Hybrid Provision of Energy based on Reliability and Resiliency by Integration of Dc Equipment

Work Package WP5

Open and secure ICT for modular resilient optimized hybrid grid

Deliverable D5.1

HYPERRIDE ICT platform specification

Funding Instrument:	Innovation Action
Call:	H2020-LC-SC3-2020-EC-ES-SCC
Call Topic:	LC-SC3-ES-10-2020 - DC – AC/DC hybrid grid for a modular, resilient and high RES share grid development
Project Start:	1 October 2020
Project Duration:	48 months
Beneficiary in Charge:	Engineering - Ingegneria Informatica SPA (ENG)
Document Identifier:	doi:10.5281/zenodo.5537587

Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the Consortium (including the Commission Services)	
CO	Confidential, only for members of the Consortium (including the Commission Services)	

This project has received funding from the European Union's
Horizon 2020 research and innovation programme under
Grant Agreement No. 957788.

Deliverable Information

Document Administrative Information	
Project Acronym:	HYPERRIDE
Project Number:	957788
Deliverable Number:	D5.1
Deliverable Full Title:	HYPERRIDE ICT platform specification
Deliverable Short Title:	ICT platform specification
Document Identifier:	HYPERRIDE-D51-ICTPlatformSpecification-final
Beneficiary in Charge:	Engineering - Ingegneria Informatica SPA (ENG)
Report Version:	v1.4
Contractual Date:	30/09/2021
Report Submission Date:	23/11/2021
Dissemination Level:	PU
Nature:	Report
Lead Author(s):	M. Mammina, A. Rossi (ENG)
Co-author(s):	H. Humer, P. Smith (AIT), F. Bellesini, E. Mancinelli (EMOT), A. Dognini, C. Joglekar, Z. Pan (RWTH)
Keywords:	Platform, Requirements, Functionalities, Security, FIWARE, interoperability, European Union (EU), H2020 Project, HYPERRIDE, GA 957788
Status:	__ draft, _x_ final, __ submitted

Change Log

Date	Version	Author/Editor	Summary of Changes Made
01/09/2021	v1.0	M. Mammina, A. Rossi (ENG)	Initial version
28/09/2021	v1.1	M. Mammina, A. Rossi (ENG), H. Humer, P. Smith (AIT), F. Bellesini, E. Mancinelli (EMOT), A. Dognini, C. Joglekar, Z. Pan (RWTH)	Draft version for review
08/10/2021	v1.2	D. Dujic (EPFL), F. Bellesini (EMOT)	Internal review and improvements
02/11/2021	v1.3	E. Mrakotsky (AIT)	Language review and improvements
23/11/2021	v1.4	G. Jambrich, T. Strasser (AIT)	Final version

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AC	Alternating Current
ACLs	Access Control Lists
API	Application Programming Interface
ABAC	Attribute-Based Access Control
B2B	Business to Business
CKAN	Comprehensive Knowledge Archive Network
DC	Direct Current
DG	Distributed Generators
EARS	Easy Approach to Requirements Syntax
EC	European Commission
EV	Electric Vehicles
GDPR	European General Data Protection Regulation
GUI	Graphical User Interface
HTTP	HyperTextTransfer Protocol
ICT	Information and Communication Technologies
IdM	Identity Manager
IEC	International Electrotechnical Commission
IoT	Internet of Things
KPI	Key Performance Indicators
LoRaWAN	Long Range Wide Area Network
LWM2M	Lightweight Machine to Machine
M2M	Machine to Machine
MCDA	Multiple-Criteria Decision Analysis
MQTT	Message Queue Telemetry Transport
NGSI	Next Generation Service Interface
NIST	National Institute of Standards and Technology
OPC UA	Open Platform Communications Unified Architecture
OPF	Optimal Power Flow
PBAC	Policy-Based Access Control
PEP	Policy Enforcement Point
PDP	Policy Decision Point
PMUs	Phasor Measurement Units
RBAC	Role-Based Access Control
SGAM	Smart Grid Architecture Model
SIEM	Security Information and Event Management
SSO	Single Sign-On
UML	Unified Modeling Language
VPS	Virtual Private Server
WP	Work Package
XACML	eXtensible Access Control Markup Language
XML	Extensible Markup Language

Executive Summary

ICT plays a crucial role in a smart grid: Digital technology allows to monitor and manage the transport of electricity from all generation sources to meet the different electricity demands of end users. A central logic allows to coordinate the needs and capacities of all generators, network operators, end users and stakeholders in the electricity market in order to:

- optimise the use and operation of resources;
- minimise costs and environmental impacts;
- maximise the reliability, stability, and resilience of the network.

Another potential approach for the sensing and control of power distribution systems could be a decentralised logic. From the ICT perspective, decentralised logic's are interesting in view of their better scalability and better control on privacy.

Measurement sensors, actuators, automation devices, information technology and communication equipment permit to exchange information and to send command, control, and automation signals from the digital management system, which represent the intelligence of the network, to the physical equipment of the infrastructure electric. The control and actuation signals sent to the physical devices of the network are no longer responding to the centralised and unidirectional logic of traditional systems only, but they are the result of advanced management logic's that are managing the flows of energy and power in real time, determining the values of "optimal" setups for distributed generation and load resources (Valenti & Graditi, 2020).

The HYPERRIDE project aims to design and implement an Open ICT Platform enabling:

- the seamless integration and management of devices, regardless of how smart they are;
- scalable and interoperable collection and management of data that support near real time observability and optimisation of the operation of modular and resilient hybrid Alternating Current (AC)/ Direct Current (DC) grids;
- the transmission of commands/setpoints for the safe and reliable operation of the grid;
- detection, prediction, prevention of technical and cyber-contingencies.

This document reports the results of the activities conducted in HYPERRIDE aiming at the elicitation and analysis of functional and non-functional requirements of the Open ICT Platform and high-level design of the platform architecture in terms of static and dynamic behaviour of the system.This document is based on the outcomes of Deliverable D2.2 "Use case description, specification and implementation roadmap report". Moreover, a questionnaire including general and customised questions has been submitted to Work Package (WP) leaders and to pilot leaders to collect expectations from the Open ICT platform in terms of functionalities, standards, data model, constraints, special technologies/tools needed, data necessary for the business processes.

A high-level description is given for each logical module that make up the architecture. Main information exchanges between architectural modules are described. Review and retrospective activities will be conducted to reflect on the iterations of requirements engineering reported in this document in the attempt to improve the process going forward and taking into account the needs that will emerge in the other WPs. New and refined requirements that will emerge in the next phases of the project will be reported in the accompanying report of Deliverable D5.6. Low level details on architectural components and their interdependencies, on processes, and

instructions for the deployment and use will be provided in the accompanying report of Deliverables D5.6 and D5.8. Furthermore, the document gives a short description of the energy services that will be evolved in HYPERRIDE according to the needs of hybrid AC/DC grids.

This document will be used as a starting point for the integration activities foreseen in later on in the project that are summarised in Deliverable D5.6 Open HYPERRIDE ICT platform (preliminary version) and following D5.8 Open HYPERRIDE ICT platform (final version).

1 Introduction

1.1 Purpose and Scope of the Document

The deep penetration of unpredictable and only partially controllable renewable energy sources is evolving the transmission and distribution of electricity networks toward the concept of smart grids. Smart grids and microgrids strongly depend on ICT, since monitoring, analysing, and managing of data are a key layer on top of which optimisation, re-configuration, fault detection, and secure management ensure the reliable, stable, safe and efficient use of the grids.

The HYPERRIDE project aims to design and implement an Open ICT platform enabling:

- the seamless integration and management of devices, regardless of how smart they are;
- scalable and interoperable collection and management of data that are supporting near real-time observability and optimisation of the operation of modular and resilient hybrid AC/DC grids;
- the transmission of commands/setpoints for the safe and reliable operation of the grid;
- detection, prediction, prevention of technical and cyber-contingencies.

The open and secure HYPERRIDE Open ICT platform will be lever to propose new ICT services necessary to set up and manage AC/DC grids.

1.2 Structure of the Document

This document constitutes the HYPERRIDE Open ICT platform specification and it reports the results of the activities conducted in WP5 Task 5.1 “Open ICT platform requirements and architecture specification”. It aims at capturing the HYPERRIDE Open ICT platform functional and non-functional requirements and at providing architecture specifications.

In Section 2, the methodological approach adopted in HYPERRIDE for the requirements elicitation and analysis is introduced and the functional and non-functional requirements of the OPEN ICT Platform are listed. Section 3 provides the overview of the platform architecture and the description of the logical modules of which the architecture is composed. Modules are mapped to the elicited requirements it is expected they meet; main interactions between the modules are described. In Section 4, results are summarised and conclusions are given. Finally, additional information is provided in Appendix A. Non-Functional Req. Minimal Checklist, Appendix B. Services Implemented in HYPERRIDE, and Appendix C. EV Smart Charging System.

1.3 Intended Audience

The intended audience of this report primarily consists of the members of the HYPERRIDE project consortium and European Commission (EC) representatives tasked with reviewing the project and its progress towards meeting the specified milestones. The document communicates the HYPERRIDE Open ICT Platform specifications and high-level design to the members of the development team as it will drive the efforts of the other technical tasks of WP5. Project stakeholders may also use it to evaluate the adequacy of the HYPERRIDE Open ICT Platform from the perspective of their individual areas of expertise.

1.4 Relations to Other Activities

The requirement elicitation process conducted in Task 5.1 and reported in this document has as one of its inputs the results of the activities carried out in WP2 Task 2.2 and reported in the Deliverable D2.2 “Use case description, specification and implementation roadmap report” (Kazmi et al., 2021) as one of its inputs.

Furthermore, the HYPERRIDE Open ICT Platform which was created for the collection of data by the field instruments will be provided to the Open ICT platform according to a specific data model. The acquired data will be leveraged for the energy services developed in WP4, such as hybrid AC/DC state estimation, Optimal Power Flow (OPF), fault location, isolation and restoration services.

2 Requirement Elicitation and Analysis

Requirement elicitation and analysis is a crucial process in the software engineering: it is intended to gain knowledge about customer needs and the environment of a software system. It moulds the shape of the desired end-product and requires effective communication and collaboration between stakeholders.

In this section, the methodology and the results of requirement elicitation and analysis activities carried out in the context of WP5 Task 5.1 are presented. Requirements have been identified taking into consideration functionalities, technologies, constraints provided by the other WPs as well.

2.1 Methodology

Software requirements describe the set of functionalities and features that users, customers, and other stakeholders expect by a target system: functional requirements (e.g., calculations, data manipulation and processing, and other specific functionalities) help to capture the intended behaviour of the system defining what the system should do. Functional requirements are supported by non-functional requirements that specify how the system should perform a certain function; they specify constraints, restrictions, i.e., standards to be used to assess the operation of a system. Functional requirements drive the application architecture, while non-functional requirements drive the technical architecture of the system.

Requirement elicitation refers to the process of researching and discovering the requirements that, in some cases, are obvious, especially when stakeholders specifically request certain features, and in other cases are latent, implicit, and, consequently, need to be elicited. Commonly used elicitation processes are brainstorming, interviews or questionnaires.

Requirement analysis is an iterative process where the gathered requirements are analysed, refined, and scrutinised to make the requirements consistent and unambiguous. Once the analysis phase is completed, the understandability of the project may improve significantly. Requirement modelling is the process of documenting the requirements, usually in different formats, such as use cases, user stories, natural-language documents, or process specification. Review and retrospective activities are conducted to reflect on the previous iterations of requirements engineering in the attempt to improve the process going forward.

In the context of the HYPERRIDE project, brainstorming, questionnaires, and interviews techniques have been adopted for eliciting the requirements. Moreover, the requirement engineering process conducted in the context of Task 5.1 (WP5) has made use of the results of the activity carried out in Task 2.2 (WP2) and reported in the Deliverable D2.2 “Use case description, specification and implementation roadmap report” (Kazmi et al., 2021). In Task 2.2, a methodology based on the well-known standard and reference architectures, namely as National Institute of Standards and Technology (NIST) (*National Institute of Standards and Technology*, n.d.) and Smart Grid Architecture Model (SGAM)(CEN-CENELEC-ETSI Smart Grid Coordination Group, 2012), has been used to identify the context, the boundaries, and actors involved and to define the use cases. The project partners have been involved in several workshops to collect the inputs that have been later analysed and reported in the form of summary use cases. Those summary use cases have been used as the basis to derive detailed pilot uses cases, which were documented using International Electrotechnical Commission (IEC) “Use Case Methodology” method with IEC 62559-2 templates.

