

Grant Agreement No.: 773715

Project acronym: RESOLVD

Project title: Renewable penetration levered by Efficient Low Voltage Distribution grids

Research and Innovation Action

Topic: LCE-01-2016-2017

Next generation innovative technologies enabling smart grids, storage and energy system integration with increasing share of renewables: distribution network

Starting date of project: 1st of October 2017

Duration: 36 months

D4.1 – Detailed description of the platform

Organization name of lead contractor for this deliverable: ICOM	
Due date:	M12 - 30 th of September 2018
Submission Date:	30 th of September 2018
Primary Authors	Isidoros Kokos, Kostas Tsakalos, Michael Houhoulis, Anna Tatsaki, Ilias Lamprinos (ICOM)
Contributors	Luisa Candido (EYPESA) Ferran Torrent (UdG) Francesc Girbau (UPC)
Version	Version 1.0 – Final Version

Dissemination Level		
PU	Public	X
CO	Confidential, only for members of the consortium (including the Commission Services)	

DISCLAIMER

This document reflects only the author's view and the Agency is not responsible for any use that may be made of the information it contains.

Deliverable reviews

Revision table for this deliverable:		
Version 0.3	Reception Date	24 of September 2018
	Revision Date	28 of September 2018
	Reviewers	Heribert Vallant (JR)
Version 1.0	Reception Date	29 of September 2018
	Revision Date	30 of September 2018
	Reviewers	Ilias Lamprinos (ICOM)

Contributions of partners

Description of the contribution of each partner organisation to the work presented in the deliverable.

Partner	Contribution
UdG	Data models of external services.
UPC	Data model of PED status.
SIN	-
JR	Final review of the document.
ICOM	Editor and main contributor.
EYPESA	Operation Application user interface design, Data Models of Legacy Systems.
CS	-

Table of contents

Acronyms and abbreviations	5
Executive Summary	7
1. Introduction.....	9
1.1. RESOLVD Project.....	9
1.2. Objectives and Methodology	9
1.3. Report structure	9
2. RESOLVD Platform.....	11
2.1. Overview	11
2.2. Design Principles	11
2.3. High Level Architecture.....	13
3. The Enterprise Service Bus	15
3.1. Overview	15
3.2. Design Principles	17
3.3. Architecture.....	19
3.3.1. Logical View.....	19
3.3.2. Process View	20
3.3.3. Physical View.....	22
3.4. Components Design	23
3.4.1. Middleware	23
3.4.1.1. Description.....	24
3.4.1.2. Internal Logic	24
3.4.1.3. Interfaces.....	24
3.4.2. Business Process Engine	24
3.4.2.1. Description.....	24
3.4.2.2. Internal Logic	25
3.4.2.3. Interfaces.....	25
3.4.3. ExtAdapter	26
3.4.3.1. Description.....	26
3.4.3.2. Internal Logic	26
3.4.3.3. Interfaces.....	26
3.4.4. Data Access <i>Manager</i>	27
3.4.4.1. Description.....	27
3.4.4.2. Internal Logic	27
3.4.4.3. Interfaces.....	27
3.4.5. Service Manager.....	27
3.4.5.1. Description.....	27
3.4.5.2. Interfaces.....	28
3.4.6. Security Access <i>Manager</i>	28
3.4.6.1. Description.....	28
3.4.6.2. Interfaces.....	28
3.4.7. System Manager.....	28
3.4.7.1. Description.....	28
3.4.7.2. Internal Logic	29
3.4.7.3. Interfaces.....	29
3.5. Graphical User Interface Design.....	29
3.6. External Interfaces Design	29
3.7. Data design	31
3.7.1. Canonical Data Model	31
3.7.2. External Data	32
4. The Data Analytics Platform.....	38

4.1.	Overview	38
4.2.	Design Principles	39
4.3.	Architecture	41
4.4.	Components Design	42
4.4.1.	Analytics Service	42
4.4.2.	Visualization Service	42
4.4.3.	Data Hub	42
4.4.4.	Access Control	42
4.5.	User Interface Design	43
4.6.	External Interfaces Design	43
4.7.	Data design	44
5.	AAA Server	47
6.	Operation Applications	51
6.1.	Overview	51
6.2.	Design Principles	52
6.3.	Architecture	52
6.4.	Internal Logic	54
6.5.	Graphic User Interface Design	54
7.	Conclusions	62
	References	63

Acronyms and abbreviations

AAA	Authentication, Authorization and Accounting
API	Application Programming Interface
BPE	Business Process Engine
CEF	Critical Event Forecaster
CEPA	Critical Event Prevention Application
CDI	Contexts and Dependency Injection (JAVA)
CIM	Common Information Model
CSS	Cascading Style Sheets
DAP	Data Analytics Platform
DBMS	Database Management System
DMS	Distribution Management System
DoW	Description of Work
DSO	Distribution System Operator
EF	Energy Forecaster
ESB	Enterprise Service Bus
FDA	Fault Detection Application
GIS	Geographic Information System
GOS	Grid Operation Scheduler
HDFS	Hadoop Distributed File System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
IPMA	Island Power Management Application
JSON	JavaScript Object Notation
JMS	Java Message Service
KPI	Key Performance Indicator
LRA	Loss Reduction Application
MDMS	Meter Data Management System
MOM	Message Oriented Middleware
NTLFDA	Non-Technical-Loss-Fraud Detection Application
PED	Power Electronic Device
PFS	Power Flow Simulator
PMU	Phasor Measurement Unit
PQM	Power Quality Monitoring
RDF	Resource Description Framework
REST	REpresentational State Transfer
RPC	Remote Procedure Call
SCADA	Supervisory Control and Data Acquisition system
SG	Smart Grid
SM	Smart Meter
SOA	Service Oriented Architecture
SoC	State of Charge
SOAP	Simple Object Access Protocol
UC	Use Case
UI	User Interface



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 773715

UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAMS	Wide Area Monitoring System
XML	Extensible Markup Language

Executive Summary

The RESOLVD project aims at increasing the observability and controllability of Low Voltage (LV) electricity distribution networks with the use of innovative ICT, power electronic and sensor infrastructures.

This report documents the design of the software platform to be used by the DSOs (namely the RESOLVD platform), providing a detailed description of its subsystems and the interactions among each other, with external systems and services, as well as with the end-users. The report presents the results of the work done in the context of task T4.1 “Enterprise Service Bus” and T4.2 “Data Analytics Platform” and is highly related to previous work of the project: D1.1 – “Use cases definition”¹, D1.2 “Functional and operational requirements” and D1.3 – “Interoperability and Integration Analysis and Requirements”.

The presentation of each level of the design follows a specific pattern. Initially, an overview of the system and constraints guiding the design process are presented, followed by the rationale of the final design: e.g. specific architectural patterns, standards, use of third party software tools/libraries. The documentation of the high-level design, presents different architectural views (where appropriate), according to ISO/IEC/IEEE 42010:2011 “Systems and software engineering—Architecture description”. Furthermore, on a subsystem basis a detailed design is presented, detailing the interfaces, internal logic and relevant data models. Unified Modelling Language notation, was utilized where possible, for describing static and/or dynamic behaviour and internal logic of the detailed design.

Initially the High-Level Architecture of the Platform is presented, depicting its main subsystems: the Enterprise Service Bus (ESB), acting as an integration middleware; the Data Analytics Platform (DAP), acting as a central data repository and data analysis and visualization provider; the Authentication, Authorization and Accounting (AAA) Server, serving as the security infrastructure; and the Operation Applications, managing advanced grid operations and providing a UI for end-users (the DSO). The architectural design decisions, mainly concern the adoption of Smart Grid standards (i.e. IEC 61968), a widely used standards aiming at interoperable communication of legacy systems in the domain of the Distribution Grids; the use of the service based architectural principles of Service Oriented Architecture (SOA), for achieving reusability and loose coupling; the adoption of a central data repository, for mitigating problems related to data accessibility and reducing the communication traffic to production infrastructure of the DSO; as well as security mechanisms, aiming to facilitate the integration of security in a network of services.

The Enterprise Service Bus (ESB) is the infrastructure enabling the seamless integration of data exchanges and services and provide loose coupling of applications. Its design was based on the functional and non-functional requirements documented in D1.2 and D1.3 and under constraints related to performance, security and data privacy. The IEC 61968 standard was used as a guideline in the design process and led in the adoption of Common Information Model (CIM), an open standard for representing information of the Power System, as the canonical data model for communications with the ESB. The “off-the-shelf” software building blocks of the design and standardized communications mechanisms are documented in the rationale of the design. UML component and deployment diagrams were used to present the logical and physical decomposition of the system, whilst activity and sequence diagrams were used to present its dynamic behavior. Our design approach presented herein includes the detailed analysis of the

¹ Access restricted to consortium members

above components, the documentation of external interfaces, the presentation of data structures of information to be transformed to the CIM and the description of the UI to be provided to the operator of the system.

The Data Analytics Platform (DAP) has the role of enabling the transparent integration of heterogeneous data technologies and vendor subsystems, as well as various data types and acts as a centralized storage of data for the project. Furthermore, it provides analytics computations and visualization of data as a service. Its design was guided by scalability, performance, availability and reliability constraints - given its role - and data security, privacy and integrity – given the nature of the data stored. The design maps to the requirements of the Data Management, Analysis and Visualization tools identified in report D1.3. Our design approach presented herein includes also the presentation of the external interfaces and the data structures managed by DAP.

The AAA Server is the security infrastructure offering Authentication, Authorization and Accounting and enabling the control of user access to network resources, as well as tracking of relevant activities. The AAA server is encapsulated behind a mediator (in our case the ESB) and provides services for authenticating and authorizing service requestors (clients) for accessing resources (services), in the context of a role-based access mapping (permissions) of services. The design rationale is to mitigate the problem of security from the service provider to the AAA Server and facilitate integration of security mechanism among the different services existing in the project. In this document, we also present of UI mock-ups for the administrator of the system.

Finally, in this document we present the Operation Applications, i.e. web-applications that enable the end-users to use the advanced functions related to grid monitoring and control developed in the project. The main features of the user interface are presented with mock-ups, indicating the depicted information and the expected behaviour of the application.

1. Introduction

1.1. RESOLVD Project

The objective RESOLVD project is to improve the efficiency and the hosting capacity of distribution networks, in a context of highly distributed renewable generation by introducing flexibility and control in the low voltage grid.

An innovative advanced power electronics device, with integrated storage management capabilities, will provide both switching and energy balancing capacities to operate the grid optimally. Continuous power flow control between storage and the grid, and also between phases, will result in a flatter and reduced demand curve at the substation level with an associated loss reduction and an improved voltage control and quality of supply.

The enhanced observability of RESOLVD, provided through cost-effective PMUs and state-of-the-art short-term forecasting algorithms that predict demand and renewable generation, will permit a reduction of uncertainty in grid operation and an increased efficiency. RESOLVD proposes hardware and software technologies to improve low voltage grid monitoring with wide area monitoring capabilities and automatic fault detection and isolation.

This improved observability and monitoring system combined with the capability of actuating on the grid will benefit from robust scheduling methods to support self-healing and grid reconfiguration, hence allowing efficient grid operation and a maximised renewable hosting capacity.

1.2. Objectives and Methodology

The objective of this report is to present the detailed design of the software platform to be used by the DSOs (namely the RESOLVD platform), presenting its architectural elements and the detailed design of their components. It addresses mostly technical staff (e.g. software architects, software developers) who are interested in understanding the design of the individual subsystems and their interoperation and will serve as a guideline during the implementation phase.

The design analysis presented herein will be realized in the context of the tasks T4.1 “Enterprise Service Bus” and T4.2 “Data Analytics Platform” and is highly related to previous work of the project. More specifically with D1.1 “Use cases definition”, which provides the definition of the Use Cases (UCs) of the project and the business and technical actors, as well as with D1.2 “Functional and operational requirements” [1], which provides the list of components to be developed or integrated in the project and their functional and non-functional requirements. It also relates to D1.3 “Interoperability and Integration Analysis and Requirements” [2], which presents the integration aspects of the project and its conceptual architecture and finally to D1.4 “Information Security requirements”, which tackles the issues related information security.

The adopted methodology involved initially the modelling of the high-level design of each system using the constraints identified as guidelines. This high-level decomposition was followed by a more detailed design of the individual sub-systems. Unified Modelling Language notation [3], was utilized where possible, for describing static and/or dynamic behaviour and internal logic during the detailed design process. The design of external software interfaces and user interfaces, as well as the design of the main data structures stored or exchanged is also presented.

1.3. Report structure

This section summarises the work presented in each of the chapters in the report:

- Chapter 1 provides an introduction to the project, the scope and objective of the report and the methodology followed;
- Chapter 2 presents the high-level architecture of the RESOLVD platform, the assembly of the different systems developed in the project, allowing integration with legacy systems and external services, the execution of complex business processes and analytic algorithms for advanced grid operations process, as well as for provision of visualization services. The chapter also details the design issues that were faced and considerations in general that influenced the design process;
- Chapter 3 focuses on the Enterprise Service Bus (ESB), the middleware responsible for message transformation, content-based routing as well as other features that allow seamless integration among enterprise applications. The chapter details the design of the ESB, as well as the rationale of the design decisions;
- Chapter 4 presents the design and relevant considerations of the Data Analytics Platform (DAP), the infrastructure that acts as a central data repository and is responsible for providing computational analysis and visualization as external services;
- Chapter 5 discusses the AAA Server, the security infrastructure utilized in the project for securing the platform from unauthorized access or malicious attempts.
- Chapter 6 presents the Operation Applications, the means of the operator (end-user) interacting with the platform and details the design of user interfaces.

2. RESOLVD Platform

This chapter provides a high-level overview of the structural and functional decomposition of the RESOLVD platform, presenting its structural elements, their roles and interrelations.

2.1. Overview

The RESOLVD platform is a software platform responsible for:

- Transparent integration with legacy systems (i.e. MDMS, SCADA, GIS), the Power Electronic Devices (PEDs), supervision and analytics services as well as external systems (i.e., Weather Forecaster, WAMS), through a common communication infrastructure;
- Integration of heterogeneous data sources and data types (e.g. smart metering data, weather station data, load consumption / generation forecasts), offering validation and homogenization of data and guaranteeing accessibility with specific QoS characteristics;
- Hosting the operational applications, which manage the business flow of advanced grid functions (DMS applications) and provide a user interface;
- Providing analytic computations as a service to other applications and presenting both stored data and results of computations in a UI, embeddable in other applications.

2.2. Design Principles

This section presents a brief analysis of the most crucial factors and decisions leading to the design that will be presented at the sub-subsequent sections. An overview of the requirements and constraints of the architectural elements of the platform is documented in D1.2 [1] and D1.3 [2].

The architectural design is based on the principles of Service Oriented Architecture (SOA). SOA provides an approach for creating service-based architectures, where each service carries out a small, well-defined set of functionality, reusable and independent. One can individually use these (reusable) services or bind them in larger aggregated (specific) ones with a process called orchestration. This approach provides a great amount of flexibility, reusability and loose coupling of applications. In the case of the project, where several functions are provided as services (e.g. supervision and analytics services), SOA provides a suitable approach.

Furthermore, the design uses the specification described in IEC 61968 "Application integration at electric utilities - System interfaces for distribution management" [4] as a guideline for integration of legacy systems of the DSO. The standard is intended to support the inter-application integration of a utility enterprise that needs to connect disparate applications that are already built or new (legacy or purchased applications) each supported by dissimilar runtime environments. IEC 61968, which is part of the Common Information Model (CIM) [5] standards series, supports exchange of applications' data on an event driven basis and implemented with middleware services (Message Brokers, Message Oriented Middleware - MOM, Message- Queuing Middleware, and Enterprise Service Buses-ESB). The design is based on the use of an ESB as a middleware.

In the example presented in Figure 1, interface adapters are used as a means to integrate many of the legacy systems with other application systems that are IEC 61968 (or CIM) compliant. Information are published through middleware services (in the standard format) enabling multiple "listeners" and better monitoring of the underlying processes. The CIM is an implementation agnostic model, defining information used by electric utilities in UML as classes along with the relationships between these classes. The exchange of information in CIM is implemented through

RDF (Resource Description Framework) models stored in a standard format, for example, XML (eXtensible Markup Language). Some sample classes of CIM related to distribution grid are presented in Figure 2.

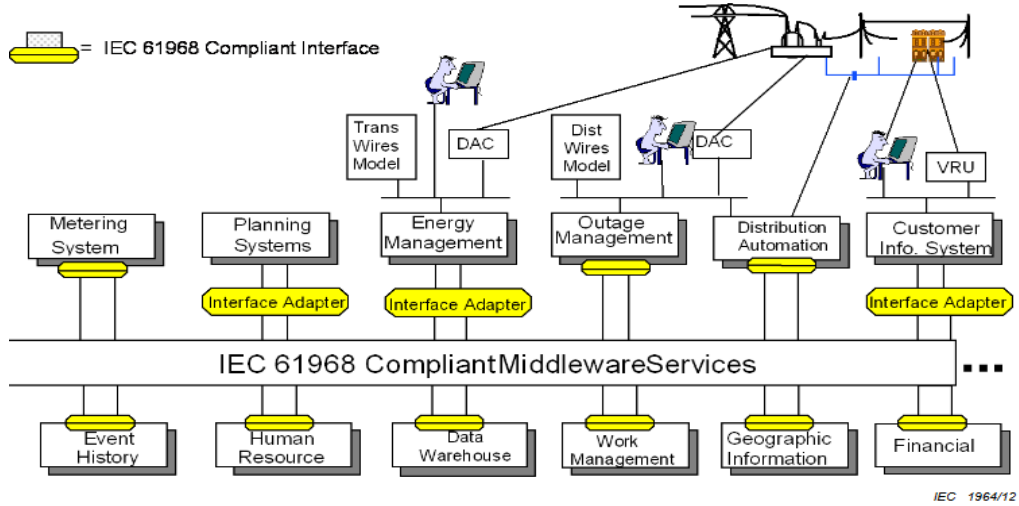


Figure 1 IEC 61968 deployment example [4]

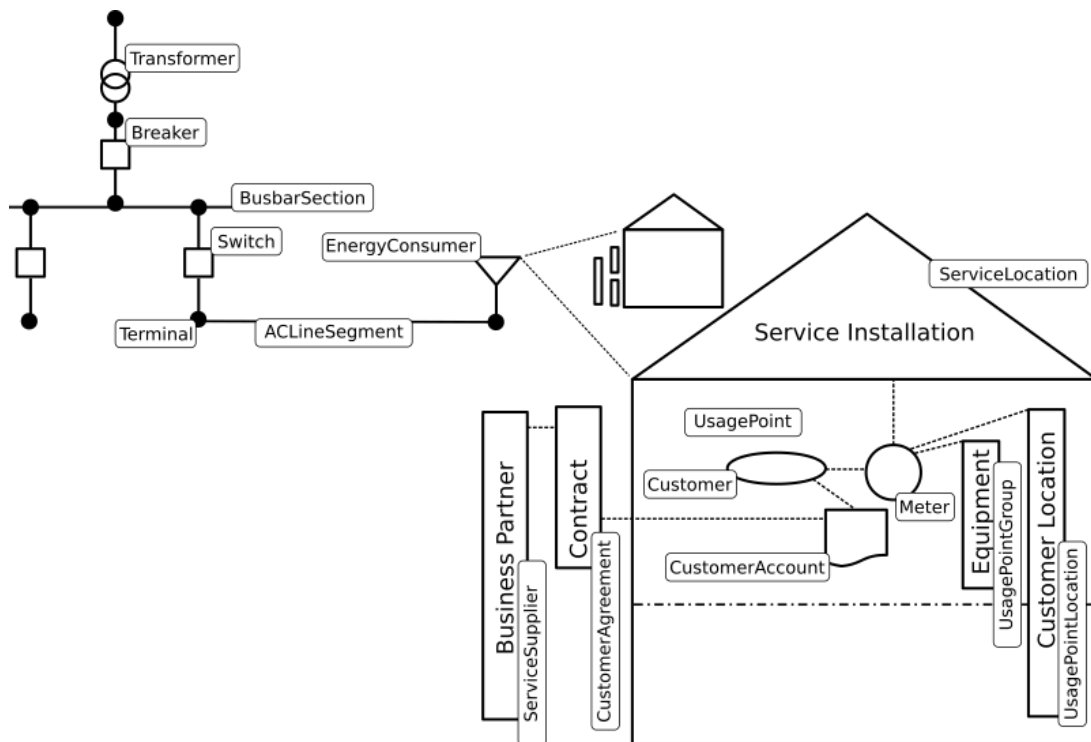


Figure 2 Sample Common Information Model classes [6]

Besides the integration aspect of integrating other systems or applications, the realization of the advanced functionalities that will be developed in the project requires access to a variety of data types and sources (from legacy or external systems). In order to mitigate problems related to data accessibility from individual systems and reduce the communication traffic to production infrastructure – since legacy systems are in operational state- the design employs a central data storage infrastructure. This infrastructure will be responsible for storing and providing access to

various project-related data - acting as an enterprise data “hub”- and facilitates their correlation as well as the meta-analysis of operation actions.

