.7.1 Messages exchanged by means of WG IOP & standards

The WG IOP will be responsible for transferring messages for the facilitation of the Demand Response campaigns managed by the WiseGRID tools - namely WiseCOOP, WiseCORP – as well as the WiseHOME product which aims to inform home residents about the energy-related behaviours.

The main interaction identified at this scope are the follow:

- WiseHOME to WiseCOOP: query for information to be visualized
- WiseCOOP to WiseHOME: response containing information for visualization
- WiseCOOP broadcasting of dynamic energy price
- WiseCORP to WiseCOOP: communication of demand flexibility potential
- WiseCOOP to WiseCORP: request to modify demand profile by given amount
- WiseCORP to WiseCOOP: confirmation of intention to modify demand profile based on request
- WiseCORP to WiseCOOP: reporting of actual demand profile modification

Demand Response related messages and information exchanges over the WG IOP will be based on the OpenADR standard.

Equipment and assets that produce or consume information in the building – such as sensors, actuators, controllable HVAC or lighting systems, etc. – will be identified using OBIS codes, as specified in the DLMS standard.

Communications regarding information exchange for visualization purposes will not rely on any standard, their structure has been developed for the purposes of the WiseGRID project.

5.7.2 Messages structure

Some examples of messages structures exchanged for the demand response including also WiseHOME tool are available in the ANNEX A, specifically 10.3 Demand Response & WiseHOME messages structure .

6 WG IOP MICRO-SERVICES

6.1 REAL TIME (RT) MONITORING

This paragraph describes the needed wrappers for the collection of the data derived by the different assets to be monitored like, smart meters, AMI, Electric Vehicles, Batteries, SCADA, etc.

Hereunder a description about each of that and their interaction and messages exchange with the WG IOP is provided.

6.1.1 EVSE wrapper

In order to establish communication among WiseEVP and EVSEs it is necessary to implement a module to wrap the exchanged messages into the proper format, in this case MQTT. With such an aim, the EVSE Wrapper has been defined.

As depicted in the picture below, this module must encapsulate all the different OCPP messages from the EVSEs, and via WG IOP these messages are sent to the Wise EVP modules.

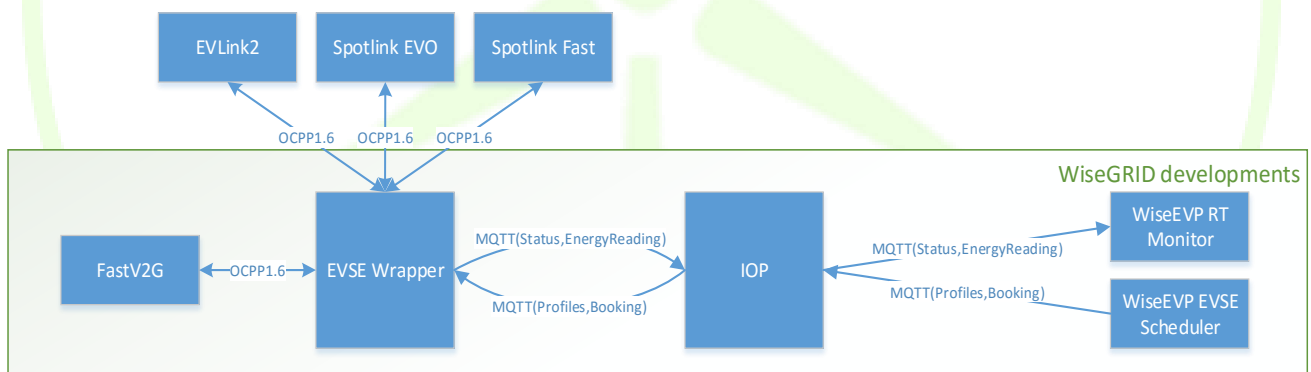


Figure 14 – Overview of communication flows from EVSEs to WiseEVP

EVSE Wrapper

Commercially available EVSEs usually implement a protocol that allows them to interact with a control system. The most extended protocol towards this end is the Open Charging Point Protocol (OCPP), which has been published in three different versions at the moment of writing (1.2, 1.5 and 1.6). An analysis of the EVSEs that will be available for demonstration of the project has shown that all of them implement protocol OCPP 1.6. A wrapper implementing this protocol to communicate with the EVSEs will be therefore developed within the project.

Taking advantage of this fact and the popularity of the OCPP protocol, a selection of OCPP 1.6 JSON messages has been also selected as the common data model to be implemented between the WiseEVP application and the EVSE Wrapper.

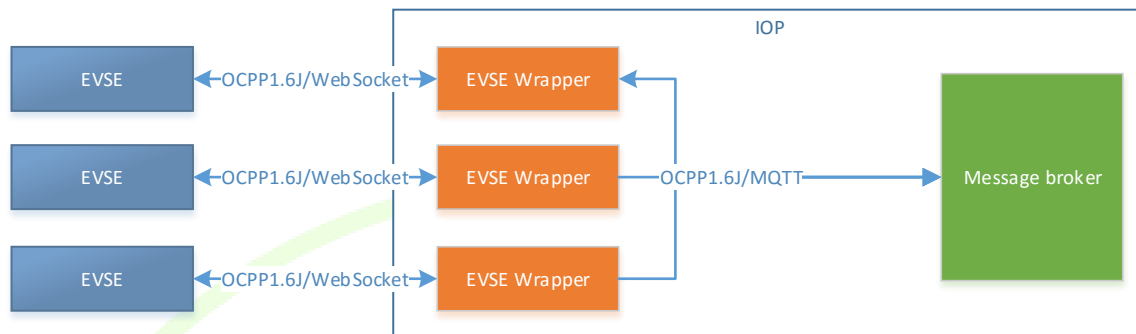


Figure 15 – Detail of protocols used for communication with EVSEs

Despite this, the architecture still allows the integration of EVSEs implementing different protocols, just by developing the corresponding wrapper to bridge the messages from the custom protocol into the selection of OCPP1.6 messages.

This wrapper implements a list of minimum necessary OCPP messages needed to successfully monitor and command a Charging Station.

In the next list are enumerated the mandatory messages:

- HeartBeat: needed to track the communication 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 a characteristics or charging sessions.
- StopTransaction: needed to modulate the energy demand of the system.

And in the next list are enumerated the encouraged messages:

- BootNotification.
- DataTransfer.
- DiagnosticStatusNotification
- FirmwareStatusNotification.
- ReserveNow.

Definition of these wrapper messages structure are available in the ANNEX A 10.4

6.1.2 EV Wrapper

Fleet managers usually deploy custom systems to monitor their fleet of vehicles and retrieve data relevant to their business, such as location, travelled distances or state of charge. The approach taken in the project to publish this information to the WiseGRID IOP (thus making it available to the WiseEVP application) is to build different wrappers that will map the information retrieved from this kind of systems (as already deployed in the pilot sites) into a custom defined data model.

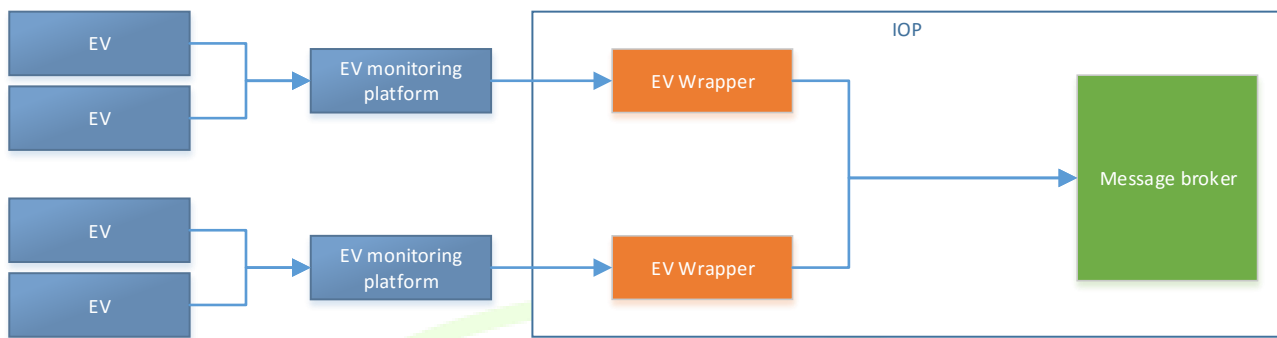


Figure 16 – Overview of the communication flows for EV monitoring

A custom data model has been defined within the project in order to retrieve the necessary information from the different EV monitoring platforms. Definition of these wrapper messages structure are available in the ANNEX A , paragraph “ 10.5 EV Wrapper messages”.

6.1.3 Wrappers for energy and grid monitoring

6.1.3.1 Unbundled Smart Meters (SMX) inside WiseGRID project

Real-time monitoring is usually implemented through standard remote terminal units (RTUs), or through more advanced solutions like digital protection relays which have integrated RTU-like functionalities.

However, in the new Smart grid paradigm, which has the biggest impact in low voltage, there are few, if none RTUs or digital protection relays in place, however each end-user has a smart meter to support energy measurement and billing.

The Smart Meter is then the most suitable equipment to be used also for monitoring purposes in to the grid, considering that privacy policy is as well ensured. That makes the SMART Meter to become a dual used device; both for billing and also for commercial & operational (SCADA for business and grid safety & security) purposes.

In WiseGRID it is used the potential of real-time monitoring through the Unbundled Smart Meter (USM) architecture, which is developed in H2020 project Nobel Grid and which is used, adapted and improved in WiseGRID for the purpose of the new services. It is useful to understand and take the opportunity for using such meters not only for billing but also for technical: monitoring and automations purpose and also for finding additional options of new types of transactions within evolving market segments. The concept considers a flexible Smart Meter which has two essential parts:

- A metrology part of the Smart Meter, named Smart Metrology Meter or SMM, which is metrologically enforced and records energy data which can be used for billing, thus for legal operation regarding the energy quantities
- A business-oriented part, which is flexible and upgradable and which can support various business agents, various communication connections and various protocols, named Smart Meter eXtension (SMX).

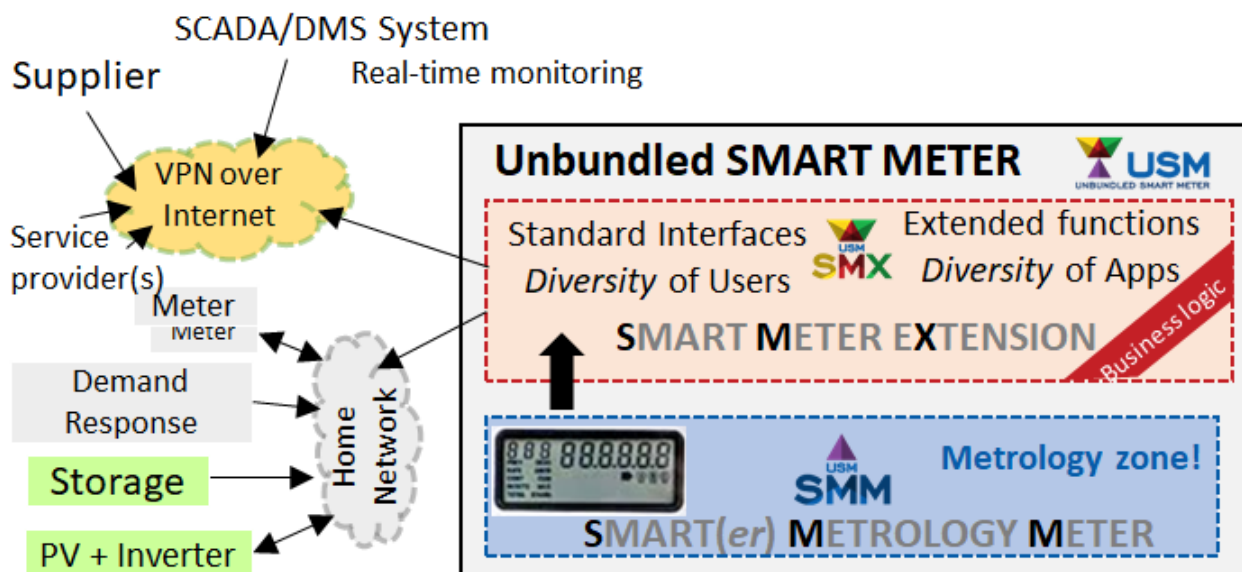


Figure 17 – Unbundled Smart Meter architecture

The Unbundled Smart meter is used in one of the two options, either with an existing meter acting as a stand-alone SMM plus a Smart Meter Extension (SMX) connected to it, or as an all-in-one USM designed from scratch, which is the SLAM model developed also in Nobel Grid.

The figure shows that SMX is able to serve SCADA/DMS systems (grid system operations), by providing real-time monitoring data.

Such data wrapping can be implemented with different types of protocols. The most flexible protocol which can be used, which includes information for mapping to both DLMS/COSEM related applications (mostly for billing), such as AMR systems, but also to SCADA systems, which may be CIM (Common Information Model) oriented systems is MQTT.

In the ANNEX A specifically in the paragraph 10.6 is available an example of how SMX provides DLMS/COSEM publications.

It can be seen that real-time measurements, such as voltages on each phase, are packed in a JSON format which contain the COSEM code of voltages, timestamp of the measurement and also the CIM code associated to the measurements.

Such multi-protocol mapping allows that the same information can be sent to any interested and eligible actor, and that at the level of the receiver it can be made the appropriate mapping for most suitable data usage.

With this solution, relevant data can be sent for monitoring purposes e.g. each 1 to 10 seconds, to a Smart Meter Wrapper (SMW) having its own MQTT broker where the SMX is publishing, depicted in Figure 18 below.

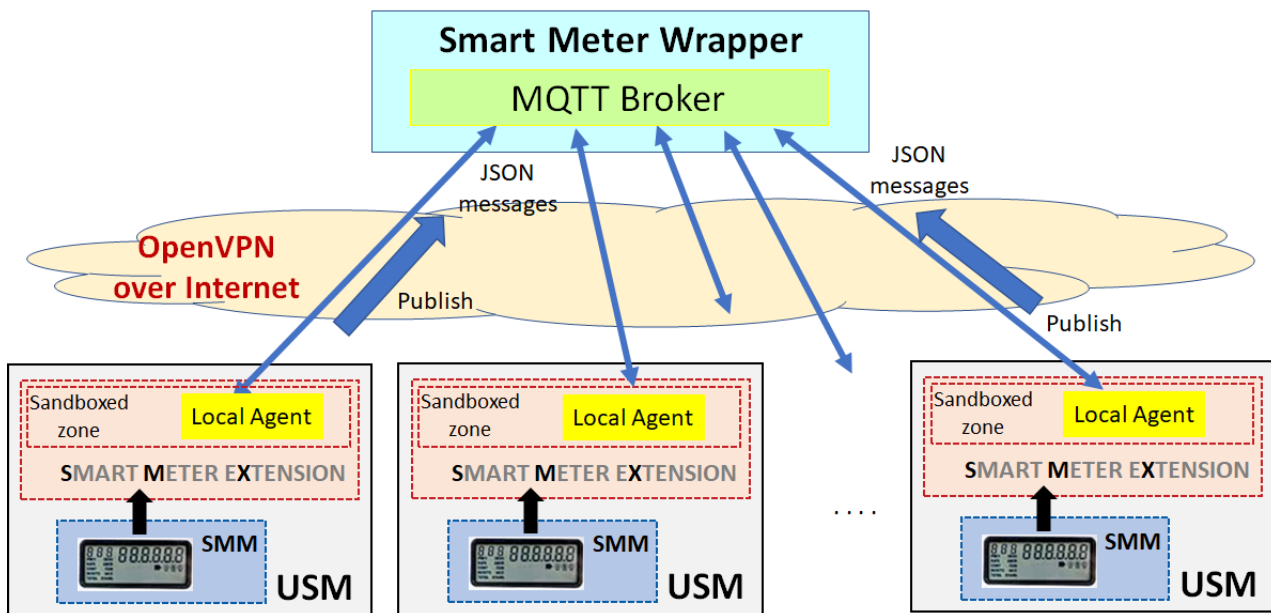


Figure 18 – Each USM is subscribing to the broker of the Smart Meter Wrapper (SMW)

In the picture can be seen that local agents in USM are publishing the data in the MQTT broker of SMW. The communication is made in a secure OpenVPN specific for the SMW operation.

The second possibility is that SMW does not have a broker but instead is subscribing to each of the USM's, in order to receive real-time data. It can be seen that an MQTT broker is needed in each USM, and that data in the broker is given by a local agent, as per Figure 19 below.

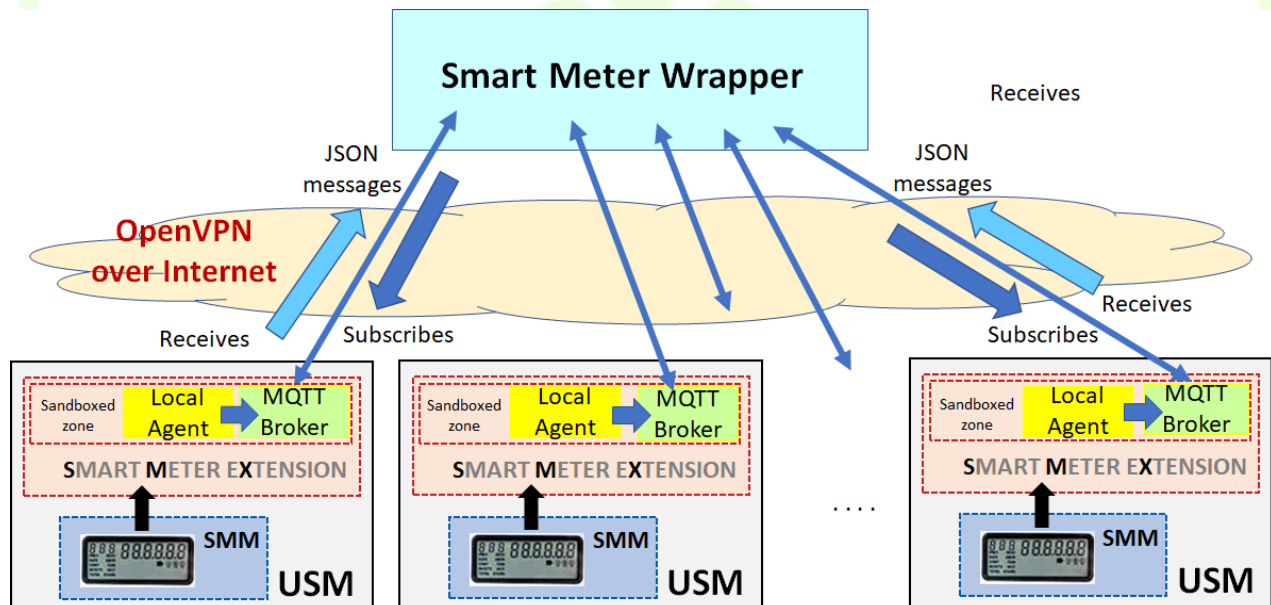


Figure 19 – Smart Meter wrapper subscribing to each USM

In the figure the communication is also over the secure OpenVPN.

In all situations it is advisable that the process which is exposed to Internet shall run in a sandboxed environment, e.g. in a Docker session, also suggested in the two figures above.

Each variant has advantages and disadvantages:

Variant 1 (Broker in SMW):

Advantages:

- No broker for SMW communication is needed in the USMs, which makes a lighter footprint of the used resources in the deployed USMs;
- The SMW need only one connection, to its broker, in order to get access to all data from the Smart Meters (USMs)

Disadvantages:

- Data in the MQTT broker of the SMW may be exposed to other end-users (all have access to the same broker, and may have access to other data in the broker); a good policy for setting the rights of each actor need to be implemented in the broker

Variant 2 (Brokers are in USM):

Advantages:

- Different end-users have by design no access to the streamed data of the others; only SMW has access to each USM broker, with a different client for each USM. In this way, a better privacy policy can be guaranteed by the SMW owner.

Disadvantages:

- We need a broker in each USM, which will take some hardware resources
- It is needed in SMW a client for each USM, which may ask for a higher hardware requirement at this level

As pointed before, the flexibility of USM and the possibility to make upgrades in time allow that both solutions can be implemented at initial deployment phase or during the operational phase.

From a practical point of view, SMX devices provide transparent communication with smart meters of different vendors, giving energy readings in a unified format. Within the WiseGRID project, the presented Variant 1 will be implemented, thus allowing SMX devices to directly Interact with the IOP using MQTT. SMX regularly publishes the energy readings and subscribes to topics where configuration updates shall be received.

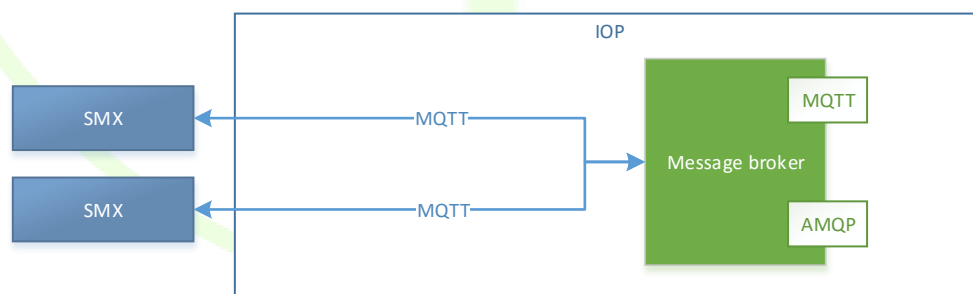


Figure 20 – SMX Wrappers communication schema

Translation to CIM MeterReading Objects

In order to comply with the CIM common data model adopted by the project, a module exists in the WG IOP with that subscribes to the SMX energy reading MQTT topics, translates those items into the corre-

sponding CIM messages on-the-fly and republishes data under a different MQTT topic. These messages are JSON and contain the whole list of readings referred to a single timestamp (example is given below).