Starting from a shared knowledge of the project goal and pilot expectations formalised in Deliverable D2.2, the brainstorming activity has resulted in a list of ideas about the Open ICT platform, spontaneously contributed by the participating project members. A questionnaire was properly prepared, including general questions and customised questions, and was submitted to WP leaders and to pilot leaders to collect expectations in terms of functionalities, standards, data models, constraints, special technologies/tools needed, as well as data necessary for the business processes. Moreover, the pilot leaders have been requested to identify possible physical risks and the information necessary to identify abnormal operation of the pilot. Interviews to WP leaders and pilot leaders have allowed to clarify any doubts on the information collected through the questionnaire and to elicit latent/implicit requirements.

The Natural Language Specification was chosen to formalise the consolidated requirements. The requirements are written in normal plain text and expressed using the Easy Approach to Requirements Syntax (EARS) (Mavin & Novak, 2009), which defines a syntax based on a set of patterns for writing requirements using natural language as follows:

- Ubiquitous requirements: define fundamental properties of the system that have no pre-conditions or trigger.
  For example, the meeting scheduler shall display the scheduled meetings.
- State-driven requirements: designate properties that must be satisfied while a pre-condition holds.
  For example, WHILE the meeting room is available, the meeting scheduler shall allow the meeting initiator to book the meeting room.
- Event-driven requirements: specify properties that must be satisfied once a condition holds.
  For example, WHEN the meeting room is booked, the meeting scheduler shall display the meeting room as unavailable.
- Option requirements: refer to properties satisfied in the presence of a feature.
  For example, WHERE a meeting is rescheduled the meeting scheduler shall inform all participants.
- Unwanted behaviour requirements: define the required system response to an unwanted external event.
  For example, IF the meeting room is unavailable THEN the meeting scheduler shall forbid booking this meeting room (Bennaceur, Tun, Yu, & Nuseibeh, 2018).

The requirements have been then analysed, consolidated, categorised, and refined (describing them at a lower level of abstraction). A unique identifier is assigned to each requirement. Moreover, the requirement is classified and prioritised. The following definitions are intended as a guideline to prioritise requirements.

- Priority 1 – The requirement is a “must have” functionality.
- Priority 2 – The requirement is needed for improved processing, and the fulfilment of the requirement will create immediate benefits.
- Priority 3 – The requirement is a “nice to have” functionality.

In case the timescale does not allow for their implementation, requirements with priority 2 and 3 will be the first to be removed.

2.2 Functional Requirements

Table 1 lists the Open ICT Platform functional requirements. If new requirements emerge in future phases of the project, they will be reported in the accompanying report of Deliverable D5.6.

Table 1: Open ICT Platform Functional Requirements.

ID	Category	Description	Priority
FR 01	Data Acquisition	The platform shall have an interface to the field instruments in order to retrieve measurement data from the field.	1
FR 02	Data Acquisition	Data from electric equipment and field devices shall be acquired directly from the apparatuses or indirectly from SCADA and/or other data acquisition systems.	1
FR 03	Seamless Integration	The platform shall be able to use a uniform language in order to allow the seamless and transparent integration of electric equipment and field devices, regardless of the protocol or data format.	1
FR 04	Data Acquisition	The platform shall be able to receive real-time measurements from smart meters, monitoring devices, and automatic reading meters.	1
FR 05	Data Processing	When low latency is requested, the platform shall be able to perform real-time message processing in-stream.	3
FR 06	Data Processing	The platform shall be able to support the reconciliation and harmonisation of the monitored data to be provided as input to the HYPERRIDE energy services according to the interoperability data model schema.	2
FR 07	Commands/Setpoints Forwarding	The platform shall provide an interface for the transmission of commands/setpoints to the electrical equipment for the safe and reliable operation of the grid.	1
FR 08	Data Processing	The platform shall provide short to mid-term persistence to the collected data.	1
FR 09	Data Persistence	The platform shall be able to provide long term persistence to the collected data.	1
FR 10	Data Usage	The platform shall provide an interface to access historical monitoring data.	1
FR 11	Data Acquisition	The platform shall provide an interface for uploading open datasets in different file formats, (e.g., CSV, XLS, JSON, PDF, etc.)	1
FR 12	Data Acquisition	The platform shall support different types of datasets (historical, live, etc.).	1
FR 13	Data Persistence	The platform shall store open datasets in a file store.	1
FR 14	Data Persistence	A tabular dataset shall be stored in a datastore respecting the structure of the dataset records.	1
FR 15	Data Access	The platform shall provide an interface for accessing open datasets.	1
FR 16	Data Search	The platform shall enable users to search for datasets with various filters.	1
FR 17	Data Access	The platform shall control access to dataset on the basis of access rights set by the data owner.	1

ID	Category	Description	Priority
FR 18	Data Update	The platform shall inform users for updates on datasets they use.	1
FR 19	Data Visualisation	The platform shall permit to visualise an open dataset.	1
FR 20	Data Update	A user having the right to view a dataset can opt to be notified of changes.	1
FR 21	Data Export	The platform shall allow to export an open dataset in different file formats (JSON, XLS, CSV).	1
FR 22	Data Acquisition	The platform shall support the collection of information about detected anomalies that identify problems with devices or assets.	1
FR 23	Data Acquisition	The platform shall support the collection of information about intrusions detected in the network.	1
FR 24	Anomalies Processing	The platform shall identify the root cause of anomalies due to either faults or cyber-attacks.	1
FR 25	Anomalies Processing	When root cause is identified, the platform shall suggest countermeasures.	2
FR 26	Data Acquisition	The platform shall support the collection of information about maintenance interventions in the grid.	1
FR 27	Data Acquisition	The platform shall support the collection of information about the weather forecast from a third-party provider.	1
FR 28	Data Acquisition	The platform shall support the collection of information about the production/consumption forecast.	1
FR 29	Data Persistence	The platform shall be able to store component reliability information for all the interested parties to access at any time.	2
FR 30	Data Acquisition	The platform shall provide an interface for collecting component reliability information.	1
FR 31	Registration	The platform shall allow users (individual/services/applications) to register for an account.	1
FR 32	Authentication	The platform shall identify users (individual/services/applications) before allowing them to use its functions.	1
FR 33	Authentication	The platform shall verify the identity of users (individual/services/ applications) before allowing them to use its functions.	1
FR 34	Authorisation	The platform shall allow to assign access privileges to authenticated users (individual/services/applications).	1
FR 35	Authorisation	The platform shall allow to revoke access privileges to users (individual/services/applications).	1
FR 36	Authorisation	The platform shall check user's access privileges before allowing them to use protected resources.	1
FR 37	Data Integrity	The platform shall prevent the intentional corruption of data/context information collected via unauthorised creation, modification, or deletion.	1
FR 38	Authentication	The platform shall detect attempted accesses that fail identification requirements.	2
FR 39	Authentication	The platform shall detect attempted accesses that fail authentication requirements.	2

ID	Category	Description	Priority
FR 40	Authorisation	The platform shall detect attempted accesses that fail authorisation requirements.	2
FR 41	Data Access	Data shall only be accessible by authenticated and authorised entities.	1
FR 42	Data Processing	The platform shall allow to configure a set of metrics/KPIs based on the historical data.	1
FR 43	Data Processing	The platform shall compute the metrics/KPIs and find the values required for the calculation.	1
FR 44	Data Visualisation	The platform shall display metrics/KPIs results.	1
FR 45	Data Visualisation	The platform shall be able to display customisable time series.	1
FR 46	Data Acquisition	The platform shall include a UI for keeping track of human interventions in the grid.	3
FR 47	Data Persistence	The platform shall record any human intervention in the grid.	3
FR 48	Message Forwarding	The platform shall be able to forward a message from the sender to the recipient.	1

2.3 Non-Functional Requirements

Table 2 lists the Open ICT Platform non-functional requirements. The categorisation for non-functional requirements adopted in HYPERRIDE is reported below:

- Security
- Audit
- Performance
- Capacity
- Availability
- Reliability
- Integrity
- Recovery
- Compatibility
- Maintainability
- Usability
- Documentation
- Legal and Regulatory

A minimal checklist for their identification is reported in Appendix A. Non-Functional Req. Minimal Checklist.

Table 2: Open ICT Platform Non-Functional Requirements.

ID	Category	Description	Priority
NFR 01	Availability	The platform shall be accessible and usable upon demand by an authorized system entity.	1
NFR 02	Usability	The platform shall report errors and security breaches in a timely and automated manner.	1
NFR 03	Usability	User interfaces shall allow actors to easily interact with platform tools.	1
NFR 04	Security	The platform may not grant access to its resources until the user creates a strong password.	2
NFR 05	Security	The user shall change the initially assigned login authentication information (password) immediately after the first successful login.	2
NFR 06	Performance	The system shall be able to scalable depending on the demands related to data ingestion, processing, and storage.	1
NFR 07	Performance	The system shall be able to exchange data with a great number of devices without altering the performance.	1
NFR 08	Performance	Communications shall go through secure TLS channels for guaranteeing confidentiality of information exchanged.	1
NFR 09	Availability	Critical nodes within the platform as shall be replicated and redundant.	1
NFR 10	Legal and Regulatory	The platform must handle data in full compliance with the European General Data Protection Regulation (GDPR).	1
NFR 11	Legal and Regulatory	Avoid and prevent any unnecessary collection, use, and storage of personal data.	1
NFR 12	Security	The platform shall prevent a party to one of its interactions from denying having participated in all or part of an interaction (non-repudiation).	1
NFR 13	Security	The platform shall protect the privacy of the content.	1
NFR 14	Security	The platform shall protect the integrity of the communications.	1
NFR 15	Security	The platform should have the capability to protect personal data using techniques such as anonymisation or pseudonymisation.	1
NFR 16	Security	The platform shall prevent unauthorised actions from being hidden.	1
NFR 17	Security	The platform should have the capability to protect critical applications (or services) from unauthorized or unwanted usage via suitable technologies, such as firewalls and Access Control Lists (ACLs).	1
NFR 18	Security	Only authorised personnel are allowed physical access to computers and the network.	1
NFR 19	Reliability	Backups shall be taken regularly.	1
NFR 20	Security	The platform should have the capability to ensure the confidentiality and integrity of data that is communicated using the public Internet.	1
NFR 21	Compatibility	The Open ICT Platform shall provide reconciliation and harmonisation tools.	1

ID	Category	Description	Priority
NFR 22	Usability	The Open ICT Platform shall support bi-directional data communication.	1
NFR 23	Compatibility	The Open ICT Platform shall allow interoperability with other IT platforms.	1
NFR 24	Security	The platform shall be able to apply privacy policies on different segments of data.	3

 doi:10.5281/zenodo.5537587

3 Overarching Architecture

This section provides an overview of the high-level architecture of the HYPERRIDE Open ICT Platform and its tools. Low level details on architectural components and their interdependencies on processes, and instructions for the deployment and use will be provided in the accompanying report of Deliverables D5.6 and D5.8.

3.1 Methodology

The process that, starting from the elicitation of the requirements, leads to the design of the Open ICT Platform architecture and then to its implementation can be described synthetically as follows (see also Figure 1):

- Goals are achieved through use cases, which describe the behaviour the system should have.
- Use cases are enabled by functional requirements, which describe the functionalities the software system must offer.
- Functional requirements lead to the design and implementation of the software architecture.
- Non-functional requirements describe how functional requirements must work.
- Constraints restrict how functional requirements may be implemented.

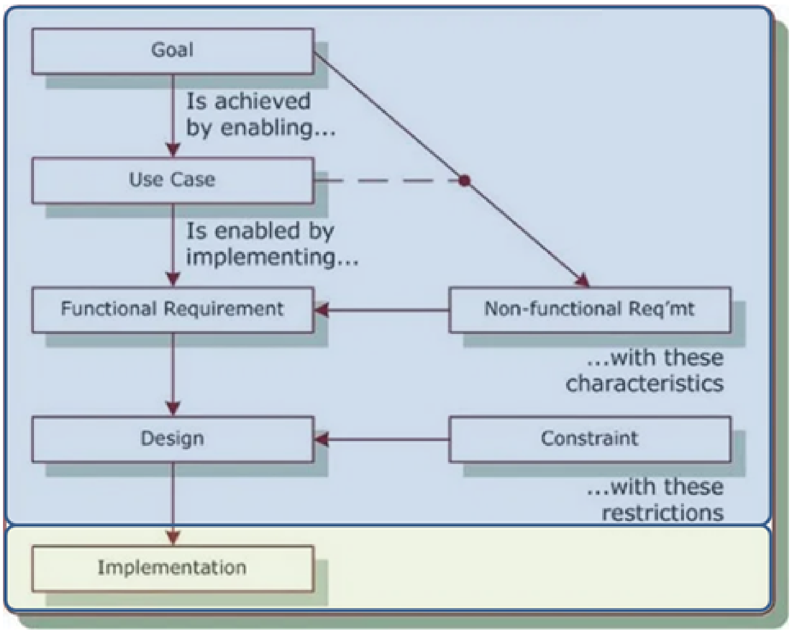

Figure 1: Methodological approach (Lucassen & Dalpiaz, 2016).