Finally, for facilitating the integration of security mechanisms in the network of the different services developed or integrated, a centralized authentication, authorization and accounting authority will be developed.

2.3. High Level Architecture

Figure 3 presents the architectural elements of RESOLVD platform and their interconnections, briefly explained below and presented in more detail in Table 1 and in the next chapters.

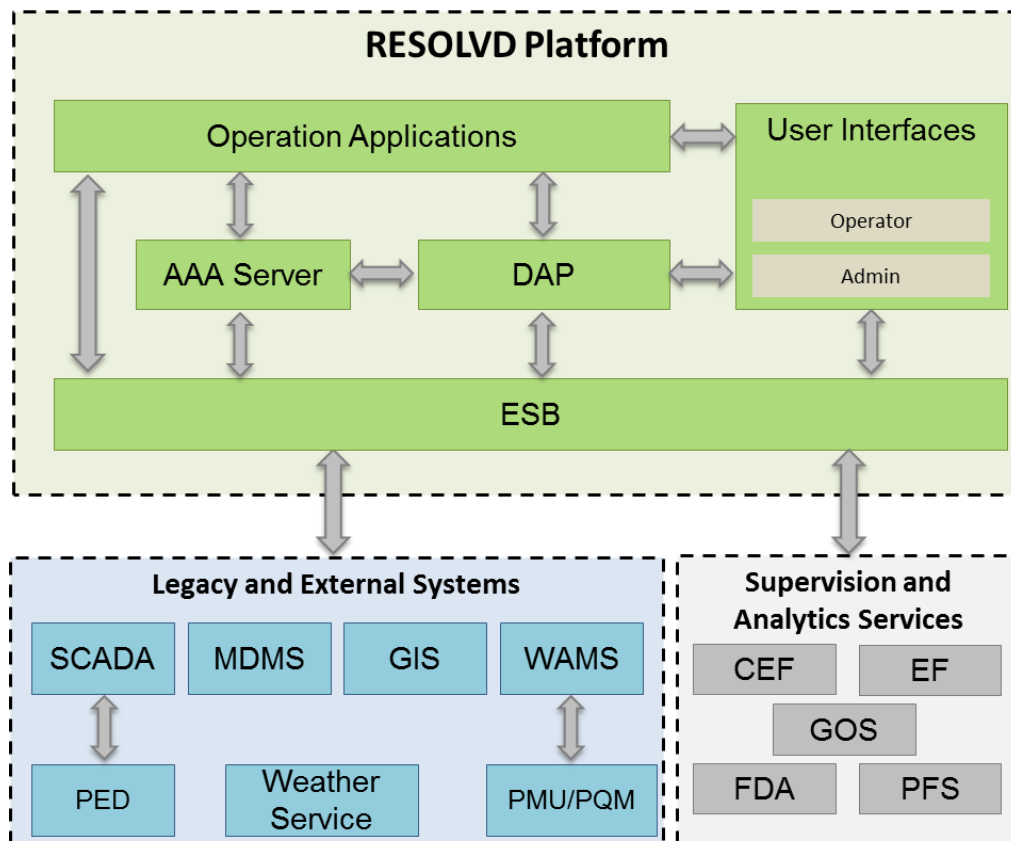


Figure 3 RESOLVD Platform

The RESOLVD platform is composed of the following architectural elements:

- The *Enterprise Service Bus (ESB)*, acting as an integration middleware;
- The *Data Analytics Platform (DAP)*, acting as a central data repository and data analysis and visualization provider;
- The *AAA Server*, serving as the security infrastructure;
- The *Operation Applications*, managing advanced grid operations and providing a UI.

Table 1 RESOLVD Platform Components

Component Name	Description
<i>Enterprise Service Bus</i>	The Enterprise Service Bus (ESB) is the means to connect service consumers with service providers, enabling them to route service calls, transform messages, mediate between communication protocols and data models and orchestrate

	<p>complex business processes. In the context of the project, the ESB will be used for interfacing and interaction among the various legacy system as well as applications and services that will be implemented in the project, through standard interfaces.</p> <p>The ESB will be responsible for the orchestration of the execution of the various modules, as well as the coordination of data exchanges with different systems (GIS, MDMS, DAP, WAMS etc.).</p>
<i>Data Analytics Platform</i>	<p>The DAP consists of the infrastructure that is responsible for data management, analysis and visualization.</p> <p>On one hand, it will integrate and homogenize data from third party applications and services via appropriate interfaces, which may differ among different data providers. It will act as a centralized storage of data from legacy systems and results of computation algorithms, allowing the decoupling of the legacy systems from the operation of the advanced functionalities offered by the operation applications.</p> <p>DAP will also be able to host the calculations of key performance indicators e.g. regarding the impact of the grid operations, being able to provide them as a service.</p> <p>Finally, DAP will provide visualizations, enabling end users to view both raw data stored on DAP as well as the insights gained from analytic computations.</p>
<i>AAA Server</i>	<p>The AAA Server will be utilized for the integration of security mechanisms in a network of services. This security infrastructure will provide authentication, authorization, and accounting and will safeguard the platform's resources/services.</p>
<i>Operation Applications</i>	<p>The Operation Applications aim to provide an intuitive user interface for the DSO, for performing advanced grid operations. The user (grid operator) will be able to manage different business processes related to the advanced grid operation, as well as visualize the results of each step e.g. problems detected, proposed solutions, impact of preventive actions performed.</p>

The platform interacts with external systems, integrated through the ESB:

- *Supervision and Analytics services*, that support Operation Applications, providing advanced functionalities as a service;
- *Supervisory Control and Data Acquisition system (SCADA)* for accessing grid monitoring and configuration information as well as accessing control operations;
- *Meter Data Management System (MDMS)* for accessing data from smart meter devices;
- *Geographic Information System (GIS)* for accessing grid topology;
- *Power Electronics Device (PED)*; this device performs the grid control actions and delivers data from its connection point, which is integrated through the SCADA;
- *Wide Area Monitoring System (WAMS)*, for accessing information of PMU and PQM devices and the results of fault detection algorithms;
- *Weather Service*, for accessing weather related information (history and forecasts).

3. The Enterprise Service Bus

In chapter, the design of the ESB is presented. The chapter is organized as follows:

- Initially, an overview of the ESB is presented based on the requirements and constraints documented in previous work of the project (D1.2 [1] and D1.3 [2]). Towards this, a high-level view of the ESB is also presented;
- The rationale of design is presented, documenting technologies, architectural patterns and standards that were followed in the design process, as well as the “off-the-shelf” hardware and software building blocks;
- An architectural description of the ESB is presented, detailing its internal components and providing different architectural views;
- Finally, the detailed design of the internal components of ESB is documented, presenting their roles, interfaces, internal logic and data models.

3.1. Overview

The ESB is a middleware that provides message transformation and content-based routing capabilities as well as other features that allow seamless integration among enterprise applications. Important benefits are the loose coupling of applications, the fact that it comprises a single point of communication, the need for minimal integration coding, its scalability, the support of standard interfaces and protocols, as well as the advanced monitoring capabilities. In contrary to more traditional Enterprise Application Integration (EAI) approaches of monolithic stacks (such as hub and spoke architecture), the ESB -based on the bus concept found in hardware architecture- enables a distributed deployment.

The ESB supports intelligent routing, data transformation, synchronous and asynchronous communications, complex event processing, service orchestration and business process automation, among others. Deliverable D1.2 [1], presents in detail the requirements related to the ESB. In accordance with these requirements, a top view of the system is presented in Figure 4 below as a UML component diagram. Table 2 presents a brief explanation of the interactions indicated in this figure.

The ESB design is constrained by the following:

- **Communication Security:** Several of the legacy systems to be integrated by the ESB are deployed in the control center or in the field, inside a Virtual Private Network (VPN) of the DSO. In order to achieve a higher level of security in communications and facilitate accounting, an adapter shall be installed inside the VPN. The adapter handles connections from and to the private network of the DSO, as a single point of interaction, provides the necessary data transformation for interactions with the legacy systems of the DSO as well functionalities related to service registration, enabling easier integration.
- **SG Standards compliance:** The ESB will follow the IEC 61968 specification (part of CIM) for data exchanges with the legacy systems of the DSO. The design is constrained by the relevant constraints of the specification;
- **Performance:** The services integrated through the ESB have different performance requirements, since some concern data exchanges, whilst other complex business flows. The aim of the design it to be able to comply with these requirement through a scalable infrastructure.
- **Data Privacy:** Sensitive data (regarding grid operation) will be transmitted through, thus data confidentiality must be ensured.

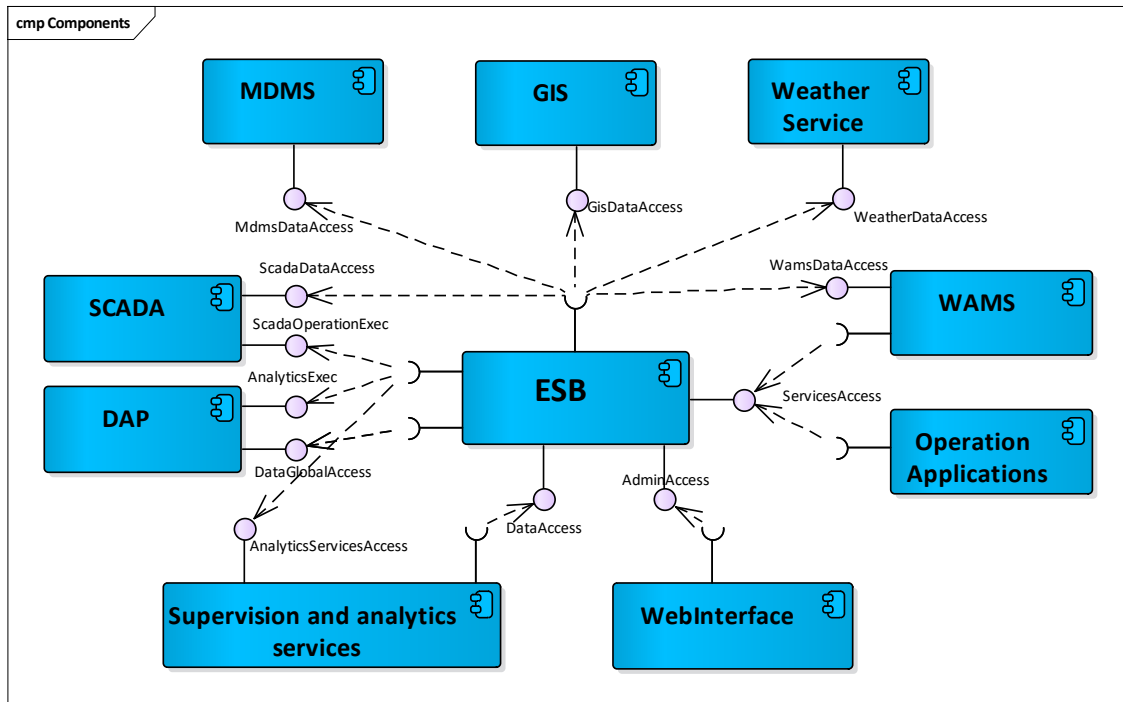


Figure 4 ESB high-level overview

Table 2 ESB Interfaces

Type	Interface Name	Description
Required	MmdsDataAccess	Through this interface, the ESB has access to MDMS data, including generation and consumption measurements reported by smart meters.
	GisDataAccess	This interface provides access to grid topology data from the GIS.
	WeatherDataAccess	Through this interface, weather forecasts and historical weather data are provided.
	ScadaDataAccess	Provides access to the ESB for data stored in the SCADA (i.e. grid configuration, PED status).
	ScadaOperationExec	Provides access to operations exposed by the SCADA
	WamsDataAccess	Provides access to the ESB for data stored in the WAMS, regarding PMU/PQM devices measurements.
	DataGlobalAccess	This interface of DAP, enables the ESB to perform data access, manipulation and storage operations to the central data repository.
	AnalyticsExec	This interface provides the ESB with the ability to trigger the execution of analytics services provided by DAP.
	AnalyticsServiceAccess	This interface provides the ability to access and integrate the services related to analytics and supervision.
Provided	ServicesAccess	Interface for exposing individual or aggregated services integrated through the ESB.
	Data Access	Exposes the data-messaging interface (CIM).
	AdminService	This interface provides a management operations for the administrator of the ESB.

3.2. Design Principles

The ESB acts as a standards-based distributed integration platform, combining various types of messaging paradigms, transformation, routing and web services. The initial intention is to support REST (REpresentational State Transfer) as well as implementation of the Remote Procedure Call (RPC) i.e. Simple Object Access Protocol (SOAP).

REST is an architectural style that defines a set of constraints to be used for creating web services commonly used in the development of Web services. Web Services that conform to the REST architectural style, (RESTful web services) allow the requestor to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations. This allows a decoupled architecture and lightweight communications between communicating parties.

SOAP is a messaging protocol specification allowing different systems to communicate, typically using Hypertext Transfer Protocol (HTTP) and the Extensible Markup Language (XML). Commonly regarded as the protocol of the choice for inter-application and business-to-business communications, many ESBs in the past were based on combination of SOAP/XML for information exchange.

Aiming to achieve interoperability both with legacy systems as well as the future services implementation, both technologies shall be supported. Toward this scope, functionalities of legacy applications will be “wrapped” as services if necessary. As already mentioned, the ESB will follow the specification described in IEC 61968. Each flow related to data exchange from/to the ESB with the legacy systems shall be in a CIM compliant data format. In the occasion where the system communicating with the ESB does not support the format, an adapter will be responsible for translating it.

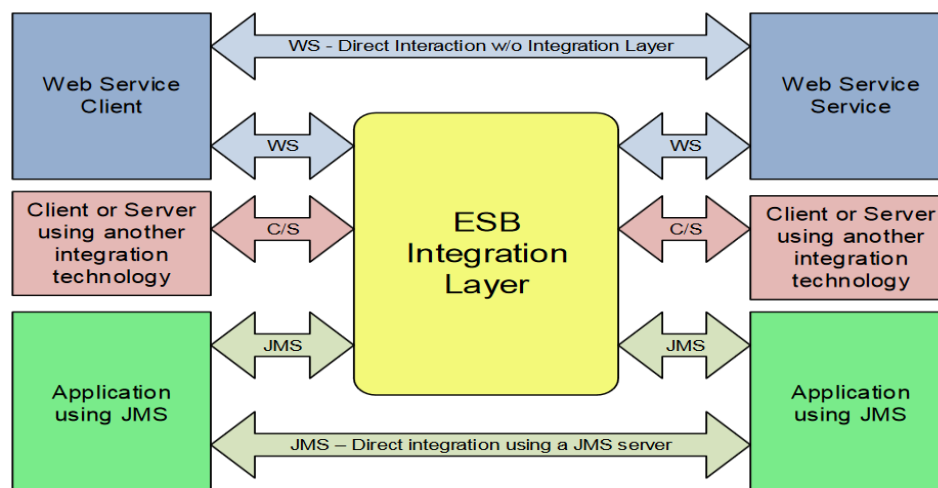


Figure 5 Scope of IEC 61968-100 [7]

Furthermore, “Part 100: Implementation profiles” [7] of the standard’s series, specifies an implementation profile for the application of the other parts of IEC 61968 using common integration technologies, including Java Message Service (JMS) and web services. This standard also provides guidance with respect to the use of Enterprise Service Bus (ESB) technologies. As presented in Figure 5, the integration layer, enables different technologies between receiver and transmitter, as well as provides support for one-to-many information exchanges (publish/subscribe integration patterns) and key functionality such as delivery guarantees.

Java Message Service is an API for message-oriented middleware concerning message

exchange between two or more clients. JMS supports request/reply, publish/subscribe and point to point messaging patterns. This messaging standard allows application components based on Java to create, send, receive, and read messages, allowing loosely coupled, reliable and secure communication. JMS is typically used within a secure, private enterprise network. In cases where this is not true, JMS can be configured to use SSL/TLS and/or use client authentication.

Another important decision related to the ESB design concerns the definition of how much of a transaction is orchestrated by the middleware, and what is required from the participating systems (or services). If an endpoint system handles the orchestration, the middleware's role is limited to that of a message broker.

Following the SOA pattern described above, the endpoint systems will be considered offering the minimal set of functionalities as services, whilst the ESB will handle integration in two different tiers: one concerns the mediation of messages towards integration of different services and the other concerns the orchestration of multiples services as a single aggregated service, following a business process flow.

In terms of using "off-the-shelf" building blocks of the ESB, our approach is that, instead of a full-blow open source solution such as Apache ServiceMix or MuleESB, a lightweight approach will be followed, with the utilization of Apache Camel [8] and add-on components.

More specifically the ESB will be composed of:

- **Apache Camel**, at the core of the ESB, as a message-oriented middleware, acting as a routing and mediation engine. Apache Camel is an open source framework for message-oriented middleware with a rule-based routing and mediation engine that provides a Java object-based implementation of the Enterprise Integration Patterns using an application programming interface to configure routing and mediation rules. The available documentation [9] on the most common enterprise integration patterns, as well as support of open-source frameworks facilitates their use for designing integration solutions. Apache Camel will be the main building block of the ESB solution designed. In Figure 6, the architecture of Apache Camel is presented. Camel utilizes different endpoints to send and receive messages with camel components, which act as the connectors with all other systems. The variety of components supported by Camel makes it interoperable with different type of technologies (DBs, transport protocols, data format etc.). At its core, Camel processors connect the different endpoints, providing message routing, manipulation (transformation, enrichment etc.) and monitoring abilities. Apache Camel uses URIs so that it can easily work directly with any kind of transport or messaging model such as HTTP, ActiveMQ, JMS etc. Adapters can be used for different kinds of communication, supporting a variety of protocols and data formats. It also employs a domain specific language (DSL) for the definition of complex routing rules and business processes, while being agnostic on the data that is processed.
- **Apache ActiveMQ** [10] as a message broker, acting as a transfer agent. ActiveMQ is an open source message broker written in Java together with a full Java Message Service (JMS) client. As a messaging component, it can provide the ability to communicate asynchronously with a range of other software and it is easily integrated with Apache Camel (through the dedicated component), towards acting an interface for message exchanges. ActiveMQ implements the Advanced Message Queueing Protocol (AMQP), an open standard application layer that allows messaging. AMQP is identical to the use of JMS, using queues in cases where the clients use the JMS API, making its use transparent from the perspective of the client-side.
- **jBPM** [11] as a business process engine, for business process management and

execution. jBPM (Java Business Process Model) is an open-source workflow engine, developed by Red Hat Software, that can execute business processes described in Business Process Model and Notation (BPMN) 2.0 [12], able to be integrated with Apache Camel. This component supports the definition processes that drives the business processes (orchestrations), providing a flexible and easy to use tool for the creation and management of such processes.