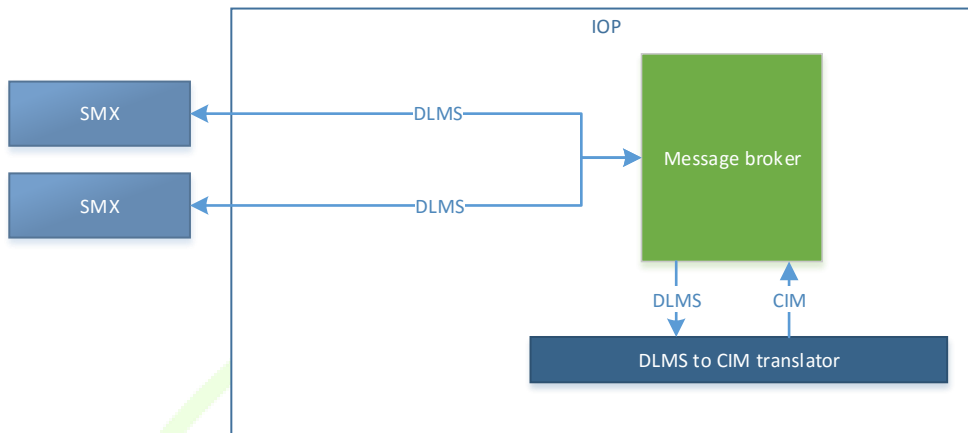


Figure 21 – DLMS to CIM translator

The specific message is available in the ANNEX A , 10.7 Translation to CIM MeterReading Objects message.

6.1.3.2 Integration of AMI systems

AMI systems typically collect data from the deployed smart meters on a regular basis (e.g. 24 hour demand curves retrieved once per day). In order to integrate this data with the ecosystem of WiseGRID applications, it will be published following the exact same mechanisms (MQTT publications) and data model (CIM MeterReadings) as defined for the Unbundled Smart Meters, thus making the actual source of information transparent to the application.

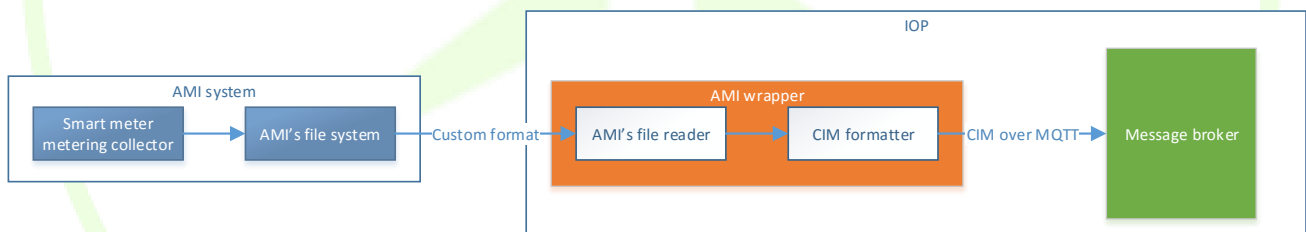


Figure 22 – AMI Wrappers communication schema

Since source data will be provided in a custom format (depending on the vendor of the deployed AMI system), different AMI wrappers will need to be developed for each one of the pilot sites.

6.1.3.2.1 SITEL STGTedis AMI (Crevillent)

STGTedis is the commercial solution already used by the DSO in Crevillent to collect data from all smart meters installed in the town. The tool is already configured to read hourly curves from each of the devices every 24 hours.

```
<Report IdRpt="S02" IdPet="0" Version="3.1.c">
  <Cnc Id="SAGxxxxxxxxxx">
    <Cnt Id="ZIVxxxxxxxxxx" Magn="1">
      <S02 Fh="20180118040000000W" Bc="00" AI="933" AE="0"
R1="1054" R2="0" R3="0" R4="0"/>
    </Cnt>
  </Cnc>
</Report>
```

```

<S02 Fh="20180118050000000W" Bc="00" AI="703" AE="0" R1="714"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118060000000W" Bc="00" AI="609" AE="0" R1="591"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118070000000W" Bc="00" AI="624" AE="0" R1="613"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118080000000W" Bc="00" AI="644" AE="0" R1="600"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118090000000W" Bc="00" AI="848" AE="0" R1="566"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118100000000W" Bc="00" AI="1078" AE="0"
R1="578" R2="0" R3="0" R4="0"/>
<S02 Fh="20180118110000000W" Bc="00" AI="834" AE="0" R1="917"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118120000000W" Bc="00" AI="690" AE="0" R1="714"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118130000000W" Bc="00" AI="671" AE="0" R1="675"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118140000000W" Bc="00" AI="628" AE="0" R1="685"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118150000000W" Bc="00" AI="668" AE="0" R1="571"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118160000000W" Bc="00" AI="962" AE="0" R1="546"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118170000000W" Bc="00" AI="747" AE="0" R1="846"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118180000000W" Bc="00" AI="625" AE="0" R1="673"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118190000000W" Bc="00" AI="587" AE="0" R1="604"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118200000000W" Bc="00" AI="660" AE="0" R1="541"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118210000000W" Bc="00" AI="977" AE="0" R1="573"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118220000000W" Bc="00" AI="683" AE="0" R1="779"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180118230000000W" Bc="00" AI="516" AE="0" R1="542"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180119000000000W" Bc="00" AI="522" AE="0" R1="556"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180119010000000W" Bc="00" AI="503" AE="0" R1="549"
R2="0" R3="0" R4="0"/>
<S02 Fh="20180119020000000W" Bc="00" AI="604" AE="0" R1="533"
R2="0" R3="0" R4="0"/>
</Cnt>

```

```
</Cnc>
</Report>
```

This files will be automatically generated for a selection of customers in a folder accessible via SFTP in a daily manner. The specific AMI wrapper for this pilot site will:

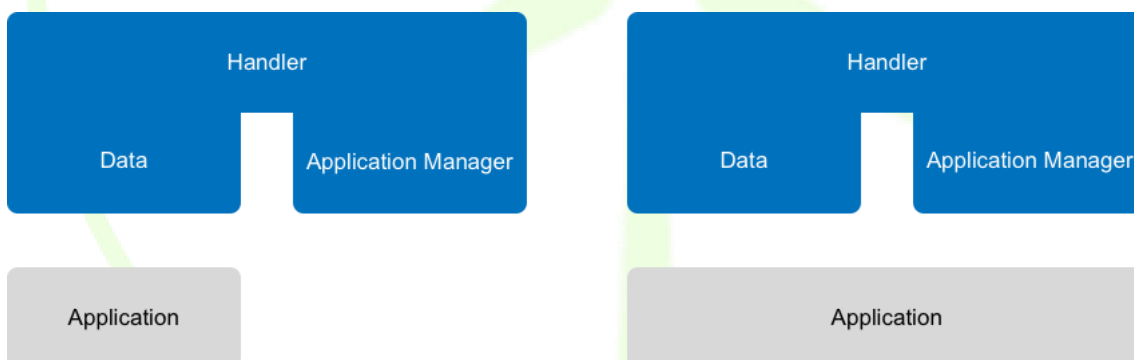
- Periodically connect to the SFTP server.
- Process the available files.
 - by translating the contents to CIM Energy Reading messages and
 - publishing those messages to the IOP using MQTT.
- Delete processed files.



Figure 23 – Integration scheme of AMI system for Crevillent pilot site

6.1.3.2.2 LoRa Gateway (Flanders)

Apart from the primary SMX datasource, a LoRaWAN based gateway on the new Flanders Digital Meters (rollout due 1 January 2019) is in procurement process. The exact outcome is on this moment not definitive, but the gateways will likely be connected to the Things Network (<https://www.thethingsnetwork.org/docs/applications/apis.html>). Two API based data-integration options are available of which one will be implemented:



- 1) Data API
The Data API allow you to receive events and messages from devices as well as send messages to devices. You can work with the Data API using SDKs or directly through MQTT.
- 2) Application Manager API
The Application Manager API lets you manage applications, gateways and devices. You can work with the Application Manager API using SDKs or directly through HTTP.

6.1.3.2.3 Terni

In the pilot site of Terni, a set of existing smart meters – currently integrated within their AMI system, will be configured to communicate with SMX devices, as described in section 6.1.3.1.

The setup is depicted in the figures below:

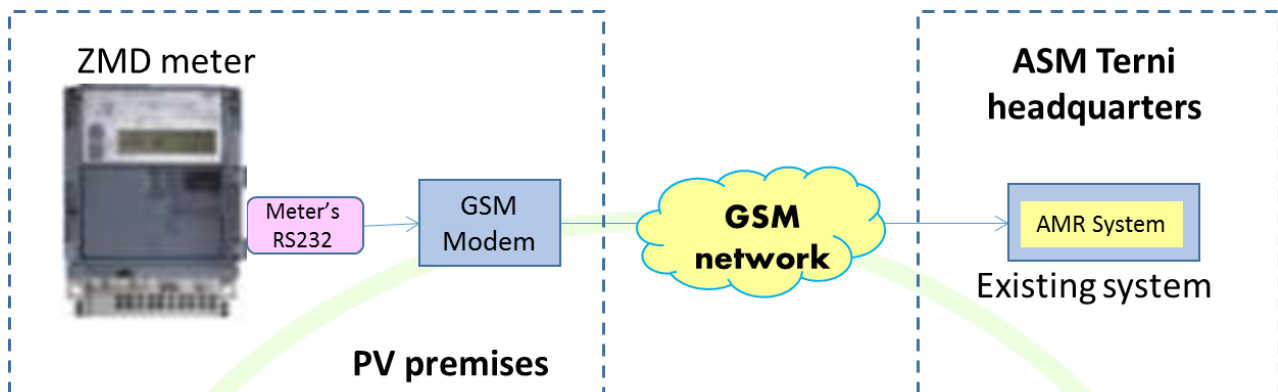


Figure 24 – Initial connection of the meters in Terni to the AMR system

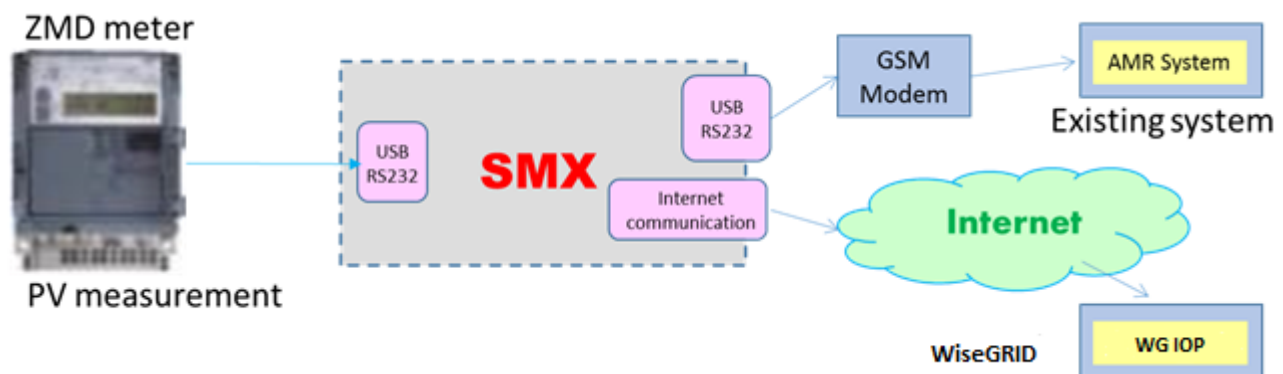


Figure 25 – Functional connection of SMX to the meters in Terni and to the different actors

Figure 24 shows the initial setup, with the ZMD smart meter read directly by the ASM Terni AMR system, by using the RS232 serial interface of the meter and the GSM Modem to connect to the AMR (Automatic Meter Reading) system.

The solution for integrating ZMD meters with PV measurement in SMX and further applications is presented in Figure 25. For this, the meter's RS232 serial interface is connected to an SMX USB/RS232 interface and the GSM modem connects to a second USB/RS232 interface. Moreover, an Internet connection is available for connection to WiseGRID applications via the WiseGRID IOP. SMX implements the corresponding wrapper in charge of reading each second the data from the meter and converting it to the Common Data Model before publishing information to the WiseGRID IOP.

6.1.3.2.4 Mesogia

With regards to the AMI wrapper for the Greek demo sites, data from the smart meters available re collected offline from the AMI systems following the procedure described below.

- Data from the smart meters available are exported manually from the AMI system periodically every week in the form of CSV files.
- CSV file are then compressed and uploaded to an FTP server in a specific directory with a pre-defined formatting of the CSV file.
- A micro-service is triggered automatically upon the upload of the CSV files.

- CSV files are decompressed and data checking and cleansing, as well as missing values analysis and correction is performed.
- Data from each customer are then used to construct the respective JSON file to be send automatically to the WiseGRID IOP

The typical structure of a JSON file is shown below.

```
{
  "meterId": "1010",
  "timestamp": "2018-04-16T20:09:06+00:00",
  "values": {
    "voltage": "223.9",
    "current": "10.2",
    "pf": "98.2",
    "active": "120",
    "reactive": "20"
  }
}
```

6.1.3.3 SCADA

In the context of the WiseGRID project, SCADA systems monitor the status of the different elements in the distribution grid under control of the DSO. The information that is particularly relevant to the WiseGRID Cockpit comprises:

- Electric measurements from substations: electric parameters of the buses of the grid are typically measured by equipment located in the substations. These measurements will be published to the WG IOP using the already described CIM MeterReading messages.
- Status of safety elements(switches). Algorithms to be executed within the WiseGRID Cockpit need to be aware of the current status and changes in those elements. The status will be read from the corresponding SCADA and published to the WG IOP using CIM Switch messages.

```
{
  mRID: "SW00001"
  normalOpen: true
}
```

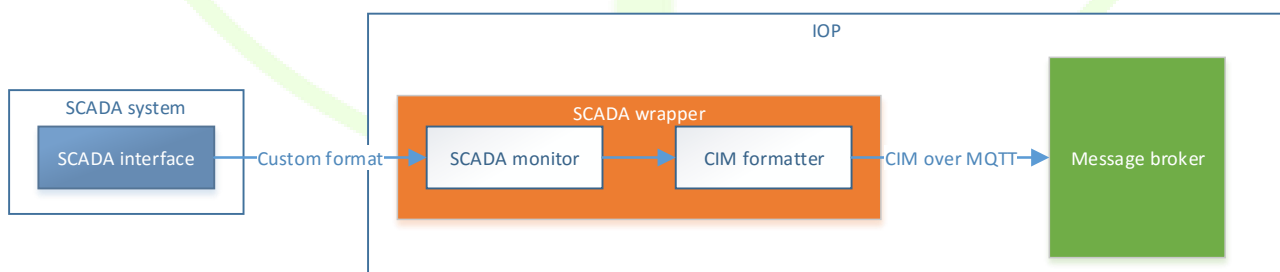


Figure 26 – SCADA Wrappers communication schema

Different SCADA wrappers will need to be developed for each one of the pilot sites.

6.1.4 Battery wrapper

For the communication of the batteries to the WG IOP and to the WG StaaS/VPP the MQTT protocol is used. It allows the use of publish/subscribe communication mechanism which enables asynchronous communication for monitoring and control purposes.

In WiseGRID batteries regularly publish their real-time status to the MQTT broker implemented within the WG IOP. Then the data are further transmitted to the RT monitor of WG StaaS/VPP and stored in a database. The implemented structure is also used to send control data from the scheduler in WG StaaS/VPP to the battery storage systems. In order to do so the scheduler publishes the control commands and the battery storage systems subscribe to the specific topics.

In order to translate vendor-specific protocols to a defined common data model that has been designed by the partners battery wrappers are implemented. This way any battery deployed in WiseGRID can be monitored and controlled in the same way, no matter which vendor it belongs to. The specification of the mentioned data model can be found in the deliverable D6.2 “Storage as a service and Innovative optimized solutions”. Whereas AMP and VARTA implemented the Battery Wrapper directly on their storage devices, BYES included it in the SMX which is also used for the smart metering. The overall structure of the communication paths is shown in the following figure.

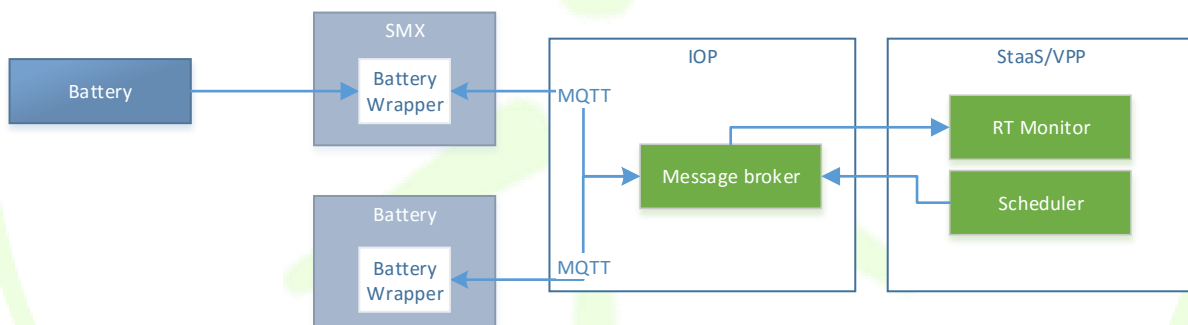


Figure 27 – Overview of communication between batteries and WG StaaS/VPP via WG IOP

Details regarding the vendor specific implementation of the MQTT clients and battery wrappers can be found in deliverable D6.2 [17]. Specification of these messages can also be found in ANNEX A, “Battery Wrapper messages”

6.1.5 Domestic/building Assets wrapper

The communication between the BMS wrapper and the rest of the WiseGRID application needs to be standardized to have the best simplified programming on the rest of the application modules. To keep a standardized method, the MQTT communication has been chosen for the BMS wrapper. All kind of communications that can be encountered, such as Modbus TCP, OPC UA or BACnet are “translated” into MQTT communication.

The Figure 28 depicts the building assets considered in the BMS. Each wrapper communicates with the different equipment and will give the needed information to the RT monitor.

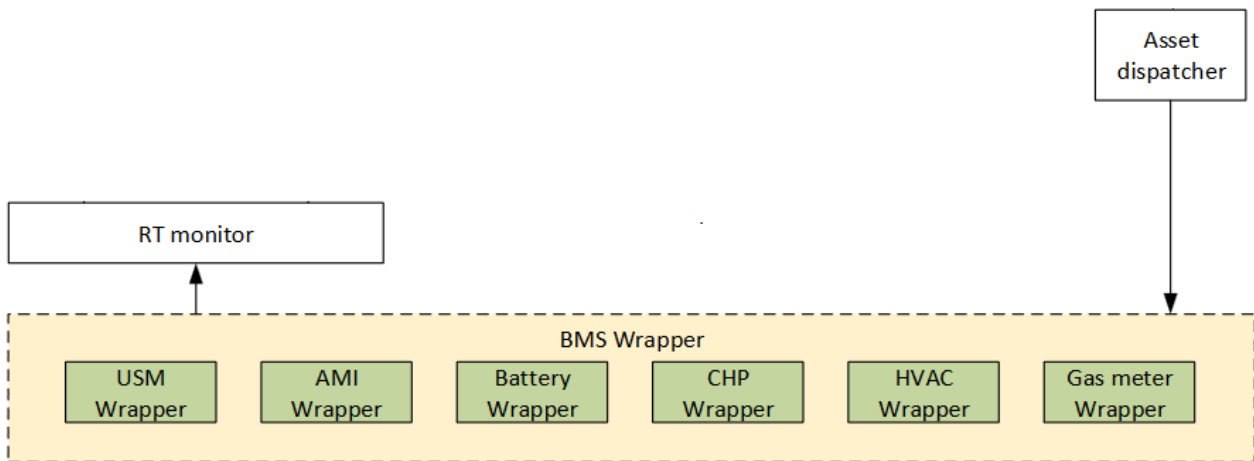


Figure 28 – Building assets integration architecture

The communication is reached by introducing a common gateway linked to the IOP application:

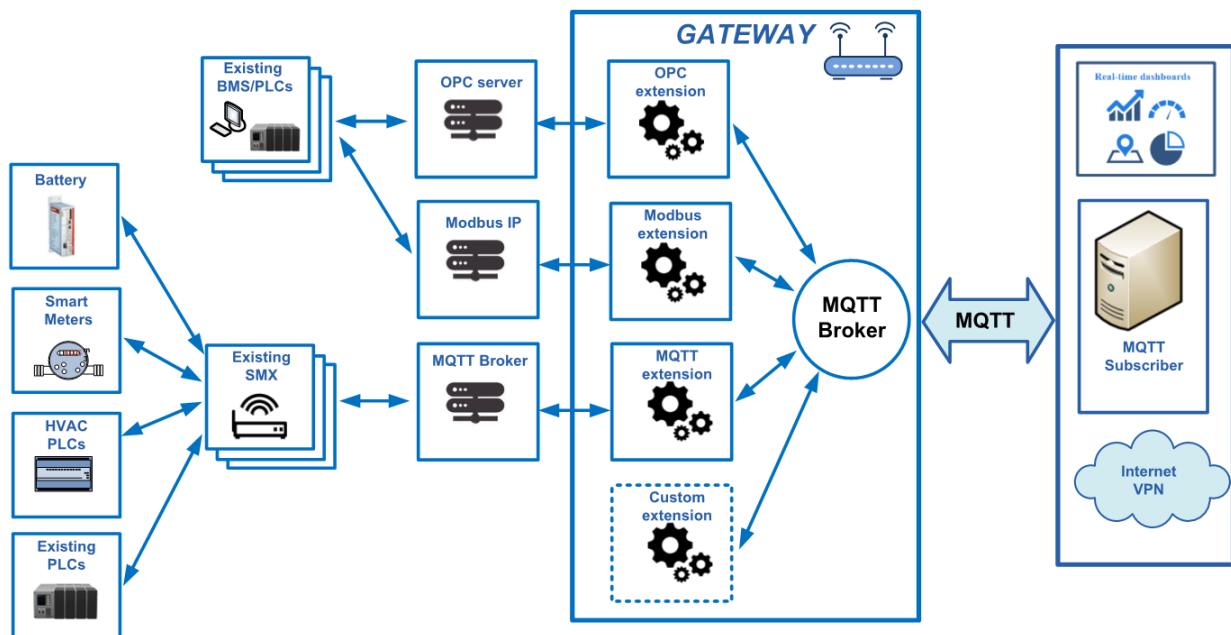


Figure 29 – BMS wrapper architecture

For some pilot sites on the WiseGRID project, SMX (PLC from Nobelgrid project) are available and already linked to some equipment. On the WiseGRID project, those equipment are reused since MQTT communication are already developed. On a more general way, the SMX are reused for new equipment if it is technically possible.