A layered architectural pattern has been adopted to provide a general view of the high-level HYPERRIDE Open ICT Architecture. A layered architecture abstracts the view of the system as whole while providing enough detail to understand the roles of individual layers and the relationships in between them. Modules or components with similar functionalities are organised into multiple horizontal layers: the pattern does not fix the number and leaves it upon the preferences of the architect. Each layer plays a specific role in the whole system and contributes to the operation of the layer above it. Each layer depends on the layers beneath it and is completely independent of the layers on top of it. In this way, these layers can be used strictly, where the layer only knows of the layer beneath it, or in a flexible manner, and access all layers beneath it. This leads to layers of isolation: changes made in one layer of the architecture generally do not impact or affect components in other layers: the change is isolated to the components within that layer, and possibly another associated layer. Separation of concerns among modules/components is the best feature of a layered architecture: modules/components within a specific layer are only dealing with the logic that pertains to that layer, making the subsequent phases of development easier.

The architectural modules will be further detailed by describing the architectural components and showing how components are wired together by using Unified Modeling Language (UML) component diagrams.

The dynamic behaviour of the architecture will be described making use of UML sequence diagrams, which depicts the interaction between the parties involved in a process, showing the message interchange in sequential order.

The deployment diagram will be used to map the software architecture created at design stage to the physical system architecture that executes it; it also determines how the software is deployed on the underlying hardware. A guide for the proper deployment will be provided as well.

This document focuses on high-level information of the HYPERRIDE Open ICT Platform architecture, while possible updates on the architecture and all details on components and processes will be provided in the D5.6 accompanying report, and instructions for the deployment will be provided in Deliverable D5.8 accompanying report.

3.2 Overview

The overarching view of the HYPERRIDE Open ICT Platform architecture and integrated tools, depicted in Figure 2, comprises three horizontal layers, namely:

- The *Presentation Layer* is the frontend layer of the architecture and is responsible for handling all user interface and browser communication logic. Presentation layer components will implement the functionalities required to allow users to interact with the system.
- The *Knowledge Layer* represents the underlying domain, mostly consisting of context information and data.
- The *Acquisition and Interoperability Layer* main role is to capture data from different devices and, when requested, to convert those data in standardised context information.

In addition, the architecture includes a cross-cutting *Security Layer*. It is responsible for all concerns related to security, both cyber and physical. From the cyber point of view, it provides for authenticated and authorised access to the system resources. From the physical point of view, it supports situation awareness and provides insights about the root cause of the alerting events generated by the tools that provide for the continuous monitoring of the system behaviour.

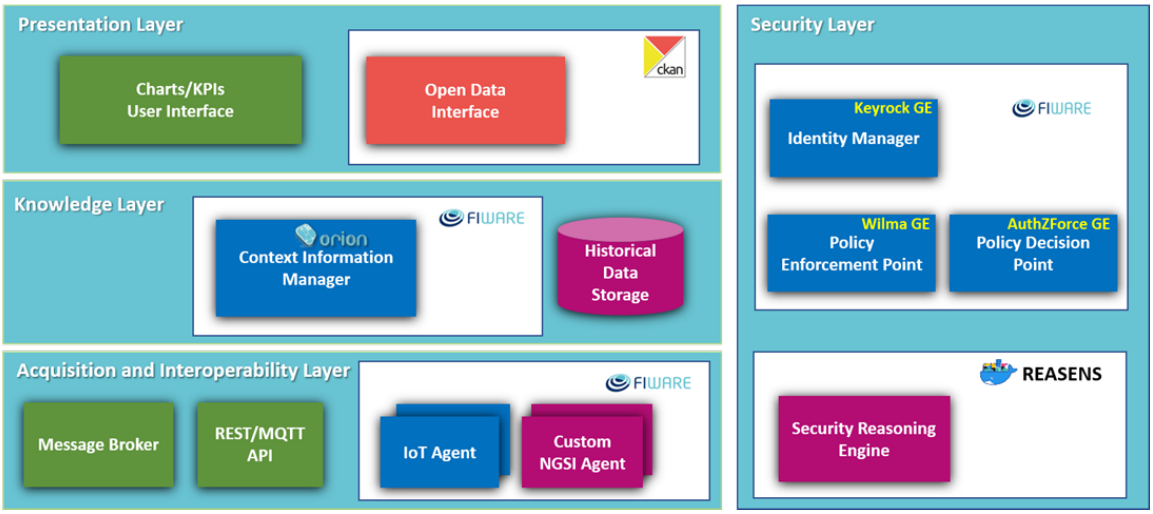

Figure 2: Overarching HYPERRIDE Open ICT Platform Architecture and Integrated Tools.

3.3 Main Modules

In this section, an overview of the main functionalities, main inputs and expected outputs for each architectural logical module is given.

3.3.1 Context Information Manager

The central block is the Context Information Manager, which is represented by the Orion Context Broker from FIWARE[1] (*Future Internet PPP: Led by industry, driven by users Addressing the challenge of Internet development in Europe*, n.d.). The Context Information Manager allows to model, manage, and gather context information at large scale enabling context-aware applications.

Technically, it is a publish-subscribe system, which holds the current state of the application. The changing of the subscribed context is notified. In addition, the Orion Context Broker can manage the whole lifecycle of context information including updates, queries, and registrations. The Context Broker is based on a Mongo NoSQL database with a REST API using the Open Mobile Alliance's Next Generation Service Interface (NGSI) protocol (*NGSI Context Management - Approved Version 1.0*, 2012).

The FIWARE NGSI API defines:

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

[1] FIWARE is a curated framework of open-source platform components that can be assembled together and with other third-party platform components to accelerate the development of Smart Solutions.

 doi:10.5281/zenodo.5537587

The main elements in the NGSI data model are context entities, attributes and metadata, as shown in Figure 3.

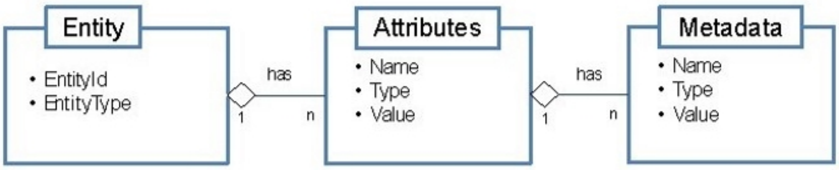

*Figure 3: NGSI data model (*FIWARE-NGSI v2 Specification*, n.d.).*

An entity represents a thing, i.e., any physical or logical object (a sensor, a person, a room, an issue in a ticketing system, etc.). It is characterised by an identifier and a type describing the "thing" represented by the entity. Each entity can have one or more context attributes, or properties:

- a name (what kind of property),
- an attribute type (the NGSI value type of the attribute value),
- an attribute value (the actual data),
- and metadata (describing properties of the attribute value).

Context metadata has:

- a metadata name (describing the role of the metadata),
- a metadata type (describing the NGSI value type of the metadata value), and
- a metadata value containing the actual metadata.

3.3.2 Message Broker

For the sake of completeness, a message broker has been included in the Open ICT Platform to enable applications, systems, and services using data format/protocols for which any NGSI Agent has been conceived to communicate with each other and exchange information anyway. The Message Broker mediates communications, minimising the mutual awareness that applications/system/services should have of each other in order to be able to exchange messages, effectively implementing decoupling. This allows interdependent applications/system/services to talk with one another directly, even if they were written in different languages or implemented on different platforms.

Message Broker is in charge of storing, routing, and delivering messages to the appropriate destinations. It acts as an intermediary for the communications, allowing senders to issue messages without knowing where the receivers are, whether or not they are active, or how many of them there are.

In order to provide reliable message storage and guaranteed delivery, the Message Broker stores and orders the messages in a message queue until the consuming applications can process them. Messages remain in the queue until receipt is confirmed. The asynchronous messaging feature guarantees that messages will be delivered only once and in the correct order relative to other messages. This prevents the loss of valuable data and enables systems to continue functioning even in the case of the intermittent connectivity or latency issues.

doi:10.5281/zenodo.5537587

3.3.3 REST/MQTT API

An Application Programming Interface (API) is a set of rules that defines how applications or devices can connect and communicate with each other. It is a mechanism that enables an application or service to access a resource within another application or service. The application or service doing the accessing is called the client and the application or service containing the resource is called the server. A REST API is an API that conforms to the design principles of the REST, or representational state transfer architectural style, sometimes referred to RESTful APIs; the state of a resource at any particular instant, or timestamp, is known as the resource representation. REST APIs communicate via HyperTextTransfer Protocol (HTTP) requests to perform standard database functions like creating, reading, updating, and deleting records within a resource.

REST APIs are used for synchronous communications: once the request is sent, the caller waits for the resource which is given after the needed elaborations by the responder. In case of asynchronous communications, MQTT protocol can be supported. MQTT implements a classic publish/subscribe pattern. In this case, the caller does not wait for the response, but subscribes to the notification of a particular resource and becomes a subscriber. Once a publisher creates or updates a particular resource, all the subscribers are notified by the platform. An MQTT message consists of a topic and a payload. The topic is the string representation of the path of the resource to which subscribers are registered, while the payload contains the resource itself, sent by the publishers. Resources are generally codified by means of descriptive/markup languages, such as XML, but the most adopted standard is JSON.

3.3.4 IoT Agent

An IoT Agent is a component that lets a group of devices send their data to and have it managed by a Context Broker using their own native protocols. IoT Agents allow to simplify the management and the integration of devices by collecting data from devices through heterogeneous protocols and translating them into the standard platform language, that is in NGSI entities; NGSI also allows to send commands to devices.

IoT Agents should also be able to deal with security aspects of the platform (authentication and authorisation of the channel). NGSI Agents may also facilitate the integration of vertical solutions (smart meters, smart industry, smart building, smart home, energy storages, e-vehicles, etc.) and of other sources of data (e.g., energy management systems, social networks, etc.) with the Open ICT Platform. IoT Agents already exist or are in development for many IoT communication protocols and data models. Available IoT Agents are presented in the FIWARE catalogue (*FIWARE catalogue*, n.d.):

- IoT Agent for JSON – a bridge between HTTP/Message Queue Telemetry Transport (MQTT) messaging (with a JSON payload) and NGSI;
- IoT Agent for Lightweight Machine to Machine (LWM2M) – a bridge between the Lightweight M2M protocol and NGSI;
- IoT Agent for Ultralight – a bridge between HTTP or MQTT messaging (with an UltraLight2.0 payload) and NGSI;
- IoT Agent for Long Range Wide Area Network (LoRaWAN) – a bridge between the LoRaWAN protocol and NGSI;

- IoT Agent for Open Platform Communications Unified Architecture (OPC UA) – a bridge between the OPC UA protocol and NGSI;
- IoT Agent for Sigfox – a bridge between the Sigfox protocol and NGSI.

3.3.5 Custom NGSI Agent

A custom NGSI Agent is an IoT Agent, in which the protocols of devices are not defined in the FIWARE catalogue. The custom NGSI Agent has the same function as IoT Agent. It provides the possibility of customisation and improves the flexibility of selecting devices. One example is the XML protocol, in which the custom IoT Agent is created, based on the IoT Agent Node.js library and the message can be read and sent to the Orion Context Broker.

3.3.6 Identity Manager

The Identity Manager (IdM) module includes the functionalities needed to manage identities and automate identity-related business processes that improve security. It manages specific permissions and policies to authenticate users before having access to resources integrated into the Open ICT platform. The IdM holds all user information and offers a Single Sign-On (SSO) service for the applications so that they do not need to maintain user information (e.g., no private credentials), and one user account can be used for all applications using the platform.

Along with two more security modules, namely the Policy Enforcement Point (PEP) (see Section 3.3.7), which protects a resource by enforcing access control, and the Policy Decision Point (PDP) (see Section 3.3.8), which evaluates the policy and makes an access determination, the IdM enables the OAuth2-based Business to Business (B2B) authorisation security to services and applications. OAuth2 is the open standard for access delegation to grant access rights. B2B OAuth2 usually represents an application that calls another application or service without end user intervention; a client (business client application or "client" as in OAuth spec) makes a call to a service, business service or "resource server" as in OAuth spec), and requests some business information, passing the access token. Since there is no end user intervention, the client needs to be pre-authorised to have access to the resource. The IdM is essentially an OAuth2 authorization server and therefore supports authentication for the entire platform.