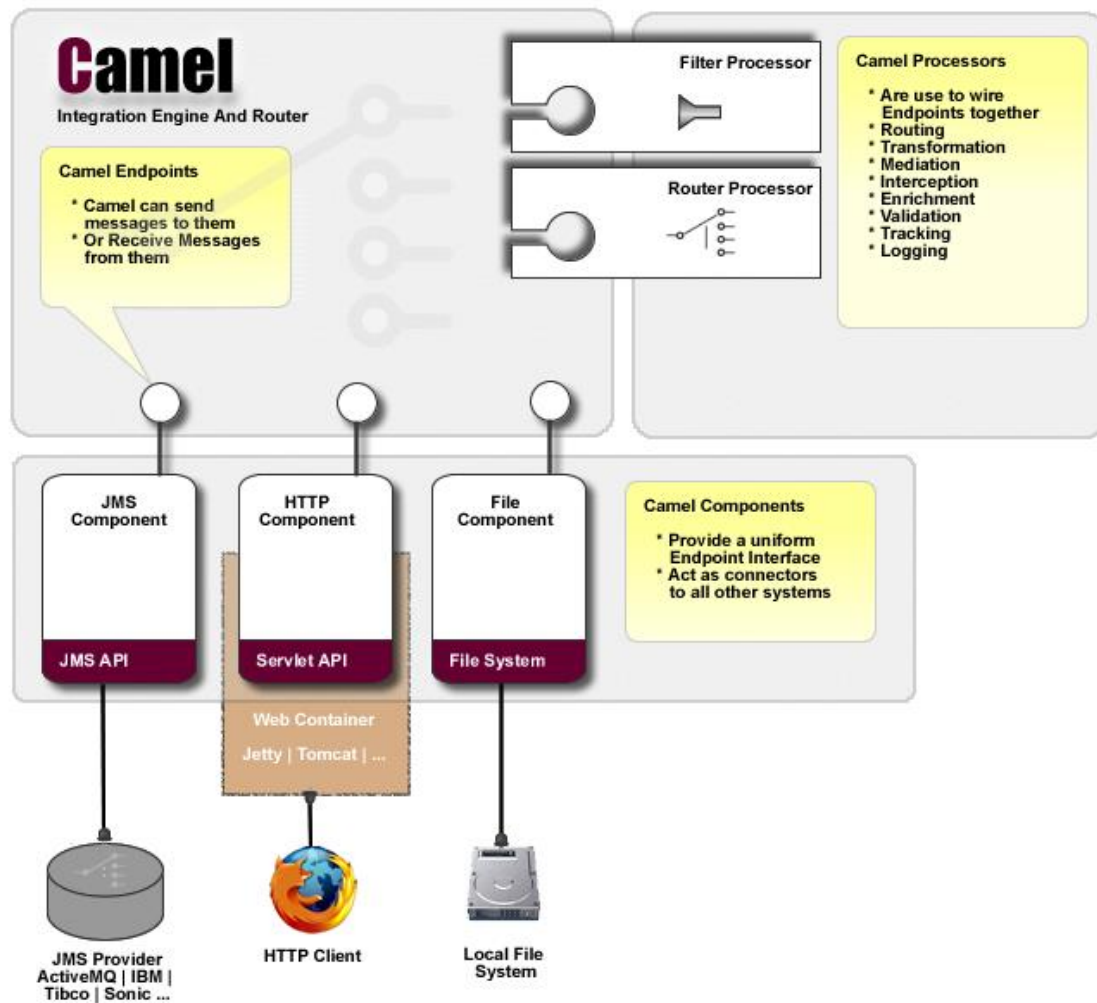


Figure 6 Architecture of Apache Camel [8]

The system will be deployed in a server with either Linux or Microsoft Windows Operating Systems. Both relational (MySQL) and non-relational (NoSQL) data databases shall be supported, with the use of appropriate connectors/adapters.

3.3. Architecture

3.3.1. Logical View

Figure 7 illustrates the internal components of the ESB, which are briefly explained in Table 3 and further analysed in the following sections.

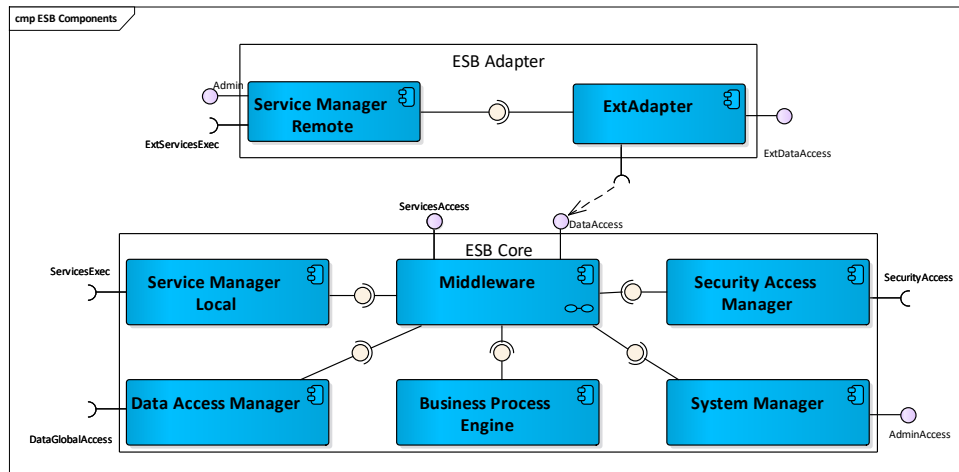


Figure 7 ESB Component diagram

Table 3 ESB's main components

Component Name	Description
<i>Middleware</i>	This exposes the interfaces for integrating data exchanges and composite functions. It provides the following core functionalities: <ul style="list-style-type: none"> • Transport protocol conversion; • Message transformation; • Message routing; • Message enhancement.
<i>Business Process Engine (BPE)</i>	Provides service choreography implementing the business flow part of Operation Applications (see chapter 0).
<i>ExtAdapter</i>	Handles communications related to data access, with the legacy systems of the DSO and external systems in general; Implements a messaging client so that these systems can connect to the Middleware's messaging interface.
<i>Data Access Manager</i>	Provides an interface for accessing and manipulating data.
<i>Service Manager Local</i>	Responsible for keeping the service registry and for integrating external services.
<i>Service Manager Remote</i>	Responsible for keeping the service registry at the remote location and integrating external services.
<i>System Manager</i>	Manages operations related to the configuration, monitoring and administration of the system.
<i>Security Access Manager</i>	Handles security integration with AAA server.

3.3.2.Process View

The ESB mediates messages through provision of communication endpoints, routes information and supports multiple transport protocols and transformation abilities. In Figure 8, the activity diagram of the message handling process is presented. In the initial step, the message is received and is directed to the message router. A routing service will forward to the right output destination of the message, whilst a translation processor will undertake the task of converting among the different data models and data formats supported by the sender/receiver of the message, and in many occasions to standardized data formats.

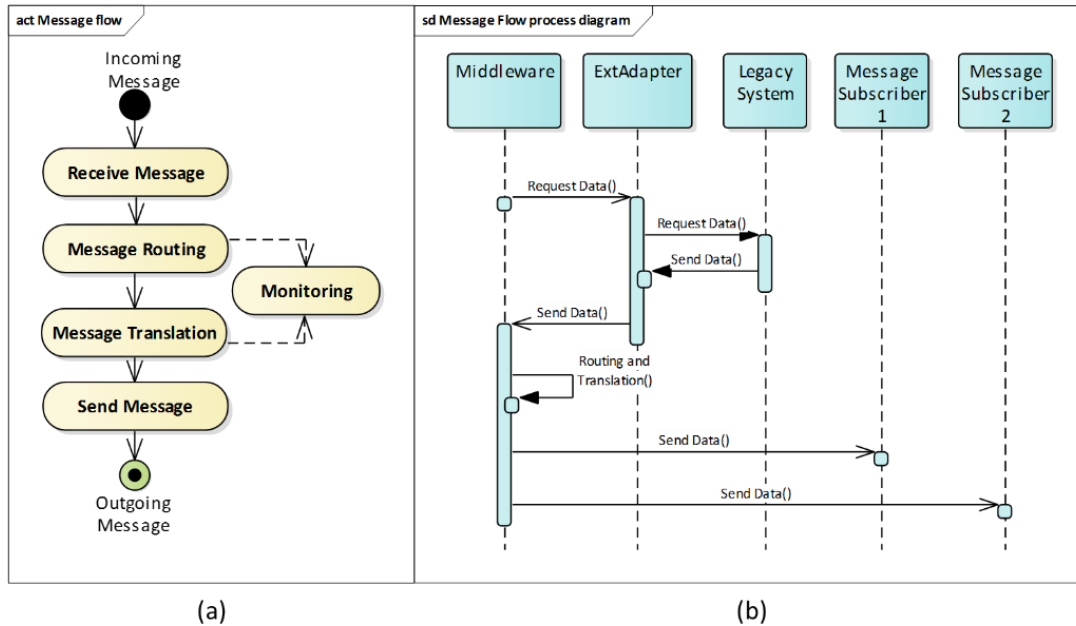


Figure 8 Message flow activity diagram (a) and process diagram (b)

Moreover, the ESB has the role of the orchestrator of business processes, handling the routing of a message through a set of services to produce the desired result. Orchestrating of such operations means that when a client of a service (e.g. the operator of an application or an internal time-trigger of an application) triggers the business process, the ESB is responsible for delivering the end-result. The ESB must invoke the right services and provide the result to the original requestor. Figure 9 presents the sequence diagram of the main flow of an application orchestration process.

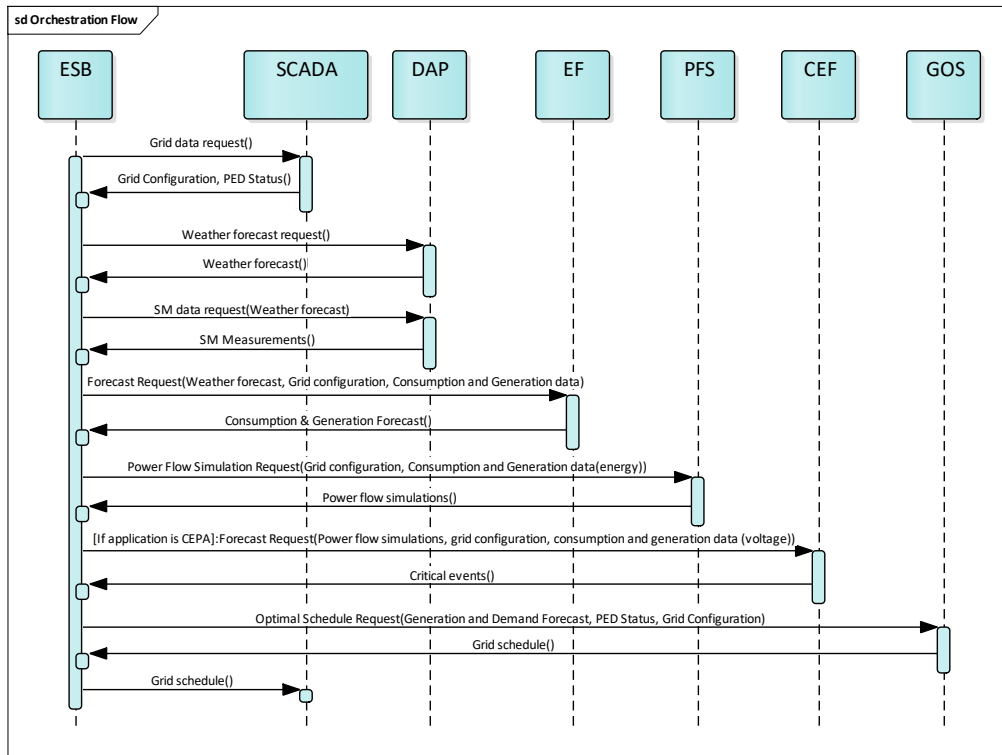


Figure 9 Sequence diagram of orchestration

In Figure 10, an execution flow of the BPE is instantiated (using BPMN notation), depicting how the external requestor triggers a process, how the ESB picks up the message and starts the BPM process and finally how the flow is executed through a series of actions (service invocation etc.) and the result is returned to the original requestor.

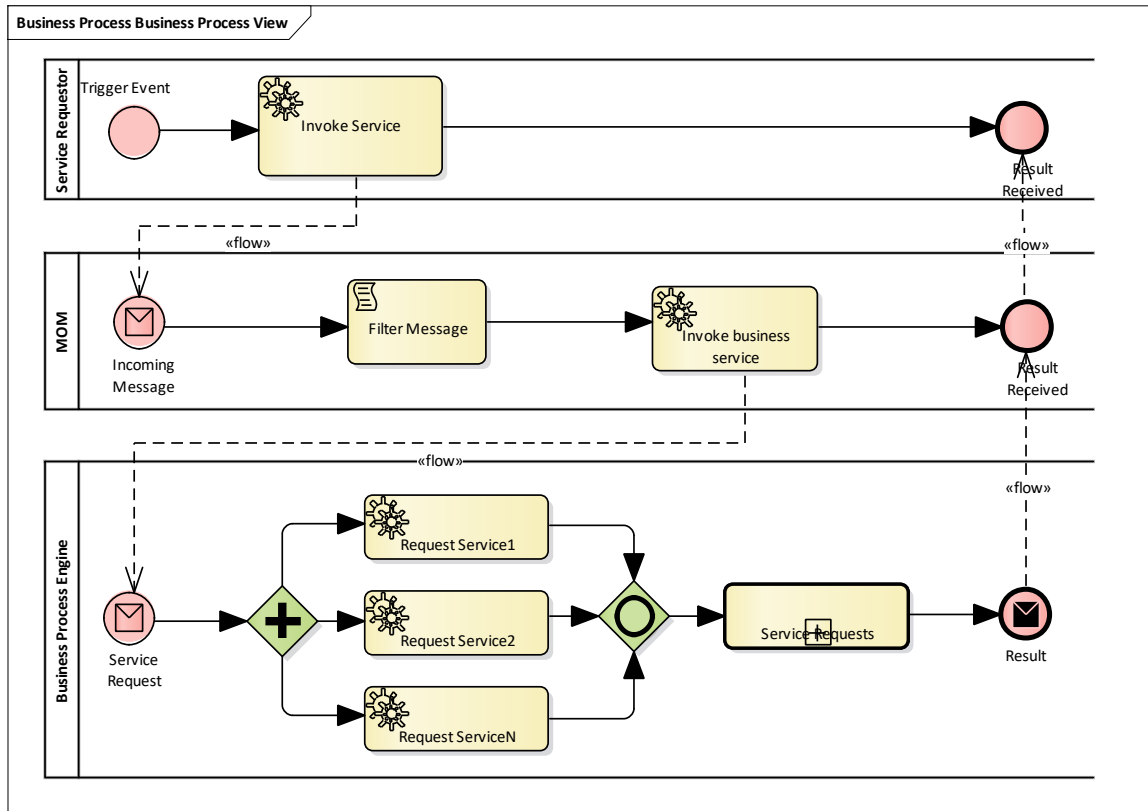


Figure 10 Business process engine execution flow

3.3.3. Physical View

Figure 11 shows the ESB decomposition in terms of distributed subsystems with a possible instantiation for the scope of the project. The subsystems are the following:

- **ESB Adapter:** Responsible for transporting and optionally translating the message, as well as for managing service integration at remote locations such as keeping the services registry and integrating external services. Security measures for this component will be established on both network (e.g. restricted access) and application level (e.g. certificate).
- **ESB Core:** Providing the core functions of the middleware related to message mediation, business process management and service integration, as well as security management (through an external AAA server) and system monitoring.

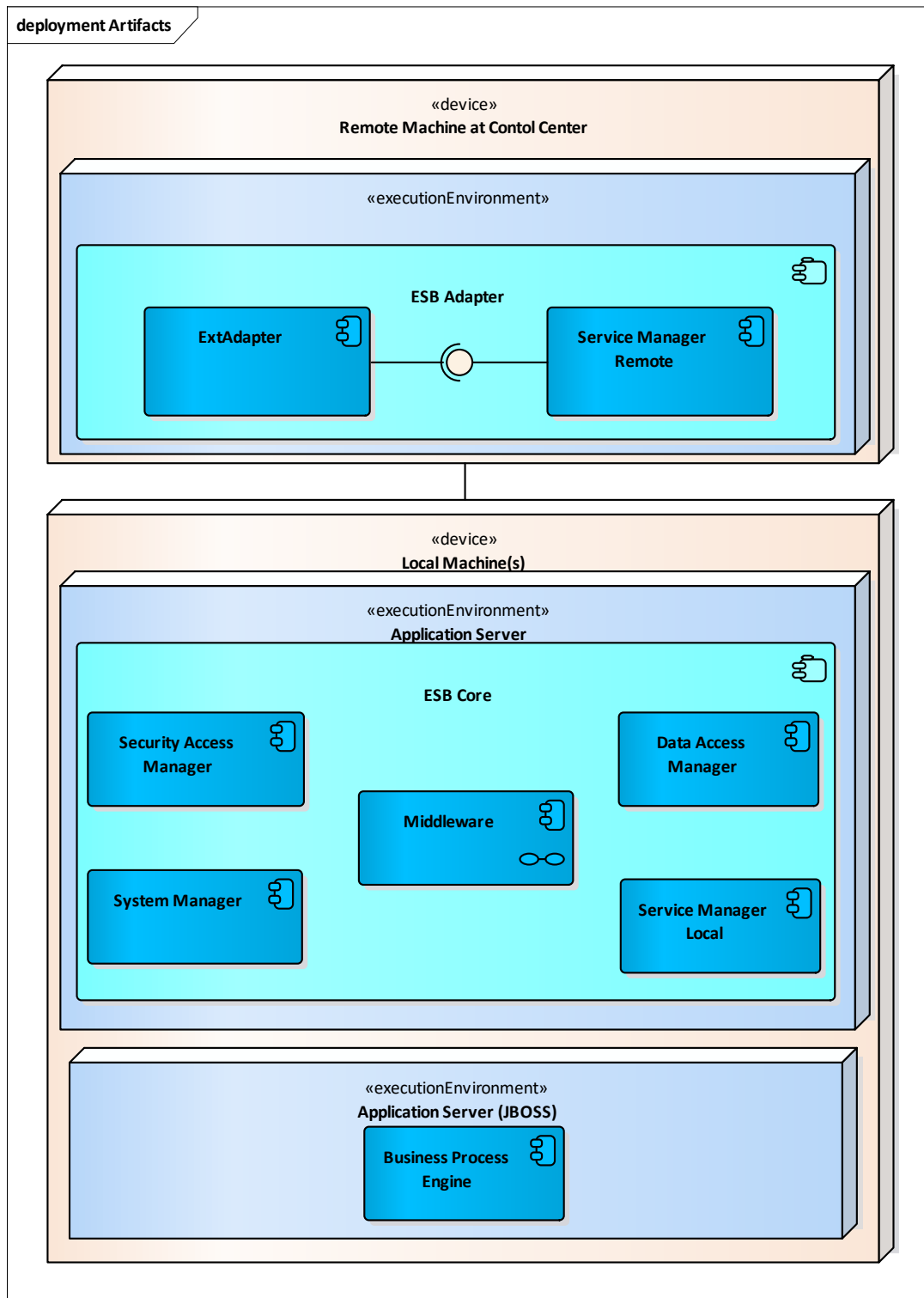


Figure 11 ESB Deployment diagram

3.4. Components Design

This section provides the low-level design for each of the components of the ESB identified in the previous section.

3.4.1. Middleware

3.4.1.1. Description

A Message Oriented Middleware (MOM) consists of the infrastructure that supports the exchange of messages between the different sub-systems of the ESB, as well as external, enabling their seamless integration. When using a MOM, a service requestor calls the system to send a message to a destination managed by the provider, which invokes provider services to route and deliver the message. In the case of asynchronous communication, the client can continue its work, until the response is retrieved. This enables loosely coupled components, which are easier to maintain and eliminates any problems related to intermittent connectivity in the network. The Middleware component acts as a MOM and provides persistent storage for backing up the message queue.

The routing logic of the MOM can expand from content-based routers, where the content of a message is inspected in order to define the routes, to dynamic-router where routing logic is modified through control signals (see EAI patterns [9]). The message route could exist in messaging layer itself or be provided by the service requestor.