More detail about the building assets integration and the messages structure exchanged is available in the D7.1 [18] .

6.2 ANCILLARY SERVICES MARKET

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

1. Flexibility requests generation
2. Reception of flexibility offers
3. Offer selection
4. Offer revocation

6.2.1 Flexibility requests generation

Upon forecast of a congestion problem in the grid, the DSO evaluates the necessary actions (increase/decrease of demand in a certain area of the grid) and publishes accordingly a flexibility request. IOP is configured in such a way the request reaches each one of the aggregators participating in the platform.

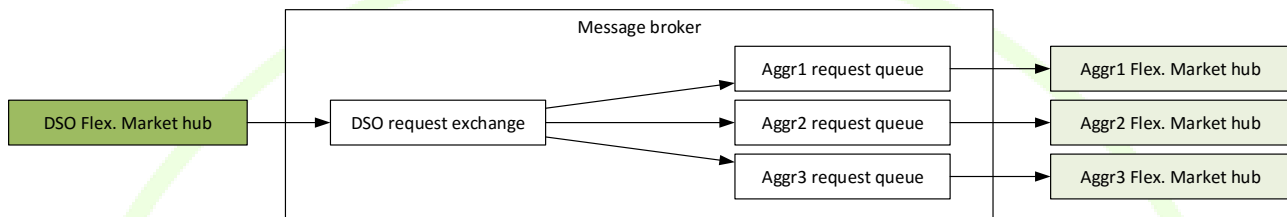


Figure 30 – Flexibility requests generation

AMQP exchange name: “flexReqs”

Message properties:

Reply_to: “flexOffers”

Correlation_id

Message payload: FlexRequest

Table 3 – Flexibility request properties

Section	Parameter	Description
FlexRequest		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.
	FlexRequest.PTU-Duration	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. The project will use PTU duration of 15 minutes.
	FlexRequest.Period	Day (in yyyy-mm-dd format) the PTUs referenced in this Flex* message belong to.
	FlexRequest.TimeZone	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. Time-Zone will be a fixed value per pilot site.
	FlexRequest.CongestionPoint	Entity Address of the Congestion Point this Flex* message applies to.
	FlexRequest.Sequence	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.
	FlexRequest.ExpirationDateTime	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. DSO will only process offers received before this expiration timestamp.
PTU		The PTU element represents one or more Program Time Units.
	PTU.Disposition	Indication whether the Power specified for this PTU represents available capacity or a request for reduction/increase (Valid value: "Requested"). WiseGRID Cockpit will only produce requests for reduction/increase of power.
	PTU.Power	Power specified for this PTU in Watts. A positive value indicates that power flows towards the Prosumer (consumption), a negative value indicates flow towards the grid (production).
	PTU.Start	Number of the first PTU this element refers to. The first PTU of a day has number 1.
	PTU.Duration	The number of the PTUs this element represents. Optional, default value is 1.

Reception of flexibility offers

Each one of the aggregators will process the request, evaluate the feasibility of responding to it (accordingly to the available resources), and finally post an offer, describing up to which extent they can support the DSO, and the associated price to that action

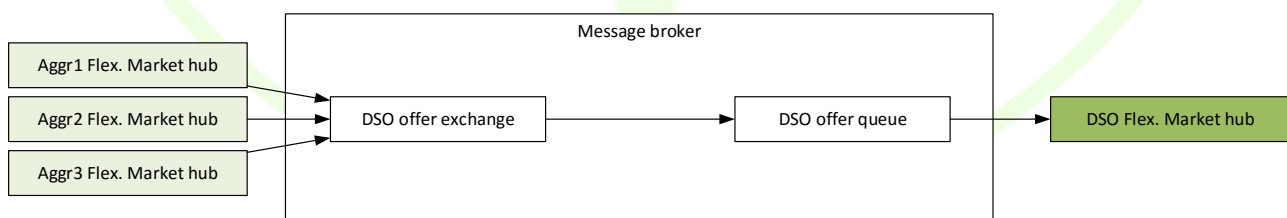


Figure 31 – Reception of flexibility offers

Queue: "FlexOffers" (accordingly to the "reply_to" property of the FlexRequest)

Message properties:

reply_to: name of the queue where the aggregator expects the offer or rejection

Correlation_id: correlation id of the flex request

Table 4 – Flexibility offer properties

Section	Parameter	Description
FlexOffer		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.
	FlexOffer.FlexRequestSequence	Sequence number of the FlexRequest message this request is based on. The combination of FlexRequestOrigin and FlexRequestSequence should be unique.
	FlexOffer.Currency	ISO 4217 code indicating the currency that applies to the prices listed for each PTU (EUR for all pilot sites)
	FlexOffer.PTU-Duration	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. The project will use PTU duration of 15 minutes.
	FlexOffer.Period	Day (in yyyy-mm-dd format) the PTUs referenced in this Flex* message belong to.
	FlexOffer.TimeZone	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. TimeZone will be a fixed value per pilot site.
	FlexOffer.CongestionPoint	Entity Address of the Congestion Point this Flex* message applies to.
	FlexOffer.Sequence	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.

	FlexOffer.ExpirationDateTime	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. DSO will only process offers received before this expiration timestamp.
PTU		The PTU element represents one or more Program Time Units.
	PTU.Power	Power specified for this PTU in Watts. A positive value indicates that power flows towards the Prosumer (consumption), a negative value indicates flow towards the grid (production).
	PTU.Start	Number of the first PTU this element refers to. The first PTU of a day has number 1.
	PTU.Duration	The number of the PTUs this element represents. Optional, default value is 1.
	PTU.Price	The price offered or accepted for supplying the indicated amount of flexibility in this PTU.

6.2.2 Offer selection

DSO waits until the validity time of the request is reached, and evaluates the set of received offers. Upon decision, the order/rejection to the corresponding aggregators is sent.

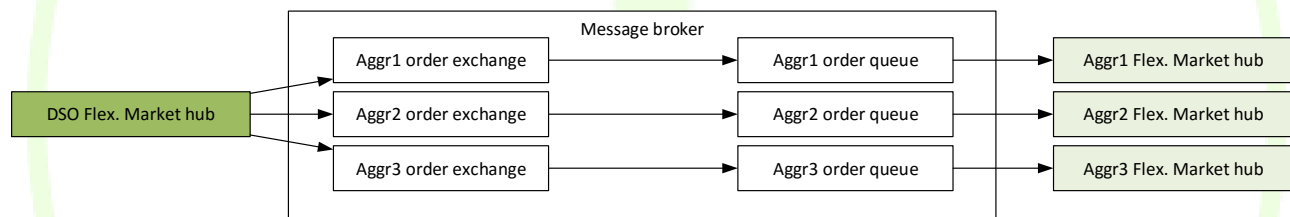


Figure 32 – Offer selection

Queue: accordingly to reply_to parameter of the offer

Payload: FlexOrder

Table 5 – Flexibility order properties

Section	Parameter	Description
FlexOrder		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.
	FlexOrder.Currency	ISO 4217 code indicating the currency that applies to the prices listed for each PTU (EUR for all pilot sites)
	FlexOrder.PTU-Duration	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 val-

		ue, making this assumption explicit in each message is important for validation purposes, allowing implementations to reject messages with an errant PTU duration. The project will use PTU duration of 15 minutes.
	FlexOrder.Period	Day (in yyyy-mm-dd format) the PTUs referenced in this Flex* message belong to.
	FlexOrder.TimeZone	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. TimeZone will be a fixed value per pilot site.
	FlexOrder.CongestionPoint	Entity Address of the Congestion Point this Flex* message applies to.
	FlexOrder.Sequence	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.
	FlexOrder.ExpirationDateTime	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. DSO will only process offers received before this expiration timestamp.
	FlexOrder.OrderReference	Order number assigned by the BRP or DSO originating the FlexOrder. To be stored by the Aggregator and used in the settlement phase.
PTU		The PTU element represents one or more Program Time Units.
	PTU.Power	Power specified for this PTU in Watts. A positive value indicates that power flows towards the Prosumer (consumption), a negative value indicates flow towards the grid (production).
	PTU.Start	Number of the first PTU this element refers to. The first PTU of a day has number 1.
	PTU.Duration	The number of the PTUs this element represents. Optional, default value is 1.
	PTU.Price	The price offered or accepted for supplying the indicated amount of flexibility in this PTU.

6.2.3 Offer revocation

Aggregator may revoke a previously sent offer, even if an order has been already posted based on this offer, if the application time has not been reached yet. DSO will reevaluate the alternatives if an order gets cancelled.

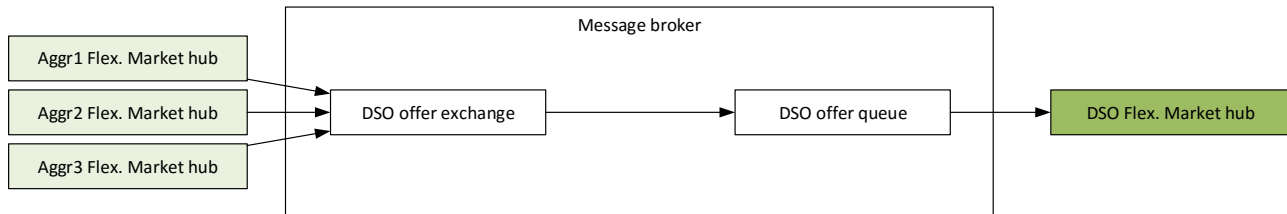


Figure 33 – Offer revocation

Queue: “FlexOffers” (accordingly to the “Reply_to” property of the FlexRequest)

Payload: FlexOfferRevocation

Table 6 – Flexibility offer revocation properties

Section	Parameter	Description
FlexOfferRevocation		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.
	FlexOfferRevocation.Sequence	Sequence number of the FlexOffer message that is being revoked.

Realization

The implementation of the workflow within the set of WiseGRID applications implies the development of 2 generic modules implementing the logic and allowing applications to interact in the ancillary services market:

6.2.4 DSO Flex. Market hub

This module will implement the DSO-side logic (to be used in the WG Cockpit). This module will receive congestion forecast details from the corresponding module of the WiseGRID Cockpit, and will trigger and handle the market-based request of ancillary services.

1. Congestion forecast is identified, and details are sent to the DSO Flex.Market Hub:
 - a. Location
 - b. Curve with required demand/production reduction
2. DSO Flex. Market hub produces the corresponding FlexReq
3. Upon reception of FlexOffers, the module will trigger a simulation (as per T4.1) in order to select the appropriate set of offer.
4. The required FlexOrder messages are sent and the process finishes

The results of each one of the steps will be stored within the WG Cockpit operational database.

6.2.5 Aggr. Flex. Market hub

This module implementing the aggregator-side logic (to be used in the WG StaaS/VPP, WiseCOOP and WiseEVP). Will provide a simple API to external modules to provide flexibility and pricing curves.

1. The market hub periodically receives the curves of forecasted available flexibility from the corresponding module of the application, with an associated price
2. Upon reception of a request (and upon updates on the available flexibility), the module checks feasibility, composes and sends offers back to the DSO
3. Upon reception of an order, the flexibility to be provided is sent to a configured AMQP queue of the internal ESB of the application, where the proper module implementing the control of the assets will be listening.

The formal definition of the messages described in this section is available in the ANNEX A , 10.9 Ancillary Services Market message specification.

6.3 WEATHER FORECASTING

The weather forecast provider is a module of the WG IOP implementing a wrapper of the public OpenWeatherMap [19] and SolCAST [20] APIs for easy integration of weather information with the WiseGRID tools.

6.3.1 Weather forecast

Provides weather forecast in the given location for the next 10 days, with a granularity of 3 hours.

AMQP queue: "weatherforecastprovider"

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>)

Message payload examples

```
{
  "query": "forecast",
  "city": "Valencia,ESP"
}
```

Or

```
{
  "query": "forecast",
  "lat": 39.46975,
  "lon": -0.37739
}
```

Or

```
{
  "query": "forecast",
  "cityId": "6362115"
}
```

City codes available here: <http://bulk.openweathermap.org/sample/city.list.json.gz>

Response example:

```
[
{
  "timestamp" : "2018-01-01T00:00:00.000Z",
  "temp" : 279.47,
  "pressure" : 977.52,
  "humidity" : 100,
  "clouds" : 92,
  "wind" : {
    "speed" : 2.67,
    "deg" : 16.506
  },
  "dhi" : 283,
  "azimuth" : 72,
  "zenith" : 37
},
...
]
```

The response includes values for:

- Temperature (K)
- Pressure (mbars)
- Humidity (%)
- Cloudiness (%)
- Wind speed (km/h)
- Wind direction (degrees respect to N)
- Diffuse Horizontal Solar Irradiance (W/m2)
- Solar azimuth angle (degrees). Zero means true north. Varies from -180 to 180. Positive is anticlockwise (west). A value of -90 means the sun is in the east.
- Solar zenith angle (degrees). Zero means directly upwards/overhead). Varies from 0 to 180. A value of 90 means the sun is at the horizon.
-

6.3.2 Current weather

Provides information about the current weather in the given location

AMQP queue: "weatherforecastprovider"

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>)

Message payload examples

```
{  
  "query": "weather",  
  "city": "Valencia, ESP"  
}
```

Or

```
{  
  "query": "weather",  
  "lat": 39.46975,  
  "lon": -0.37739  
}
```

Or

```
{  
  "query": "weather",  
  "cityId": "6362115"  
}
```

City codes available here: <http://bulk.openweathermap.org/sample/city.list.json.gz>

Response example:

```
{  
  "temp" : 280.15,  
  "pressure" : 1007,  
  "humidity" : 87,  
  "clouds" : 75,  
  "wind" : {  
    "speed" : 5.1,  
    "deg" : 350  
  },  
  "dhi" : 283,  
  "azimuth" : 72,  
  "zenith" : 37  
}
```

The response includes the same elements as the ones provided by the forecasting service.

6.4 ENERGY FORECASTING

For the energy forecasting within the WiseGRID project, it is going to be implemented two services to forecast the demand and the generation of energy. With the aim of providing forecasts of demand and energy generation, it have been developed two algorithms energy demand and generation demand, respectively. To perform their forecasts, both algorithms take into account historical data and additional data, such as a calendar (with information of working days, holidays and weekends), exogenous variables (weather) and some client parameters.

These services must communicate with the rest of the WiseGRID tools via WG IOP, and in consequence they have to make use of RabbitMQ services. Due to this fact, the forecast server will include an AMQP client, to route client request to the RPC forecasting functions, and to call the RPC function, which obtains the weather forecast. In the image down below it is defined the structure of the forecast server, and the communication scheme with the WG IOP.

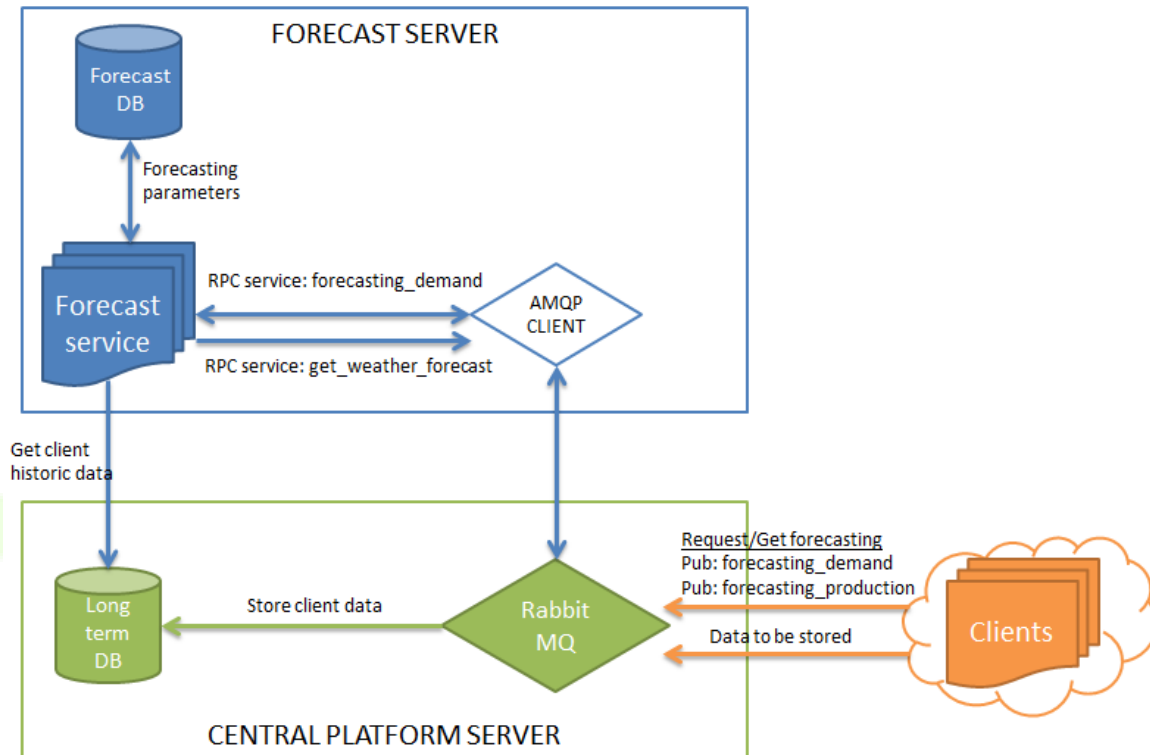


Figure 34 – Forecast server structure integration schema

As last image shows, the forecast database stores all the historical data about client's consumptions and energy generation, as well as historical weather values to be taken into account in the forecasts. At the same time, the long term data base is needed to store all the necessary historical data to perform the forecasts.

The forecasting algorithm has been implemented under CNEA (Cascaded Neuro-Evolutionary Algorithm) forecasting model premises, which are based on neural networks algorithms. The CNEA forecasting model consists on some SVM (Support Vector Machine) in cascade, with an optimal chose of input models, number of neurones and evolutionary training process. The iterative CNEA algorithm uses a SVM to forecast the power of the first hour ($h+1$), and this result is used as an input to the next SVM to forecast the power during the next hour ($h+2$). The algorithm repeats this process until the last forecasting slot is achieved. In consequence, it is required 24 SVM, executed in cascade to provide a forecast of 24 values.

To generate each forecast, the forecasting module needs to be provided an input of three values. These three values are the aggregator identifier, the period of time between two consecutive forecasting values, and the total window horizon for the forecast output.

At the same time, the algorithm has set their internal parameters that are automatically optimized to obtain the best perform. Within these parameters it is possible to find the number of past days to take into account for training, number of past hours for prediction and variables related to internal supervised learning and optimization algorithms.

For the production forecasting, in addition to all the necessary information stated in the previous paragraphs, it is necessary to be provided of weather forecast information. This information it will be

provided by the weather forecasting service. The rest of the necessary information for the forecast it will be retrieved from the forecast database and the long term database.

Both, demand and production forecast, are provided to WG IOP via the usage of a wrapper. This wrapper integrates these algorithms, providing their output to the WG IOP.

6.5 TARIFF PROVIDER

The tariff provider module is one of the horizontal modules developed within the project, 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.

6.5.1 Communication mechanism

The tariff provider module implements the asynchronous RPC messaging pattern by using AMQP protocol, and will be directly accessible from the WG IOP Message Broker. Further details are provided in the following section.

6.5.2 Tariff model

Tariffs are modelled by following a custom model based on work already taken with success in the previous FP7 project BESOS [21]. The main components of this model are hereby described.

A tariff plan is defined as a set of prices that are applied at specific periods of time. The current data model defines separately the rules that establish when a type of price is applied and the prices –and additional information– for each tariff plan that uses that price schema.

Two types of objects are thus defined:

- Rule
- Plan

6.5.2.1 Rule

Each Rule object defines a period of time when a type of price (e.g. peak, base, off-peak, etc.) is applied. Depending on the type of tariff and the complexity of application of a type of price, a specific Rule object can define periods lasting from minutes to always being applied. Moreover, a type of price may need one or several rules to fully describe its times of application.

6.5.2.1.1 Definition

The full –although expandable– Rule object has the following fields:

Table 7 – Rule objects

Field	Type	Description
ruleType	String	Type of tariff this rule applies to. Each tariff plan has a type of tariff associated (fixed, two periods, three periods, etc.) that defines the rules to apply each price.
priceType	String	Type of price to be applied during the period (15 minutes, hour, etc.). Here a name for each type of price is set, while the actual value of each price (in currency) will be defined on each tariff plan.
startMonth	Integer (1-12)	Number of the month when the rule starts being applied.
endMonth	Integer	Number of the month (inclusive) when the rule stops being applied.