In the context of a smart grid, sensors represent a massive security risk, since potential attackers can get easily access to their hardware. Thus, while protecting the communication channel (e.g., with TLS or VPN), enforcing authentication and authorisation is crucial too. The IdM responsible for managing accounts for sensors and for creating permission rules that allows the sensor to only send data of a specific type to a specific service endpoint is eliminating the chance to spoof some other sensor identity. The same approach can be adopted for other field devices.

Since, as stated in the Grant Agreement, a FIWARE-compliant version of the reference Open ICT platform and tools will be implemented, it has been agreed to integrate the FIWARE Generic Enabler KeyRock Identity Manager into the HYPERRIDE Open ICT platform (FIWARE, 2021a). Keyrock is essentially an OAuth2 authorisation server. The main identity management concepts within Keyrock are:

- Users (“resource owners” role in the OAuth2 framework):
  - have a registered account in Keyrock;
  - have credentials;
  - can manage organisations and register applications.
- Organisations:
  - are a group of users that share resources of an application (roles and permissions);
  - users can be members or owners (manage the organisation).
- Applications (“clients” role in the OAuth2 framework):
  - request protected user data;
  - are able to authenticate users using their OAuth credentials (ID and secret) which unequivocally identify the application;
  - define roles and permissions to manage authorisation of users and organisations;
  - can register Pep Proxy to protect backends;
  - can register IoT Agents.

Keyrock provides both a GUI and an API interface.

3.3.7 Policy Enforcement Point

The PEP performs the actual authentication and optional authorisation checks. As anticipated in section section 3.3.6, the PEP interacts with the IdM and the PDP, by requesting and subsequently enforcing authorisation decisions. The Identity Management must be used to create and manage users and applications and to configure roles and permissions for them.

The PEP plays the role of the so-called “resource server” according to OAuth2. The resource server hosts information that is sensitive and, therefore, can only be accessed by authorised requests. In fact, the PEP adds security by transparently acting on behalf of the actual service/application hosting the security information that needs to be protected.

The PEP proxy needs to be informed about addresses and ports of the service/application it is protecting and about of the other security components (IdM and PDP). When a new application is registered in the IdM, a pair of credentials for the PEP proxy is generated and those credentials are used by the PEP to authenticate with the IdM. This process allows to check the authentication. The authorisation decisions can be part of the application logic or can be moved at platform layer security by deploying authentication based on permissions by providing information on user’s roles and request details to the PDP, which verify whether this information fits with its security policies and decides whether access should be granted or not.

FIWARE GE PEP Wilma will be integrated into the reference Open ICT platform (FIWARE, 2021b). Wilma acts as a proxy that enforces access control to services and applications: only permitted users can access to applications or RESTful services. Wilma is a backend component, without frontend interface.

3.3.8 Policy Decision Point

The PDP provides authorisation decisions based on various attributes given by the PEP about each incoming access request, and policies that define multiple rules. It verifies whether these attributes (and therefore the access request) satisfy certain conditions. By replacing all the attribute references in the policy with these input values, PDP is able to evaluate the policy and determine whether the access should be granted.

FIWARE provides a PDP reference implementation called AuthZForce (FIWARE, 2021c), which provides an API to get authorisation decisions based on authorisation policies, and authorisation requests from the PEP. The API follows the REST architecture style, and complies with XACML v3.0 (OASIS, 2021). eXtensible Access Control Markup Language (XACML) is an OASIS standard for authorisation policy format and evaluation logic, as well as for the authorisation decision request/response format. The standard defines a declarative fine-grained, attribute-based access control policy language, an architecture, and a processing model describing how to evaluate access requests according to the rules defined in policies.

As a published standard specification, one of the goals of XACML is to promote common terminology and interoperability between access control implementations by multiple vendors. XACML is primarily an Attribute-Based Access Control (ABAC) system, also known as a Policy-Based Access Control (PBAC) system, where attributes (bits of data) associated with a user or action, or a resource are inputs into the decision of whether a given user may access a given resource in a particular way. Role-Based Access Control (RBAC) can also be implemented in XACML as a specialisation of ABAC.

Although policies may be edited by the IdM user interface, they need to be stored in the PDP as well. Policies are created automatically when defining a role in the IdM (formally written in XACML by the system); however, users are allowed to write custom rules using the Extensible Markup Language (XML).

3.3.9 Security Reasoning Engine

The main responsibilities of the Security Reasoning Engine module are the root-cause analysis for identifying the root causes of faults or problems and the identification of countermeasures to prevent similar incidents. The REASENS Framework has been identified as starting point for the root-cause analysis. It is a hierarchical REAsoning system that enables the collection of events from distributed and heterogeneous SENSors. Its purpose is to support reasoning (analyses) about the potential root causes of events that are generated by sensors, e.g., to determine whether they pertain to a fault or a cyber-attack. The REASENS framework will be extended to enable the ingestion of events from the systems that perform anomaly detection in AC/DC networks. Further, suitable causal models will be developed that can be used to reason about their root causes, detection that leverages state estimation algorithms that will be developed in WP5.

The REASENS framework is based on a micro-service and event-driven architecture, which integrates different security sensors and enables reasoning about the monitored system state, to enable situation awareness. A microservice architecture facilitates the integration of independent (standalone) applications, written in different programming languages. The event-driven architecture enables the system to be dynamically updated whenever something changes without the need to periodically query the state of all the subcomponents. Examples of security sensors include anomaly detection systems that monitor both host and network activity, net-

 doi:10.5281/zenodo.5537587

work intrusion detection systems to check if the network has been manipulated; and host intrusion detection systems to alert on suspicious activities within the hosts. In addition, sensors to ensure system safety could be deployed locally to monitor physical process sensor measurements (anomaly detection) or to check whether control commands are safe, given the current system state (hazardous control detection).

At the time of writing, there are two ways to deploy the REASENS Framework: using the first option, sensors are communicating using the MQTT protocol with a more centrally located component (server), which performs high level alert correlation based on the input from the security sensors. Sensors publish events to an MQTT broker, which are then normalized by a parser and published further to reasoning and complex processing components, e.g., an Evidential Network or Recurrent Neural Networks, or other alert correlation algorithms. Secondly, an implementation has been developed that uses the Elastic Stack[2]. Using this deployment model, sensors place events onto an Apache Kafka-based message queue[3], which distributes them to other components in the architecture. After being processed by Kafka, events can then be normalised using Logstash and forwarded to a reasoning engine or a Security Information and Event Management (SIEM) solution, for example. The normalised events as well as raw messages from the sensors can be logged into a (Elasticsearch-based) database and accessed via a (Kibana-based) Graphical User Interface (GUI). The GUI offers a web interface (dashboard) with presentation of the system state, logging of the security events, alarms (generated by engines), register and configure sensors etc. The REASENS architecture allows for different types of sensors and reasoning engines which could subscribe to messages sent from selected security sensors. In the current implementation, an Evidential Network is deployed as a reasoning engine.

3.3.10 Historical Data Storage

The Historical Data Storage module provides long term persistence to the data. The Context Information Manager module presented in Section 3.3.1, for example, is enabled to store data in the short to medium; therefore, the Historical Data Storage module gives the opportunity to create a historical view of the context.

According to the nature of the data to be stored, a particular kind of database is more appropriate than others. As an example, for monitoring data, the most suitable kind of database is timeseries-based. In this kind of databases the timestamp plays a fundamental role: all measurements have at least a timestamp, which is indexed, and one or more fields to be stored, which are not indexed. Optionally, also tags can be given, which are indexed as well. In this way, this kind of databases is particularly suitable to store and query data by means of time ranges and the optional tags. One example of such a database is influxDB.

Another possible option for long-term storage are No-SQL ones. These databases have the possibility to store and query documents which are not related to particular tables, as it happens in the relational ones, such as MySQL or Postgres. Documents are instead stored in collections and stored and retrieved by a unique id. One example of such a kind of database is MongoDB.

The Historical Data Storage is also conceived as DataStore for open datasets.

[2] https://www.elastic.co/elastic-stack/
[3] https://kafka.apache.org/

3.3.11 Open Data Interface

The Open Data Interface module is in charge of the publication, management, and consumption of open data, including both static and dynamic datasets: it allows to catalogue, upload, and manage open datasets and data sources and also supports searching, browsing, visualising, or accessing open data.

A Comprehensive Knowledge Archive Network (CKAN) based open data portal will be integrated in the HYPERRIDE Open ICT Platform (*CKAN code architecture*, 2018). CKAN is a data management system for powering data hubs and data portals developed and managed by the Open Knowledge Foundation. It is a reference software for the publication of institutional, governmental, and private open data, being widely used also by companies that provide users with information related to the services provided (transport, services to citizens, utilities in general). It is an open-source software, whose core technology is maintained by an active community, while it is modified and extended by an even larger community of developers, who are contributing to a growing library of CKAN extensions.

The units of data published in CKAN are called "datasets" (or packages). A dataset is a collection of data, like measurement temperature readings from sensors. A dataset contains:

- "Metadata", that is information about the data, for example, the title and publisher of the dataset, the date, the formats, etc.
- A number of "resources", which contain the data. A resource can be a CSV or Excel spreadsheet, XML file, PDF document, image file, linked data in RDF format, etc., since CKAN does not mind what format the data is. CKAN can either store the resource internally or store it as a link, in case the dataset is elsewhere on the web.

A dataset can contain any number of resources; for example, different resources could contain the data for different years, or the same data but in different formats. It also offers a powerful API that allows third-party applications and services to be built around it. When enabled, CKAN FileStore allows users to upload data files to CKAN resources. While the FileStore provides the storage of whole files with no way to access or query parts of that file (the file can be only downloaded as a whole), the DataStore is like a database where data elements are accessible via a simple web API and queryable. The DataStore API allows tabular data to be quickly and easily stored inside CKAN DataStore.

3.3.12 Charts/KPIs User Interface

The Charts/Key Performance Indicators (KPI) User Interface module takes raw data, metrics, or KPIs and displays this information in simple charts and graphs on dedicated dashboards. This module provides a simple tool for tracking and analysing performance towards prefixed goals, thus making data-driven decisions more efficient.

According to the data and KPI to be visualised, there are plenty of tools which can be used for this purpose. For example, they can display in a graphical way metrics and raw data taken from the historical series stored in the Historical Data Storage module. This kind of tools allows also to fulfil a series of requirements, such as selection of time intervals and ranges, interpolation methods, and so on. An example of a tool that is well suited to represent time series is Grafana.

3.4 Main Processes

In this section, the main interactions between modules are graphically described through UML sequence diagrams.

3.4.1 Interactions with the Context Information Manager

The sequence diagram in Figure 4 describes the different NGSI interactions an IoT Agent makes with the Context Information Manager, specifically the Orion Context Broker, using as an example an OMA Lightweight Machine to Machine (M2M) device.

If a device connects to the IoT Agent, configurations and device provisioning information must be provided to the IoT Agent. Whenever a device is registered, the IoT Agent reads the device entity information from the request or, if that information is not in the request, from the default values for that type of device. When a request for data (lazy attribute) arrives to the Context Broker, it forwards the request to the Context Provider of that entity, in this case the IoT Agent. The IoT Agent will in turn ask the device for the information needed, transform that information to a NGSI format and return it to the Context Broker.

The latter will forward the response in a transparent way to the caller. Commands are modelled as updates over a lazy attribute. As in the case of the lazy attributes, updates over a command will be forwarded by the Context Broker to the IoT Agent, who will interact with the device to perform the requested action. Parameters for the command will be passed inside the command value. Whenever a device proactively sends a message to the IoT Agent, it should transform its data to the appropriate NGSI format and send it to the Context Broker as an update Context request (*iotagent-node-lib Architecture*, n.d.).

3.4.2 Interactions with Security Components

The FIWARE security framework, which is made of Keyrock IdM, Wilma PEP and AuthZForce PDP, enables controlled and secure access to platform resources (Peter Salhofer, 2020). Instead of allowing direct access to a sensitive service, a client application interacts with the proxy (see Figure 5). The proxy checks authentication with the IdM and forwards the request to the actual resource server, (called “Back-end Apps”) if security constraints are met.

In this scenario (called Level 1 security in FIWARE) only the authentication is checked, but it is not tested, whether the application is allowed to perform the current action. In this framework, there are two options to deploy authorisation:

- Application layer security (Level 1 Security);
- Platform layer security (Level 2 or Level 3 Security).