The MOM can transform messages en-route to match the requirements of the sender or of the recipient. In the context of RESOLVD, messages concerning the integration of data from external systems will comply with the CIM - with the use of adapters where necessary.

3.4.1.2. Internal Logic

The component design is based on the following artefacts:

- camel-core – the basic module of apache camel.
- camel-jms / activemq-camel (or other mq library supporting JMS) as JMS component.

Table 4 presents the main modules of the component.

Table 4 Modules of Middleware component

Component Name	Description
<i>MOM</i>	This module provides a rule-based routing and mediation engine offering the core functionalities of the Middleware.
<i>Message Broker</i>	A building block of the Middleware is the message transfer agent (message broker). It acts as the connection of the Middleware with the external world and provides a solid implementation of messaging interface.
<i>Adapter</i>	This module is responsible for communicating with the back-end and transforming data from a certain format to the canonical format. The adapter can also handle an activity flow related to security (e.g. authentication, decryption, verify or sign document) and handle error cases.

3.4.1.3. Interfaces

Provided interfaces are presented in §3.6.

3.4.2. Business Process Engine

3.4.2.1. Description

An orchestration consists of a series of ordered operations or transactions that implement a business process. The role of the business process engine is to be able to interpret business process into a series of service calls using simple conditional logic and data aggregation techniques, in order to complete a business flow.

3.4.2.2. Internal Logic

The design of BPE follows a modular approach, utilizing the following artefacts:

- jBPM libraries, as a business process engine;
- camel-jbpm (Camel add-on), providing integration with jBPM;
- custom modules, enabling interaction among them and with the Middleware.

In Figure 12 a structure of the BPE is illustrated, adapted from [13], depicting its main modules and their interrelations.

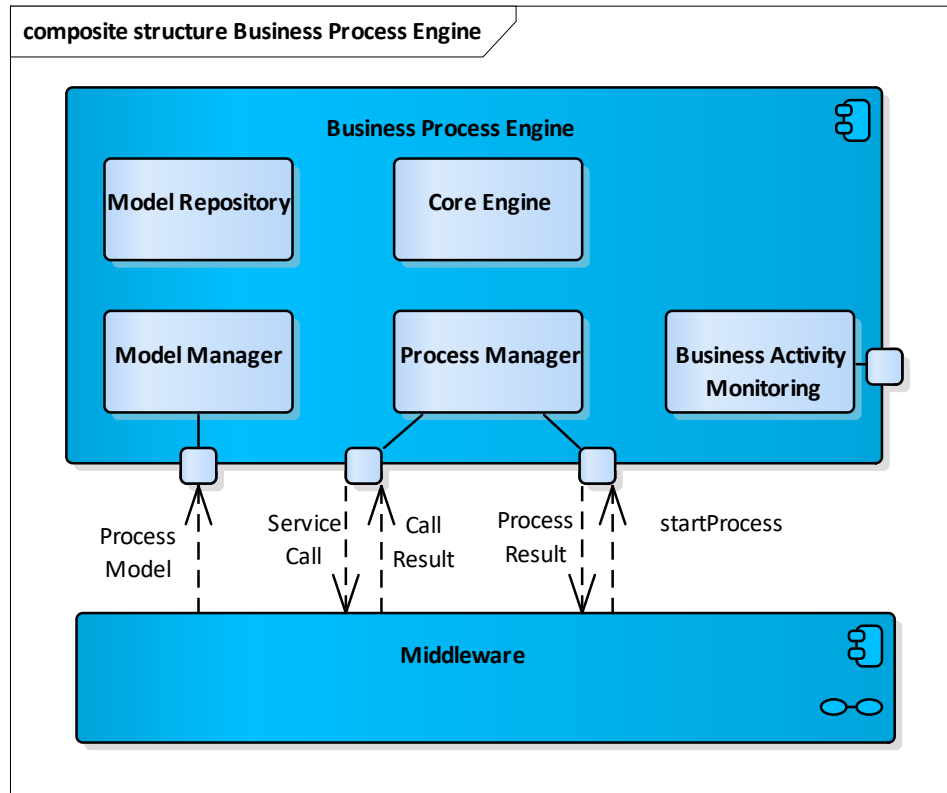


Figure 12 Business process engine composite structure

The following table provides a brief description of the main modules of the BPE component.

Table 5 Modules of BPE component

Module Name	Description
<i>Core Engine</i>	This module is responsible for handling the steps of the business process. It is deployed as a service and can be integrated through Java API or CDI services, as well remotely through a REST and JMS API.
<i>Business Activity Monitoring</i>	Keeps track of the business processes and provides monitoring information of the status of the business process.
<i>Process Manager</i>	Handles events (both internal and external), either by invoking business processes in the Core Engine or by invoking services in the Middleware and providing the result to the Core Engine.
<i>Model Manager</i>	Provides an interface for storing the process models in the model repository.
<i>Model Repository</i>	Responsible for storing the models (in XML format) of business processes and the history of business processes' invocation.

3.4.2.3. Interfaces

The functions provided through the interface of the BPE, are presented in Table 6.

Table 6 Business Process Engine interface functions

Method	Parameters	Response	Description
<i>startProcess</i>	Process id (Parameters)	Process Instance	Start a new process instance. Returns an object representing the instance of the process that was initiated.
<i>abortProcess</i>	Process Instance ID	Success Code	Aborts the process instance with the given id. If the process instance has been completed or aborted, or cannot be found, it returns an error.
<i>signalEvent</i>	Event Type (Process Instance ID)	-	Signals the engine that an event has occurred. The type parameter defines which type of event and the event parameter can contain additional information related to the event.
<i>getProcess</i>	Process Instance ID	Process Instance	Returns the process instance with the given id.
<i>storeModel</i>	XML file	Process Id	Stores a model of a process as XML file. Returns the process id upon success, or an error code.

3.4.3.ExtAdapter

3.4.3.1. Description

The ExtAdapter is a messaging client that handles communications with legacy systems of the DSO and external systems in general, that do not support messaging or the canonical data model format (CIM). It provides the ability of both push/pull mechanism in the communications with such systems, whilst gathered information will be provided to the message queue (or relevant topic) of a message communication thread. In the case of pull mechanism, the component will be able to communicate with the Service Manager Remote component, in order to have access to the location of the services. The pull mechanism could be triggered either periodically, through an internal process, or by an external request of the Middleware.

3.4.3.2. Internal Logic

The following table provides a brief description of the main modules of the ExtAdapter component.

Table 7 Modules of ExtAdapter component

Module Name	Description
<i>Mediator</i>	Responsible for providing a web interface for receiving data from external systems (push), as well as invoking services for accessing data (pull) upon trigger.
<i>CimTransform</i>	This module can transform data from external systems to CIM format, using a predefined mapping of ontologies and attributes.
<i>Messaging Client</i>	Undertakes the role of the JMS client for exchanging messages with the Middleware.
<i>Scheduler</i>	This module triggers the process of accessing data from external systems (in pull mechanism).
<i>Data Manager</i>	Responsible for communicating with a local database and storing the message queue.

3.4.3.3. Interfaces

The ExtAdapter will provide a web interface for exchanging data with external systems, as well as a messaging interface (compliant with IEC 61968-100 standard) for communicating with the

Middleware.

The integration between the ExtAdapter and non-compliant interfaces can be based on a variety of integration mechanisms, which may include (but not limited to):

- Web services
- HTTP
- Java Database Connectivity (JDBC)
- File Transfer Protocol (FTP)

3.4.4. Data Access Manager

3.4.4.1. Description

Data Access Manager component is in charge of storing and retrieving data from an external data repository (i.e. DAP), whilst all data provided for storage or retrieved comply with the predefined data model. The component represents the service layer that manages data provided through:

- “Bulk” data interface (DataAccess) between legacy systems and external services;
- Services interface (ServiceAccess);
- Middleware related meta-data (logs, events etc.); and
- Data requested by external applications and services.

3.4.4.2. Internal Logic

The component design is based on the following Apache Camel add-ons:

- custom modules for accessing the short-term store through a web API;
- camel-jdbc, for accessing management databases (through JDBC);
- camel-hdfs2, providing the functionalities for reading and writing messages from/to the bulk-store (HDFS file systems).

The main modules of the component are presented in Table 8, below.

Table 8 Modules of Data Access Manager component

Module Name	Description
<i>Message Database Connector</i>	Manages connections to the short-term store, which keeps all the messages currently processed by the middleware. Messages contain both data and context information (related to routing).
<i>Management Database Connector</i>	Manages connections to the data store for meta-information.
<i>Bulk-Storage Connector</i>	Manages connections to the database that keeps records of all events taking place and acts as the long-term storage for all messages.

3.4.4.3. Interfaces

The *Data Access Manager* will support interface for data manipulation both for relational (SQL queries) as well as for non-relational database (NoSQL queries), through proper connectors.

3.4.5. Service Manager

3.4.5.1. Description

The *Service Manager* component keeps the registry of services. Information such as Service UID, description, URL, class (group), version, call method, exceptions, security (e.g. authorisation), permissions (i.e. roles of AAA server - §5) are managed by this component. The component is

also responsible for integrating external applications, by invoking external services.

The *Service Manager Remote* component has the same functionality with the *Service Manager Local* Component. Their differentiation concerns the deployment location.

3.4.5.2. Interfaces

The *Service Manager* interface provides the methods presented in Table 9.

Table 9 *Services Manager Local* interface

Method	Parameters	Response	Description
<i>addService</i>	<i>Service Description</i>	Service ID	Add a new service in the registry providing any relevant parameters.
<i>updateService</i>	Service ID <i>Service Description</i>	Success Code	Updates an existing service.
<i>deleteService</i>	Service ID	Success Code	Deletes a service from the registry.
<i>getServices</i>	-	List of Service objects	Returns the list of services.
<i>invokeService</i>	Service ID Call Parameters (Session Ticket)	String	Invokes a service from the one registered in the registry.

3.4.6. Security Access Manager

3.4.6.1. Description

This component handles the integration of the ESB and the AAA server, providing an interface for authentication, authorization and service registry synchronization.

3.4.6.2. Interfaces

Table 10 presents the *Security Access Manager* interface's methods.

Table 10 *Security Access Manager* interface

Method	Parameters	Response	Description
<i>authenticate</i>	Username Password	Session ticket	Exposes the method of authentication via the AAA server.
<i>authorize</i>	Session ticket, Service ID	Success Code	Exposes the method of authorization via the AAA server. Returns an error code if the authorization fails.
<i>synchronize</i>	List of <Service Id, Group Id, Role Id>	Success Code	Provides a batch synchronization of the service registry between AAA server and Service Manager.

3.4.7. System Manager

3.4.7.1. Description

The *System Manager* component is responsible for managing operations related to the configuration, monitoring and administration of the system. The component will provide a UI for performing the relevant operations (see §3.5).

3.4.7.2. Internal Logic

The component design is based on the following Apache Camel add-ons:

- camel-core, providing the *Control Bus* enterprise integration pattern for logging and monitoring the operation of the ESB;
- jBPM libraries for monitoring of the business process;
- custom modules for integration and UI functionalities.

3.4.7.3. Interfaces

Provided interfaces are presented in §3.6.

3.5. Graphical User Interface Design

The ESB will provide a user interface environment for managing and parameterising its functionalities and monitoring its operation. The main aspects of the design approach for the user interface of the ESB are as follows:

- Login page: The user will use a login page for being authenticated, defending any unauthorized access to the system. By inserting the username and the password and successfully logging in, one the user will have to access the overall system functionalities based on the roles assigned to the user account².
- Managing Services: The services' management administrator will have the ability to define service, by giving all the necessary information (name, description, version, URL etc.). The service definition will be stored in the services repository. The user will be prompted to select the class (group) of service described and edit the permissions for accessing the service. The service by default will have the permission inherited by its class (group). The user will also have the ability to manage the routing rules through this interface.
- Business Process Monitoring: Any business process has a predefined set of steps in order to be completed. The user will have the ability to monitor the status of the operation and the session variables as well. In case the process is in waiting mode, requiring user input, the administrator will be able to update the variables in order to move to the next step of the process. The administrator will also have the ability to see the list of current or past processes, providing information on their status (pending, completed, active, aborted or suspended), execution time, duration and relevant key performance indicators. Finally, the administrator will have the ability to pause or stop the execution of a process.
- Managing Business Process Models: The administrator will have the ability to create, update or delete a business process. All created business process model, will be stored in the model repository and can be instantiated using its unique identifier.
- Managing Communications: A web based administration tool will enable the parameterization and monitoring of the message broker.

3.6. External Interfaces Design

The following external interfaces will be provided by the ESB:

- *Data Access*: Supports messaging communications (IEC 61968-100 standard);
- *Services Access*: Exposes the services (i.e. orchestration services) as a web service;
- *Admin*: Exposes the management of services and business processes as a web service.

² Additional measures like two-factor authentication will be specified in deliverable D4.5.

There are several integration patterns described in IEC 61968-100 [7] for implementing communications. The most basic ones, concern:

- Synchronous request/reply, through web services or JMS queues;
- Asynchronous request/reply, through web services (using callback) or JMS queues;
- Publish/subscribe, through web services (using callback) or JMS topics³.

Services Access and *Admin* interfaces' provided methods are explained in Table 11 and Table 12.

Table 11 Services Access interface

Method	Parameters	Response	Description
invokeService	Service ID Call Parameter (Session Ticket)	Service Response	Calls a service with the provided parameter and returns the response.
getServiceInfo	Service ID (Session Ticket)	Service Description	Provides the description of a service.
getServiceStatus	Service ID (Session Ticket)	Service Status	Provides the status of a service.

Table 12 Admin interface

Method	Parameters	Response	Description
registerService	Service Description (Session Ticket)	Service ID	Add a new service in the registry providing any relevant parameters.
updateService	Service ID Service Description (Session Ticket)	Success Code	Updates the details of an existing service in the registry.
deleteService	Service ID (Session Ticket)	Success Code	Deletes a service from the registry.
getServiceList	(Session Ticket)	List of Service objects	Returns the list of services.
addRoute	XML file (Session Ticket)	Success Code	Creates a specific routing design from/to an endpoint(s) and returns the route id or an error code.
deleteRoute	XML file (Session Ticket)	Success Code	Deletes a specific route.
startRoute	route ID (Session Ticket)	Success Code	Initiates a specific route.
stopRoute	route ID (Session Ticket)	Success Code	Stops a specific route.
getRouteStatus	route ID (Session Ticket)	Route Status	Provides the status of a specific route.
addBusinessProcess	XML file	Business Process ID	Save a business process in the repository.
deleteBusinessProcess	Business Process ID	Success Code	Deletes a business process from the repository.

³ Topics are used when the destination of a message is potentially more than one process, whilst Queues are used when the destination of a message is at most one process.

getBusinessProcessList	-	List Business Process Descriptions	Returns the list of the business processes.
getBusinessProcessStatus	Business Process ID	Business Process Status	Return the status of the business process e.g. execution state, current step of workflow, execution start time, duration.

3.7. Data design

3.7.1. Canonical Data Model

Data provided through the DataAccess interface must be compliant with the information model IEC61968 (referred here as CIM). Adapters which convey with the messaging standard defined in IEC 61968-100, are responsible for transforming the data to proper data format if necessary, whilst the actual message is transferred inside the payload of the message. Figure 13 presents the top structure (message envelope) of CIM messages, which is further explained below.

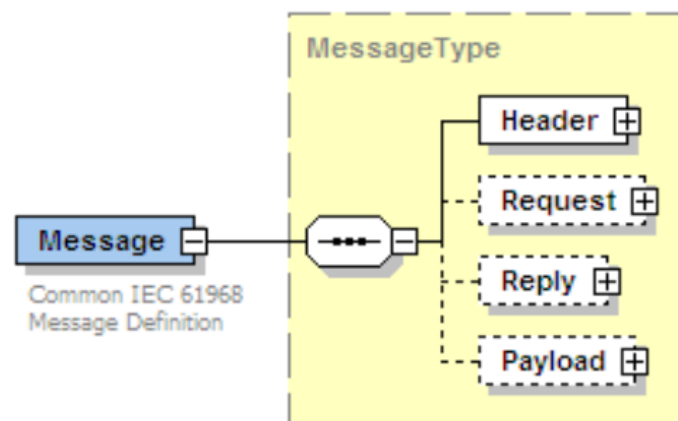


Figure 13 IEC61968 Message Structure [14]

- Header holds meta-data about the message:
 - Noun: Identifies the CIM profile being exchanged. Verb: Identifies the action to be taken (e.g. *create, change, cancel*).
 - Message Id: Unique identifier of the message.
 - Correlation Id: Unique identifier used to map related messages.
 - Reply Address: Address to specify URL for asynchronous replies.
 - Revision: Version of standard (IEC61968) used.
- Request is used to convey request parameters for request messages e.g. Start and End Time of interest (time-based query), type of request, among other.
- Reply captures result/error state for response messages.
- Payload holds the CIM information object being exchanged. It will correspond to the CIM profile identified in the header.

3.7.2.External Data

A description of data structures integrated through the ESB is presented in this section, detailing the source and format of the data, as well as a small description. For business process information, the metadata field provides a relation of the data with the information provided in the original request.

Table 13 presents the structure of the data provided by the MDMS, which concern smart meter measurements. The mapping to the CIM format can be done using the MeterReadings schema defined by IEC 61968-9.

Table 13 Smart meter data

Source description
Metered electricity consumption by commercial and industrial customers. Provided by the MDMS.
Data format
CSV or XLSX file with header row, containing the following information: <ul style="list-style-type: none"> • Meter identifier • Secondary Substation name • Time of reading as yyyy-MM-dd. HH:mm:ss • Type of measurement (e.g.Hourly) • Season as summer (E) or winter (H) • Imported active energy (accumulative) in Wh • Exported active energy (accumulative) in Wh • Reactive energy of quadrant 1 (Imported inductive energy) in +Var_i • Reactive energy of quadrant 2 (Exported capacitive energy) in -Var_c • Reactive energy of quadrant 3 (Exported inductive energy) in -Var_i • Reactive energy of quadrant 4 (imported capacitive energy) in +Var_c • Voltage of phase 1 in V • Voltage of phase 2 in V • Voltage of phase 3 in V • Voltage of phase N in V

Table 14 and Table 15 present the data provided by the GIS. These concern secondary substation data, including information of transformers, lines and supply points (customer connection points), as well as switchgear data. The CIM classes needed to model will be based on SecondarySubstation, PowerTransformer, Line, ACLineSegment, Switch, Location, ConnectivityNode.

Table 14 GIS Secondary Substation data

Source description
Information of transformer element in the grid as well as its interconnection with lines (and line segment), supply points and meters. Provided by the GIS.
Data format
XLSX file with header row, with multiple sheets, containing the following information. Sheet with output of transformer, containing: <ul style="list-style-type: none"> • Exit no. of transformer • Unique ID of node

<ul style="list-style-type: none"> Line voltage in V
Sheet with line segments connected to the transformer, containing:
<ul style="list-style-type: none"> Exit no. of transformer Unique Id of line segment Line voltage in V Size in m2 Material Type (Cu or Al) Deployment Type (aerial or ground) Length in meters Starting node ID Ending node ID Coordinates of starting point Coordinates of end point
Sheet with supply points connected to the transformer, containing:
<ul style="list-style-type: none"> Exit no. of transformer Unique ID of node Unique ID of supply point Unique ID of meter Contracted power of meter Line voltage in V Type of meter (three-phase or single phase)

Table 15 GIS Switchgear data

Source description
Information of switchgear element in the grid. Provided by the GIS.
Data format
XLSX file with header row, containing the following information. <ul style="list-style-type: none"> Unique ID of node Name Network Substation Voltage level Type of element (switchgear, disconnector etc.)

Table 16 and Table 17 present the data provided by the SCADA. The former describes the information of the status of switchgears (grid configuration), whilst the latter describes the information of the status of PEDs. The CIM classes needed to model will be based on SecondarySubstation, PowerTransformer, Line, ACLineSegment, Switch, Location, ConnectivityNode, as well as custom attributes and classes for the modelling of PED attributes.