	(1-12)	
startDay	Integer (1-31)	Day of the month when the rule starts being applied.
endDay	Integer (1-31)	Day of the month (inclusive) when the rule stops being applied.
startHour	Integer (0-23)	Hour of the day when the rule starts being applied.
endHour	Integer (0-23)	Hour of the day (inclusive) when the rule stops being applied.
startMinute	Integer (0-59)	Minute of the hour when the rule starts being applied.
endMinute	Integer (0-59)	Minute of the hour (inclusive) when the rule stops being applied.
saturday	Boolean	Indicates whether the rule is applied on Saturdays.
sunday	Boolean	Indicates whether the rule is applied on Sundays.
nationalHoliday	Boolean	Indicates whether the rule is applied on national holidays.
specialHoliday	Boolean	Indicates whether the rule is applied on special holidays (used in Spain for January 6th, open to other meanings).
dst	Boolean	Indicates whether the rule is applied during Daylight Savings Time (i.e. Summer time).

Considerations:

- Only the first three fields (ruleType, priceType, and site) are required – although site could eventually be set as optional further in the project.
- The field ruleType is not called tariffType for uncoupling reasons as, in practice, one rule can apply to several tariff types.
- When one of the other fields is not defined, that specific criteria is not taken into consideration, e.g.:
 - If saturday: true, the rule only applies to Saturdays.
 - If saturday: false, the rule applies on any day of the week except on Saturdays.
 - If saturday is not defined, the rule will be applied based on a different criteria, not taking into account whether it is Saturday or not.
- All criteria defined is considered using a logical AND, e.g.:
 - If {sunday: true, startHour: 11, endHour: 13}, the rule applies from 11 am to 1 pm on Sundays.
 - If {sunday: true, nationalHoliday: true, startHour: 11, endHour: 13}, the rule applies from 11 am to 1 pm on Sundays that are also national holidays.
 - If the rule must apply from 11 am to 1 pm on Sundays and/or on national holidays, two different rules must be defined:
 - {sunday: true, startHour: 11, endHour: 13}

- {nationalHoliday: true, startHour: 11, endHour: 13}

- The recommended approach is to build a truth table per tariff type that maps every possible variable to a price type, then grouping them to find the minimum set of rules that fully defines the application times for each price (see examples below).

6.5.2.1.2 Examples

Based on a subset of tariff types in Spain, some examples are provided:

Single-price tariff

For a tariff type (e.g. "2.XA") that sets one unique price (e.g. "p1") for every time of day and every day of the year, a single rule would be enough to define it:

{ruleType: "2.XA", priceType: "p1"}

As no filtering criteria is defined, no differentiation is performed, thus being the price "p1" always applied.

Two-period tariff

Having the following tariff where two periods of time are defined and they move one hour depending on the time of the year:

	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	20-21	21-22	22-23	23-24
WINTER																								
SUMMER																								

And having:

- "summer" represents Daylight Savings Time period
- "winter" represents the rest of the year
- Red squares represent peak hours ("p1")
- Green squares represent off-peak hours ("p2")

Three rules per period would be necessary to fully define the schema ("site" is omitted for legibility reasons):

{ruleType: "2.XDHA", priceType: "p2", dst: false, endHour: 11}

{ruleType: "2.XDHA", priceType: "p1", dst: false, startHour: 12, endHour: 21}

{ruleType: "2.XDHA", priceType: "p2", dst: false, startHour: 22}

{ruleType: "2.XDHA", priceType: "p2", dst: true, endHour: 12}

{ruleType: "2.XDHA", priceType: "p1", dst: true, startHour: 13, endHour: 22}

{ruleType: "2.XDHA", priceType: "p2", dst: true, startHour: 23}

Please note that startHour cannot be greater than endHour, which makes it necessary to define three rules per period instead of two.

Six-period tariff

Finally, having the following tariff type with six different price types:

	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	20-21	21-22	22-23	23-24
JAN					P6					P2		P1				P2				P1				P2
FEB					P6					P2		P1				P2				P1				P2
MAR					P6					P2		P1				P2				P1				P2
APR					P6											P5								
MAY					P6											P5								
JUN					P6					P4		P3				P4				P3				P4
JUL					P6					P4		P3				P4				P3				P4
AUG					P6					P4		P3				P4				P3				P4
SEP					P6					P4		P3				P4				P3				P4
OCT					P6					P4		P3				P4				P3				P4
NOV					P6					P2		P1				P2				P1				P2
DEC					P6					P2		P1				P2				P1				P2

Where:

- The table rows represent the twelve months of the year.
- Additionally, P6 is applied on Saturdays, Sundays and holidays.

P6 can be defined as ("site" is omitted):

```
{ruleType: "6.XTD", priceType: "p6", nationalHoliday: true}
```

```
{ruleType: "6.XTD", priceType: "p6", specialHoliday: true}
```

```
{ruleType: "6.XTD", priceType: "p6", saturday: true}
```

```
{ruleType: "6.XTD", priceType: "p6", sunday: true}
```

```
{ruleType: "6.XTD", priceType: "p6", nationalHoliday: false, specialHoliday: false, saturday: false, sunday: false, endHour: 7}
```

While P1 is defined as:

```
{ruleType: "6.XTD", priceType: "p1", nationalHoliday: false, specialHoliday: false, saturday: false, sunday: false, endMonth: 3, startHour: 10, endHour: 13}
```

```
{ruleType: "6.XTD", priceType: "p1", nationalHoliday: false, specialHoliday: false, saturday: false, sunday: false, endMonth: 3, startHour: 18, endHour: 21}
```

```
{ruleType: "6.XTD", priceType: "p1", nationalHoliday: false, specialHoliday: false, saturday: false, sunday: false, startMonth: 11, startHour: 10, endHour: 13}
```

```
{ruleType: "6.XTD", priceType: "p1", nationalHoliday: false, specialHoliday: false, saturday: false, sunday: false, startMonth: 11, startHour: 18, endHour: 21}
```

Please note that every rule for P1 must include the variables for Saturdays, Sundays and holidays set to false, as they are relevant for defining the rule.

6.5.2.2 Plan

While Rule objects identify the types of price to be applied on each period of time, the actual prices (in currency) are further defined using Plan objects. Each tariff Plan is related to a set of rules and provides information on the prices, conditions and other settings that identify it.

6.5.2.2.1 Definition

The basic Plan object has the following fields:

Table 8 – Plan object

Field	Type	Description
planName	String	Commercial name of the plan
company	String	Name of the retailer offering the plan
tariffType	String	Type of the tariff plan (not to be confused with next field)
ruleType	String	Identifier of the rules that define the price type appliance of the plan. Must match a value set in ruleType of Rule.
site	String	Pilot site the plan applies to.
startDate	Date	Date when the plan was made available to the general public by the retailer
endDate	Date	Date when the plan will no longer be available to the general public
active	Boolean	Whether the plan is currently being considered by the system
minActivePower	Integer (kW)	Minimum active power a customer has to contract to be able to adopt the plan
maxActivePower	Integer (kW)	Maximum active power a customer has to contract to be able to adopt the plan
minKwhPerYear	Integer (kWh)	Minimum energy a customer must consume per year to be able to adopt the plan
maxKwhPerYear	Integer (kWh)	Maximum energy a customer must consume per year to be able to adopt the plan
activePowerPrices	Array of ActivePowerPrice	List of prices associated to the contracted active power (see definition and examples below)
energyConsumptionPrices	Array of EnergyConsumptionPrice	List of prices associated to the periods of consumption as defined in the Rule objects (see definition and examples below)
activePowerDiscount	Double (%)	Discount (before taxes) performed over the total cost associated to the contracted active power
energyConsumptionDiscount	Double (%)	Discount (before taxes) performed over the total cost associated to the consumption
fixedPrices	Boolean	Whether the plan has fixed prices, i.e. they do not vary on a daily basis
green	Boolean	Whether the plan has associated energy coming exclusively from green sources
url	String	URL with additional information of the plan

Price for contracted capacity

The data model can allocate two different ways to define the contracted capacity, in order to accommodate the differences found on the definition in different countries.

Model 1: Fixed price per contracted kW

A fixed price is paid for the contracted active power, calculated as:

*Price * Days * kW contracted*

Some tariff types allow contracting different powers for different periods, which coincide with the periods when the price of the energy varies. This led to the definition of the following structures for ActivePowerPrice and EnergyConsumptionPrice objects:

```
{
...
  activePowerPrices: [
    {type: "p1", price: 0.108720633},
    {type: "p2", price: 0.054407367},
    {type: "p3", price: 0.039817167},
    {type: "p4", price: 0.039817167},
    {type: "p5", price: 0.039817167},
    {type: "p6", price: 0.018167167}
  ],
  energyConsumptionPrices: [
    {type: "p1", price: 0.167879},
    {type: "p2", price: 0.135136},
    {type: "p3", price: 0.109557},
    {type: "p4", price: 0.088137},
    {type: "p5", price: 0.079538},
    {type: "p6", price: 0.063772}
  ],
...
}
```

Model 2: Price per contracted block

This different structure is used to accommodate schemes with the following peculiarities:

- A different price was set depending on the power contracted, instead of giving a fixed price to be multiplied by the kW.

- Several tariff plan presenting different modalities of prices: single price, two prices, or three prices. All three had to be considered as different plans that shared everything else (name, conditions, discounts, active power prices, etc.).

This led to the definition of the following structures for `ActivePowerPrice` and `EnergyConsumptionPrice` objects:

```
{
...
  activePowerPrices: [
    {activePower: 6900, price: 0.265973},
    {activePower: 10350, price: 0.398795},
    {activePower: 13800, price: 0.531616},
    {activePower: 17250, price: 0.664767},
    {activePower: 20700, price: 0.797589}
  ],
  energyConsumptionPrices: [
    {timePlan: "simple", type: "p1", price: 0.1635},
    {timePlan: "bi", type: "p1", price: 0.2061},
    {timePlan: "bi", type: "p2", price: 0.095},
    {timePlan: "tri", type: "p1", price: 0.3298},
    {timePlan: "tri", type: "p2", price: 0.1696},
    {timePlan: "tri", type: "p3", price: 0.095}
  ],
...
}
```

6.5.3 Implemented services

6.5.3.1 Access to tariff list

The module implements a service to query the list of registered tariffs

AMQP queue: "tariffprovider"

Message properties

- `reply_to`: name of the queue where response will be delivered
- `correlation_id`: free text for query/response correlation

Message payload example

```
{
  "query": "tariffs"
}
```

Response example

The response includes an array of tariffs, each one defined by the following elements:

- Identification of the tariff
- Company marketing the tariff
- URL where further information may be found

```
[
  {
    "tariff": "EDP 3.0A",
    "company": "EDP",
    "url": "http://www.edpenergia.es/recursosdp/doc/portal-
clientes/20130827/precios/tarifas-electricas-para-empresas.pdf"
  },
  {
    "tariff": "EDP 6.1A",
    "company": "EDP",
    "url": "http://www.edpenergia.es/recursosdp/doc/portal-
clientes/20130827/precios/tarifas-electricas-para-empresas.pdf"
  },
  ...
]
```

6.5.3.2 Access to tariff prices

The response format is a curve of hourly prices for energy, power and other fixed costs. All fixed monthly/yearly fees are provided also in the form of hourly curves. This feature simplifies the further processing of the result by the majority of external modules, which no longer require to understand the tariff model, whose analysis is performed within the tariff provider module.

- Energy price is given in C/kWh (C being the currency in which the tariff was defined). Actual costs of the period can be obtained by multiplying those values by the measured hourly consumption.
- Power price is given in C/kW (C being the currency in which the tariff was defined). Actual costs of the period can be obtained by multiplying those values by the contracted capacity.
- Other fixed costs are given in C (C being the currency in which the tariff was defined) and include extra costs associated to the energy bill, such as the renting price of the smart meters

AMQP queue: "tariffprovider"

Message properties

- `reply_to`: name of the queue where response will be delivered
- `correlation_id`: free text for query/response correlation

Message payload example

```
{
  "query": "prices",
  "from": "2018-01-01T00:00:00.000Z",
```

```

    "to": "2018-01-02T00:00:00.000Z",
    "tariff": "EDP 3.0A"
}

```

Response example

```

[
  {
    "timestamp": "2018-01-01T00:00:00.000Z",
    "energyPrice": 0.018762,
    "powerPrice": 0.0255695,
    "fixedCosts": 0.001125
  },
  {
    "timestamp": "2018-01-01T01:00:00.000Z",
    "energyPrice": 0.018762,
    "powerPrice": 0.0255695,
    "fixedCosts": 0.001125
  },
  ...
]

```

6.5.3.3 Access to tariff model

The module implements a service to query the list of registered tariffs. This opens the possibility to external modules to process the tariff in any required manner.

AMQP queue: "tariffprovider"

Message properties

- reply_to: name of the queue where response will be delivered
- correlation_id: free text for query/response correlation

Message payload example

```

{
  "query": "model",
  "tariff": "EDP 3.0A"
}

```

Response example

```

{
  "rules": [
    {
      "priceType": "p1",

```



```

        "dst" : false,
        "startHour" : 18,
        "endHour" : 21
    },
    {
        "priceType" : "p1",
        "dst" : true,
        "startHour" : 11,
        "endHour" : 14
    },
    {
        "priceType" : "p2",
        "dst" : false,
        "startHour" : 8,
        "endHour" : 17
    },
    {
        "priceType" : "p2",
        "dst" : false,
        "startHour" : 22
    },
    {
        "priceType" : "p2",
        "dst" : true,
        "startHour" : 8,
        "endHour" : 10
    },
    {
        "priceType" : "p2",
        "dst" : true,
        "startHour" : 15
    },
    {
        "priceType" : "p3",
        "endHour" : 7
    }
},
    ],

```

```

plan: {
  {
    "active" : true,
    "activePowerPrices" : [
      {
        "type" : "p1",
        "price" : 0.111585896
      },
      {
        "type" : "p2",
        "price" : 0.066951945
      },
      {
        "type" : "p3",
        "price" : 0.044633951
      }
    ],
    "company" : "EDP",
    "energyConsumptionPrices" : [
      {
        "type" : "p1",
        "price" : 0.125362
      },
      {
        "type" : "p2",
        "price" : 0.106022
      },
      {
        "type" : "p3",
        "price" : 0.08082
      }
    ],
    "fixedPrices" : true,
    "minActivePower" : 15000,
    "planName" : "Luz EDP 3 Periodos",
    "startDate" : ISODate("2016-06-30T22:00:00.000Z"),
    "url" : "http://www.edpenergia.es/es/hogares/gas-y-electricidad/precios/mercado-
  
```

```

    libre/"
  }
}
}

```

6.6 WHOLESALE MARKET PRICES PROVIDER

This module provides the wholesale market energy prices of the different countries of the pilot sites of the project.

6.6.1 Spot prices

Provides spot (day-ahead) market prices for the given time period and the selected country. Prices provided in €/MWh.

AMQP queue: "wholesalemarketprices"

Message properties

- `reply_to`: name of the queue where response will be delivered
- `correlation_id`: free text for query/response correlation (RPC pattern)

Message payload examples

```

{
  "query": "spot",
  "region": "spain",
  "periodStart": "2018-01-01T00:00:00.000Z",
  "periodEnd": "2018-01-02T00:00:00.000Z"
}

```

Response example:

```

[[{
  "value" : 4.74,
  "datetime" : : "2018-01-01T00:00:00Z",
}, {
  "value" : 3.66,
  "datetime" : "2018-01-01T01:00:00Z",
},
...
]]

```

6.6.2 Implementation specific per pilot site

6.6.2.1 Crevillent

For the spanish pilot site, the module is implemented as wrapper over the ESIOs Transparency Portal [22], This portal, managed by the spanish TSO REE, offers through a REST-based API broad information about the energy markets, actual demand and production, and different disaggregation of these elements

(accordingly to the region or the source type, for instance).

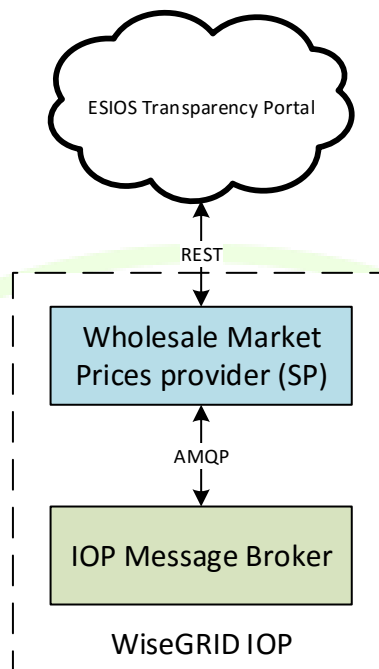


Figure 35 – Implementation of the Wholesale Market Prices provider module for Spain

6.6.2.1.1 PVPC Prices (Spain)

ESIOS transparency portal also provides information about PVPC, the state-regulated dynamic tariff in Spain. The Wholesale market prices provider will also give access to the price curves of this tariff. Prices provided in €/MWh.

AMQP queue: “wholesalemarketprices”

Message properties

- `reply_to`: name of the queue where response will be delivered
- `correlation_id`: free text for query/response correlation (RPC pattern)

Message payload examples

```

{
  "query": "pvpc",
  "periodStart": "2018-01-01T00:00:00.000Z",
  "periodEnd": "2018-01-02T00:00:00.000Z"
}
  
```

Response example:

```

[[ {
  "value" : 74.24,
  "datetime" : "2018-01-01T00:00:00Z",
}, {
  "value" : 73.05,
  "datetime" : "2018-01-01T01:00:00Z",
}
  
```

```
} ,  
...  
]
```

6.6.2.2 Flanders

For the Flanders pilot site the module is implemented as a wrapper over the European Entso-e Transparency Platform [23]. The implementation is analogue to description in section 6.6.2.1 Crevillent pilot site implementation. The python based wrapper is open-source available on github (<https://github.com/EnergieID/entsoe-py>) .

6.6.2.3 Terni

For the Italian pilot site, the module could be implemented as wrapper over the institutional GME Gestore dei Mercati Energetici S.p.A portal [2]. However, access to the data of this entity is governed by specific conditions and contractual restrictions to which the pilot of Terni at this stage of the project has not adhered and it is not clear whether this can be simple or not. An alternative solution could be to use the Spanish provider as a source as the two markets are very similar and try to make an average between Spanish prices and the historical data of Italian prices of easier access.

6.6.2.4 Mesogia and Kythnos

For the pilot sites Mesogia and Kythnos the module is implemented as a wrapper over the LAGIE official portal [24]. LAGIE is the Greek operator of the electricity market and through this portal the organisation every day provides to the public the Day-Ahead market results.

The provided data are given at files of an .xls format. These files include the next day's scheduling of the available units, the system marginal price or spot price for every hour of the next day at €/MWh, the primary and secondary reserves and various other related information.

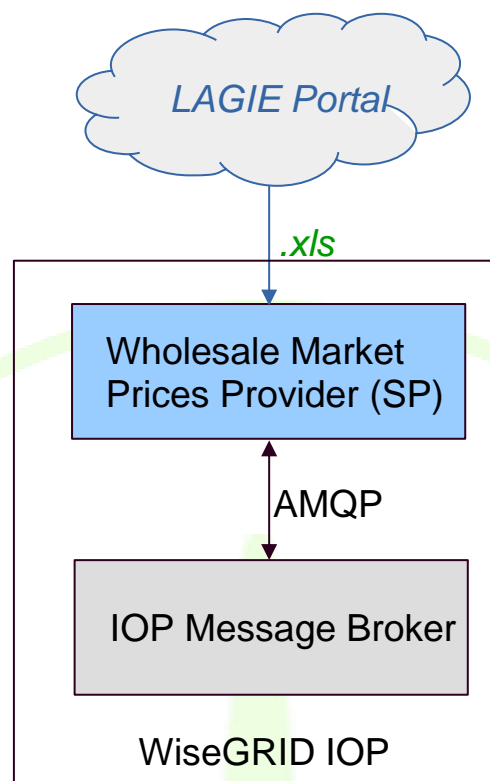


Figure 36 – Implementation of the Wholesale Market Prices provider module for Greece

7 ALGORITHMIC FRAMEWORK FOR OPTIMIZING MULTI-OBJECTIVE ENERGY SYSTEMS

7.1 DESCRIPTION

This section presents the work carried out in the context of task 4.2” Simulator for the resource portfolio and for the domains to use the services”. The main objective of the task is to work on the basis developed in task 4.1 “Development of an algorithmic framework for optimizing Multi-Objective Energy Systems” – in order to implement a tool allowing the simulation of different scenarios of interest to the DSO.

The focus on the development of the final user interface for the simulator has been put in those aspects that will be further on tested in the project by the ecosystem of applications. The targeted user is the DSO operator, who is the actor responsible of maintaining the distribution grid operating under acceptable levels and ensuring the quality of the energy. On the other hand, other actors of the grid (targeted by different applications of the project) are in position of increasing the penetration of renewable energy sources and storage (considering V2G-enabled electric vehicles as storage elements as well), and modulating the demand to support the DSO operator in its duties.

The simulator therefore allows the simulation of scenarios considering:

- Demand and production data from different sources.
- Flexibility offered by other actors of the grid, and the price associated to the activation of this flexibility.
- Storage units available at the distribution grid.