In the first case, authorisation decisions are made as part of the application logic. Alternatively, the FIWARE platform supports authentication based on so called permissions, taking security related decisions out of the application logic and it performs them on the platform layer. In this scenario, the PEP, after having checked the validity of the access-token with the IdM, makes a consecutive request to the PDP providing the current account user’s roles and the request details. The PDP checks this information with its security policies and decides whether access should be granted or not.

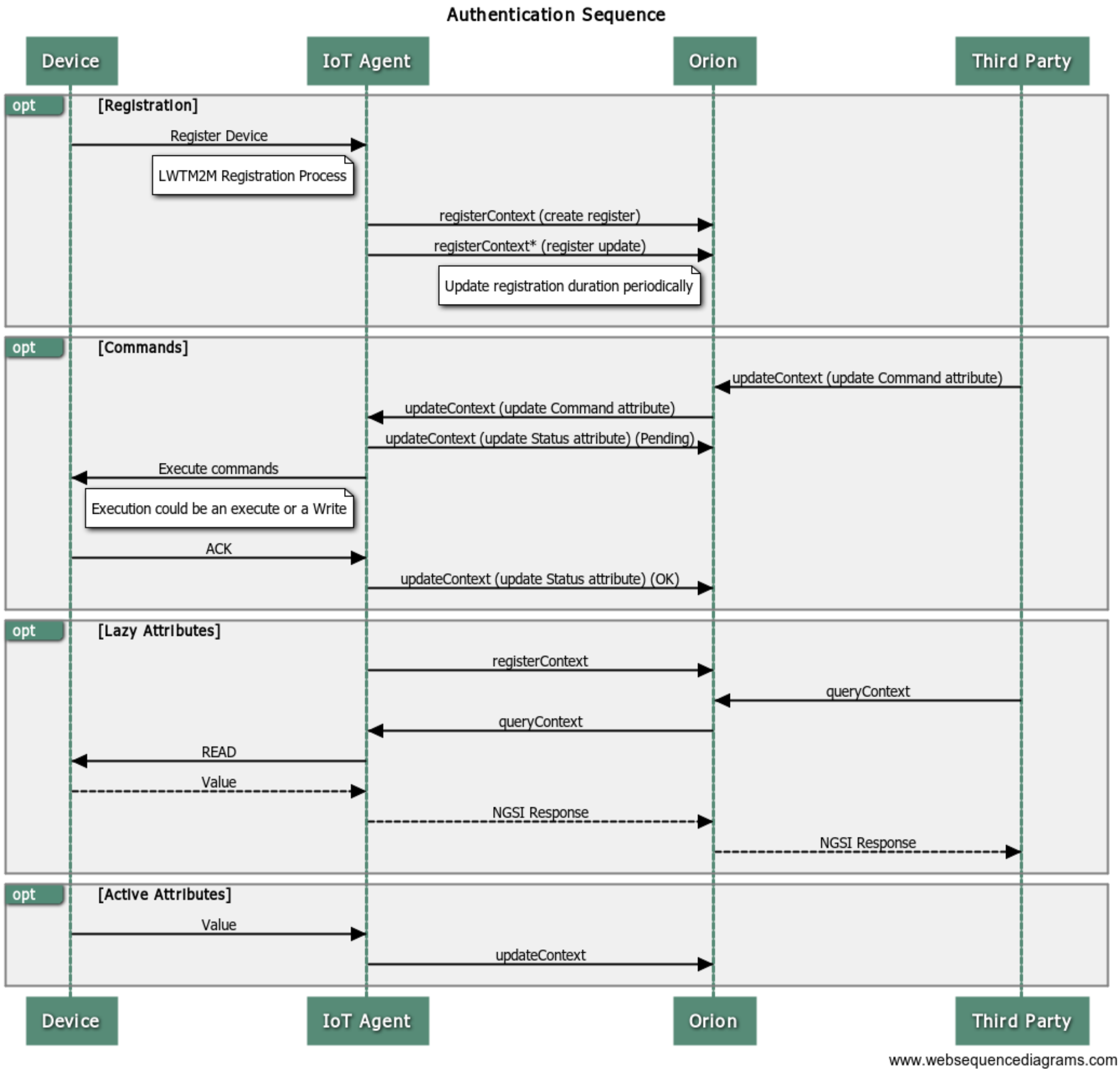

*Figure 4: NGSI interactions of an IoT Agent with the context broker (*iotagent-node-lib Architecture*, n.d.).*

In the case of Level 2 Security, when defining a permission which is always part of a specific role, through the IdM user interface simply a URL and a http verb (GET, POST, PUT, etc.) are provided (that should be granted to all users belonging to the corresponding role), while the corresponding policy in XACML is created automatically. The Level 3 Security allows users to write custom rules using XML.

3.4.3 Interactions through the Message Broker

The Message Broker implements the publish/subscribe messaging pattern. In this message distribution pattern, the producer of each message publishes it to a topic, and multiple message consumers subscribe to topics, which they want to receive messages from. A message topic provides a lightweight mechanism to broadcast asynchronous event notifications, and endpoints that allow software components to connect to the topic in order to send and receive those messages. The messages published to a topic are then distributed to those applications which are subscribed to it.

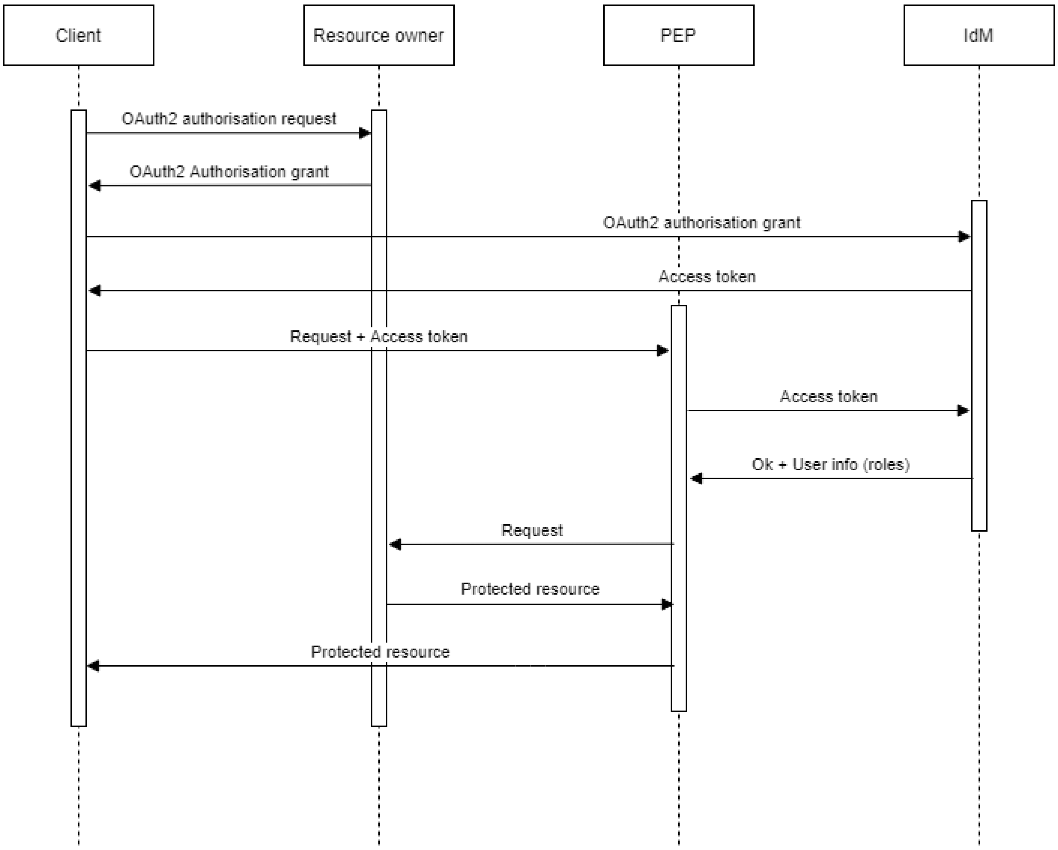

Figure 5: Authentication Level 1 Security.

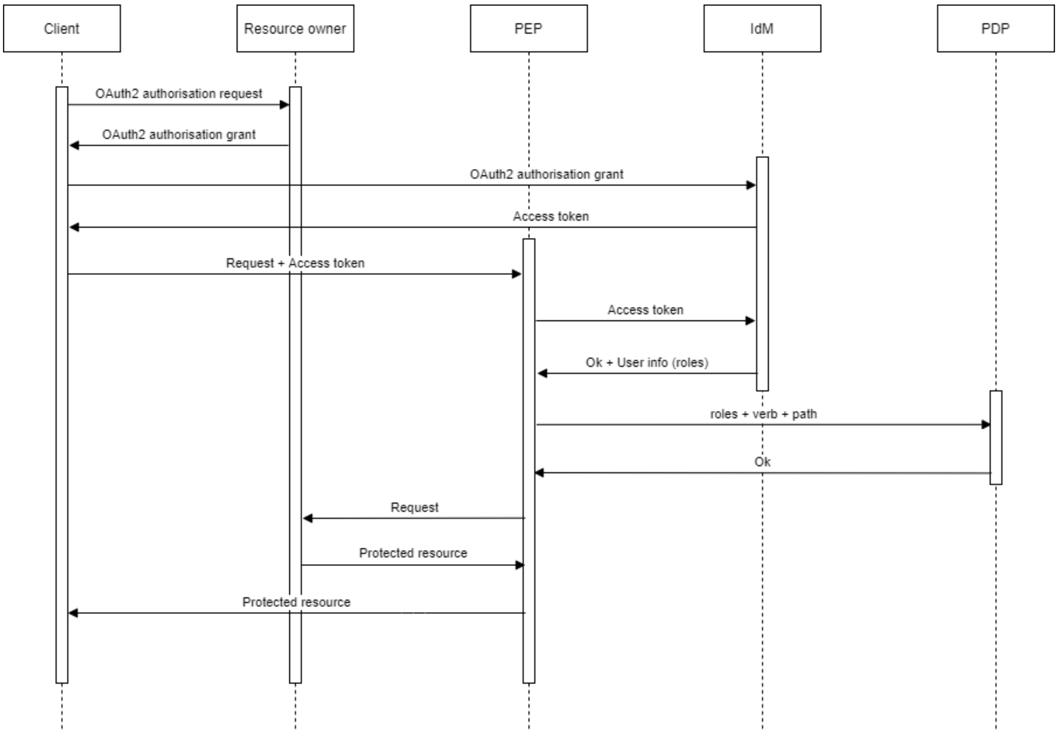

Figure 6: Platform Level Authorisation.

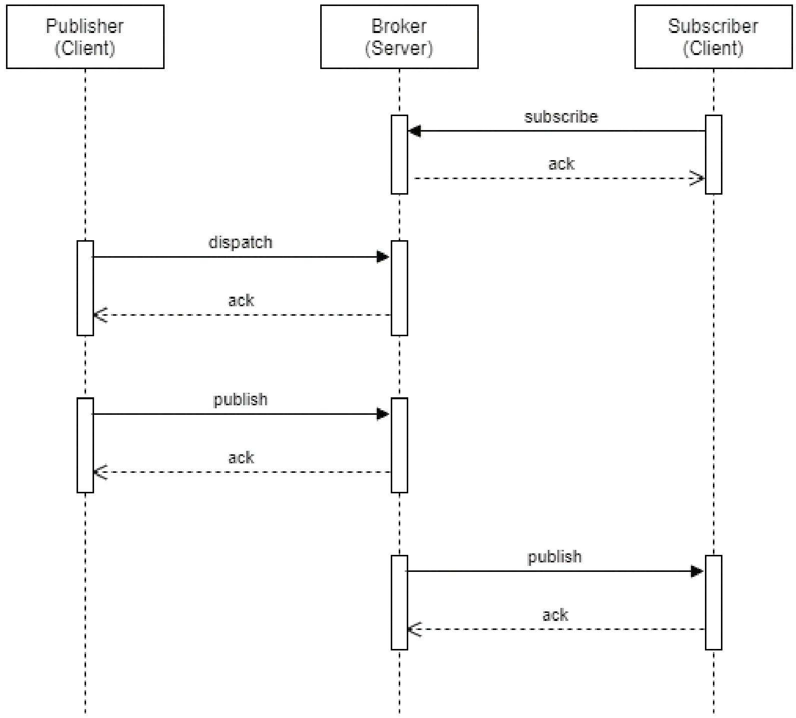

Figure 7: Message Broker: publish/subscribe pattern.

3.5 Mapping Modules-Requirements

In Table 3 each architectural module is mapped to the functional requirements it should meet.

Table 3: Mapping Modules-Requirements.