Table 16 SCADA configuration Data

Source description
Information of grid configuration (i.e. switchgear's statuses). Provided by the SCADA.

Data format

CSV file with header row containing the following information:

- Identifier of the element
- Description of the element
- Name
- Network
- Substation
- Voltage level
- Type of element (switchgear, disconnector etc.)
- State (undetermined, open, closed, or unknown)

Table 17 PED Status data

Source description

Information of PED statuses in the grid. Provided by the SCADA.

Data format

Text file containing the following information:

- Identifier of element
- Time as yyyy-MM-dd. HH:mm:ss
- Voltage Ph1 (V)
- Voltage Ph2 (V)
- Voltage Ph3 (V)
- Voltage PhN (V)
- Current Ph1 (A)
- Current Ph2 (A)
- Current Ph3 (A)
- Current PhN (A)
- Power Ph1 (KW)
- Power Ph2 (KW)
- Power Ph3 (KW)
- State of charge of battery (kWh)
- State of charge of battery (%)
- Alarm code
- Reactive compensation signal (true/false)
- Harmonic current mitigation signal (true/false)
- Currents balancing signal (true/false)

Table 18 up to Table 22 provide a description of the data provided by the integrated services of EF, PFS, CEF, GOS and FDA.

Table 18 Consumption and Generation forecast data

Source description

Consumption and Generation forecast data provided by the Energy Forecaster (EF). Details the quantity of energy imported/exported at specific time in different buses of the distribution grid.

Metadata
<ul style="list-style-type: none"> • Forecast request • Weather forecast data
Data format
<p>Text file containing the following information:</p> <ul style="list-style-type: none"> • Bus ID • Time as yyyy-MM-dd. HH:mm:ss • Forecast quantity (+/- sign defines the direction of the energy flow) • Upper bound • Lower bound • Confidence level (of the calculated upper and lower bounds) • Energy unit (by default Wh)

Table 19 Power Flow Simulation data

Source description
Power flow simulation data provided by the Power Flow Simulator (PFS).
Metadata
<ul style="list-style-type: none"> • Simulation request. • Consumption and generation data forecast used. • SCADA configuration used. • Voltage data used.
Data format
<p>Text file containing the following information:</p> <ul style="list-style-type: none"> • Node ID • Time • Voltage Ph1 (V) • Voltage Ph2 (V) • Voltage Ph3 (V) • Voltage PhN (V) • Current Ph1 (A) • Current Ph2 (A) • Current Ph3 (A) • Current PhN (A)
Notes
Time is in yyyy-MM-dd. HH:mm:ss format.

Table 20 Critical Event Forecast data

Source description
Critical event forecast data provided by the Critical Event Forecast (CEF). A list of events a specified locations of the grid (bus or line).
Metadata
<ul style="list-style-type: none"> • Forecast request

<ul style="list-style-type: none"> • Consumption and generation data forecast used. • SCADA configuration used. • Voltage data used. • Power flow simulation data used.
Data format
<p>Text file containing the following information:</p> <ul style="list-style-type: none"> • The bus ID where critical event has been forecast • Branch (line) ID where critical event has been forecast • The Phase where the critical event has been forecast • Calculation time of the forecast • Critical event forecast time, when the critical event is expected to occur • Critical event type, defining the type of critical event (congestion, over-voltage, under-voltage) • A values indicating the magnitude of the critical event • Optionally the critical event magnitude unit. By default congestion unit is % over thermal limit and voltage unit % over nominal voltage.
Notes
Date times are in as yyyy-MM-dd. HH:mm:ss format.

Table 21 Grid Operation Schedule data

Source description
Grid schedules provided by the Grid Operation Scheduler (GOS). Describes a list of actions of each grid actuator: PED or switchgear.
Metadata
<ul style="list-style-type: none"> • Schedule Request • Consumption and generation data forecast used. • SCADA configuration used. • Power flow simulation data used.
Data format
<p>Text file containing the following information:</p> <p>Switchgear Schedules:</p> <ul style="list-style-type: none"> • Switchgear ID • Switching Schedule, as a 2xN array of switching actions, specifying: <ul style="list-style-type: none"> • Date time of the action • Action code(open/close a line) <p>PED Schedules:</p> <ul style="list-style-type: none"> • PED ID: • Operation Schedule, as a 2xN array of import/export actions, specifying: <ul style="list-style-type: none"> • Date time of the action • The quantity of energy exported/imported (+/- sign defines the direction of the energy flow) • Energy unit (by default Wh)
Notes
Date times are in as yyyy-MM-dd. HH:mm:ss format.

Table 22 Fault detection alert

Source description	
Faults provided by the FDA application.	
Metadata	
<ul style="list-style-type: none"> • Schedule Request • Consumption and generation data forecast used. • SCADA configuration used. • Power flow simulation data used. 	
Data format	
<ul style="list-style-type: none"> • Fault type code, for describing fault type • Fault location, as a list of Node IDs where the fault occurred (estimation) • Fault locations, as a list of Line ID(s) where fault occurred (estimation) • Pinpoint location as a distance of the fault from the node (inline distance in meters) • Detection time of the fault • Estimation of the fault start time • Estimation of the fault end time (if occurred) 	
Notes	
Date times are in as yyyy-MM-dd. HH:mm:ss format.	

4. The Data Analytics Platform

This chapter presents the design of the DAP and is organized as follows:

- Initially, an overview of the DAP is presented based on the requirements and constraints documented in previous work of the project (D1.2 [1] and D1.3 [2]). Towards this a high-level view of the DAP is presented;
- The rationale of design is presented, documenting 3rd party software building blocks that were utilized;
- An architectural description of the DAP is modelled, detailing its internal components and providing different views for consideration.
- Finally, the detailed design of the components is documented, describing their role, interfaces, internal logic and data design.

4.1. Overview

The Data Analytics Platform (DAP) is the solid instantiation of the Data Management, Analysis and Visualization tools identified in D1.3. Its role is, on one hand, to allow the transparent integration of heterogeneous data technologies & vendor subsystems, various data types (from start metering data to models of distribution grid, load consumption / generation forecasts etc.), offering validation and homogenization of data and guaranteeing accessibility with specific QoS characteristics. DAP acts as a centralized storage of data from legacy systems and results of computation algorithms, allowing the decoupling of the legacy systems from the operation of the advanced functionalities that will be developed in the project. On the other hand, it is able to host analytics computations, providing them as a service to other applications, as well as provide data visualization (raw data or results of computations) as an embeddable artefact able to be integrated in web applications.

The DAP's design is constrained by the following facts:

- **Scalability:** The data held by the DAP is expected to increase beyond to those initially foreseen, as new legacy systems are integrated and the understanding of ways to exploit these data improves, leading to new applications being developed and executed by the DAP. A flexibility on the amount of data and type of data stored in the DAP must be established;
- **Performance:** The DAP will act as a central data repository, providing access to different applications, with different performance requirements. The DAP must be able to meet these requirements, avoiding any unnecessary delays, which can affect negatively the normal execution of the business process. Furthermore, DAP acts also as an execution platform for analytics applications and as a visualization provider. In a similar manner the quality of service of these services must be ensured;
- **Availability and Reliability:** The DAP acts as a source of data for many applications, hence, the downtime of both the DAP must be minimized. Furthermore, non-availability of visualization or analytics services from the DAP, should also be avoided, otherwise the Operation Applications, which integrated such services, will not be able to provide monitoring functionalities to the end-user;
- **Smart Grid Standards' compliance:** The communications related to data operations with DAP will follow the CIM specification (IEC 61968), given that the ESB, which mediates all relevant communication with DAP, will follow this specification;
- **Data security, privacy and integrity:** Some of the data held in DAP are of sensitive

nature (grid operation data) thus data confidentiality must be ensured. Furthermore, all data requested from the DAP must be delivered to properly authenticated and authorized recipients, over reliable and secure channels.

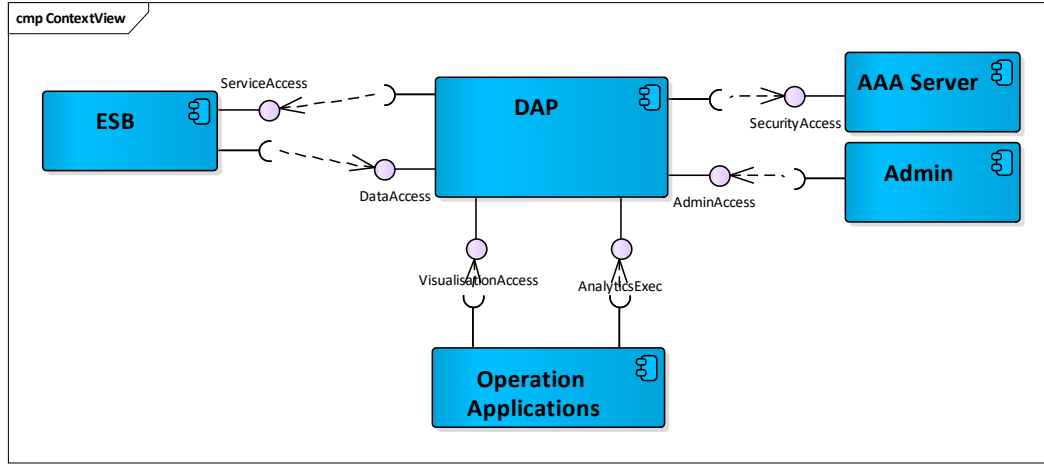


Figure 14 DAP high-level overview

A top-view of the DAP is presented as a UML component diagram in Figure 14, detailing its interaction with external systems and applications. A brief explanation of the interfaces is presented in Table 23.

Table 23 DAP Interfaces

Type	Interface Name	Description
Required	<i>ServiceAccess</i>	This interface exposes the services integrated through the ESB; DAP will use this interface for polling external data.
	<i>SecurityAccess</i>	Interface to AAA server for integrating security services.
Provided	<i>DataAccess</i>	Interface for accessing and providing data from/to DAP.
	<i>VisualisationAccess</i>	Through this interface, access to visualisations is provided.
	<i>AnalyticsExec</i>	This interface provides the ability to trigger the execution of analytics services provided by DAP.
	<i>AdminAccess</i>	Through this interface, the administrator of the system interacts with DAP in order to design the applications.

4.2. Design Principles

The data managed by the DAP have certain characteristics that could characterize them as “big data” [15]. The 3Vs (volume, variety and velocity) are three defining properties or dimensions of big data, with volume referring to the size of data, variety referring to the number of types of data and velocity referring to the speed of data processing. On one hand we have a large volume of data to be stored and a variety of types (e.g. CIM network models, grid measurements), acquired from different systems (e.g. SCADA MDMS, GIS). On top of that, we have a data ingestion that could be considered near real-time for some types of data (e.g. from SCADA). Furthermore, as already mentioned, the creation of such a platform dictates flexibility and scalability both in capacity as well as in processing power.

CIM-based network models have generally been managed in relational databases (RDBMS) and queried through complex Structured Query Language (SQL). Unfortunately, such data models are difficult to naturally represent in relational structures creating a management overhead. The object-class hierarchy of the network model is decomposed into numerous linked relational tables and sophisticated queries must be developed using large numbers of table joins. The use of a triple-store [16] or graph [17] database technology facilitates [18] the management of the data representation of such models, in its native triple format (subject-predicate-object). SPARQL [19], an RDF query language, enables retrieval and manipulation of data stored in triples.

The solid benefits of traditional relational databases include efficient storage/transactions and retrieval, guarantee of validity of data and concurrent access (ACID properties), as well as data security. But on the other hand, Distributed File Systems (or network file systems), such as the Hadoop Distributed File System (HDFS) [20], enable multiple users using different machines to share file storage and computational resources and provide performance scalability and resilience. Non-relational databases, such as HBase [21] - a distributed key-value store on top Hadoop- provide flexibility in terms of data schemas and scalability.

Based on the above, as well as considering the visualization requirements of DAP, its design will be based on the following tools/libraries:

- **Hadoop:** A set of software libraries that provide a framework for the distributed processing of massive data sets and computations across a network of computers [20]. It is based on the MapReduce programming model, designed to scale up from single to thousands of machines, each offering local computation and storage, whilst hardware failures are automatically handled by the framework. Hadoop has a rich ecosystem of modules and applications (e.g. Hive, Spark, Mahout) that can be installed on top of or alongside enabling: data manipulation operations, management of real-time data series, machine learning operations etc. Hadoop platform: includes the storing component HDFS and the processing components MapReduce and Yarn. In the case of RESOLVD project, the key attributes of Hadoop have lead to its selection for the design of the DAP, concern its scalability using commodity hardware, it's out of the box fault tolerance, as well as its support of for a variety data types by design.
- **Hive:** An open source SQL-based distributed warehouse system built on top of Hadoop framework [22]. Hive has an SQL-like declarative query language called HiveQL, providing an abstraction of the complexity of Hadoop MapReduce. HiveQL queries are compiled into map-reduce jobs that are executed using Hadoop. Hive uses a meta-store (relational DB) for storing information related to Hive tables (like their schema and location. In the context of RESOLVD, Hive will be used for batch processing of the data stored in DAP.
- **Elasticsearch:** An open-source search and analytics engine for data stored in non-relational databases [23]. It provides good scalability, has near real-time search, and supports multitenancy. Elasticsearch uses Lucene library to provide powerful full-text search capabilities. By the use of Elasticsearch big volumes of data can quickly be analyzed in near real time. Utilizing its extensive query language and by offering simple REST based APIs and the use of schema-free documents it offers an easy way to index, search, and query data. Elasticsearch for Apache Hadoop (ES-Hadoop) is a software library that offers the combination of these two technologies, allowing a bi-directional flow of data, which can be utilized for their interoperation.
- **Kibana:** An open-source, visualization tool designed as a plugin of Elasticsearch [24]. Kibana offers intuitive and interactive charts and allows the creation of dashboards [25]

and reports, which can be saved and easily accessed through a web-browser. Users can create bar, line and scatter plots, or pie charts (see Figure 15) and maps on top of large volumes of data - on top of content indexed on an Elasticsearch cluster.



Figure 15 Sample Kibana visualisations

In the context of the DAP, Hadoop will be utilized for as a long-term storage repository, Elasticsearch will be used as short-term storage and analytics engine (KPIs calculation) and Kibana as a visualization provider.

4.3. Architecture

A logical/functional decomposition of DAP is presented in Figure 16 and briefly explained below:

- The Analytics Service enables the execution of computations (e.g. KPI calculations) for internal or external components;
- The Visualization Service enables the visualization of data stored in DAP;
- The Data Hub is responsible for receiving or collecting data from various sources (e.g. MDMS, Weather Service etc.) and can be queried by internal components (e.g. Visualization Service) or external systems (i.e. the ESB) to retrieve stored information;
- The Access control component ensures authorized access to the data stored as well as the analytics and visualization services provided by the DAP.

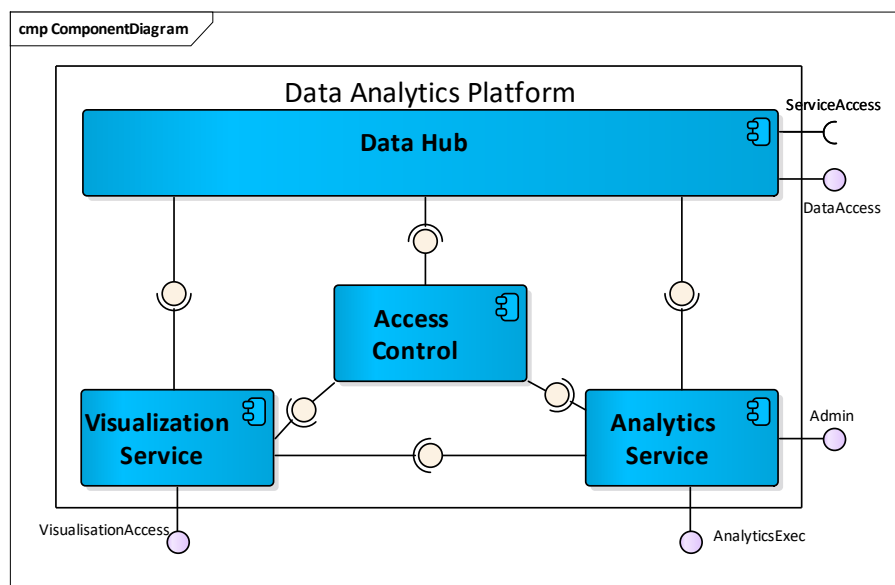


Figure 16 DAP Component diagram

4.4. Components Design

4.4.1. Analytics Service

This component is responsible for the execution of analytic computations, as well as for keeping the repository of the code to be executed. The current requirements of the project indicate DAP as the calculation engine for the KPI's that will be utilized for the statistical analysis of impacts based on grid operation actions. The component will provide an interface for the definition of the KPI's and for requesting their calculation. The Analytics Service will use Elasticsearch and Hive at its core, using domain specific language to define queries.

The methods exposed by the Analytics Service are those externally visible through the Admin Access and the AnalyticsExec interface, as presented in Table 25.

4.4.2. Visualization Service

This component provides visualisations of the data stored in the DAP, or computation of such data, as provided by the Analytics Service. Such visualization are provided to the Operation Applications as an embeddable artefact in the web tier. The Visualization Service will use Kibana at its core, for visualizing data stored in Elasticsearch and Hadoop.

Kibana visualizes data queried from the Elasticsearch database, which can be presented either as a single chart or a collection of more charts – as a Kibana dashboard. Individual dashboards can be saved and re-executed based on the current state of the data, whilst accessibility is provided by a unique URL.

Predefined visualizations patterns will be stored as templates and will be accessible through a web interface. URLs created from Kibana for accessing or parameterizing a visualisation are quite complex, hence an encapsulation will be provided by a mediator -in our case the ESB- creating a 'user-friendly' web API for the clients (Operation Applications). Furthermore, the mediator will enhance the direct access to the service with a security mechanism, which is quite useful, since the inherent security capability of Kibana is not adequate and easy to integrate.

The methods exposed by the Visualization Service, are those that are externally visible through the Visualisation Access interface, presented in Table 25.

4.4.3. Data Hub

This component will expose an API for data manipulation and retrieval. Data will be delivered to DAP from various sources and stored into two storage modules: the short-term (Elasticsearch) and the long-term (Hadoop). By using es-hadoop [26] add-on of Elasticsearch, an interface allowing the simultaneous storage of data onto both Elasticsearch and Hadoop will be achieved.

Data provided to the Data Hub will be considered as validated data (from the original source), whilst any transformations to the appropriate format will be handled by the ESB, through adapters. Hence the component will only be responsible of retrieving and storing the data.

4.4.4. Access Control

Access Control component ensures authorized access to DAP. Most services are mediated by the ESB, which is safeguarded by the AAA server. However the need for direct access of the *Visualisation Access* interface or the Admin interface imposes the need for using this component, which handles the integration of DAP with the AAA server, by invoking the relevant authorisation and authentication services.

Table 24 presents the *Access Control* interface's methods.

Table 24 Access Control interface

Method	Parameters	Response	Description
<i>authenticate</i>	Username Password	Session ticket	Exposes the method of authentication of the AAA server.
<i>authorize</i>	Session ticket, Service ID	Success Code	Exposes the method of authorization of the AAA server. Returns an error code if the authorization fails.

4.5. User Interface Design

The Visualisation Service will be able to present data stored in DAP as web pages and as embeddable components into other applications. For this purpose, Kibana dashboards, a collection of one or more charts, will be utilized. These UI elements are able to present the results of queries on the underlying Elasticsearch database, whilst Kibana provides the ability to save dashboards, which can be re-executed using the current state of the data. A unique URL is provided for each dashboard saved, whilst during re-execution different parameters (e.g. time) can be used as presented in §4.4.2.

The dashboards will be created by the administrator interface of Kibana and will be saved and made accessible through an external service, using a visualisation identifier and providing the necessary query parameters (e.g. time period, filters, aggregations).

Kibana supports a variety of chart types, whereby the visualised data can be either raw data or the result of processing (e.g. summing, averaging) or aggregation (e.g. count, location, ordering).