7.2 ALGORITHM

This section provides an overview of the developed algorithm, pointing out the relevant improvements made with respect to the version described in the previous deliverable D4.1 [25].

7.2.1 Objective Function

In D4.1 [25] it was stated that the Objective Function to be minimized could differ according to the needs of the DSO. In particular, four functions were presented as main candidates, the cost of generation and general transactions, the distribution grid losses, the state changes of switchgear and the voltage deviation from nominal value.

In the final model, those individual functions have all been included in one general Objective Function. Each of them is assigned a weight coefficient by the DSO before the execution of the model, which determines its financial significance relatively to the other three. Treating the weights as user defined parameters will adjust the model to the specific needs of the DSO each time and allow for the best network configuration under various circumstances.

As a result, the complete Objective Function is as follows:

$$\begin{aligned} \min \{ & w_c \sum_t [- \sum_{db,s} \bar{p}_{db,s,t} Q_{db,s,t} + \sum_{pb,s} \bar{p}_{pb,s,t} Q_{pb,s,t} + \sum_w c_w P_{w,t} + \sum_{pv} c_{pv} P_{pv,t} + \sum_h c_h P_{h,t} \\ & + \sum_{st} (-c_{st}^{charge} ST_{st,t}^C + c_{st}^{discharge} ST_{st,t}^D) \\ & + \sum_p (a_p (P_{p,t}^{CHP})^2 + \beta_p P_{p,t}^{CHP} + \gamma_p)] \\ & + w_l \sum_{n,m \in N_{n,t}} \frac{I_{nm,t}^2}{g_{nm}} + w_s \sum_{n,m \in N_{n,t}} s_{nm,t} + w_v \sum_{n,t} (V_{n,t} - V^{nom})^2 \} \end{aligned}$$

where:

w_c = Generation & Transaction cost weight coefficient

w_l = Grid Losses weight coefficient

w_s = Switching actions weight coefficient

w_v = Voltage deviation weight coefficient

7.2.2 Distribution Network Constraints

The network constraints for buses and lines have remained exactly similar to those described in D4.1 [25] and no further additions have been made.

7.2.3 Flexible Resources

The equations determining the dispatch of Flexible Resources have also remained the same, and subject to the price-quantity pairs that should be provided as external information.

7.2.4 Distributed Generation Units

The equations here have largely remained the same, with the exception of those concerning Wind Parks and Small Hydropower Units, the only Generation Units that inject Reactive Power in the Distribution Network. In addition to the constraints regarding the upper and lower levels of injected reactive power, it has been deemed important to include constraints disallowing the units to operate with a power factor under a certain value. This value is designed to be defined by the DSO before the execution of the model.

The added equations are as follows:

$$\begin{aligned} P_{w,t}^2 &\geq PF_{min}^2 (P_{w,t}^2 + Q_{w,t}^2), & \forall w, t \\ P_{h,t}^2 &\geq PF_{min}^2 (P_{h,t}^2 + Q_{h,t}^2), & \forall w, t \end{aligned}$$

where:

PF_{min} = the minimum value of the power factor assigned by the DSO

7.2.5 Storage Units

In addition to the equations already presented for the Storage Units, an additional equation has been considered necessary. Since it should be impossible for a Storage Unit to be charged and discharged during the same time period, equations that set the product of the charging and discharging values of each storage unit for each time period equal to zero have been added.

The added equations are as follows:

$$ST_{st,t}^C \cdot ST_{st,t}^D = 0, \quad \forall st, t$$

7.2.6 CHP Units

The equations regarding CHP units have remained exactly the same.

7.3 GAMS MODEL

The aforementioned changes are the only ones that affect the mathematical structure of the model. Apart from those, there have been some small functional changes and additions in the code in order to handle the data input and output (and its conversion to (pu) system for the execution) as well as a couple of minor changes in variable names that in no way affect the execution of the code.

In its current form, the GAMS model reads data from the input file INPUT.xls following directions included in the text file INPUTDATA_TEMPLATE.txt and creates the output file Results.xls following directions included in the text file OUTPUTDATA_TEMPLATE.txt. More details as to the content of the input and output data are included in the explanatory worksheets named "Sheet_Guide_INPUT" and "Sheet_Guide_Output" in the file INPUT.xls.

7.4 SIMULATOR INTERFACE

Within the task T4.2, the necessary modules have been implemented around the algorithm for optimizing Multi-Objective Energy Systems result of task T4.1, with the objective of making it easy to use for DSO operators. The main developments are summarized in the next picture:

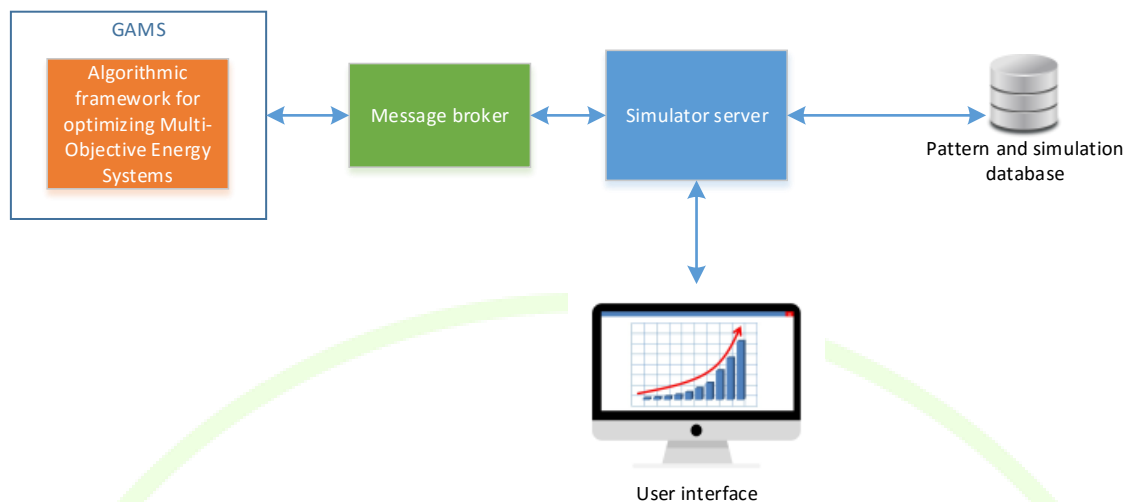


Figure 37 – Architecture of the simulator implementation

The main elements composing the simulation environment are:

- GAMS [1]: GAMS is the high-level modelling system for mathematical optimization. The algorithm developed in T4.1 has been implemented using this system.
- Message broker: element used to communicate GAMS with the simulator server. Messages exchanged between both are Excel files, as depicted in section 7.3, filled with all data necessary to trigger the simulation of an scenario and with the results of the simulations
- Simulator server: hosts the application that allows the DSO operator to define scenarios to be simulated, as well as new demand/production/flexibility patterns. It also handles the generation and interpretation of the Excel files exchanged with the GAMS system
- Pattern and simulation database: holds all the information required to create new simulation scenarios, as well as the results of already completed simulations
- User interface: interaction point of the simulator with the DSO operator

The workflow for defining and simulating a scenario is as follows:

1. The DSO operator creates a new *scenario*. A *scenario* is a set of data (load curves per bus, production curves per bus, flexibility available...) that models a certain situation of interest on the grid that will be simulated. To facilitate the task, scenarios are given a name and contain a description field (free text)

Note: for simplicity in the usage of the tool, the information about the topology of the network is just configured once during the commissioning of the simulator for a particular DSO operator. No visual tools for topology network edition are included.

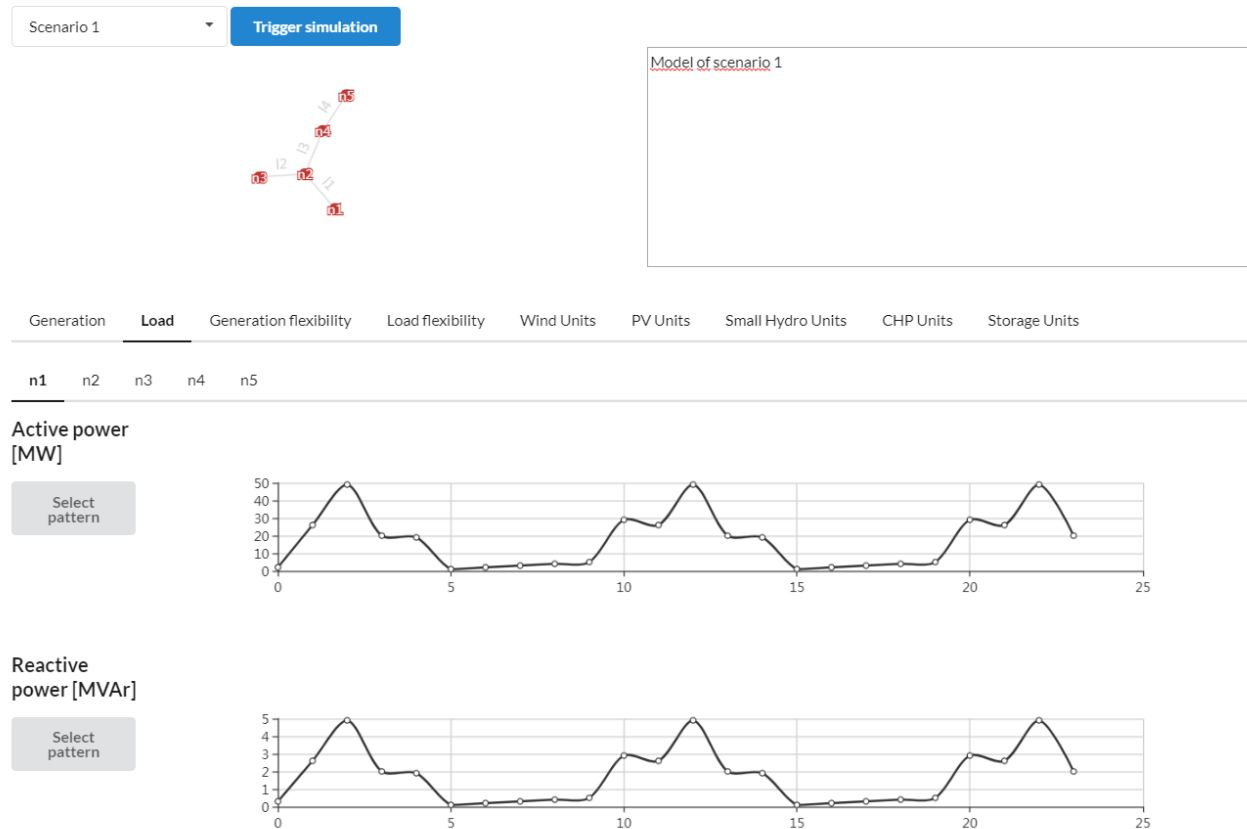


Figure 38 – Scenario creation

- Once the scenario is created, the DSO operator can populate the necessary data in order to complete the model. This data is summarized in the next table:

Table 9 – DSO data

Property modelled per bus	Curves and properties
Generation	Active power curve (MW) Reactive power curve (MVar)
Load	Active power curve (MW) Reactive power curve (MVar)
Generation flexibility	Flexibility curve (MWh) Price curve (€/MWh)
Load flexibility	Flexibility curve (MWh) Price curve (€/MWh)
Wind Units	Capacity (MW) Equivalent hours Price for energy injection Minimum/Maximum active power curves (MWh)

	Minimum/Maximum reactive power curves (MVar)
PV Units	Capacity (MW) Equivalent hours Price for energy injection Minimum/Maximum active power curves (MWh) Minimum/Maximum reactive power curves (MVar)
Small Hydro Units	Capacity (MW) Equivalent hours Price for energy injection Minimum/Maximum active power curves (MWh) Minimum/Maximum reactive power curves (MVar)
CHP Units	Ramp up Ramp down Generation cost coefficient a (€/MW ²) Generation cost coefficient b (€/MW) Generation cost coefficient c (€) Minimum/Maximum active power curves (MWh) Minimum/Maximum reactive power curves (MVar)
Storage Units	Capacity (MWh) Initial Level (MWh) Final Level (MWh) Max. Charging power (MW) Max. Discharging power (MW) Charging efficiency Discharging efficiency Charging price (€/MWh) Discharging price (€/MWh)

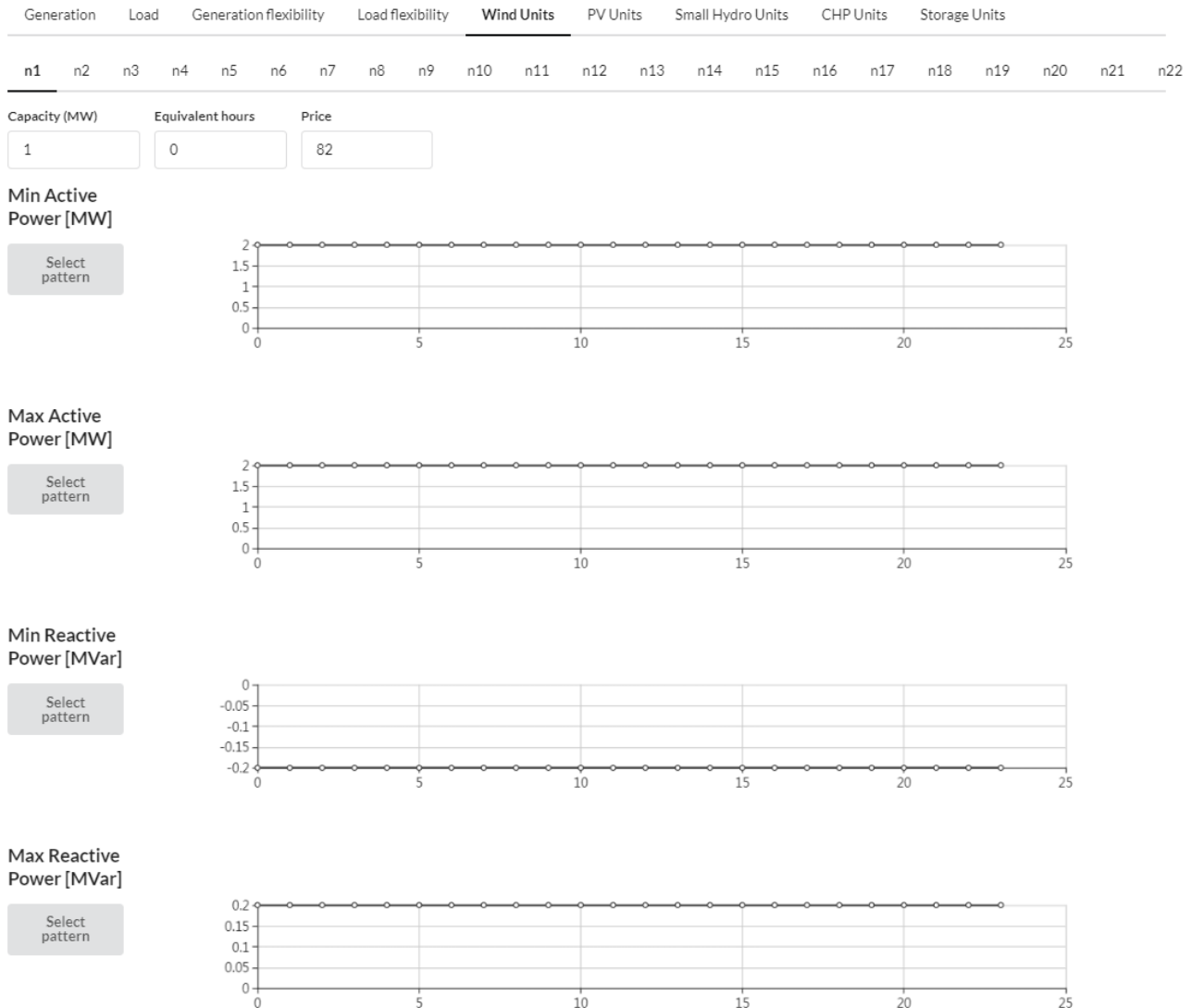


Figure 39 – Example of wind units model representation

In order to facilitate the introduction of the data, predefined patterns can be used. The simulator holds a database of patterns, which can be of two types:

- Power patterns: curves of 24 values modelling whole-day active or reactive power curves, that can be associated to both demand or production curves of the different elements.
- Flexibility patterns: curves of 24 pairs of values modelling whole-day curves of available flexibility, and its associated prices.

These patterns can be extended with custom patterns created by the DSO operator, by using the interface under *Patterns* menu.

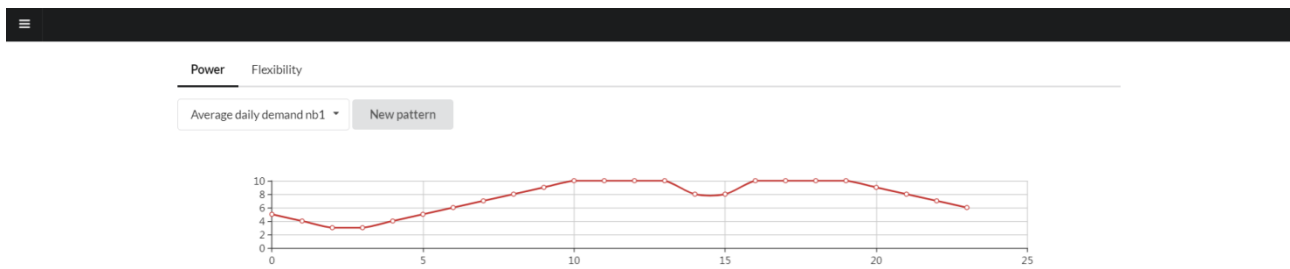


Figure 40 – Patterns

3. Once the scenario model is completed, the DSO operator can trigger the simulation. At that point, the model is transferred to GAMS and a new item under *Simulation results* menu is created.
4. Once the simulation is completed, results are reported by GAMS and can be visualized under the *Simulation results* menu. The main results provided by the model include the following elements:
 - Voltage magnitude (PU) and angle (rad) per bus;
 - Active (MW) and Reactive (MVar) Power flows per line;
 - Active (MW) and Reactive (MVar) injection from TSO through the slack bus;
 - Energy (MWh) delivered by each Flexible Load or Generation;
 - Active (MW) and Reactive (MVar) Power injection per wind park;
 - Active Power (MW) injection per PV Park;
 - Active (MW) and Reactive (MVar) Power injection per Hydro unit;
 - Active Power (MW) injection per CHP unit;
 - Energy flows (MWh) charged or discharged per time step (settlement period) and storage unit.

The simulator displays the results of interest to identify possible issues in the network for the modelled scenario, namely:

- Voltage magnitudes per bus
- Active Power flows per line

For those parameters, values provided by the GAMS model are automatically compared with operational thresholds ($\pm 10\%$ for voltage magnitude and power capacity of each of the lines). Identified issues are summarized in a table and pointed out in the different charts, giving the DSO operator a clear idea of where and which magnitude the faced issues appeared on the scenario would present.

Summary	Voltage	Power flows			
Issue type	Location	Time step	Value	Lower threshold	Upper threshold
Voltage magnitude	n1	6	1.176748344	0.9	1.1
Power flow	n1 > n2	6	10.6505448719351	0	10

Figure 41 – Summary of issues identified in the results of the simulated scenario

Summary **Voltage** Power flows

n1 n2 n3 n4 n5

Voltage [PU]

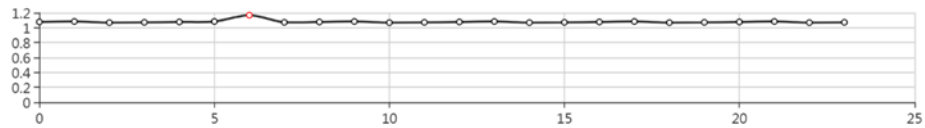


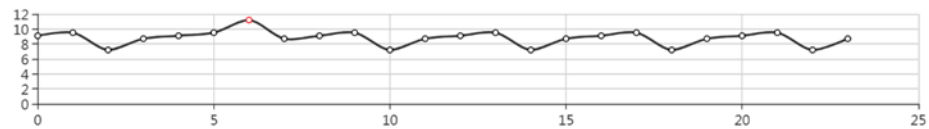
Figure 42 – Example of voltage magnitude representation with identified overvoltage

Summary **Voltage** Power flows

n1 n2 n3 n4 n5

n1 > n2

Power [MW]



n1 > n3

Power [MW]

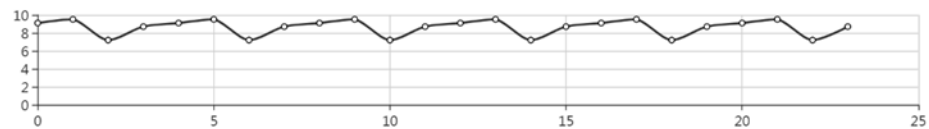


Figure 43 – Example of power flow representation with identified capacity threshold overpassed

7.5 PRACTICAL DEMONSTRATION

7.5.1 Model

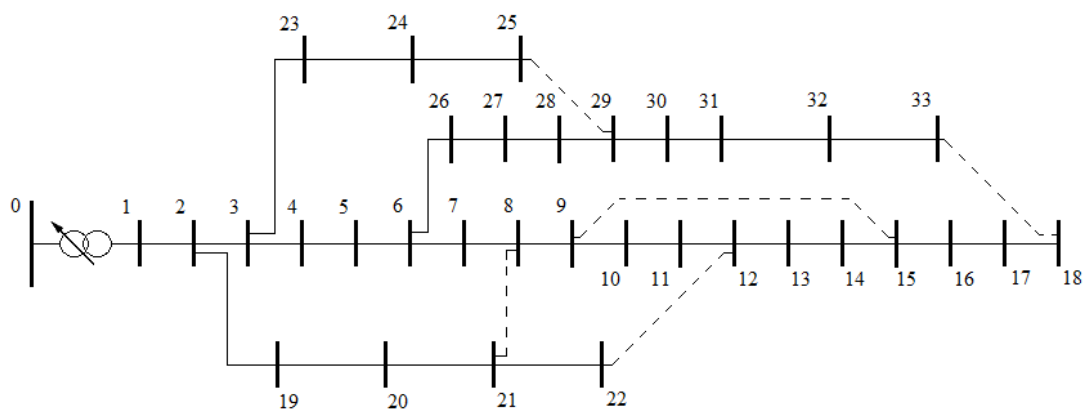


Figure 44 – Topology of the example scenario

Scenario Deliverable

Trigger simulation

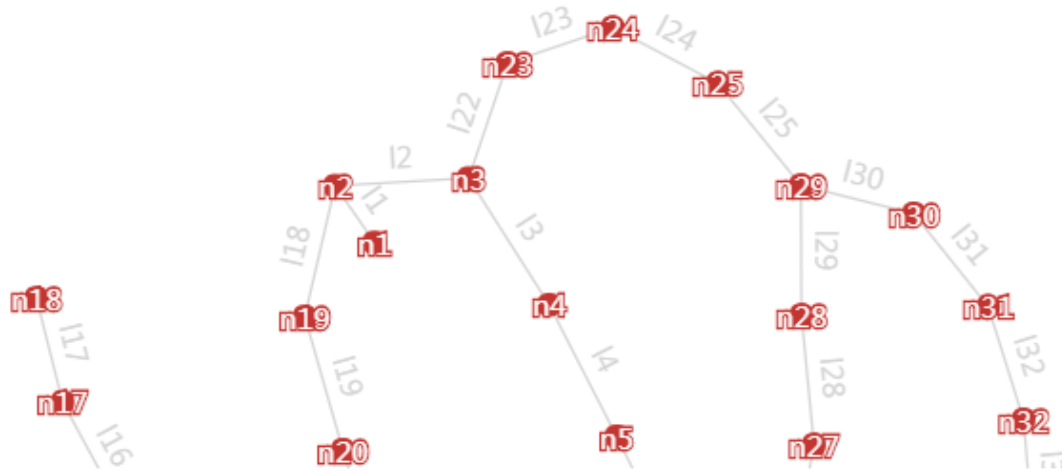


Figure 45 – Partial graph representation of the network generated automatically by the simulator

The program has been tested on the above hypothetical 33 bus system. It has a nominal voltage of 12.66 (kV) and details about its lines' operational limits (impedance and thermal limit values) are presented in the table below.