ID	Category	Description	Context Information Manager	IoT Agent	Custom NGSI Agent	Message Broker	REST/MQTT API	Identity Manager	Enforcement Point	Policy Decision Point	Security Reasoning Engine	Historical Data Storage	Open Data Interface	Charts/KPIs User Interface
FR 01	Data Acquisition	The platform shall have an interface to the field instruments in order to retrieve measurement data from the field.		x	x	x	x							
FR 02	Data Acquisition	Data from electric equipment and field devices shall be acquired directly from the apparatuses or indirectly from SCADA and/or other data acquisition systems.		x	x	x	x							

ID	Category	Description	Context Information Manager	IoT Agent	Custom NGSI Agent	Message Broker	REST/MQTT API	Identity Manager	Enforcement Point	Policy Decision Point	Security Reasoning Engine	Historical Data Storage	Open Data Interface	Charts/KPIs User Interface
FR 03	Seamless Integration	The platform shall be able to use a uniform language in order to allow the seamless and transparent integration of electric equipment and field devices, regardless of the protocol or data format used.	x	x	x									
FR 04	Data Acquisition	The platform shall be able to receive real-time measurements from smart meters, monitoring devices, and automatic reading meters.		x	x	x	x							
FR 05	Data Processing	When low latency is requested, the platform shall be able to perform real-time message processing in-stream.	x											
FR 06	Data Processing	The platform shall be able to support the reconciliation and harmonisation of the monitored data to be provided as input to the HYPERRIDE energy services according to the interoperability data model schema.	x											
FR 07	Commands/ Setpoints Forwarding	The platform shall provide an interface for the transmission of commands/setpoints to the electrical equipment for the safe and reliable operation of the grid.		x	x	x	x							
FR 08	Data Persistence	The platform shall provide short to mid-term persistence to the collected data.	x											
FR 09	Data Persistence	The platform shall be able to provide long term persistence to the collected data.										x		

ID	Category	Description	Context Information Manager	IoT Agent	Custom NGSI Agent	Message Broker	REST/MQTT API	Identity Manager	Enforcement Point	Policy Decision Point	Security Reasoning Engine	Historical Data Storage	Open Data Interface	Charts/KPIs User Interface
FR 10	Data Usage	The platform shall provide an interface to access historical monitoring data.											x	x
FR 11	Data Acquisition	The platform shall provide an interface for uploading open datasets in different file formats, e.g., CSV, XLS, JSON, PDF, etc.)											x	
FR 12	Data Acquisition	The platform shall support different types of datasets (historical, live, etc.).										x	x	
FR 13	Data Persistence	The platform shall store open datasets in a file store.											x	
FR 14	Data Persistence	A tabular dataset shall be stored in a datastore respecting the structure of the dataset records.										x		
FR 15	Data Access	The platform shall provide an interface for accessing open datasets.											x	
FR 16	Data Search	The platform shall enable users to search for datasets with various filters.											x	
FR 16	Data Search	The platform shall enable users to search for datasets with various filters.											x	
FR 17	Data Access	The platform shall control access to datasets on the basis of access rights set by the data owner.						x	x	x				
FR 18	Data Update	The platform shall inform users for updates on datasets they use.										x	x	
FR 19	Data Visualisation	The platform shall permit to visualise an open dataset.										x	x	x

ID	Category	Description	Context Information Manager	IoT Agent	Custom NGSI Agent	Message Broker	REST/MQTT API	Identity Manager	Enforcement Point	Policy Decision Point	Security Reasoning Engine	Historical Data Storage	Open Data Interface	Charts/KPIs User Interface
FR 20	Data Update	A user having the right to view a dataset can opt to be notified of changes.										x	x	
FR 21	Data Export	The platform shall allow to export an open dataset in different file formats (JSON, XLS, CSV).										x	x	
FR 22	Data Acquisition	The platform shall support the collection of information about detected anomalies that identify problems with devices or assets.		x	x	x	x							
FR 23	Data Acquisition	The platform shall support the collection of information about intrusions detected in the network.		x	x	x	x							
FR 24	Anomalies Processing	The platform shall identify the root cause of anomalies due to either faults or cyber-attacks.									x	x		
FR 25	Anomalies Processing	When root cause is identified, the platform shall suggest countermeasures.									x	x		
FR 26	Anomalies Processing	The platform shall support the collection of information about maintenance interventions in the grid.		x	x	x	x							
FR 27	Data Acquisition	The platform shall support the collection of information about the weather forecast from a third-party provider.				x	x							
FR 28	Data Acquisition	The platform shall support the collection of information about the production/consumption forecast.		x	x	x	x							

doi:10.5281/zenodo.5537587

ID	Category	Description	Context Information Manager	IoT Agent	Custom NGSI Agent	Message Broker	REST/MQTT API	Identity Manager	Enforcement Point	Policy Decision Point	Security Reasoning Engine	Historical Data Storage	Open Data Interface	Charts/KPIs User Interface
FR 29	Data Persistence	The platform shall be able to store component reliability information for all the interested parties to access at any time.										x		
FR 30	Data Acquisition	The platform shall provide an interface for collecting component reliability information.												x
FR 31	Registration	The platform shall allow users (individual/ services/ applications) to register for an account.						x						
FR 32	Authentication	The platform shall identify users (individual/ services/ applications) before allowing them to use its functions.						x	x					
FR 33	Authentication	The platform shall verify the identity of users (individual/ services/ applications) before allowing them to use its functions.						x	x					
FR 34	Authentiction	The platform shall allow to assign access privileges to authenticated users (individual/ services/ applications).						x	x	x				
FR 35	Authentication	The platform shall allow to revoke access privileges to users (individual/ services/ applications).						x		x				
FR 36	Authentication	The platform shall check user's access privileges before allowing them to use protected resources.						x		x				
FR 37	Data Integrity	The platform shall prevent the intentional corruption of data/context information collected via unauthorised creation, modification, or deletion.										x		

ID	Category	Description	Context Information Manager	IoT Agent	Custom NGSI Agent	Message Broker	REST/MQTT API	Identity Manager	Enforcement Point	Policy Decision Point	Security Reasoning Engine	Historical Data Storage	Open Data Interface	Charts/KPIs User Interface
FR 38	Authentication	The platform shall detect attempted accesses that fail identification requirements.						x						
FR 39	Authentication	The platform shall detect attempted accesses that fail authentication requirements.							x					
FR 40	Authorisation	The platform shall detect attempted accesses that fail authorisation requirements.								x				
FR 41	Data Access	Data shall only be accessible by authenticated and authorised entities.						x	x	x				
FR 42	Data Processing	The platform shall allow to configure a set of metrics/KPIs based on the historical data.												x
FR 43	Data Processing	The platform shall compute the metrics/KPIs and find the values required for the calculation.												x
FR 44	Data Visualisation	The platform shall display metrics/KPIs results.												x
FR 45	Data Visualisation	The platform shall be able to display customisable time series.												x
FR 46	Data Acquisition	The platform shall include a UI for keeping track of human interventions in the grid.												x
FR 47	Data Persistence	The platform shall record any human intervention in the grid.										x		
FR 48	Message Forwarding	The platform shall be able to forward a message from the sender to the recipient.	x	x	x	x								

doi:10.5281/zenodo.5537587

3.6 HYPERRIDE Services using the Open ICT Platform

This section provides an overview of the services that will be implemented in the context of HYPERRIDE and that will made use of data acquired through the Open ICT Platform (see Figure 8).

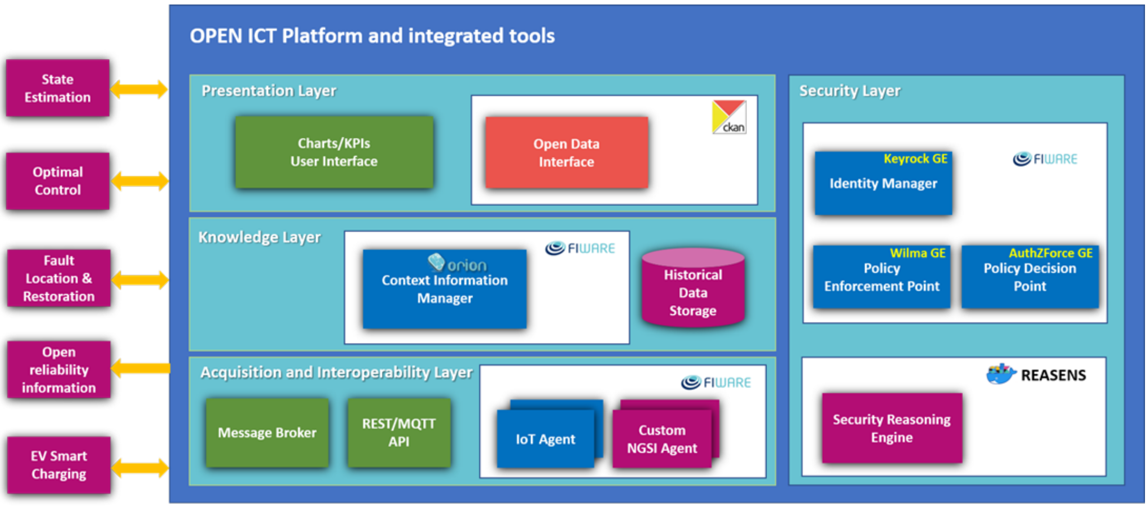

Figure 8: HYPERRIDE services using the Open ICT Platform.

3.6.1 Optimal Control

The Optimal Control service that is being deployed in the HYPERRIDE ICT platform is based on the OPF algorithm for AC/DC grids, developed in Task T4.5 “Monitoring and control automation architecture for Hybrid AC/DC Distribution Network”.

OPF is a grid management technique that consists of determining the most optimal set of variables in an electrical network to achieve the fulfilment of pre-determined criteria related to power flow results. The variables considered in the adjustment can be, for example, the power set points of Distributed Generators (DG) or power converters, the control of transformers tap-changers or the position (open - close) of switching devices. The algorithm considers objective criteria that are pre-defined, as the minimisation of power losses and the efficient exploitation of renewable energy sources. The specific OPF that is developed within the HYPERRIDE project focuses on the management of AC/DC distribution grids, hence it includes the accurate modelling of components and control systems for AC/DC power converters as well as loads and distributed generators in the DC sub-section of the network.

Inputs and Sources

The Optimal Control solution will be implemented within the HYPERRIDE ICT platform as middleware component, interconnected to other software components, using the defined data model for information exchange, and independent from specific grid topologies. The necessary inputs for the execution of the algorithm consists of static and dynamic data:

- *Static Data:* consisting of grid topology (types of nodes and lines, location and position of switching devices, location of AC/DC converters) and grid parameters (impedances/admittances of the lines, technical data of transformers and power converters). The source of these data corresponds, in the HYPERRIDE platform, to the Orion Context Information Management or the Historical Data Storage component.

- *Dynamic Data:* consisting of the actual values of power injections at the nodes, the voltages at the nodes, and the current and power flows along the electrical lines. The State Estimation component, as middleware in the HYPERRIDE platform, constitutes the source of these data.

Outputs and Destinations

The output of the OPF algorithm consists of the computed optimal parameters that satisfy grid constraints and optimise the energy management. In the implemented algorithm specifically tailored for AC/DC distribution grids, the outcomes are power setpoints for AC/DC converters. The destination of the computed commands are the actuators installed in the fields of HYPERRIDE pilot sites, reached via the software components Orion Context Information Manager and specific IoT Agent, which implements the used communication protocol.

3.6.2 State Estimation

State estimation is a technique used to acquire full knowledge of the power system in real-time conditions. Specifically, it consists of processing the available measurements, in order to provide an optimal estimate of the current operating state. Usually, the states are constituted by voltage phasors at each system bus for a given point in time. Consequently, all the other electrical quantities of the network are computed via the fundamental electro-technical functions.

In case of hybrid AC/DC grids, the main obstacle is constituted by the exchange of power flow values among the AC and DC sub-portions of the network (i.e., the modelling of losses and operating conditions and AC/DC converters). Two approaches are identified: in the first case, the two-steps algorithm initially solves the state estimations separately in AC and DC grid sub-portions and, then, it combines the outcomes to achieve an accurate estimation of the power network states. The second method relies on data provided by Phasor Measurement Units (PMUs) to linearly model the state estimation problem and solve it in a single iteration.

In the context of HYPERRIDE ICT platform, the State Estimation solution will be integrated with additional grid services, as the Optimal Control and Service Restoration middleware, which are being developed in the Tasks T4.5 “Monitoring and control automation architecture for Hybrid AC/DC Distribution Network”.