Some basic chart types that are supported:

- Line Chart: Depicting data as lines;
- Pie Chart : A pie with different slices for each bucket identified in the data;
- Area Chart: Presenting a line chart with filled areas below the lines, whilst stacking, overlapping of the areas is supported;
- Heatmap: A heatmap is a graphical representation of data in a two dimensioned coloured matrix, where colouring indicates a range of values;
- Data Table: visualizes a table of the data.

A design of the user screens (dashboards), according to the requirements set in D1.2, is presented in the Operation Applications section (see §6).

4.6. External Interfaces Design

Table 25 presents the external interfaces and underlying methods provided by DAP.

Table 25 DAP external interfaces

Interface Name	Method	Parameters	Response	Description
Data Access	retrieve	Data Query	Data Payload	Retrieves a data set based on the data query.
	store	Type, Data Payload	Success Code	Saves the provided data set.
Visualisation Access	retrieve	Visualisation Identifier, Query Params	Embeddable iframe	Returns an interactive embeddable visualisation.

Analytics Exec	execute	Process ID, Process Params	Process Result	Start execution of a calculation process.
	info	Process ID	Process Status	Retrieves the information of a calculation process and its current status.
Admin Access	add	Process Name, Process Description, Process Code,	Process ID	Stores calculation process code.
	delete	Process ID	Success Code	Deletes a calculation process.
	list	-	List (ID, name, description)	Lists all stored calculation processes.

4.7. Data design

A description of the data structures provided to the DAP is presented in the table below, providing, a short description, the format of the data and the update frequency.

Table 26 Data Types

No.	Information Object Name	Short Description	Relative Standards	Ingestion Frequency
1	Consumption & Generation Data	Energy demand and supply measurement data per customer for a specified time period	CIM XML	Medium (Once per day)
2	Consumption & Generation Data Request	Request for energy data containing location (or grid ID) and time period	CIM XML	Medium (Multiple times per day)
3	Critical Event Forecast	Voltage at each bus and current through line with a detected critical situation (e.g. overcurrent)	CIM XML	Medium (Multiple times per day)
4	Critical Event Forecast Request	Request for forecast of critical events, containing power flow simulation, SM voltage and grid configuration data	CIM XML	Medium (Multiple times per day)
5	Critical events analysis	Critical events (over/under-voltages) list including the magnitude, duration and location (bus and/or line) of the event.	CIM XML	Medium (Multiple times per day)
6	Fault detection alert	Alert of fault detection specifying fault type, component which performed the detection (WAMS or FDA), fault time (PMU data timestamp used to detect the fault), and fault localization	Proprietary	Medium (Multiple times per day)
7	Fraud detection alert	Alert of fraud specifying meter identifier and magnitude/likelihood of fraud	Proprietary	Medium (Multiple times per day)
8	Generation and Demand Forecast	Energy consumption and generation forecast as a time series, with model uncertainty,	CIM XML	Medium (Multiple times per day)

		at each specified bus and each time-slot for the requested time period		
9	Generation and Demand Forecast Request	Request for calculating and providing energy forecast for a specified period and locations (grid or model ID). The request also contains last week's consumption and generation data, weather forecast for the target time period, relevant buses for energy forecast	CIM XML	Medium (Multiple times per day)
10	Grid Configuration and Status	Data that depict the current grid topology and status (switchgear status, lines or regions that are offline due to maintenance, fault etc.)	CIM XML	High (Multiple times per hour)
11	Grid Configuration Data Request	Request for calculating and provisioning grid configuration data (switchgear status).	CIM XML	Medium (Multiple times per day)
12	Grid Operation Schedules	Set of grid operation schedules to be considered by the DMS with the estimated optimal operation of each one. Each schedule consists of a sub-schedule for each grid actuator (PED/switchgear)	CIM XML	(Medium (Multiple times per day)
13	Grid Schedule	Grid Schedule containing control commands for Switchgear and Charging/Discharging Schedules for PEDs.	CIM XML	Medium (Multiple times per day)
14	Grid Operation Schedule Request	Request for a Grid operation schedule by defining optimization objective and enclosing, grid information, expected energy generation and demand, grid actuators information (battery status, capacity, switchgear connections, etc.)	CIM XML	Medium (Multiple times per day)
15	Grid Topology	Grid topology containing electrical features of the grid (e.g. lines impedances)	CIM RDF/XML	Low (Upon change of physical element)
16	Grid Topology request	Request for acquiring the model of a desired part of the grid.	CIM XML	Low
17	Historical SM voltage and demand and generation data	Historical data from SMs containing voltage and demand / generation measurements.	CIM XML	Medium (Multiple times per day)
18	Historical SM voltage and demand and generation data request	Request for Historical SM voltage and demand and generation data request	CIM XML	Medium (Multiple times per day)
19	Historical weather data	Weather data (temperature and irradiance) at each time-slot for the specified time period.	JSON	Low
20	Historical weather data request	Request for weather data containing location and time period and time granularity.	JSON	Medium (Multiple times per day)
21	PED state	Describes the state of the PED, detailing the PED ID, voltages, current and power at the point of	CIM XML	High (Multiple times per hour)

		connection, SoC of the set of batteries (aggregated), alarms and warnings, availability and information related to the activation of main functionalities (such as Reactive compensation, Harmonic current mitigation).		
22	PED state request	Request for PED state (including PED ID)	CIM XML	Medium (Multiple times per day)
23	Power Flow Simulation request	Request containing energy demand and supply and grid configuration	CIM XML	Medium (Multiple times per day)
24	Power Flow Simulation Results	Results of power flow analysis as a time series of voltage/current values per location	CIM XML	Medium (Multiple times per day)
25	Voltage data	Voltage data measured by SMs in the specified grid and for the specified time period.	CIM XML	Medium (Multiple times per day)
26	Voltage data request	Request containing location (or grid ID) and time period	CIM XML	Medium (Multiple times per day)
27	Weather Forecast	Time series of weather forecast data (temperature and irradiance) for a specified time period.	JSON	Medium (A few times per day)
28	Weather Forecast Request	Request for weather forecast containing location, time period and time granularity.	JSON	Medium (Multiple times per day)

5. AAA Server

The AAA server is a security infrastructure offering authentication, authorization and accounting functionality and enabling the control of user access to network resources, as well as tracking of relevant activities.

The aim of the current design is to facilitate the integration of security mechanisms to services provided in a network of services, mitigating the issue of security from the service provider to the AAA server. The design supports the management of both individual services as well as aggregated ones, supporting a hierarchy-based structure.

A conceptual visualisation of the operations of AAA server is presented in Figure 17. The AAA server is encapsulated behind a mediator (in the context of the project the ESB) and is responsible for authenticating clients, authorizing access to services (resources) as well as for keeping track of the activity of clients for any malicious behaviour.

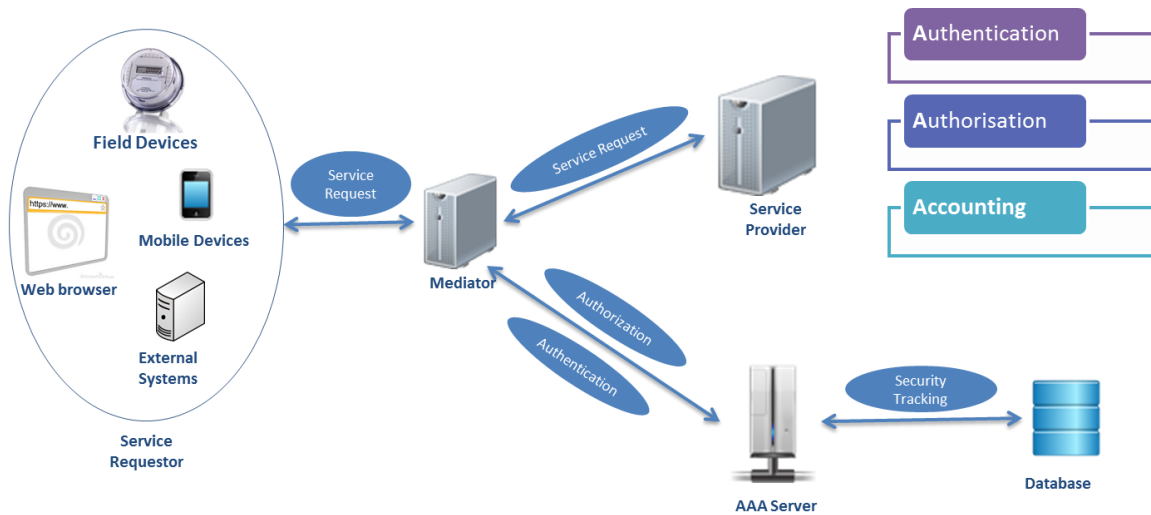


Figure 17 AAA Server high-level overview

Figure 18, presents the sequence of actions related to authenticating and accessing a service protected by the AAA server. Upon successful authentication, the service requestor will be provided with a session ticket. This ticket must be used in following communications as an authentication and authorization means for accessing a service. When requesting a service, the AAA server will validate whether the ticket is valid and whether the session is still active. If both valid and active, the mediator will invoke the requested service and the result will be provided to the service requestor. Otherwise, a relevant error code will be presented to the service requestor.

Such operation requires that the Mediator and the AAA server have a synchronized service registry, where services are registered and which roles have accessibility to each services. Therefore, the AAA will provide an interface for relevant operations, whilst the mediator will be responsible for managing the service registry.

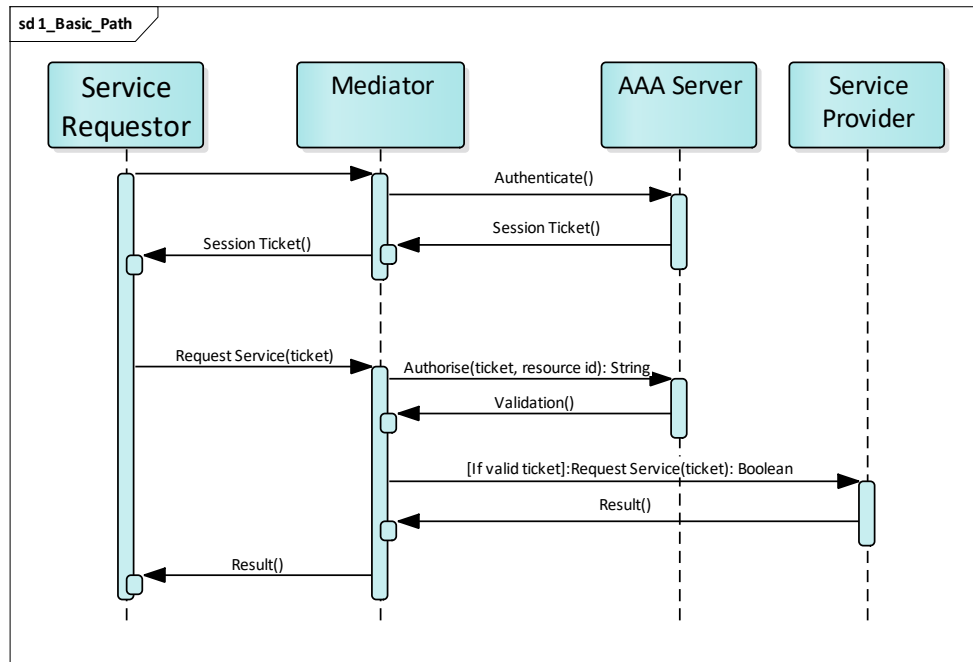


Figure 18 AAA Server: Sequence diagram

The internal logic of the AAA server is based on the following concepts:

- **Client:** A web-enabled entity that wants to access one or more services protected by the AAA server;
- **Node:** A network entity (group of resources) that uses the AAA server to be protected from unauthorized access. Consists of the node name (unique ID), type and one or more Data Nodes;
- **Data Node:** A service (resource) that uses the AAA server to be protected from unauthorized calls;
- **Roles:** Each role relates to one or more nodes and provides accessibility to the client(s) related to it. A role can have a list of permitted addresses (IPs), allowing only those to assume the specific role;
- **Permission:** Describes the relations (of accessibility) between roles and nodes.

In Figure 19 the login screen is presented, where the Administrator of the AAA server will need to provide credential for accessing the application.

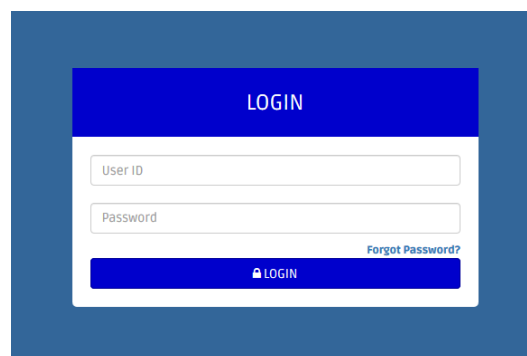


Figure 19 AAA Server UI: Login Screen

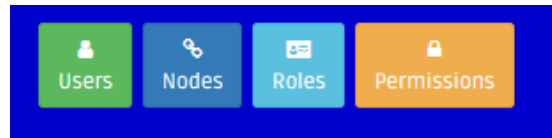


Figure 20 AAA Server UI: Menu

Figure 20 presents the menu of the application. The menu entails the basic concept presented above, the Users of the AAA server (aka clients), the Nodes, the Roles and relevant Permissions. The visualization of the Users screen, where the user is redirected when clicking on Users button, is presented in Figure 21. A list of the details of the users is presented to the administration, detailing the name, username, e-mail and type of user, as well as its status. By clicking on the view button the administration is able to view other information about the user, such as when was the last login, when was the user profile created or last updated and whether the user is registered in any roles (among other). The administrator can edit the information of the user, whilst by using the insert button, the administrator can add new user, providing the necessary information.

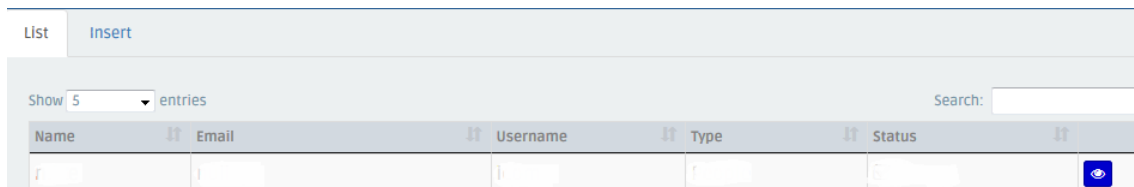


Figure 21 AAA Server UI: Clients view

In Figure 22, the Nodes registered in the application are presented. A list depicting the type of node, its alias (unique) and name, as well as any related Data Nodes. The administrator has the ability to delete a Node or insert a new one.



Figure 22 AAA Server UI: Nodes view

Figure 23, presents the Roles modelled in the system, detailing any restricted IPs, as well as the users that are assigned this Role. The administrator is able to add or delete a Role (without deleting the users) as well as to manage the restrictions and member of the Role.

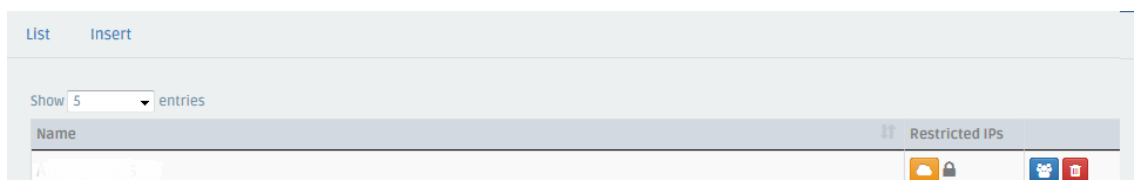


Figure 23 AAA Server UI: Roles view

The different Permissions modelled in the application are presented in Figure 24 and Figure 25. The first one presents for each Node Type, the different Roles that are registered for accessing this resource. The administrator is able to add or delete a Role, or the Permission. The second one, presents the list of roles, able to access a specific Data Node and enables the user to add/remove Permissions.

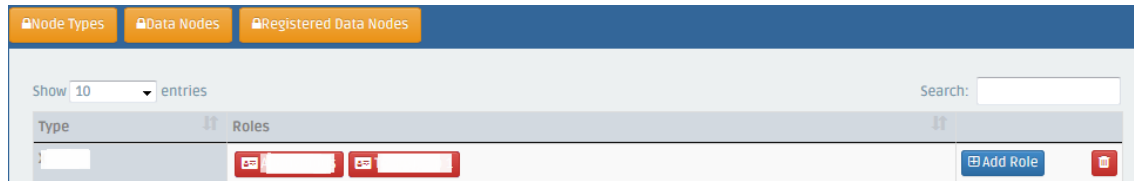


Figure 24 AAA Server UI: Permissions view (Node Types)



Figure 25 AAA Server UI: Permissions view (Data Nodes)

With regards to the external interfaces provided by the AAA server, these are summarized in Table 27.

Table 27 External interfaces provided by the AAA Server

Method	Parameters	Response	Description
authenticate	User credentials	Session ticket	The authentication function returns a session ticket if the provided credentials are correct. Otherwise, an error code is provided.
authorize	Session ticket, Resource ID	Boolean	Authorization function returns a session ticket if the client (identified by the session ticket) can access the resource. Otherwise, an error code is provided.
registerResource	Resource ID, List of Role ID	Boolean	This method can be used for adding a new service in the registry along with the list of role ids that have access to this service.
updateResource	Resource ID, List of Role ID	Boolean	This method can be used for updating an existing service in the registry along with the role ids that have access to this service.
deleteResource	Resource ID	Boolean	This method can be used for deleting a service from the registry.
getRoles	-	List of Role IDs and Description	Returns the description of the existing roles in the AAA server.
getResources	Role ID	List of Resource ID	This method returns list of resources accessible by a specific role.

6. Operation Applications

6.1. Overview

The Operation Applications provide the human-machine interface to enable the user (control centre operator) to monitor and control the advanced grid operations developed in the project, whilst on the other hand to communicate with the back-end systems for triggering these operations. More specifically the following features are provided.

- User interface for management the business flows;
- Triggering of business processes (services);
- Ability to set alarms on the results of processes;
- Manual or automatic mode of execution of grid control actions (schedule);
- Presents the list of business context “events” (i.e. critical events, faults, requests for loss reduction, self-healing and island management) of the application and their attributes;
- Visualizes the workflow model and information for further correlation, concerning:
 - Events
 - Consumption trends
 - Power flow simulation results
- Suggested and implemented grid operation schedules;
- Provides a statistical analysis of :
 - Events
 - Grid operation schedules implemented
- Utilization of the storage (batteries);

The applications provide two levels of analysis:

- Historical events, which were projected, detected or triggered and occurred (or not) in the past;
- Upcoming events, which are projected or planned to occur in the future.

Also correlation on whether grid operations schedules were proposed and implemented, or not, for the above type of events will be provided.

The applications developed will be the following:

- **Critical Event Prevention Application (CEPA):** This application will detect and present critical events related to the grid operation. The business workflow could be automatically triggered in a periodic manner (using a scheduler) or upon change of initial conditions (i.e. grid configuration modification, new energy forecast is available, PED schedule is modified);
- **Loss Reduction Application (LRA):** This application will trigger the loss reduction process and present the power losses reductions achieved. The business workflow could be triggered manually by the operator, or automatically in a periodic manner (using a scheduler);
- **Self-Healing Application (SHA):** Responsible for managing the self-healing process for resolving a fault situation in the distribution grid. Its workflow is triggered by the fault detection application.
- **Island Power Management Application (IPMA):** This application will manage the operation of an islanded part of the grid, through the available resources (PEDs), ensuring continuity of supply and power quality. Its workflow can be triggered manually, by an operator.

Detailed description of the requirement of the applications is presented in deliverable D1.2 [1].

6.2. Design Principles

Operation Applications will be developed as web-based applications, offering the required features. Hence, the client - the user interface and client side logic – will run in a web browser providing ease of accessibility to the user. The implementation environment will be the J2EE platform, supporting the implementation of distributed multi-tier applications.

Furthermore, the following tools will be utilized/integrated in the final solution.