Table 10 – Line operational limits

Lines	R	X	Smax
I1	0.0922	0.047	10
I2	0.493	0.2511	10
I3	0.366	0.1864	10
I4	0.3811	0.1941	10
I5	0.819	0.707	10
I6	0.1872	0.6188	10
I7	0.7114	0.2351	8
I8	1.03	0.74	8
I9	1.04	0.74	8
I10	0.1966	0.065	8
I11	0.3744	0.1238	8
I12	1.468	1.155	8
I13	0.5416	0.7129	6
I14	0.591	0.526	6
I15	0.7463	0.545	6
I16	0.289	1.721	6
I17	0.732	0.574	6
I18	0.164	0.1565	6
I19	1.5042	1.3554	8
I20	0.4095	0.4784	8
I21	0.7089	0.9373	8
I22	0.4512	0.3083	8
I23	0.898	0.7091	8
I24	0.896	0.7011	8
I25	0.203	0.1034	8
I26	0.2842	0.1447	8
I27	1.059	0.9337	8
I28	0.8042	0.7006	8
I29	0.5075	0.2585	6
I30	0.9744	0.963	6
I31	0.3105	0.3619	6
I32	0.341	0.5302	6

The simulation has been executed for a 4 hour period in 1 hour intervals. All topological information has been modelled as part of the configuration of the tool. Including the reference bus, the allowed operational values for the buses' voltage, the minimum allowed power factor for any connected units of distributed generation that inject reactive power to the system, as well as the weights for each of the 4 partial objectives of the program's objective function.

Regarding uncontrolled load and generation, for the whole 4 hour period, all buses have been assumed to be connected to non-controllable load, both active and reactive in nature.

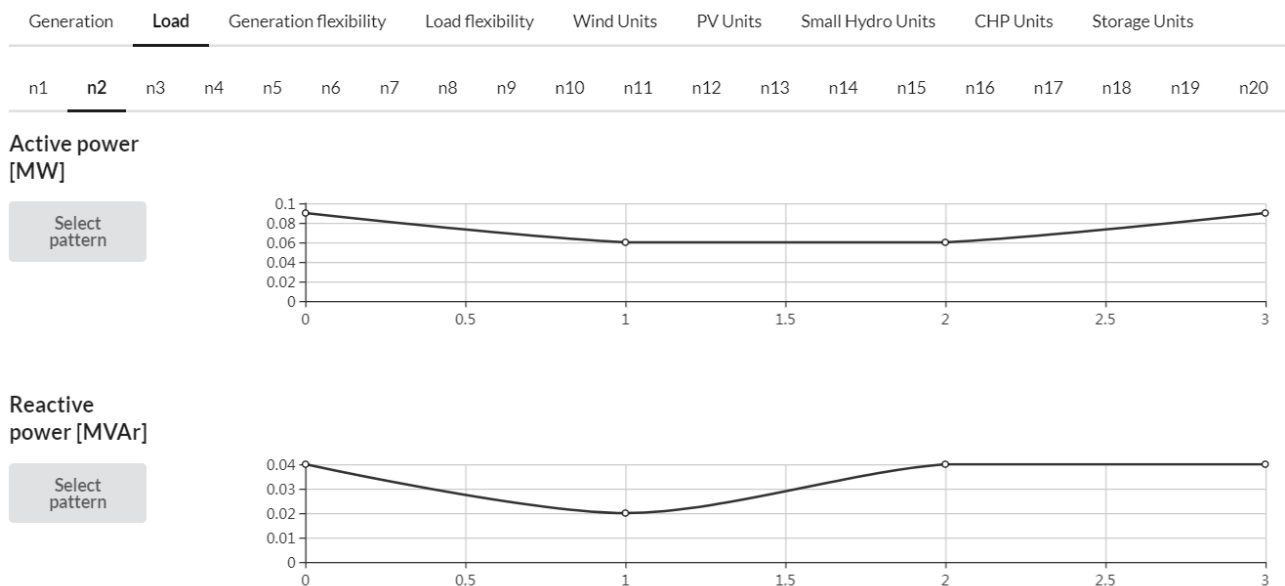


Figure 46 – Load curves for bus 2

Four production units of each type have been also modelled. The following table shows the list with the buses where those are connected:

Table 11 – Buses connected

Production unit	Connected buses
Flexible Generation	2, 10, 12, 20
Flexible Load	3, 8, 10, 15
Wind Units	4, 15, 27, 33
PV Units	11, 13, 15, 17
Small Hydro Units	8, 14, 20, 24
CHP Units	15, 18, 20, 30
Storage Units	7, 20, 22, 28

Furthermore, for this particular test case some additional choices for certain unit parameters will be presented and explained below:

- Equality hours: For the Wind, PV and Hydro units, the “EqHours” parameter, whose role is to limit the curtailment of the active power injection in the network, has been set to 0, essentially imposing no limits on the units curtailment.

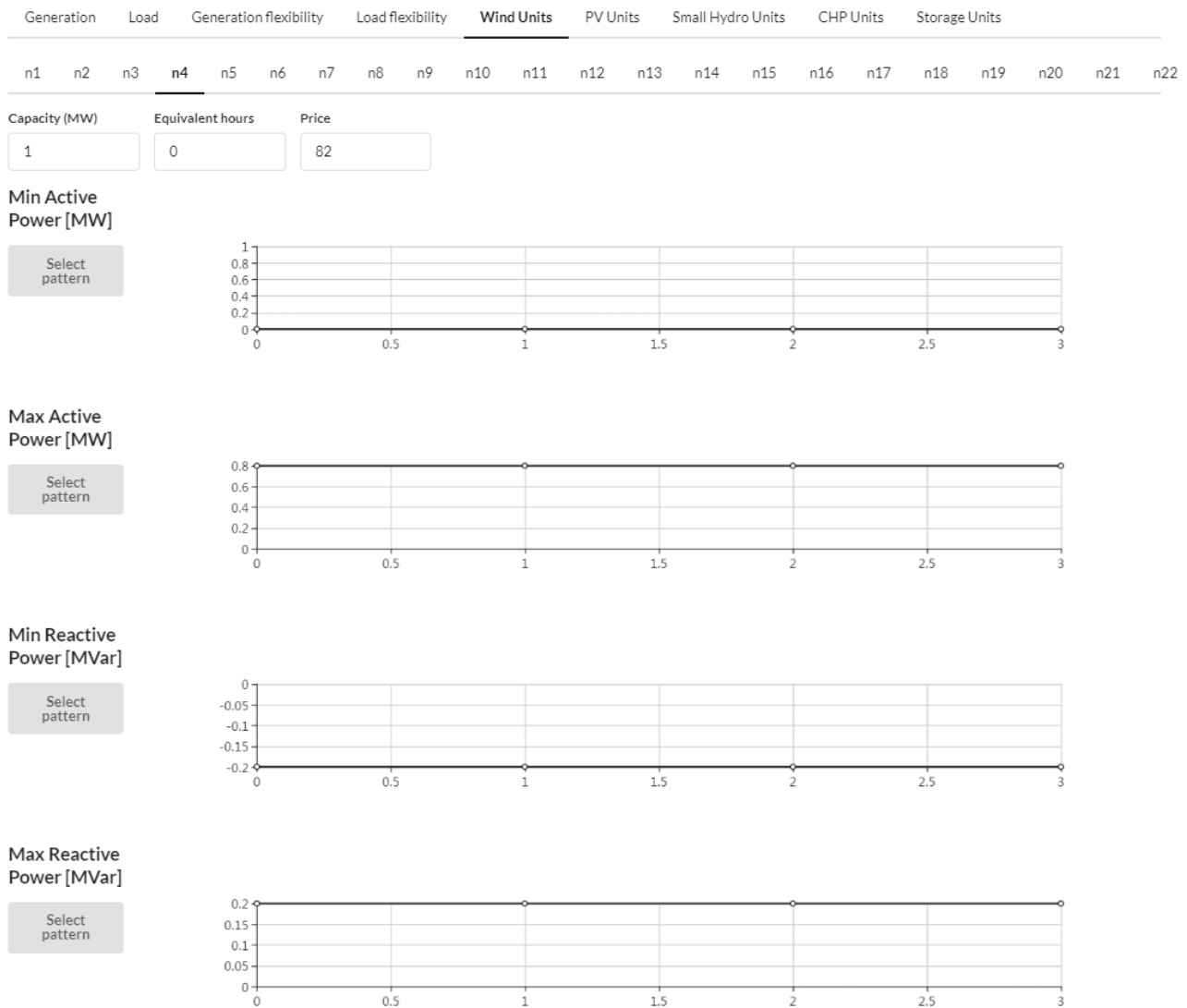


Figure 47 – Overview of the characteristics modelled for the Wind units connected to bus 4

- Offer Steps: For the Flexible Generation and Load units, the offer steps, which determine the number of price/quantity pairs in the respective unit diagrams, have been considered the same for all units.

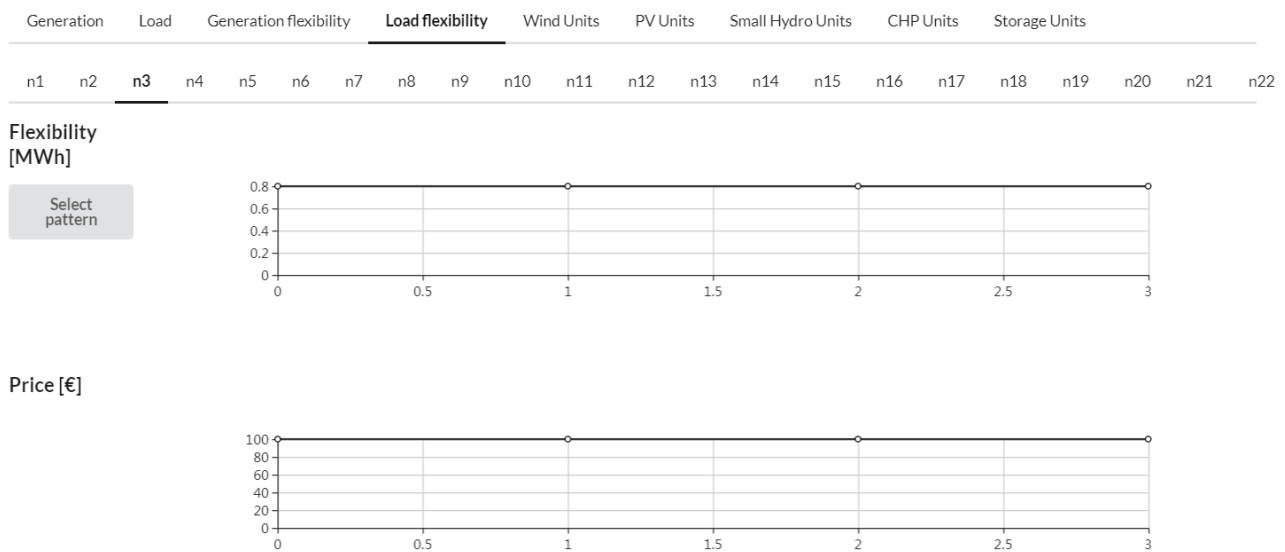


Figure 48 – Overview of the characteristics modelled for the Flexible Loads connected to bus 3

7.5.2 Results

After the simulation gets successfully executed, the curves depicting the expected voltage magnitudes for each bus and the expected power flows for each line are represented. With the parameters given to this model, no operational threshold violation has been identified.

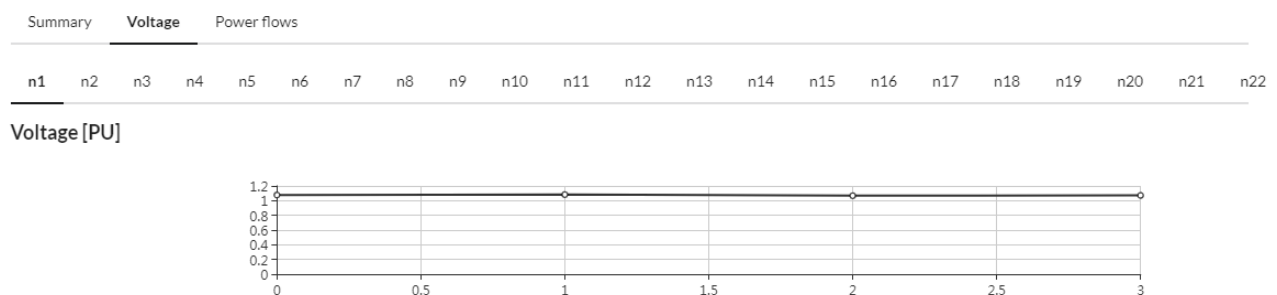


Figure 49 – Overview of voltage magnitude for bus 1

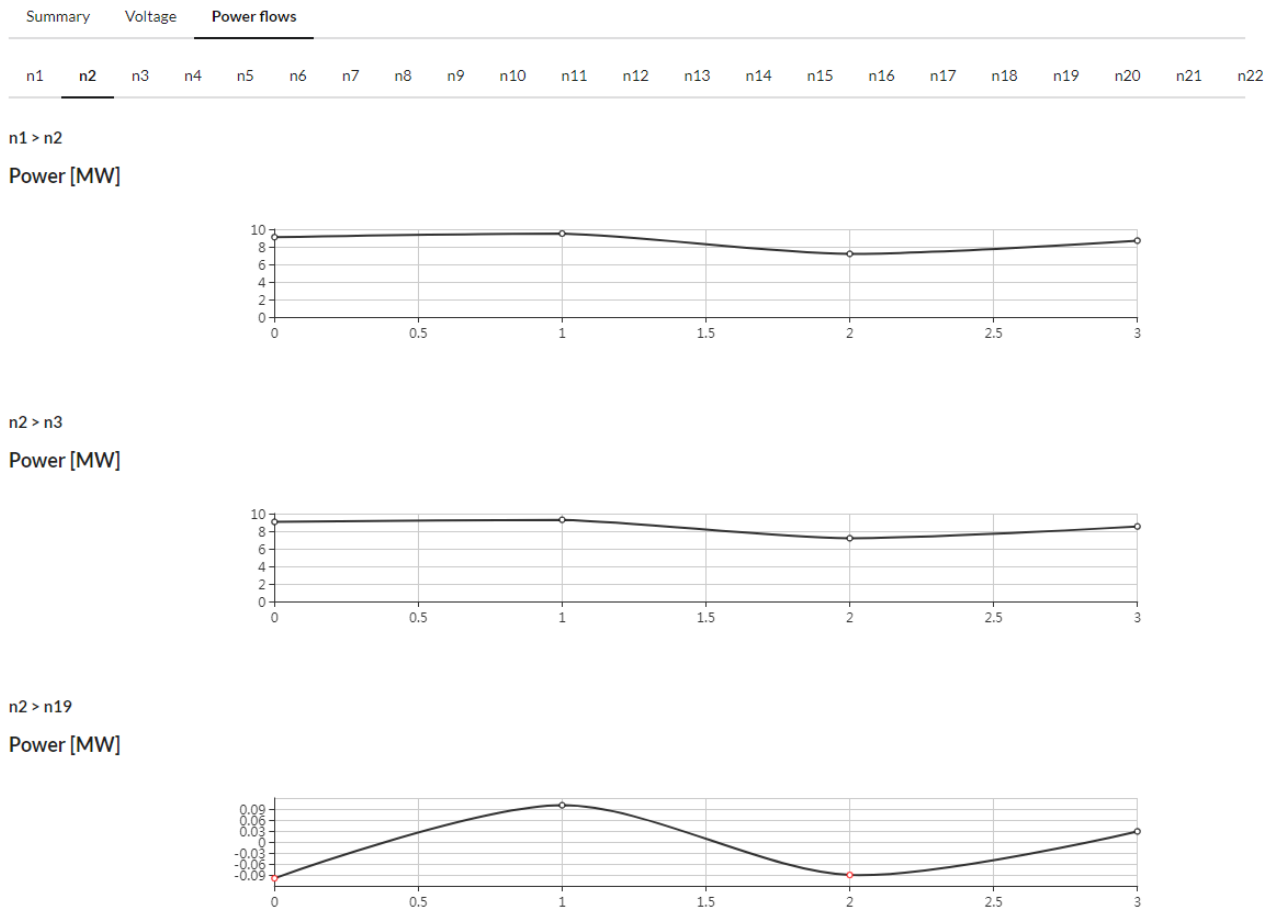


Figure 50 – Overview of power flows involving bus 2

8 CONCLUSIONS

The document presents WG IOP architecture as designed to allow the message exchange between tools and services of the WiseGRID project. The message exchange among those components and the WiseGRID tools via WG IOP is secured by two different security levels, the first one by an Authorization Control and the second one by Authentication Control

Regarding the WG IOP framework, the core of its architecture is the open source multiprotocol RabbitMQ message broker by which all messages are exchanged, being a kind of “mailman”. RabbitMQ is using mostly two protocols: AMQP and MQTT; used by the WISEGRID project tools and services to exchange messages through the WG IOP.

MQTT (MQ Telemetry Transport or Message Queue Telemetry Transport) is a standard ISO protocol (ISO/IEC PRF 20922) designed for low impact and limited band.

AMQP (Advanced Message Queuing Protocol) is a binary, application layer protocol, designed for a wide variety of messaging applications and communication patterns. It is reliable and interoperable.

The standards and message structure between WG IOP and other WiseGRID tools are described in dedicated sections.

WiseGRID is a project mostly focused in the Distribution Grid, then USEF framework will be considered the framework as identifies different flexibility services for the DSO which provide value by helping the DSO to optimise its performance and efficiency in managing the distribution grid [2]. It will focus mainly on:

- Congestion management
- Voltage problems
- Grid capacity management
- Controlled islanding
- Redundancy (n-1) support.

Another standard makes reference to the data model used in WiseGRID as the Common Information Model (CIM). The third standard to be used in WiseGRID is OpenADR [10] (Automated Demand Response).

Conclusion is that advanced technologies are considered for development of WG IOP.

Interactions between each WiseGRID tool and WG IOP elements and various modules is detailed for each tool in dedicated subchapters (standards and message structures).

Within microservices of IOP are presented: Real time monitoring (EVSE wrapper, Smart Meter wrapper, Battery wrapper and Domestic/building Assets wrapper), Weather forecasting and Energy forecasting, Tariffs and Wholesale prices for energy, ancillary services Market.

Putting all above together, this document also presents the Algorithmic framework for optimizing Multi-Objective Energy Systems within four levels: Description, Algorithm, Simulator tool and Practical demonstration.

The simulator allows the simulation of scenarios considering:

- Demand and production data
- Flexibility offered by other actors of the grid, and the price associated to the activation of this flexibility

Main four criteria are presented to optimize DSO operations:

- Minimize the cost of generation and general transactions,
- Minimize the distribution grid losses,
- Optimize the state changes of switchgear
- Optimise the voltage deviation from nominal value.

Detailed equations are presented for calculations within each defined criteria.

The document presents a clear solution for the necessary modules to be implemented around the algorithm for optimizing Multi-Objective Energy Systems, with the objective of making it easy to use for DSO operators. That would enable the DSO to operate their grids more secure and efficient for the grid users (clients) benefits.