Inputs and Sources

As well as the Optimal Control, also the inputs of the State Estimation are divided among static and dynamic data:

- *Static Data:* the provision of these data occurs only once, when a new network is interconnected. They consist of grid topology (types of nodes and lines, location of AC/DC converters), grid parameters (impedances/admittances of the lines, technical data of transformers and power converters), and the accuracies of measurement devices. In case of strict necessity, due to an unfavourable condition of sensing infrastructures, also forecasts uncertainties are included. The source of these data corresponds, in the HYPERRIDE platform, to the Orion Context Information Management, which stores according to the standardised data models, or the Historical Data Storage component.

- *Dynamic Data:* corresponding to the measurements provided from the field devices (transducers, current and voltage transformers, smart meters) via the IoT Agent and adapted to NGSI to interface FIWARE components. The measurements consist of current and power injections at the nodes, voltages at the nodes, the current and power flows along the electrical lines. Eventually, forecasted power consumed by loads or injected by distributed generators (from forecasting load/generation profiles) may be included as measurements. The source of these data corresponds to the Historical Data Storage component.

Outputs and Destinations

The outputs of State Estimation solution are the states (quantification of electrical network variables, e.g., nodes voltages or branch currents) computed at each node with the indication of their computed accuracies. Consequently, the remaining operating conditions of the power grid are computed from the aforementioned states. The configuration of the HYPERRIDE ICT platform foresees the interconnections of several grid services to achieve their primary functions: the outcomes of State Estimation are directly used by the Optimal Control and the Service Restoration middleware.

3.6.3 Service Restoration

The occurrence of faults, as short-circuit, in the electrical networks requires fast and efficient countermeasures, to reduce the dangerous impacts (for users and devices) and to return quickly to necessary operating conditions. Particularly, after the localisation and isolation of a faulted area, the distribution grid needs to be reconfigured in order to re-energise the disconnected nodes; this procedure relates to the Service Restoration solution. In case of AC/DC distribution grids, the additional adjustment of AC/DC power converters setpoints constitutes a fundamental operation in improving the power flow.

The Service Restoration solution, developed and used within HYPERRIDE tasks, is a middleware software component particularly tailored to the fault management for AC/DC distribution grids, at Medium Voltage level. The re-energising of disconnected nodes relies on the reconfiguration of grid topology (by closing normally open tie-switches) and the adjustment of AC/DC converter setpoints. The solution makes use of two additional HYPERRIDE grid services: the State Estimation and the Optimal Control solutions (described in the paragraphs above). Moreover, the Service Restoration solution considers as reconfiguration criteria the number of restored loads, their criticality, the power losses, and the priority of tele-controlled switches; the solution is computed by implementing a Multiple-Criteria Decision Analysis (MCDA) approach.

Inputs and Sources

Similar to other HYPERRIDE grid services the Service Restoration solution is executed by considering two different groups of input data: static and dynamic:

- *Static Data:* as for the State Estimation solution, the provision of these data occurs only once, when a new network is interconnected. They consist of grid topology (types of nodes and lines, location and type of switching devices and location of AC/DC converters), grid parameters (impedances/admittances of the lines, technical data of transformers and power converters, criticality and technical data of loads and distributed generators), and the accuracies of measurement devices and forecasts. The source of these data corresponds, in the HYPERRIDE platform, to the Orion Context Information Management, which stores according to the standardised data models, or the Historical Data Storage component.

- *Dynamic Data:* They are corresponding to the measurements provided from the field devices (transducers, current and voltage transformers, smart meters) and the grid statuses via the IoT Agent, and are adapted to NGSI to interface FIWARE components. The measurements consist of current and power injections at the nodes, voltages at the nodes, and the current and power flows along the electrical lines. Eventually, forecasted power consumed by loads or injected by distributed generators (from forecasting load/generation profiles) are included as measurements. Additionally, the positions of the switching devices and the tripping conditions are necessary data to execute the Service Restoration solution. The source of these data corresponds to the Historical Data Storage component.

Outputs and Destinations

The destinations of Service Restoration outcomes are different, at the various steps of the service. The algorithm structures candidate topologies for the network reconfiguration, which are then optimised by the adjustment of AC/DC converter setpoints: this is carried out via the execution of State Estimation and the OPF algorithms. The outcomes of the OPF service constitute then an internal input to the Service Restoration.

The final outcome of the Service Restoration corresponds to the operating (opening - closing) commands for switching devices, and the setpoints of AC/DC power converters. The destinations of these data are the actuators (protection relays and converter controllers) installed in the field of HYPERRIDE pilot sites, reached via the software components Orion Context Information Manager and a specific IoT Agent, which implements the used communication protocol.

3.6.4 Open Reliability Information

The goal of the Open Reliability Information system is to provide a common platform for storing and sharing component reliability information. The shared data will consist of system and subsystem reliability and maintenance statistics, information on system structure and operation conditions as well as estimations on data quality. The data will be used in quantitative reliability and availability assessments.

The input for this database (see Figure 9) comes on the one side from literature and product specification, on the other side from data collections of existing real components (devices).

The Open Reliability Information system can benefit from observed information from real life by interpreting *fault tracking, logbooks,* and *observations*.

On the other side the information system can provide statistics on probabilities of failures on system components under different environmental conditions.

The statistics can be evaluated by a single component instance or over a larger group of similar components, either provided by the own organisation/company or shared between many co-

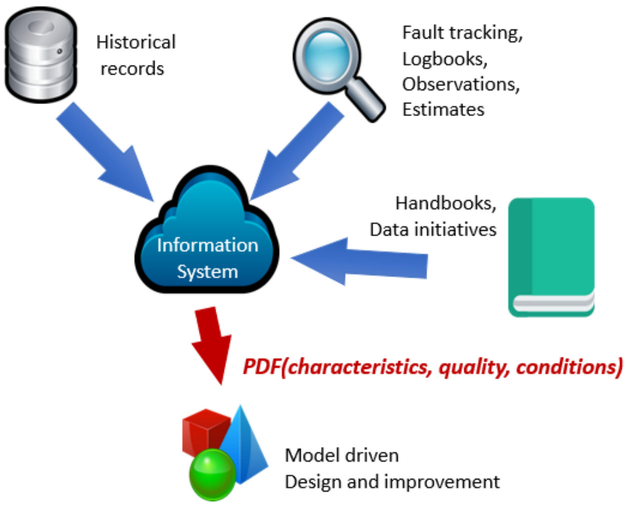

Figure 9: Context of Open Reliability Database.

operating organisations. This possibility allows to provide statistics on a bigger set of samples while respecting privacy of individual records.

Inputs and Sources

Inputs can be the information on the stability and reliability of the system, coming from fault tracking or logging. Examples of such information are:

- Identification of the system or subsystem
- Environmental conditions (temperature, radiation, electric field intensity, . . .)
- Failure mode
- Failure costs
- Time of interruption
- Maintenance activities
- Changed spare parts
- Running time of component.

It is a necessary precondition, that both sides are reaching an agreement on the semantic of the exchanged information and both sides have the same picture on the structure of the systems and subsystems. The system relies on the methods of international standard *ISO 14224:2016* (“Collection and exchange of reliability and maintenance data for equipment‘”). It is possible to support also an interface for configuring the reliability information system by a REST interface.

Outputs and Destinations

The outcome of the Open Reliability Information system would be the Probability Density Function (PDF) on a single component or on a group of components.

3.6.5 EV Smart Charging

Electric Vehicles (EV) smart charging services are provided to increase grid operation efficiency. Through a real-time monitoring system and a remote management system, the EV smart charging stations can be used to provide energy flexibility to the electricity grid, so as to avoid reverse power flow due to the distributed renewable energy plants.

Inputs and Sources

The EV smart charging service will receive inputs from the optimal control service to modulate charging station power output during charging sessions, as well as start or stop charging sessions.

Outputs and Destinations

The EV smart charging service will provide outputs to the state estimation service, delivering real-time data collected from the charging stations and EV in order to assess electricity grid conditions and forecast potential energy flexibility provisioning. In Appendix B. Services Implemented in HYPERRIDE, details are given on the EMOT EV Smart Charging System that will be integrated in the HYPERRIDE Open ICT Platform in the Terni pilot site.

3.7 Next Steps

The adopted approach to the requirement analysis is an "iterative" process, rather than a "waterfall" one. For this reason, the work about the architecture will continue also beyond the submission of the current deliverable, to further refine and optimise the architecture with the emerging needs inside the project. This report, in this perspective, is the first consolidated starting point for the actual implementation of the platform which will take into account the complete list of requirements which will emerge also beyond the submission of this report itself.

The work started from Deliverable D2.2 use cases to start eliciting the requirements presented, limited to the ICT specific needs. The evolution of the use cases will be taken into account, for a possible further refinement of the ICT requirements as well. At the same time, the "HYPERRIDE services" presented here will be integrated and evolved according to the needs coming from the pilots. The platform has been designed as generic as possible right to avoid limitations and to be ready to integrate theoretically anything which may emerge as necessary in the course of the project.

In order to further refine and specialise on the ICT requirements, a series of calls will be arranged with the main stakeholders:

- the penetration test team, for the specifics of the cyber-security aspects;
- the REASENS team, for its support as reported in Deliverable D2.3 (Stöckl et al., 2021) to cascading-effect mitigation tool developed in WP4;
- the pilot leaders, to follow the ongoing refinement of the specific use cases.

Once the requirements of the pilots are clarified, new non-functional requirements for the ICT part may be identified and reported as contribution to D6-7-8.1 deliverables. The specialisation of the “HYPERRIDE Services” will be studied in WP4 and integrated in WP5. The possible updates on the architecture and all details on components and processes will be provided in Deliverable D5.6 accompanying report, in which the final specifications will be developed into a first prototype, while instructions for the deployment will be provided in Deliverable D5.8 accompanying report. Moreover, the specific needs of hybrid AC/DC grids, in terms of cyber-security and mitigation of cascading effects, will be addressed in the next deliverables pertaining to Task 5.3, starting from Deliverable D5.3, due at the end of project month 24.

4 Conclusions

The HYPERRIDE project envisions the field implementation of DC and hybrid AC/DC grids by identifying and providing solutions to overcome barriers for a successful roll-out of new infrastructure concepts throughout Europe.

In this vision, an Open ICT Platform has been conceived for the scalable and seamless integration and management of devices and electricity equipment and collection of data that are supporting near real-time observability and optimisation of the operation of modular and resilient hybrid AC/DC grids.

This document gives a brief introduction of the methodologies adopted to produce the results. The results of the elicitation and the analysis of functional and non-functional requirements at the basis of the HYPERRIDE Open ICT Platform have been reported. Functional and non-functional requirements have been deduced starting from a shared knowledge of the project goals and pilot expectations formalised in Deliverable D2.2 and coming from the answers WP leaders and pilot leaders had given in a survey that had been properly prepared to collect expectations in term of functionalities, standards, data models, constraints, special technologies/tools needed, and data necessary for the business processes.

A specifications review and retrospective activities that will be conducted in the next phases of the project will bring out new and refined requirements that will be reported in the accompanying report of Deliverable D5.6.

The core part of the report relates to the description of the Open ICT Platform architecture: it provides a general overview of the whole architecture and the high-level description of each functional module that has been identified. A layered architectural pattern has been adopted to provide a general view of the high-level HYPERRIDE Open ICT Platform:

- The *Presentation Layer* is the frontend layer of the architecture and is responsible for handling all user interface and browser communication logic. Presentation layer components will implement the functionalities required to allow users to interact with the system.
- The *Knowledge Layer* represents the underlying domain, mostly consisting of context information and data.
- The *Acquisition and Interoperability Layer*'s main role is to capture data from different devices and, if requested, to convert those data in standardised context information.

The main interactions between modules have been illustrated through sequence and activity diagrams. Moreover, an overview of those energy services has been provided that will be evolved in the context of HYPERRIDE for the safe and reliable operation of hybrid AC/DC grids and that will make use of data acquired through the Open ICT Platform.

The specifics of the cyber-security aspects will be provided in Deliverable D5.3. Possible updates on the architecture and all details on components and processes will be provided in the D5.6 accompanying report, while instructions for their deployment will be provided in Deliverable D5.8 accompanying report.

References

Bennaceur, A., Tun, T., Yu, Y., & Nuseibeh, B. (2018). *Requirements engineering*.