- **Leaflet:** An open source JavaScript library used to build web-mapping applications known for its simplicity, performance and usability [27]. It supports different GIS formats, such as GeoJSON, KML and WMS and takes advantage of features of HTML5 and CSS3. In the context of this design, it is utilized for building interactive maps, utilizing the GeoJSON format for data exchanges. GeoJSON - based on the JavaScript Object Notation (JSON) - is an open standard format designed for representing simple geographical features, along with their non-spatial attributes. The web application will encapsulate the map UI and display it to the user. The user will be able to interact with the map through the dedicated UI elements, which will be parameterized by the web application, whilst visualized data (layers) will also be provided by the application.
- **Kibana:** As already presented (see §4.2), Kibana is an open-source visualization tool. A visualization artefact provided by a Kibana instance deployed in the DAP (see chapter 0) will be integrated in the web application, allowing the visualization of stored information as well as user interaction with the graphs. All business process related data as well as relevant legacy and external system/services data are stored in the DAP. Hence, this design provides an easy, straightforward and service-based approach of visualization.

The web pages presented to the user will consist of widgets from different sources. As illustrated in Figure 26, the web client will seamlessly integrate application code, with the artefacts of Leaflet and Kibana.

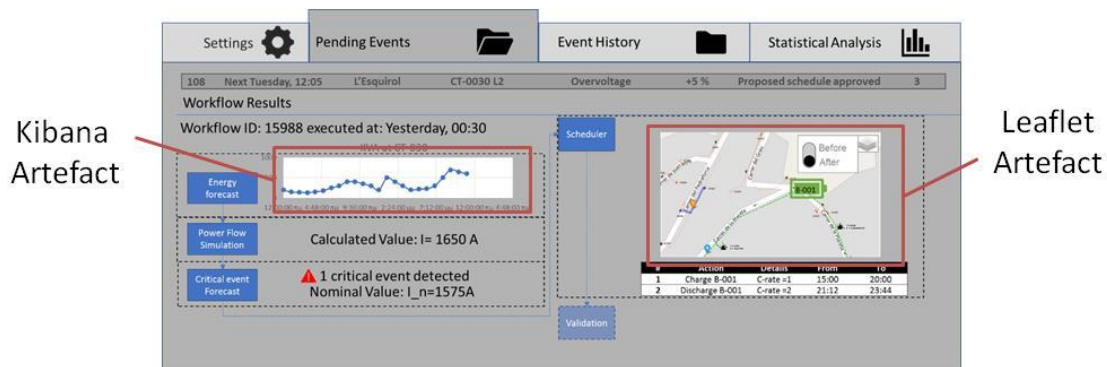


Figure 26 Example of artefact integration in UI

6.3. Architecture

An architectural view of the web application is presented in Figure 27.

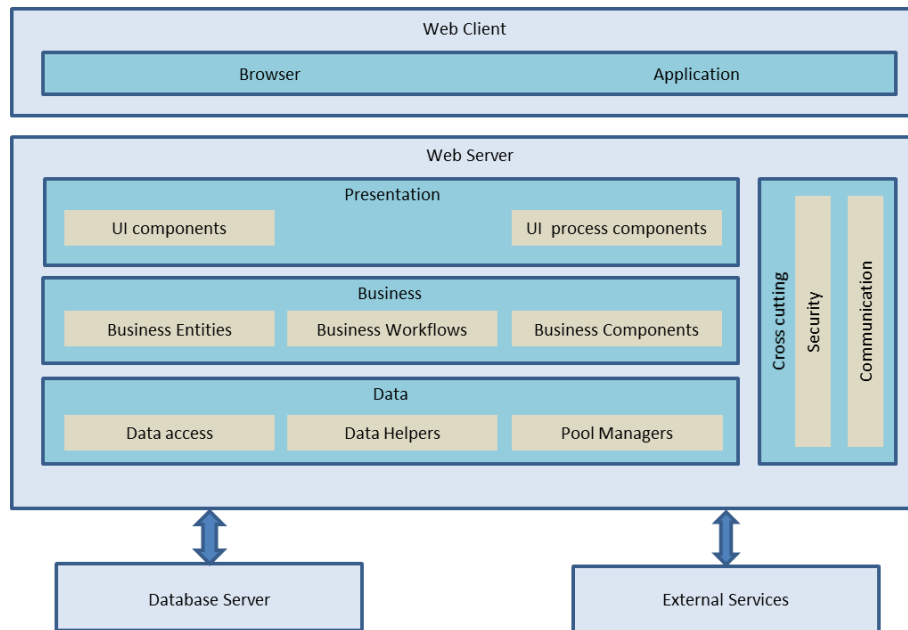


Figure 27 Operation application component view

The web architecture consists of the following components:

Table 28 Operation application components

Component Name	Description
<i>Client</i>	A web browser able to communicate (locally or remotely) with the web server.
<i>Web Server</i>	<p>A machine where the web application is hosted. The design of the web application is based on a classic three-tier architecture:</p> <ul style="list-style-type: none"> The first tier is the presentation tier, which consists of the UI components such as HTML, servlet, JSP or JavaScript. The second tier is the business logic tier, which consists of business workflows, business components and business entities, which are based on the J2EE standards. The third tier is the Data tier, which consists of Data access components, Data helpers and pool managers in order to achieve communication between the application server and the database server. <p>It also includes cross-cutting component related to security and communication:</p> <ul style="list-style-type: none"> The security component is able to check for the authority of any access to UI components, business logic components or Data components. The communication component is able to control the communication between layers.
<i>Database Server</i>	A repository of the data related to the operation of the application.

<i>External Services</i>	Services integrated to the application (i.e. Kibana visualisation services, ESB, AAA server, Leaflet plugins)
--------------------------	---

6.4. Internal Logic

The internal logic of the application is based on the following concepts:

- **UI Components:** The UI design has its basis on:
 - Templates: Text files (e.g. XML) that describe the layout of every web page that will appear on the client's screen. The latter is divided logically in parts, which are referred as visual components;
 - XSL drivers: Files based on XSL metadata, which include directives about the specific format of any visual component that will be presented on the client's screen;

All the above are part of the first tier, being orchestrated by the classes placed in the second tier;

- **Business Components:** Java classes including algorithms for calculating, summarizing, organizing and collecting any type of data. These components are able to connect with data collectors or database helpers in order to eventually transfer data from/to the components of the third tier, which is the Data layer;
- **Business entities:** Java classes having special features, inherited from other basic classes. Following the Java Enterprise Edition standards, some business classes should include all the Java Beans features by inheriting them from basic Bean classes;
- **Business Workflows:** Workflows imprinted either in java classes as program coding, or in special drivers including specific directions, which should be translated in order to feed decision makers classes;
- **Data access components:** These components enable communication with local or remote databases. Their instantiation enables them to access different types of databases based on the provided configuration;
- **Data helpers:** Java classes which are providing methods that will facilitate any action related to database information access;
- **Pool managers:** These java classes provide all the necessary functionality for managing and using data pools. Data pools are dedicated abstract components, able to better-organize database connections.

6.5. Graphic User Interface Design

This section presents indicative use interface screens (though mock-ups) as well as an analysis of the depicted information and the expected behaviour of the applications. Since the information depicted among the different applications are quite similar, the list of provided view is not exhaustive and aims to present the different type of interactive elements or workflow steps in the applications.

In Figure 28, the main menu of the application is presented in a tabular view, presenting the four main operations: Settings, Pending events, Event history and Statistical analysis. This menu will be used across all applications, whilst event will represent the domain specific event for each applications (i.e. critical event, loss reduction request, self-healing request, island power management request).

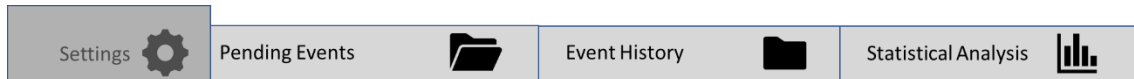
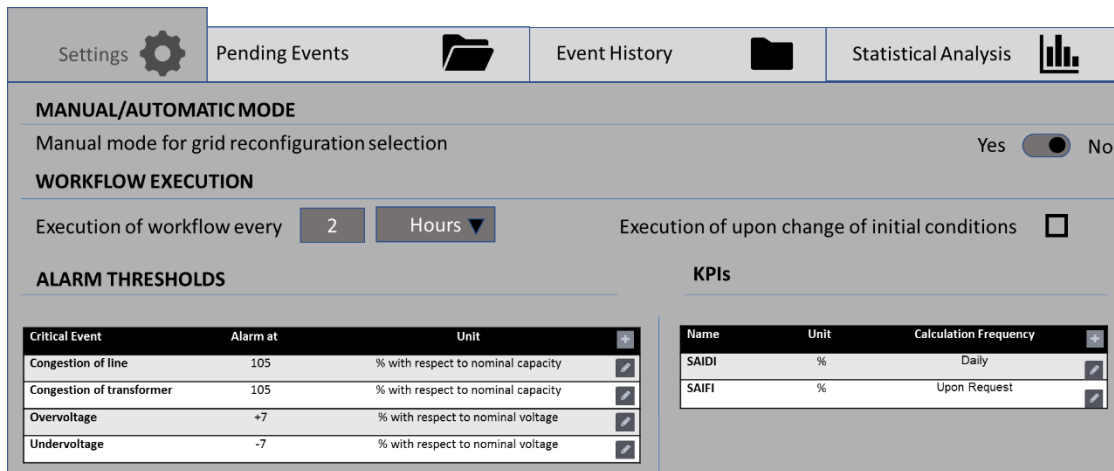


Figure 28 Main Menu

Figure 29 presents a mock-up of the settings page of CEPA, where the user is able to set the mode of operation of grid re-configuration selected. If manual mode is selected, upon execution of the business flow and calculation of the optimal scheduled, the business process will pause. The user must select one of the proposed schedules and resume the workflow. Otherwise, the application will automatically select one and continue the flow. Furthermore, the user is able to select the trigger event of the workflow execution, as well as alarm thresholds on critical events detected and KPIs to be calculated and visualized in the workflow analysis.

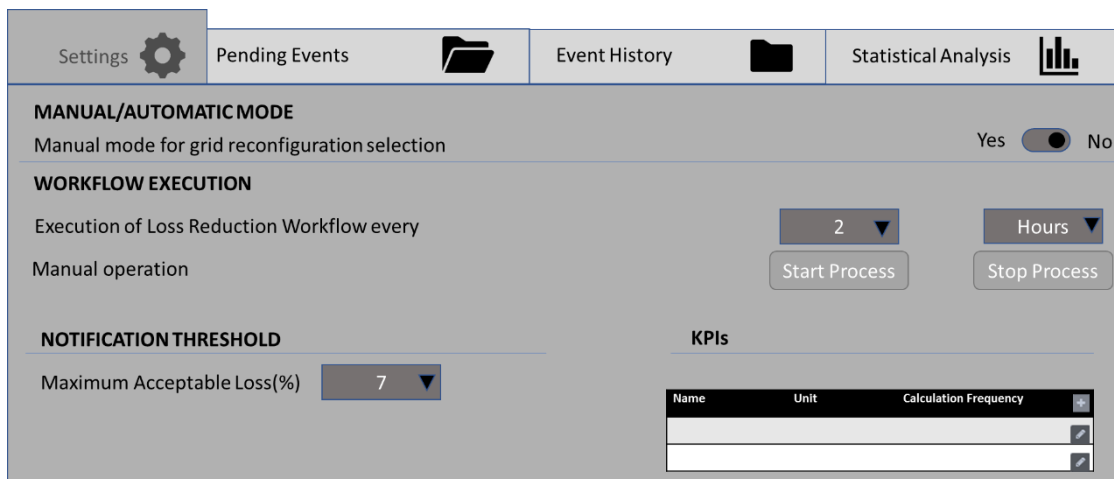


Critical Event	Alarm at	Unit
Congestion of line	105	% with respect to nominal capacity
Congestion of transformer	105	% with respect to nominal capacity
Overvoltage	+7	% with respect to nominal voltage
Undervoltage	-7	% with respect to nominal voltage

Name	Unit	Calculation Frequency
SAIDI	%	Daily
SAIFI	%	Upon Request

Figure 29 CEPA: Settings view

On the other hand, Figure 30, presents a mock-up of the settings page of LRA, illustrating the case where the user is able to manually control the execution of the workflow.



Name	Unit	Calculation Frequency

Figure 30 LRA: Settings view

In Figure 31, Figure 32 and Figure 33 different views of the upcoming critical events page are presented for CEPA:

- The first one, presents the overall status of the upcoming events using charts concerning status of schedule dispatch and its correlation with the severity of events.
- The second figure displayed a list of detected future critical event, detailing their

information: date/time, location, element affected, type, status and severity. Different colours categorize among the different event types and, for example, magnitude/severity of an issue. Furthermore, filtering and sorting of information is supported.

- Finally, the third figure depicts a map view of the critical event in the list. The map presents the grid nodes (lines, transformers, switchgear, PEDs) on a geographical map. The different colours highlight the level of severity of overload (congestion) of the lines. The map is composed of different layers. With each layer providing the different types of nodes. The nodes that are affected by critical events are highlighted and the user can click on a node to see information about the event. Information presented in the map are filtered by the list of critical events.

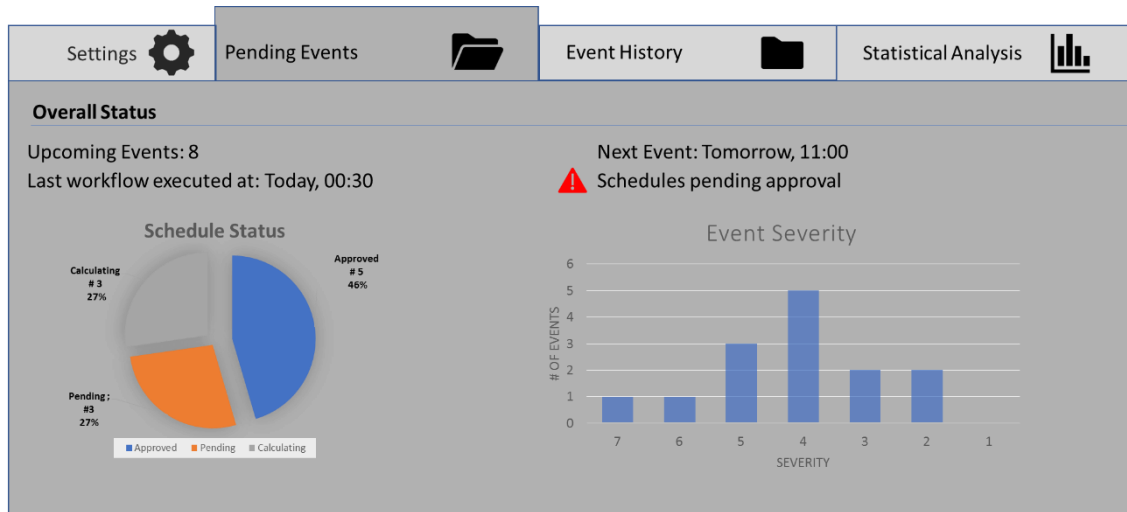
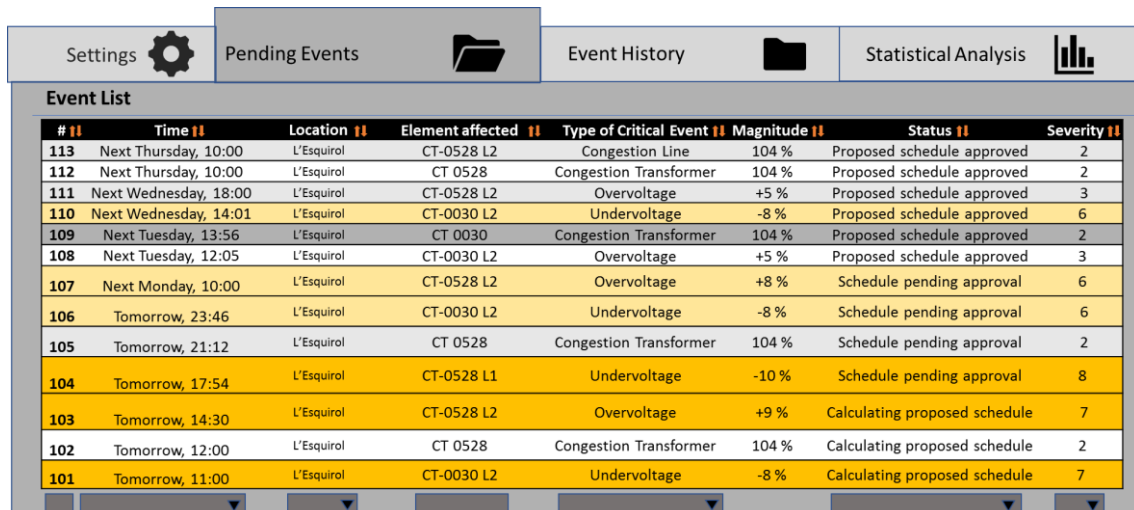


Figure 31 Upcoming critical events (Overview)



Event List

#	Time	Location	Element affected	Type of Critical Event	Magnitude	Status	Severity
113	Next Thursday, 10:00	L'Esquirol	CT-0528 L2	Congestion Line	104 %	Proposed schedule approved	2
112	Next Thursday, 10:00	L'Esquirol	CT 0528	Congestion Transformer	104 %	Proposed schedule approved	2
111	Next Wednesday, 18:00	L'Esquirol	CT-0528 L2	Overvoltage	+5 %	Proposed schedule approved	3
110	Next Wednesday, 14:01	L'Esquirol	CT-0030 L2	Undervoltage	-8 %	Proposed schedule approved	6
109	Next Tuesday, 13:56	L'Esquirol	CT 0030	Congestion Transformer	104 %	Proposed schedule approved	2
108	Next Tuesday, 12:05	L'Esquirol	CT-0030 L2	Overvoltage	+5 %	Proposed schedule approved	3
107	Next Monday, 10:00	L'Esquirol	CT-0528 L2	Overvoltage	+8 %	Schedule pending approval	6
106	Tomorrow, 23:46	L'Esquirol	CT-0030 L2	Undervoltage	-8 %	Schedule pending approval	6
105	Tomorrow, 21:12	L'Esquirol	CT 0528	Congestion Transformer	104 %	Schedule pending approval	2
104	Tomorrow, 17:54	L'Esquirol	CT-0528 L1	Undervoltage	-10 %	Schedule pending approval	8
103	Tomorrow, 14:30	L'Esquirol	CT-0528 L2	Overvoltage	+9 %	Calculating proposed schedule	7
102	Tomorrow, 12:00	L'Esquirol	CT 0528	Congestion Transformer	104 %	Calculating proposed schedule	2
101	Tomorrow, 11:00	L'Esquirol	CT-0030 L2	Undervoltage	-8 %	Calculating proposed schedule	7

Figure 32 CEPA: Upcoming critical events (List view)

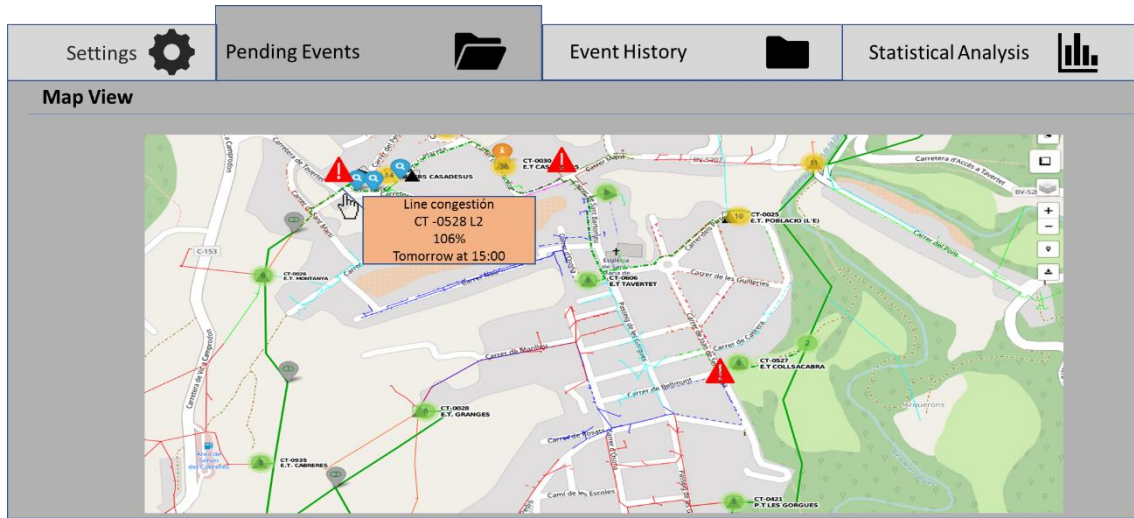


Figure 33 CEPA: Upcoming critical events (Map view)

Figure 34 and Figure 35 present different situations of the workflow associated to a schedule dispatch of a critical event in CEPA. The user can access this view by clicking on a specific event from the list. The user will be presented with the energy forecast at the specific node of the grid, and the outcome of the critical event forecast, presenting both the estimated value of the grid element and the nominal one. Furthermore, the schedule (to be) implemented or the list of proposed schedules for the user to select (in manual mode) is presented. An interactive map presents the impact of each schedule to the grid.