The workflow for defining and simulating a scenario is as follows:

1. The DSO operator creates a new scenario.
2. For the created scenario, the DSO operator can populate the necessary data (complete model).

For facilitating the introduction of the data, predefined patterns are used. The simulator holds a database of patterns, which correspond to:

- Power patterns: curves of 24 values modelling whole-day active or reactive power curves, that can be associated to both demand or production
- Flexibility patterns: curves of 24 pairs of values modelling whole-day curves of available flexibility, and its associated prices

These patterns can be extended with custom patterns created by the DSO operator.

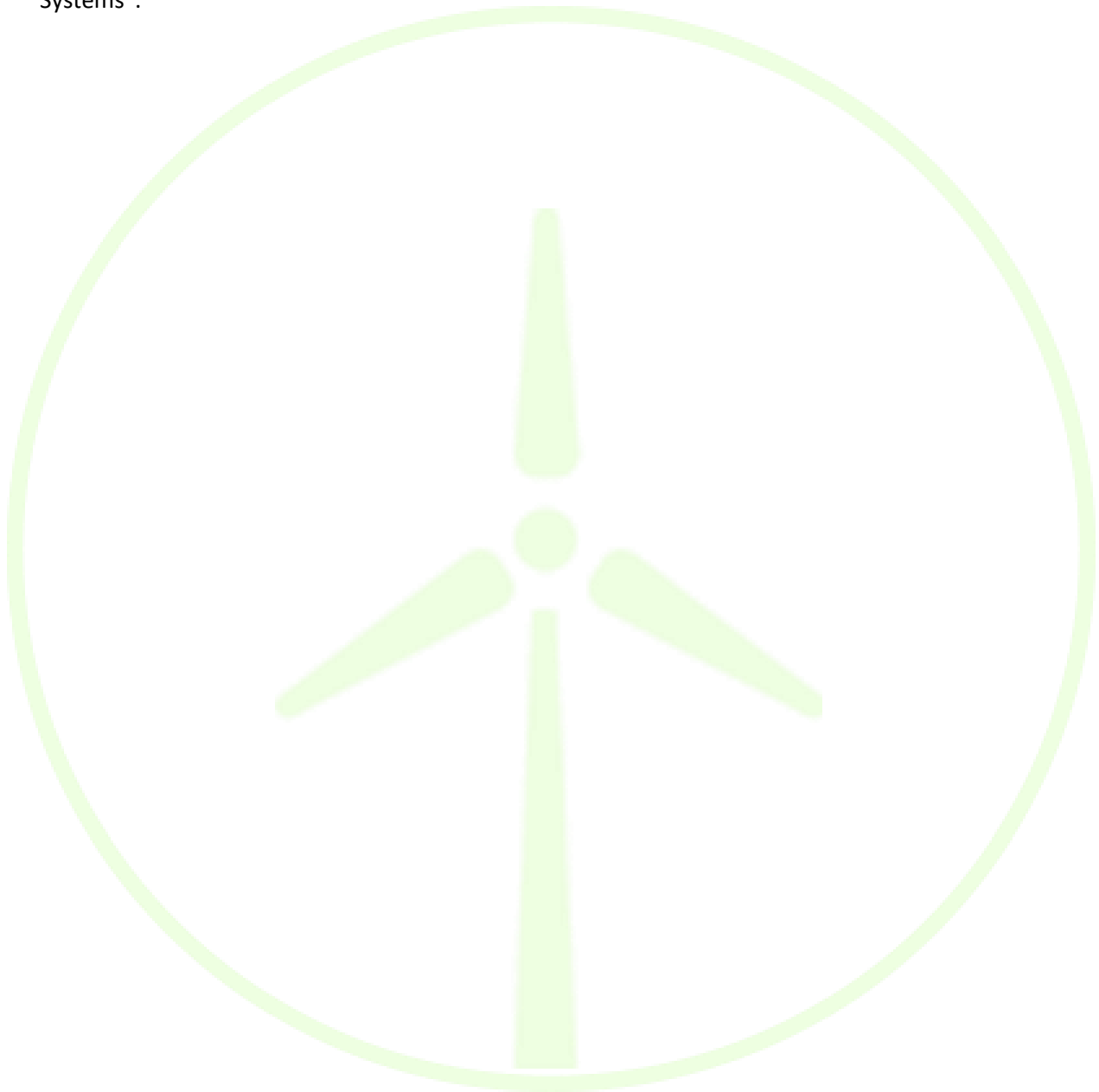
3. Once the scenario model is completed, the DSO operator can trigger the simulation. At that moment, the model is transferred to GAMS and a new item under Simulation results menu is created.
4. Once the simulation is completed, results are reported by GAMS and can be visualized under a Simulation predefined results menu.

9 REFERENCES AND ACRONYMS

9.1 REFERENCES

- [1] "<https://www.gams.com/>," [Online].
- [2] [Online]. Available: <https://www.rabbitmq.com/authentication.html>.
- [3] "<https://www.rabbitmq.com/access-control.html>," [Online].
- [4] [Online]. Available: <https://www.rabbitmq.com/>.
- [5] [Online]. Available: <http://mqtt.org/>.
- [6] [Online]. Available: <https://www.amqp.org/>.
- [7] [Online]. Available: <https://www.usef.energy/>.
- [8] NOBEL GRID project.
- [9] *IEC 61970: Energy management system application program interface, Common Information Model (CIM) for Energy Management..*
- [10] [Online]. Available: <http://w3.siemens.com/smartgrid/global/en/products-systems-solutions/control-center-solutions/grid-control-platform/about-spectrum-power/pages/common-information-model.aspx>.
- [11] [Online]. Available: <https://www.entsoe.eu/major-projects/common-information-model-cim/Pages/default.aspx>.
- [12] "Introducción al modelo CIM de los sistemas de energía eléctrica".
- [13] [Online]. Available: <http://www.openadr.org/>.
- [14] [Online]. Available: http://www.openadr.org/assets/docs/openadr_primer.pdf.
- [15] "<https://www.usef.energy/>," [Online].
- [16] "D13.1 WiseGRID Cockpit design".
- [17] "D6.2 Storage as a service and Innovative optimized solutions".
- [18] "D7.1 WiseCOOP and WiseCORP Apps Design".
- [19] "<https://openweathermap.org/>," [Online].
- [20] "<https://solcast.com.au/>," [Online].
- [21] "<http://besos-project.eu/>," [Online].

- [22] "<https://www.esios.ree.es/es/>," [Online].
- [23] [Online]. Available: <https://transparency.entsoe.eu/>.
- [24] "<http://www.lagie.gr/en/market/electricity-power-market-participation/day-ahead-market-results/>," [Online].
- [25] "D4.1 Designing and Modelling of an algorithmic framework for optimizing Multi-Objective Energy Systems".



9.2 ACRONYMS

Table 12 – List of Acronyms

Acronyms List	
AMQP	Advanced Message Queuing Protocol
AMR	Automatic meter reading
API	Application Programming Interface
BACnet	Building Automation and Control network
CIM	Common Information Model
DSO	Distribution System Operator
EV	Electrical Vehicle
EVSE	Electric Vehicle Supply Equipment
IOP	Intoperable Platform
JSON	JavaScript Object Notation
MQTT	Message Queue Telemetry Transport
OCPP	Open Charge Point Protocol
OPENADR	Automated Demand Response
PV	Photovoltaic
PVPC	Voluntary Price for the Small Consumer (in Spain)
RES	Renewable Energy Source
RESCO	Renewable Energy Service Company
RTU	remote terminal units
SCADA	Supervisory Control And Data Acquisition
SMM	Smart Metrology Meter
SMW	Smart Meter Wrapper
SMX	Smart Meter Extension
StaaS	Storage as a Service
TCP	Transmission Control Protocol
USEF	Universal Smart Energy Framework
USM	Unbundled Smart Meter
VPP	Virtual Power Plant
WG	WiseGRID

10 ANNEX A

10.1 RESCO TOOL MESSAGES STRUCTURE

WG RESCO to Forecast module: query for information to visualize

The WG RESCO send a request via IOP into two specific AMQP queue: “forecasting_demand” and “forecasting_production” that contains a payload whit information about the “Client_id, Latitude, Longitude, Horizon and Period” for which to perform forecasts. The WG RESCO is listening to the corresponding response queue and it is able to read it and elaborate it content (timestamp and related consumption/production numeric value in KW).

Here is an example of queue payload and related response.

Message payload examples for Demand Forecast

```
{
  "Client_id": 1,
  "Latitude" : 42.61
  "Longitude" : 12.45
  "Horizon": 1,
  "Period": 60
}
```

Response example to read from the IOP:

```
{
  "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"
```

Message payload examples for Production Forecast

```
{
  "Client_id": 1,
  "Latitude" : 42.61
  "Longitude" : 12.45
  "Horizon": 1,
  "Period": 60
}
```

Response example to read from the IOP:

```
{
  "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
  },
}
```

WG RESCO billing module to WG tools

The WG RESCO provide information related the billing for a specific supply point and a certain contract typology to the WiseHOME and WiseCORP tools. The message is sent by the RESCO tool every time a billing is elaborated into a specific queue.

```
{ "contract_id": 1,  
  "Bill_Start_date_period" : dd/mm/yyyy  
  "Bill_End_date_period" : dd/mm/yyyy  
  "Euro_Energy_price_kw": 2,  
  "Consummed_energy_kw": 60,  
  "Produced_energy_kw": 60  
}
```

10.2 STAAS/VPP TOOL MESSAGES STRUCTURE

This sub-section illustrates some representative messages that will be communicated by the StaaS/VPP tool in order to communicate with the batteries.

Exemplary payload for battery storage system data:

```
{  
  "SN": "0123456789A",  
  "FWVersion": "1.0",  
  "Power": 1.0,  
  "Capacity": 1.0,  
  "PVPower": 1.0  
}
```

Exemplary payload for command to a battery storage system

```
{
  "Id": "b246798a-4574-4c02-9a3c-946407242f49",
  "Command": 0,
  "Params": {
    "WorkingMode": 1
  }
}
```

Exemplary payload for acknowledgement

```
{
  "Id": "b246798a-4574-4c02-9a3c-946407242f49",
  "Result": 1,
  "Details": {
  }
}
```

Exemplary payload for active power control within control loop

```
{
  "Id": "b246798a-4574-4c02-9a3c-946407242f49",
  "Command": 3,
  "Params": {
    "ControlMode" : 1,
    "active" : true,
    "P" : 1234
  }
}
```

Exemplary payload for the status of the battery storage system

```
{
  "MeterActivePower": 1.0,
  "MeterReactivePower": 1.0,
  "InverterActivePower": 1.0,
  "InverterReactivePower": 1.0,
  "InverterPVPower": 1.0,
  "ExternalPV": 1.0,
  "InverterBatteryPower": 1.0,
  "BatterySOC": 1.0,
  "MeterGridFrequency": 1.0,
  "Temperatures": 1.0,
  "CosPhi": 1.0,
  "ChargeDisp": 1.0,
  "BatterySOH": 1.0,
  "BatteryVoltage": 1.0,
  "MeterGridVoltage": 1.0,
  "DischargeDisp": 1.0,
  "Status": 1,
  "Alarms": 1,
  "InverterPVVoltage": 1.0,
  "WorkingMode": 1,
  "Load": 1.0
}
```

10.3 DEMAND RESPONSE & WISEHOME MESSAGES STRUCTURE

This sub-section illustrates some representative messages that will be communicated by WiseHOME, WiseCORP and WiseCOOP over the WG IOP. The list is not exhaustive, but it provides a view on the message structure in sufficient detail.

WiseHOME to WiseCOOP: query for information to visualize


```
{
  "header": {
    "Sender": "001DK1004",
    "Recipient": "ECOPOWER",
    "conversationID": "ECPDK1004001",
    "messageType": "REQUEST"
  },
  "body": [
    {
      "reference": "USER",
      "assetKey": "DK1004",
      "dataType": "TIMESERIES",
      "metricType": "ENERGY_CONSUMPTION",
      "startTime": "2017-07-18T10:45:01.898Z",
      "endTime": "2017-07-18T11:15:01.842Z",
      "sampleTime": "15 MIN"
    },
    {
      "reference": "USER",
      "assetKey": "DK1004",
      "dataType": "CUMULATIVE_DATA",
      "metricType": "ENERGY_COST",
      "startTime": "2017-07-18T10:45:01.898Z",
    }
  ]
}
```

For more details about the specifications of this message exchange please refer to deliverable D11.1 “WiseHOME app design”.

WiseCOOP to WiseHOME: response containing information for visualization

The following is a sample response with limited amount of data in order to illustrate the main principles.

```
{
  "header": {
    "Sender": "ECOPOWER",
    "Recipient": "001DK1004",
    "conversationID": "ECPDK1004",
    "messageType": "RESPONSE"
  },
  "body": [
    {
      "reference": "USER",
      "assetKey": "DK1004",
      "dataType": "TIMESERIES",
      "metricType": "ENERGY_CONSUMPTION",
      "endTime": "2017-07-18T11:15:01.842Z",
      "sampleTime": "15 MIN",
      "metricTimeseries": [
        {
          "value": 1.49,
          "timestamp": "2017-07-18T10:45:01.908Z"
        },
        {
          "value": 0.44,
          "timestamp": "2017-07-18T11:00:01.833Z"
        },
        {
          "value": 2.07,
          "timestamp": "2017-07-18T11:15:01.842Z"
        }
      ]
    },
    {
      "reference": "USER",
      "assetKey": "DK1004",
      "dataType": "CUMULATIVE_DATA",
      "metricType": "ENERGY_COST",
      "startTime": "2017-07-18T10:45:01.898Z",
      "value": 250.10
    }
  ]
}
```

For more details about the specifications of this message exchange please refer to deliverable D11.1 “WiseHOME app design”.

WiseCOOP broadcasting of dynamic energy price

The following message will be used to broadcast variable electricity pricing information to all interested parties. The DR Framework supports the communication of different prices to various buildings, a feature

that will most likely not be used during WiseGRID on demand of the pilot site partners who opt for a homogeneous price for all their retail/LV customers.

```
{
  "market": "ToU, CPP or RTP?",
  "timestampCreated": "datetime",
  "AssetPricingList": [
    {
      "assetId": "asset01",
      "dsmPriceList": [
        {"interval": "0",
          "value": "xxx"},
        {"interval": "1",
          "value": "xxx"},
        .
        {"interval": "23",
          "value": "xxx"},
      ]
    },
    {
      "assetId": "asset02",
      "dsmParticipation": [
        {"interval": "0",
          "value": "xxx"},
        {"interval": "1",
          "value": "xxx"},
        .
        {"interval": "23",
          "value": "xxx"},
      ]
    },
    ...
  ]
}
```

More details about this message specifications be found in deliverable D10.2 Demand Response Framework specifications.

WiseCORP to WiseCOOP: communication of demand flexibility potential

Message Type: JSON message with the list of provision of flexibilities for each setpoint, per interval and per asset:

```
[
{
  "assetKey": "asset01",
  "devices": [
    {
      "deviceId": "0-1-160-7-0-1",
      "deviceType": "HVAC",
      "flexibilityList": [
        {
          "setpoint": 22,
          "interval": 1,
          "comfort": 0.85,
          "flexAmount": 0.120
        },
        {
          "setpoint": 22,
          "interval": 2,
          "comfort": 0.83,
          "flexAmount": 0.120
        },
        {
          "setpoint": 22,
          "interval": 3,
          "comfort": 0.84,
          "flexAmount": 0.120
        },
        {
          "setpoint": 22,
          "interval": 4,
          "comfort": 0.82,
          "flexAmount": 0.120
        },
        .....
      ]
    },
    {

```

```

"deviceId": "0-1-163-7-0-1",
"deviceType": "LIGHT",
"flexibilityList": [
  {
    "setpoint": 0.85,
    "interval": 1,
    "comfort": 0.92,
    "flexAmount": 0.001
  },
  {
    "setpoint": 0.85,
    "interval": 2,
    "comfort": 0.92,
    "flexAmount": 0.001
  },
  {
    "setpoint": 0.85,
    "interval": 3,
    "comfort": 0.92,
    "flexAmount": 0.001
  },
  {
    "setpoint": 0.85,
    "interval": 4,
    "comfort": 0.92,
    "flexAmount": 0.001
  },
  ....
]
}
]
}
]

```

More details about this message specifications be found in deliverable D10.2 Demand Response Framework specifications.

WiseCOOP to WiseCORP: request to modify demand profile by given amount

The JSON body for **explicit DR broadcasting (eiEvent)** to assets should have the following format:

```
{
  "eiEventDescriptor": {
    "eventID": "1",
    "createdDateTime": "2012-12-13T12:12:12"
  },
  "eiEventSignals": {
    "eiEventSignal": [
      {
        "signalID": "17",
        "startTime": "2012-12-13T12:00:00",
        "activePeriod": "PT15M",
        "eiTarget": {
          "venID": "assetID",
          "aggregatedPnode": "8"
        }
      },
      {
        "signalID": "97",
        "startTime": "2012-12-13T12:15:00",
        "activePeriod": "PT15M",
        "eiTarget": {
          "venID": "12345",
          "aggregatedPnode": "1.0"
        }
      },
      {
        "signalID": "107",
        "startTime": "2012-12-13T12:20:00",
        "activePeriod": "PT15M",
        "eiTarget": {
          "venID": "assetID",
          "aggregatedPnode": "-0.4"
        }
      }
    ],
    "numDataSources": "3",
    "activePeriod": "PT45M",
    "eiTarget": {
      "venID": "assetID",
      "aggregatedPnode": "8.6"
    }
  }
}
```

The above JSON object is broadcasted as a list of objects, each one is dedicated to a specific asset, defined by the unique asset key (**venID**) and represents an explicit DR request to the specific asset (building). More details about this message specifications be found in deliverable D10.2 Demand Response Framework specifications.

WiseCORP to WiseCOOP: confirmation of intention to modify demand profile based on request

The following message is the **response (EiEventResponse)** of each asset to the DR triggering depending on whether the asset opts in or out of the DR trigger:

```
{
  "eventID": "123",
  "venID": "assetID",
  "signalIDs": ["17", "97", "107"],
  "optType": "true" OR "false"
}
```

More details about this message specifications be found in deliverable D10.2 Demand Response Framework specifications.

WiseCORP to WiseCOOP: reporting of actual demand profile modification

While the following JSON object is the **reported result – actual flexibility delivered (EiEventReport)** of the explicit DR signal:

```
{
  "reportID" : "1233",
  "reportName" : "reportName1",
  "eventID" : "123",
  "venID" : "assetID",
  "aggregatedPnode" : "8.6",
  "reportedSignals" : [
    {
      "signalID" : "17",
      "aggregatedPnode" : "8"
    },
    {
      "signalID" : "97",
      "aggregatedPnode" : "1.0"
    },
    {
      "signalID" : "107",
```

```
"aggregatedPnode" : "-0.4"  
  }  
}
```

More details about this message specifications be found in deliverable D10.2 Demand Response Framework specifications.