CEN-CENELEC-ETSI Smart Grid Coordination Group. (2012). *Smart Grid Reference Architecture.* Retrieved from https://ec.europa.eu/energy/sites/ener/files/documents/xpert_group1_reference_architecture.pdf

CKAN code architecture. (2018). Retrieved from https://docs.ckan.org/en/2.9/contents.html

FIWARE. (2021a). *Identity Manager - KeyRock.* Retrieved from https://fiware-idm.readthedocs.io/en/latest/index.html

FIWARE. (2021b). *Pep proxy - wilma.* Retrieved from https://fiware-pep-proxy.readthedocs.io/en/latest/

FIWARE. (2021c). *Welcome to authzforce's official documentation.* Retrieved from https://authzforce-ce-fiware.readthedocs.io/en/latest/

FIWARE catalogue. (n.d.). Retrieved from https://www.fiware.org/developers/catalogue/

FIWARE-NGSI v2 Specification. (n.d.). Retrieved from http://fiware.github.io/specifications/ngsiv2/stable/

Future Internet PPP: Led by industry, driven by users Addressing the challenge of Internet development in Europe. (n.d.). Retrieved from https://www.fi-ppp.eu/

iotagent-node-lib Architecture. (n.d.). Retrieved from https://iotagent-node-lib.readthedocs.io/en/latest/architecture/index.html

Kazmi, J., Strasser, T. I., Smith, P., Stöckl, J., Jambrich, G., Dognini, A., ... Aghaie, H. (2021). *Use case description, specification and implementation roadmap report.* Deliverable D2.2, HYPERRIDE Consortium. doi:10.5281/zenodo.4772166

Lucassen, & Dalpiaz. (2016). The use and effectiveness of user stories in practice. *Requirements Engineering: Foundation for Software Quality*.

Mavin, H., Wilkinson, & Novak. (2009). Easy approach to requirements syntax (ears). *17th IEEE International Requirements Engineering Conference*, 317–322. doi:10.1109/RE.2009.9

National Institute of Standards and Technology. (n.d.). Retrieved from https://www.nist.gov/

NGSI Context Management - Approved Version 1.0. (2012). Retrieved from http://www.openmobilealliance.org/release/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf.

OASIS. (2021). *extensible access control markup language (xacml) version 3.0.* Retrieved from http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

Peter Salhofer, P. D. (2020). Evaluating the fiware platform. In *53rd hawaii international conference on system sciences.* doi:10.24251/HICSS.2020.809

Stöckl, J., Jambrich, G., Milnera, M., Kapeller, J., Fuchs, N., Smith, P., ... Norrga, S. (2021). *Enabling technologies requirements and specification report.* Deliverable D2.3, HYPERRIDE Consortium. doi:10.5281/zenodo.4772129

Valenti, M., & Graditi, G. (2020). *Le smart grid per un futuro energetico sostenibile e sicuro.* doi:10.12910/EAI2020-043

Appendix A. Non-Functional Req. Minimal Checklist

An example of a minimal checklist for the elicitation of non-functional requirements is reported below:

- Security
  - Login requirements - access levels, CRUD levels.
  - Password requirements - length, special characters, expiry, recycling policies.
  - Inactivity timeouts – duration’s, actions.
- Audit
  - Audited elements – what business elements will be audited?
  - Audited fields – which data fields will be audited?
  - Audit file characteristics - before image, after image, user and time stamp, etc.
- Performance
  - Response times - application loading, screen open and refresh times, etc.
  - Processing times – functions, calculations, imports, exports.
  - Query and Reporting times – initial loads and subsequent loads.
- Capacity
  - Throughput – how many transactions per hour does the system need to be able to handle?
  - Storage – how much data does the system need to be able to store?
  - Year-on-year growth requirements.
- Availability
  - Hours of operation – when is it available? Consider weekends, holidays, maintenance times, etc
  - Locations of operation – where should it be available from, what are the connection requirements?
- Reliability
  - Mean Time Between Failures – What is the acceptable threshold for down-time?
  - Mean Time To Recovery – if broken, how much time is available to get the system back up again?
- Integrity
  - Fault trapping (I/O) – how to handle electronic interface failures, etc.
  - Bad data trapping – data imports, flag-and-continue or stop the import policies, etc.
  - Data integrity – referential integrity in database tables and interfaces.
  - Image compression and decompression standards.
- Recovery

  - Recovery process – how do recoveries work, what is the process?
  - Recovery time scales – how quickly should a recovery take to perform?
  - Backup frequencies – how often is the transaction data, set-up data, and system (code) backed-up?
  - Backup generations - what are the requirements for restoring to previous instance(s)?
- Compatibility
  - Compatibility with shared applications – What other systems does it need to talk to?
  - Compatibility with 3rd party applications – What other systems does it have to live with amicably?
  - Compatibility on different operating systems – What does it have to be able to run on?
  - Compatibility on different platforms – What are the hardware platforms it needs to work on?
- Maintainability
  - Conformance to architecture standards – What are the standards it needs to conform to or have exclusions from?
  - Conformance to design standards – What design standards must be adhered to or exclusions created?
  - Conformance to coding standards – What coding standards must be adhered to or exclusions created?
- Usability
  - Look and feel standards - screen element density, layout and flow, colours, UI metaphors, keyboard shortcuts
  - Internationalisation / localisation requirements – languages, spellings, keyboards, paper sizes, etc
- Documentation
  - Required documentation items and audiences for each item.

Appendix B. Services Implemented in HYPERRIDE

This appendix provides details on the automation services implemented in the context of the HYPERRIDE project to provide innovative solutions enabling resilient autonomous self-healing as well as protection of hybrid AC/DC grids, while improving network observability through real-time system awareness.

B.1 Open Reliability Information

The services to the reliability database can be accessed by a REST interface. The definition of the interface is available as Swagger interface[4]. The following types of services are currently available:

- Organisation administration: defining an update companies and organisation, participating in the open reliability database.
- User administration: administration of users for each organisation.
- Taxonomy and meta catalogue administration: defining and updating the meta catalogue, defining taxonomies
- Equipment class administration: defining the characteristic of an equipment class (attributes, rules, bounding borders, etc.)
- Location definition: definition of the system, where parts are installed.
- Equipment definition and maintenance: definition of components and associating them to equipment classes and installation points.
- Inserting of maintenance activities and failure events for equipment components.
- Retrieving statistics.

The main focus for interfacing will be the insertion of maintenance and failure events. Detailed definitions will follow in a later phase.

[4] https://service.ait.ac.at/aries_service/swagger/ (username/password can be provided upon request)

Appendix C. EV Smart Charging System

EMOT EV smart charging system for HYPERRIDE project is hosted into a Virtual Private Server (VPS) whose details are:

- CPU: 2 core 3.1 GHz;
- HDD: 50 GB;
- RAM: 4 GB;
- S.O.: Ubuntu 16.04 LTS.

Into the VPS, run simultaneously EV Wrapper Server, OCPP server and API REST. The image below describes the EMOT network topology, which is divided into three main networks:

- Green network, related to EMOT headquarters and charging stations;
- Red network, related to electric vehicles;
- Orange network, related to EMOT VPS.

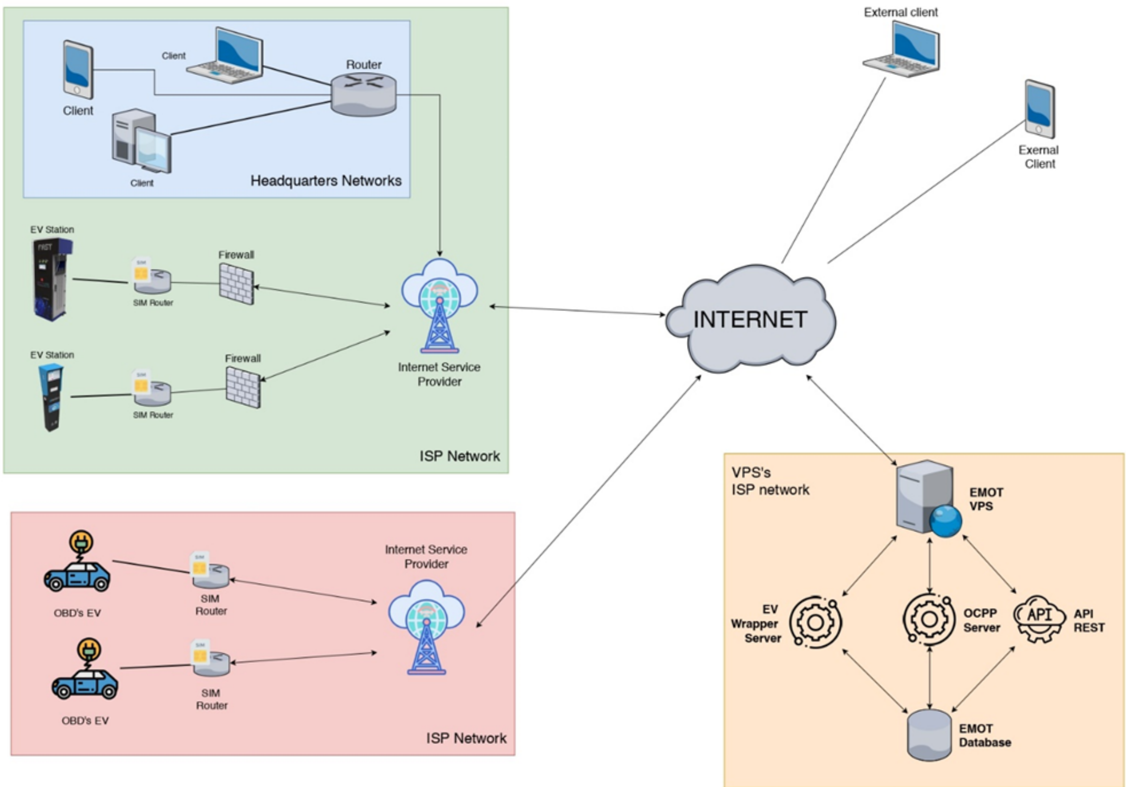

Figure 10: EMOT network topology.

EMOT charging stations are exchanging data through a modem connected to a single-board computer, a Raspberry Pi 3, with a CPU of quad-core ARM Cortex A53 1.2 GHz, a SD of 16 GB, a RAM of 1 GB and a Raspbian Stretch 4.14 S.O.; charging station protocols are OCPP (application protocol for communication between charging stations and EMOT central management system) and websocket (computer communications protocol, providing full-duplex communication channels over a single TCP connection). EMOT OCPP server accepts communications and data exchange only with the client program that is installed in the charging station computer. OCPP server accepts the connection by the client only and exclusively if a valid authentication key is used at the time of the request. Charging station data format is JSON and the sampling rate is one second. Charging station data collected are: Charging Station ID, Charging Station Electric Current Real-Time Value, Charging Station Voltage Real-Time Value, Socket ID, Socket Status, Charging Session ID, Charging Session Start Time, Charging Session End Time, Charging Session Energy Value, Charging session Cost Value. From the actuation point of view, EMOT charging stations are designed to receive commands via EMOT API, to remotely modify charging station power output or to start and stop a charging session.

Regarding EV monitoring, EMOT use an on-board diagnostic (OBD) device to retrieve data from the EV; OBD is an IoT component, based on a Raspberry Pi 3 and Carberry; Carberry represents the link between car electronics and Raspberry Pi, which allows the development of end-user applications. OBD utilise a TCP/IP communication to a TCP/IP server. The network connectivity of the OBD device is via data SIM (UMTS), thanks to a Raspberry module that works as a modem, and the server is a python software; OBD protocol is MQTT and the sampling rate is 5 seconds. The OBD connects to the diagnostic interface from which it is able to extract the information from the electric vehicle control unit using the CAN-bus protocol. The output data format of the OBD is an ASCII string; when the data is sent to the server, it is reorganised into a wrapper, thus obtaining a grouping of the data in JSON format. EV data collected are:

- Measure ID: unique identifier of a specific measurement;
- Vehicle ID: unique identifier of a specific EV;
- Brand: EV manufacturer name;
- Model: EV model name;
- Battery Power (kW): maximum EV charging power value;
- Battery Capacity (kWh): maximum EV battery energy capacity value;
- Connector Type: EV charging connector type name;
- Autonomy (km): real-time EV kilometers autonomy value;
- Odometer (km): real-time EV odometer value;
- State of Charge percentage: real-time EV State of Charge percentage value;
- Timestamp: record of the time of measurement event.

Consortium

Disclaimer

All information provided reflects the status of the HYPERRIDE project at the time of writing and may be subject to change.

Neither the HYPERRIDE Consortium as a whole, nor any single party within the HYPERRIDE Consortium warrant that the information contained in this document is capable of use, nor that the use of such information is free from risk. Neither the HYPERRIDE Consortium as a whole, nor any single party within the HYPERRIDE Consortium accepts any liability for loss or damage suffered by any person using the information.

This document does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of its content.

Copyright Notice

This project has received funding from the European Union's
Horizon 2020 research and innovation programme under
Grant Agreement No. 957788.