Figure 34 CEPA: Upcoming critical events (Pending Event Workflow: Schedule Approved)

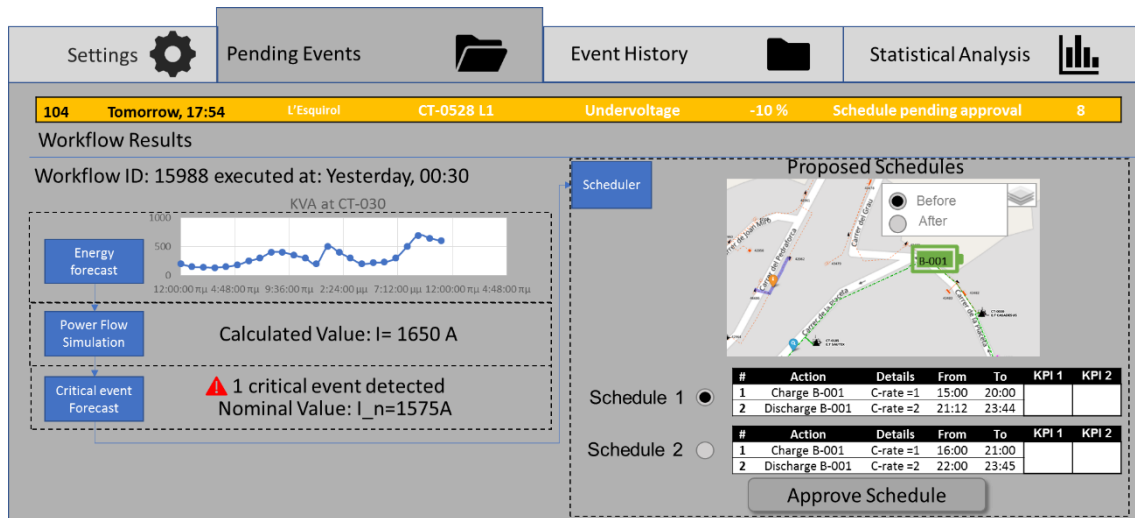


Figure 35 CEPA: Upcoming critical events (Pending Event Workflow: Schedule Pending)

On the other hand, Figure 36 presents the workflow of a loss reduction event in LRA, with a schedule pending for approval. One must notice the absence of the critical event forecast step. Expect from LRA, the same workflow (but with different trigger events and optimization parameters) is followed by IPMA and SHA as well.

Figure 37 visualises the Critical event history page of CEPA, presenting the list of past critical events and relevant information in a similar manner as “Pending events” page. The user is able to narrow down the scope of the analysis by selecting a specific time range. The user can add, edit and view notes of a specific record in the table, using the dedicated button.

In Figure 38 the workflow of a past event of CEPA is presented, in a similar manner to pending events (see Figure 34). In this case, the validation step is also presented, depicting the validation measurement or KPI(s) for the event.

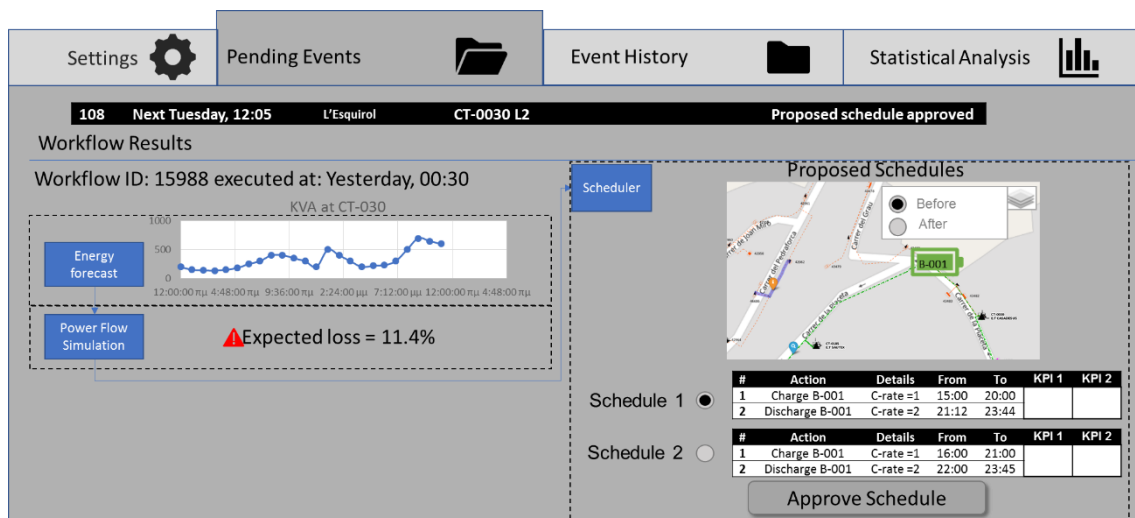


Figure 36 LRA: Upcoming critical events (Pending Event Workflow: Schedule Pending Approval)

Settings

Pending Events

Event History

Statistical analysis

From

01/01/2018

To

01/08/2018

#	Time	Location	Element affected	Type of Critical Event	Magnitude	Status	Severity
98	Yesterday, 23:45	L'Esquirol	CT-0528 L2	Congestion Line	104 %	Proposed schedule approved, C.E. avoided	2
97	Yesterday, 22:00	L'Esquirol	CT 0528	Congestion Transformer	104 %	Proposed schedule approved, C.E. avoided	2
96	Yesterday, 16:30	L'Esquirol	CT-0528 L2	Overvoltage	+5 %	Proposed schedule approved, C.E. avoided	3
95	Yesterday, 20:45	L'Esquirol	CT-0030 L2	Undervoltage	-8 %	Proposed schedule approved, C.E. avoided	6
94	Yesterday, 16:15	L'Esquirol	CT 0030	Congestion Transformer	104 %	Proposed schedule approved, C.E. avoided	2
93	Yesterday, 12:00	L'Esquirol	CT-0030 L2	Overvoltage	+5 %	Proposed schedule approved, C.E. avoided	3
92	Two days ago, 18:00	L'Esquirol	CT-0528 L2	Overvoltage	+8 %	Proposed schedule approved, C.E. avoided	6
91	Two days ago, 12:00	L'Esquirol	CT-0030 L2	Undervoltage	-8 %	Proposed schedule approved, C.E. avoided	6

Figure 37 CEPA: Critical event history (List view)

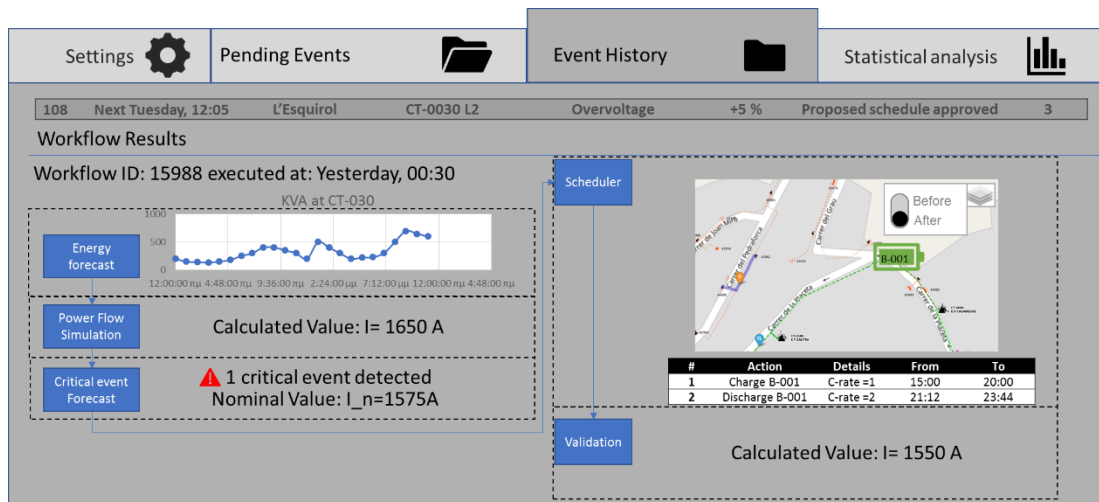


Figure 38 CEPA: Critical event history (Event view)

Figure 39, Figure 40 and Figure 41 present some views of the statistical analysis page of CEPA:

- In the first figure, three graphics present different types of analysis of critical events, based on: status type and severity;
- The second one presents two maps analysing the critical events in terms of most frequent event type and severity per location;
- The third one presents the frequency of occurrence of critical events (of any type) and status of events per location.

The user is able to narrow down the scope of the analysis by selecting a specific time range.

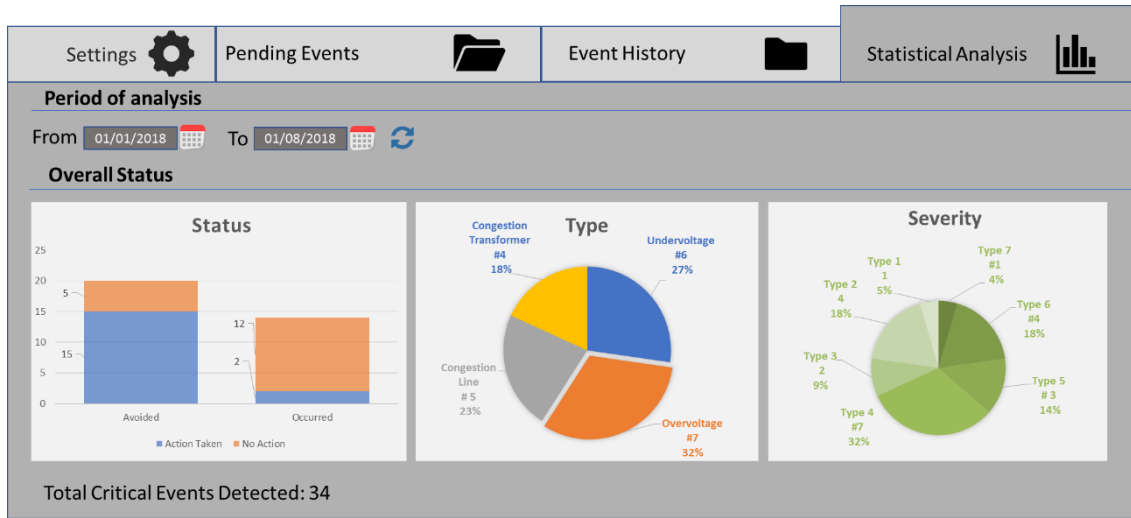


Figure 39 CEPA: Statistical Analysis Overview

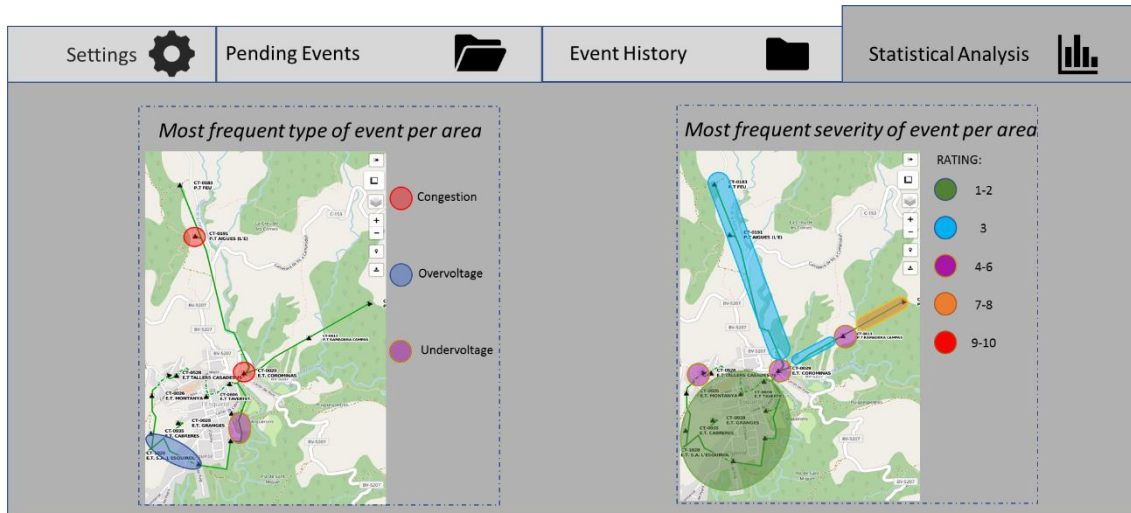


Figure 40 CEPA: Statistical Analysis map view (1 of 2)

In Figure 42 a list of critical events (for the defined time period) per element is presented for CEPA, specifying the type and location and the frequency of occurrence of critical events, as well as the most frequent occurring event.

Finally, Figure 43 presents the list of grid actuators i.e. PEDs and switchgears for CEPA, detailing their location, number of control actions and most frequent critical event tackled.

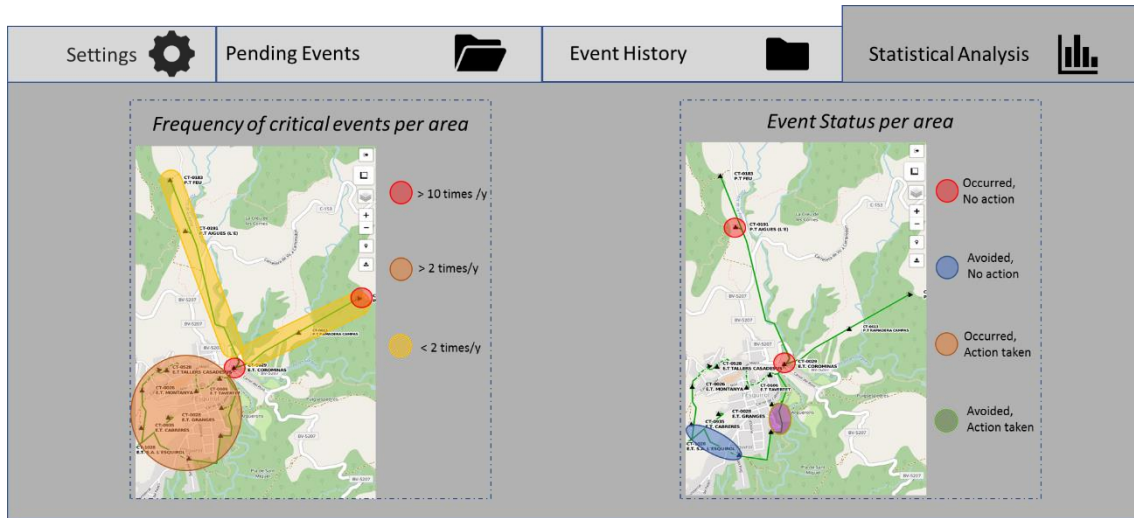


Figure 41 CEPA: Statistical Analysis map view (2 of 2)

Element	Type of element	Location	Frequency of CE (times/y)	Most frequent CE
CT 0528	Transformer MV/LV	L'Esquirol	12	Undervoltage
CT-0528 L2	Cable (d=x)	L'Esquirol	12	Undervoltage
CT 0030	Transformer MV/LV	L'Esquirol	11	Overvoltage
CT-0030 L2	Cable	L'Esquirol	10	Congestion Transformer
CT 0090	Transformer MV/LV	L'Esquirol	7	Undervoltage
CT-0030 L1	Cable	L'Esquirol	5	Congestion Transformer

Figure 42 CEPA: Statistical Analysis grid elements

Name	Location	No. of actions	Most frequent CE	Battery Usage (%)
B001	L'Esquirol	12	Undervoltage	
B002	L'Esquirol	12	Undervoltage	
B003	L'Esquirol	11	Overvoltage	

Name	Location	No. of actions	Most frequent CE
SW01	L'Esquirol	10	Congestion Transformer
SW02	L'Esquirol	7	Undervoltage
SW03	L'Esquirol	5	Congestion Transformer

Figure 43 CEPA: Statistical Analysis PED and Switchgear

7. Conclusions

This report presented the results of the design process performed in the context of tasks T4.1 “Enterprise Service Bus” and T4.2 “Data Analytics Platform”, also leveraging the outcomes of other deliverables of the project (D1.1, D1.2, D1.3 and D1.4). During this work, the main architectural elements of the RESOLVD platform were analysed. A top-level view of the functional decomposition of the platform was initially presented in order to provide the general context. Each architectural component’s design was further analysed, providing different architectural views, describing the interfaces, as well as the data design.

This work will serve as a basis for the final implementation of the components in the context of T4.1 “Enterprise Service Bus”, T4.2 “Data Analytics Platform” and Task 4.4 “Service integration”.

References

- [1] RESOLVD, D 1.2 - Functional and operational requirements.
- [2] RESOLVD, D1.3 – Interoperability and Integration Analysis and Requirements.
- [3] “<https://www.omg.org/spec/UML>”.
- [4] “IEC 61968-1:2012 Application integration at electric utilities - System interfaces for distribution management - Part 1: Interface architecture and general recommendations”.
- [5] EPRI, “Common Information Model Primer, Third Edition,” 2015.
- [6] [Online]. Available: https://wiki.openelectrical.org/index.php?title=Common_Information_Model.
- [7] IEC, “IEC 61968-100:2013 Application integration at electric utilities - System interfaces for distribution management - Part 100: Implementation profiles”.
- [8] [Online]. Available: <http://camel.apache.org/architecture.html>. [Accessed 09 2018].
- [9] B. W. Gregor Hohpe, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley.
- [10] [Online]. Available: <https://activemq.apache.org/>.
- [11] [Online]. Available: <https://www.jbpm.org/>.
- [12] “<https://www.omg.org/spec/BPMN/2.0/>,” [Online].
- [13] JBoss jBPM Team, “jBPM Documentation,” [Online]. Available: http://docs.jboss.org/jbpm/release/7.0.0.Beta3/jbpm-docs/html_single/. [Accessed 09 2018].
- [14] CIM University, “IEC 61968-100 Overview,” Austin, TX, 2011.
- [15] [Online]. Available: https://en.wikipedia.org/wiki/Big_data.
- [16] [Online]. Available: <https://en.wikipedia.org/wiki/Triplestore>.
- [17] [Online]. Available: https://en.wikipedia.org/wiki/Graph_database.
- [18] R. B. Melton et al., “Leveraging Standards to Create an Open Platform for the Development of Advanced Distribution Applications,” *IEEE Access*, vol. vol. 6, pp. pp. 37361-37370, 2018.
- [19] [Online]. Available: <https://www.w3.org/TR/rdf-sparql-query/>.
- [20] [Online]. Available: <https://hadoop.apache.org/>.
- [21] [Online]. Available: <https://hbase.apache.org/>.
- [22] [Online]. Available: <https://hive.apache.org/>.
- [23] [Online]. Available: <https://www.elastic.co/products/elasticsearch>.
- [24] [Online]. Available: <https://www.elastic.co/products/kibana>.

[25] [Online]. Available: <https://www.elastic.co/guide/en/kibana/current/dashboard.html>.

[26] [Online]. Available: <https://www.elastic.co/products/hadoop>.

[27] [Online]. Available: <https://leafletjs.com/>.