10.4 EVSE WRAPPER MESSAGES

The EVSE Wrapper will communicate with the WG IOP and the WiseEVP using a selection of OCPPv1.6 JSON messages. The selected messages allow the implementation of the different use cases considered for the WiseEVP. This selection includes the following messages:

MQTT is foreseen as the protocol 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 [17]) 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

10.4.1 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:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "BootNotificationRequest",
  "type": "object",
  "properties": {
    "chargePointVendor": {
      "type": "string",
      "maxLength": 20
    },
    "chargePointModel": {
      "type": "string",
      "maxLength": 20
    },
    "chargePointSerialNumber": {
      "type": "string",
      "maxLength": 25
    },
    "chargeBoxSerialNumber": {
      "type": "string",
      "maxLength": 25
    }
  }
}
```

```

    },
    "firmwareVersion": {
      "type": "string",
      "maxLength": 50
    },
    "iccid": {
      "type": "string",
      "maxLength": 20
    },
    "imsi": {
      "type": "string",
      "maxLength": 20
    },
    "meterType": {
      "type": "string",
      "maxLength": 25
    },
    "meterSerialNumber": {
      "type": "string",
      "maxLength": 25
    }
  },
  "additionalProperties": false,
  "required": [
    "chargePointVendor",
    "chargePointModel"
  ]
}

```

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:

```

{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "MeterValuesRequest",
  "type": "object",

```

```

"properties": {
  "connectorId": {
    "type": "integer"
  },
  "transactionId": {
    "type": "integer"
  },
  "meterValue": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "timestamp": {
          "type": "string",
          "format": "date-time"
        },
        "sampledValue": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "value": {
                "type": "string"
              }
            }
          },
          "context": {
            "type": "string",
            "enum": [
              "Interruption.Begin",
              "Interruption.End",
              "Sample.Clock",
              "Sample.Periodic",
              "Transaction.Begin",
              "Transaction.End",
              "Trigger",
              "Other"
            ]
          }
        }
      }
    }
  }
},

```

```

"format": {
  "type": "string",
  "enum": [
    "Raw",
    "SignedData"
  ]
},
"measurand": {
  "type": "string",
  "enum": [
    "Energy.Active.Export.Register",
    "Energy.Active.Import.Register",
    "Energy.Reactive.Export.Register",
    "Energy.Reactive.Import.Register",
    "Energy.Active.Export.Interval",
    "Energy.Active.Import.Interval",
    "Energy.Reactive.Export.Interval",
    "Energy.Reactive.Import.Interval",
    "Power.Active.Export",
    "Power.Active.Import",
    "Power.Offered",
    "Power.Reactive.Export",
    "Power.Reactive.Import",
    "Power.Factor",
    "Current.Import",
    "Current.Export",
    "Current.Offered",
    "Voltage",
    "Frequency",
    "Temperature",

```

```

        "SoC",
        "RPM"
    ]
},
"phase": {
    "type": "string",
    "enum": [
        "L1",
        "L2",
        "L3",
        "N",
        "L1-N",
        "L2-N",
        "L3-N",
        "L1-L2",
        "L2-L3",
        "L3-L1"
    ]
},
"location": {
    "type": "string",
    "enum": [
        "Cable",
        "EV",
        "Inlet",
        "Outlet",
        "Body"
    ]
},
"unit": {
    "type": "string",
    "enum": [
        "Wh",
        "kWh",
        "varh",
        "kvarh",
        "W",
        "kW",
        "VA",

```

```

    "Celcius",
    "Fahrenheit",
    "Percent"
  ],
  "required": [
    "value"
  ]
},
{
  "required": [
    "timestamp",
    "sampledValue"
  ]
},
{
  "required": false,
  "required": [
    "connectorId",
    "meterValue"
  ]
}
}

```

Topic: **status**

{

```

"$schema": "http://json-schema.org/draft-04/schema#",
"title": "StatusNotificationRequest",
"type": "object",
"properties": {
  "connectorId": {
    "type": "integer"
  },
  "errorCode": {
    "type": "string",
    "enum": [
      "ConnectorLockFailure",
      "EVCommunicationError",
      "GroundFailure",
      "HighTemperature",
      "InternalError",
      "LocalListConflict",
      "NoError",
      "OtherError",
      "OverCurrentFailure",
      "PowerMeterFailure",
      "PowerSwitchFailure",
      "ReaderFailure",
      "ResetFailure",
      "UnderVoltage",
      "OverVoltage",
      "WeakSignal"
    ]
  },
  "info": {
    "type": "string",
    "maxLength": 50
  },
  "status": {
    "type": "string",
    "enum": [
      "Available",
      "Preparing",
      "Charging",
      "SuspendedEVSE",

```

```

        "SuspendedEV",
        "Finishing",
        "Reserved",
        "Unavailable",
        "Faulted"
    ]
},
"timestamp": {
    "type": "string",
    "format": "date-time"
},
"vendorId": {
    "type": "string",
    "maxLength": 255
},
"vendorErrorCode": {
    "type": "string",
    "maxLength": 50
}
},
"additionalProperties": false,
"required": [
    "connectorId",
    "errorCode",
    "status"
]
]
}

```

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:

```

{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "StartTransactionRequest",
    "type": "object",
    "properties": {

```



```

    "connectorId": {
      "type": "integer"
    },
    "idTag": {
      "type": "string",
      "maxLength": 20
    },
    "meterStart": {
      "type": "integer"
    },
    "reservationId": {
      "type": "integer"
    },
    "timestamp": {
      "type": "string",
      "format": "date-time"
    }
  },
  "additionalProperties": false,
  "required": [
    "connectorId",
    "idTag",
    "meterStart",
    "timestamp"
  ]
}

```

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:

```

{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "StopTransactionRequest",
  "type": "object",
  "properties": {
    "idTag": {

```

```

        "type": "string",
        "maxLength": 20
    },
    "meterStop": {
        "type": "integer"
    },
    "timestamp": {
        "type": "string",
        "format": "date-time"
    },
    "transactionId": {
        "type": "integer"
    },
    "reason": {
        "type": "string",
        "enum": [
            "EmergencyStop",
            "EVDDisconnected",
            "HardReset",
            "Local",
            "Other",
            "PowerLoss",
            "Reboot",
            "Remote",
            "SoftReset",
            "UnlockCommand",
            "DeAuthorized"
        ]
    },
    "transactionData": {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "timestamp": {
                    "type": "string",
                    "format": "date-time"
                },
                "sampledValue": {

```

```

    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "value": {
          "type": "string"
        },
        "context": {
          "type": "string",
          "enum": [
            "Interruption.Begin",
            "Interruption.End",
            "Sample.Clock",
            "Sample.Periodic",
            "Transaction.Begin",
            "Transaction.End",
            "Trigger",
            "Other"
          ]
        },
        "format": {
          "type": "string",
          "enum": [
            "Raw",
            "SignedData"
          ]
        },
        "measurand": {
          "type": "string",
          "enum": [
            "Energy.Active.Export.Register",
            "Energy.Active.Import.Register",
            "Energy.Reactive.Export.Register",
            "Energy.Reactive.Import.Register",

```

```

"Energy.Active.Export.Interval",

"Energy.Active.Import.Interval",

"Energy.Reactive.Export.Interval",

"Energy.Reactive.Import.Interval",

"Power.Active.Export",

"Power.Active.Import",

"Power.Offered",

"Power.Reactive.Export",

"Power.Reactive.Import",

"Power.Factor",
"Current.Import",
"Current.Export",
"Current.Offered",
"Voltage",
"Frequency",
"Temperature",
"SoC",
"RPM"
    ],
    "phase": {
        "type": "string",
        "enum": [
            "L1",
            "L2",
            "L3",
            "N",
            "L1-N",
            "L2-N",
            "L3-N",
            "L1-L2",
            "L2-L3",
            "L3-L1"
        ]
    }
}

```

```

    ],
    "location": {
      "type": "string",
      "enum": [
        "Cable",
        "EV",
        "Inlet",
        "Outlet",
        "Body"
      ]
    },
    "unit": {
      "type": "string",
      "enum": [
        "Wh",
        "kWh",
        "varh",
        "kvarh",
        "W",
        "kW",
        "VA",
        "kVA",
        "var",
        "kvar",
        "A",
        "V",
        "K",
        "Celcius",
        "Fahrenheit",
        "Percent"
      ]
    }
  },
  "required": [
    "value"
  ]
}

```

```

    },
    "required": [
        "timestamp",
        "sampledValue"
    ]
}

}

},
"additionalProperties": false,
"required": [
    "transactionId",
    "timestamp",
    "meterStop"
]
}

```

10.4.2 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:

```

{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "ReserveNowRequest",
  "type": "object",
  "properties": {
    "connectorId": {
      "type": "integer"
    },
    "expiryDate": {
      "type": "string",
      "format": "date-time"
    },
    "idTag": {
      "type": "string",
      "maxLength": 20
    }
  }
}

```

```

    },
    "parentIdTag": {
      "type": "string",
      "maxLength": 20
    },
    "reservationId": {
      "type": "integer"
    }
  },
  "additionalProperties": false,
  "required": [
    "connectorId",
    "expiryDate",
    "idTag",
    "reservationId"
  ]
}

```

Topic: **cancelreservation**

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

QoS-Level: 1

E.g. MQTT/EVSE/EVSE00001/cancelreservation

Payload JSON Schema:

```

{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "CancelReservationRequest",
  "type": "object",
  "properties": {
    "reservationId": {
      "type": "integer"
    }
  },
  "additionalProperties": false,
  "required": [
    "reservationId"
  ]
}

```

Topic: setchargingprofile

Central System can send a SetChargingProfile.req to a Charge Point, to set a charging profile.

QoS-Level: 1

E.g. MQTT/EVSE/EVSE00001/setchargingprofile

Payload JSON Schema:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "SetChargingProfileRequest",
  "type": "object",
  "properties": {
    "connectorId": {
      "type": "integer"
    },
    "csChargingProfiles": {
      "type": "object",
      "properties": {
        "chargingProfileId": {
          "type": "integer"
        },
        "transactionId": {
          "type": "integer"
        },
        "stackLevel": {
          "type": "integer"
        },
        "chargingProfilePurpose": {
          "type": "string",
          "enum": [
            "ChargePointMaxProfile",
            "TxDefaultProfile",
            "TxProfile"
          ]
        },
        "chargingProfileKind": {
          "type": "string",
          "enum": [
            "Absolute",
            "Recurring",

```



```

        "Relative"
    ]
},
"recurrencyKind": {
    "type": "string",
    "enum": [
        "Daily",
        "Weekly"
    ]
},
"validFrom": {
    "type": "string",
    "format": "date-time"
},
"validTo": {
    "type": "string",
    "format": "date-time"
},
"chargingSchedule": {
    "type": "object",
    "properties": {
        "duration": {
            "type": "integer"
        },
        "startSchedule": {
            "type": "string",
            "format": "date-time"
        },
        "chargingRateUnit": {
            "type": "string",
            "enum": [
                "A",
                "W"
            ]
        }
    }
},
"chargingSchedulePeriod": {
    "type": "array",
    "items": {
        "type": "object",

```

```

        "properties": {
            "startPeriod": {
                "type": "integer"
            },
            "limit": {
                "type": "number",
                "multipleOf" : 0.1
            },
            "numberPhases": {
                "type": "integer"
            },
            "required": [
                "startPeriod",
                "limit"
            ]
        },
        "minChargingRate": {
            "type": "number",
            "multipleOf" : 0.1
        },
        "required": [
            "chargingRateUnit",
            "chargingSchedulePeriod"
        ]
    },
    "required": [
        "chargingProfileId",
        "stackLevel",
        "chargingProfilePurpose",
        "chargingProfileKind",
        "chargingSchedule"
    ]
},
"additionalProperties": false,

```

```
"required": [
    "connectorId",
    "csChargingProfiles"
]
```

10.5 EV WRAPPER MESSAGES

For the communication between the EV Wrappers and the WG 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.

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
  "connectors" : [
    {
      "type" : "chademo",
      "power" : "22" //kW
    }
  ],
  "capacity" : 41, //kWh
}
```

Accepted values for connector types are:

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
}
```

10.6 DLMS/COSEM PUBLICATIONS OF SMX TO WG IOP EXAMPLE

Each individual SMX publishes under a particular (configurable) topic. The usual topic has the following structure:

[SMX unique id]/SMX/

Example: BBB5979/SMX/

MQTT topics are based on OBIS codes, as defined by DLMS/COSEM standard. Those topics are appended to the base topic, as detailed in the following list, conforming a “DLMS over MQTT” channel. For convenience, SMXs also publish the corresponding CIM code for each data item.

```
{
  "LD01": {
    "1-1-14-7-0-255": {
      "-2": "49.96",
      "-5": "2018/01/20 08:03:08:650",
      "Description": "Frequency",
      "OBIS": "1-1-14-7-0-255",
      "CIM": "0.0.0.12.0.1.15.0.0.0.0.0.0.0.224.0.33.0",
      "Unit": "Hz"
    },
    "1-1-32-7-0-255": {
      "-2": "234.75",
      "-5": "2018/01/20 08:03:08:650",
      "Description": "U1",
      "OBIS": "1-1-32-7-0-255",
      "CIM": "0.0.0.0.0.1.54.0.0.0.0.0.0.0.128.0.29.0",
      "Unit": "V"
    },
    "1-1-52-7-0-255": {
      "-2": "233.63",
      "-5": "2018/01/20 08:03:08:650",
      "Description": "U2",
      "OBIS": "1-1-52-7-0-255",
      "CIM": "0.0.0.0.0.1.0.0.0.0.0.0.0.0.64.0.29.0",
      "Unit": "V"
    },
    "1-1-72-7-0-255": {
      "-2": "235.12",
      "-5": "2018/01/20 08:03:08:650",
```

```

        "Description": "U3",
        "OBIS": "1-1-72-7-0-255",
        "CIM": "0.0.0.0.0.1.0.0.0.0.0.0.0.32.0.29.0",
        "Unit": "V"
    },
    "1-1-36-7-0-255": {
        "-2": "40.0",
        "-5": "2018/01/20 08:03:08:650",
        "Description": "P1",
        "OBIS": "1-1-36-7-0-255",
        "CIM": "0.0.0.12.1.1.37.0.0.0.0.0.0.128.0.36.0",
        "Unit": "W"
    },
    "1-1-56-7-0-255": {
        "-2": "40.0",
        "-5": "2018/01/20 08:03:08:650",
        "Description": "P2",
        "OBIS": "1-1-56-7-0-255",
        "CIM": "0.0.0.12.1.1.37.0.0.0.0.0.0.64.0.36.0",
        "Unit": "W"
    },
    "1-1-76-7-0-255": {
        "-2": "40.0",
        "-5": "2018/01/20 08:03:08:650",
        "Description": "P3",
        "OBIS": "1-1-76-7-0-255",
        "CIM": "0.0.0.12.1.1.37.0.0.0.0.0.0.64.0.36.0",
        "Unit": "W"
    },
    "_MeterPoint": "PointOfCommonCouplin-PCC",
    "_MeterType": "USM"
},
"_Actor_type": "Prosumer",
"_GPS": "??? ??? Greece/Kithnos",
"_Project": "H2020 WiseGRID",
"_RBAC": "SMXcore.2018.03.17.000",
"_Service": "Providing real-time values for SCADA service",

```

```
"_User_name": "*****",
"_SMXtimestamp": "2018/01/20 08:03:08:650"
}
```

Example: Phase 1 voltage complete topic: G3M40000001/SMX/LD01/1-1-32-7-0-255/-2

SMX also announces itself by regularly publishing its configuration using CIM message on topic MQTT/config/[SMX unique id]

10.7 TRANSLATION TO CIM METERREADING OBJECTS MESSAGE

```
{
  "Meter" : {
    "mRID" : "BBB5976",
    "reason" : "",
    "remark" : "",
    "value" : "ok"
  },
  "Readings" : [{
    "ReadingType" : "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1",
    "reportedDateTime" : "2018-02-19T08:45:10.000Z",
    "timeStamp" : "2018-02-19T08:45:19.970Z",
    "value" : "02/19/2018 08:45:20",
    "ReadingQualities" : {
      "comment" : "Current time"
    }
  }, {
    "ReadingType" : "0.26.0.1.15.1.12.0.0.0.0.0.0.0.0.224.3.73.0",
    "reportedDateTime" : "2018-02-19T08:45:10.000Z",
    "timeStamp" : "2018-02-19T08:45:20.249Z",
    "value" : 0.3,
    "ReadingQualities" : {
      "comment" : "Reactive Energy QI - Total"
    }
  }, {
    "ReadingType" : "0.26.0.1.16.1.12.0.0.0.0.0.0.0.0.224.3.73.0",
    "reportedDateTime" : "2018-02-19T08:45:10.000Z",
    "timeStamp" : "2018-02-19T08:45:20.269Z",
    "value" : 0.3,
```

```

        "ReadingQualities" : {
            "comment" : "Reactive Energy QII - Total"
        }
    }, {
        "ReadingType" :
"0.26.0.1.17.1.12.0.0.0.0.0.0.0.0.224.3.73.0",
        "reportedDateTime" : "2018-02-19T08:45:10.000Z",
        "timeStamp" : "2018-02-19T08:45:20.289Z",
        "value" : 0.3,
        "ReadingQualities" : {
            "comment" : "Reactive Energy QIII - Total"
        }
    }, {
        "ReadingType" :
"0.0.0.12.0.1.15.0.0.0.0.0.0.0.0.224.0.33.0",
        "reportedDateTime" : "2018-02-19T08:45:10.000Z",
        "timeStamp" : "2018-02-19T08:45:20.480Z",
        "value" : 50.09,
        "ReadingQualities" : {
            "comment" : "Instantaneous net Frequency any phase"
        }
    }, {
        "ReadingType" :
"0.0.0.0.0.1.54.0.0.0.0.0.0.0.0.128.0.29.0",
        "reportedDateTime" : "2018-02-19T08:45:10.000Z",
        "timeStamp" : "2018-02-19T08:45:20.500Z",
        "value" : 235.87,
        "ReadingQualities" : {
            "comment" : "Instantaneous voltage L1"
        }
    }, {
        "ReadingType" :
"0.26.0.1.19.1.12.0.0.0.0.0.0.0.0.224.3.73.0",
        "reportedDateTime" : "2018-02-19T08:45:10.000Z",
        "timeStamp" : "2018-02-19T08:45:20.558Z",
        "value" : 45680.878,
        "ReadingQualities" : {
            "comment" : "Reactive Energy - Total"
        }
    }, {

```



```

        "ReadingType" : "0.26.0.1.1.1.12.0.0.0.0.0.0.0.0.224.3.72.0",
        "reportedDateTime" : "2018-02-19T08:45:10.000Z",
        "timeStamp" : "2018-02-19T08:45:20.569Z",
        "value" : 12354.467,
        "ReadingQualities" : {
            "comment" : "Active Energy + Total"
        }
    }, {
        "ReadingType" : "0.0.0.0.0.1.4.0.0.0.0.0.0.0.128.0.5.0",
        "reportedDateTime" : "2018-02-19T08:45:10.000Z",
        "timeStamp" : "2018-02-19T08:45:20.581Z",
        "value" : 5.35,
        "ReadingQualities" : {
            "comment" : "Instantaneous current L1"
        }
    }, {
        "ReadingType" : "0.26.0.1.1.1.12.0.0.0.0.0.0.0.0.224.3.73.0",
        "reportedDateTime" : "2018-02-19T08:45:10.000Z",
        "timeStamp" : "2018-02-19T08:45:20.593Z",
        "value" : 34567.678,
        "ReadingQualities" : {
            "comment" : "Reactive Energy + Total"
        }
    }
]
}

```

10.8 BATTERY WRAPPER MESSAGES

The aim of this interface with the WiseGrid application is to have a common data model for the battery storage system without regarding the supplier of the system.

This interface is an MQTT interface and the structure is defined as follow:

Base topic: MQTT/battery/[UID of the battery]

Base topic uniquely identifies the battery in the whole system

E.g. MQTT/battery/BAT0001

A “battery” in this context can be an individual, fully-integrated battery system (like VARTA pulse) or consist of several sub-units. In the latter case, SOC and other battery-specific parameters must represent an appropriate collective value for the complete system.

Whether the six BYES batteries at Terni are to be connected to Staas-VPP as one system with one UID or as six individual units with six individual UIDs is beyond the scope of this document and to be decided somewhere else.

The UID of each battery is configured during commissioning of the system or hard-coded into the system. There is not going to be a mechanism to dynamically assign or change UIDs via MQTT.

Topic for battery system data: **system**

Refer to Data Model page „Status & System“ for details.

The battery publishes upon connection to the MQTT broker a JSON message to this subtopic informing of the system information. This publication must be flagged as “retained” (i.e. broker will send it to any new subscriber automatically)

QoS-Level: 1

E.g. MQTT/battery/BAT0001/system

Exemplary payload (refer to Data Model for details):

```
{
  "SN": "0123456789A",
  "FWVersion": "1.0",
  "Power": 1.0,
  "Capacity": 1.0,
  "PVPower": 1.0
}
```

Topic for battery status: **status**

Refer to Data Model page „Status & System“ for details.

The battery publishes upon changes (or periodically) a JSON message to this subtopic informing of the status. This publication may be flagged as “retained” (i.e. broker will send the last status to any new subscriber automatically)

The battery must configure a *MQTT LWT message* updating the status to “disconnected” when the battery disconnects from the MQTT broker.

QoS-Level: 0

E.g. MQTT/battery/BAT0001/status

Exemplary payload (refer to Data Model for details):

```
{
  "MeterActivePower": 1.0,
  "MeterReactivePower": 1.0,
  "InverterActivePower": 1.0,
  "InverterReactivePower": 1.0,
  "InverterPVPower": 1.0,
  "ExternalPV": 1.0,
}
```

```

    "InverterBatteryPower": 1.0,
    "BatterySOC": 1.0,
    "BatterySOH": 1.0,
    "BatteryVoltage": 1.0,
    "MeterGridVoltage": 1.0,
    "MeterGridFrequency": 1.0,
    "Temperatures": 1.0,
    "CosPhi": 1.0,
    "ChargeDisp": 1.0,
    "DischargeDisp": 1.0,
    "Status": 1,
    "Alarms": 1,
    "InverterPVVoltage": 1.0,
    "WorkingMode": 1,
    "Load": 1.0
}

```

Topic for battery commands: **command**

Refer to Data Model page „Command“ for details.

The battery subscribes to this topic, where the control system will publish the commands to be executed by the battery. Publications on this topic shall not be marked as “retained” (i.e. the batteries will only receive commands in real-time)

QoS-Level: 1

E.g. MQTT/battery/BAT0001/command

Exemplary payload (refer to Data Model for details):

Params field must be adapted depending on the specific command parameters.

Id field uniquely identifies the command (to be used to acknowledge the execution)

```

{
  "Id": "b246798a-4574-4c02-9a3c-946407242f49",
  "Command": 0,
  "Params": {
    "WorkingMode": 1
  }
}

```

Topic for battery command acknowledgements: **command/ack**

Upon reception of a command, the battery publishes the result of the execution by publishing a message to this topic

QoS-Level: 1

E.g. MQTT/battery/BAT0001/command/ack

Exemplary payload (refer to Data Model for details):

Id field matches the *Id* field of the received command

Result field is a Boolean indicating result of the execution

Details field may accommodate further details of the result of the execution, depending on the command

```
{
  "Id": "b246798a-4574-4c02-9a3c-946407242f49",
  "Result": 1,
  "Details": {
  }
}
```

10.9 ANCILLARY SERVICES MARKET MESSAGE SPECIFICATION

10.9.1 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]{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:schema>
```

```

<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}\.[0-9]{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 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:complexType>

```



```

        </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: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>
```

```
<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 yyymmddHHMMSSsss (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>

```

10.9.2 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: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>

```

10.9.3 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 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="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: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>

```

10.9.4 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>
<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="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:complexType>
</xs:element>

```

```

</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>

```

10.9.5 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: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="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>

```

10.9.6 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 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="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>

```

10.9.7 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:restriction>
  </xs:simpleType>

```

```

    <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>

```

10.9.8 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:restriction>
  </xs:simpleType>

```

```

    <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: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>

