Title:	Document Version:
D2.1 Reference Architecture and Energy Services v1.0	1.0

Project Number:	Project Acronym:	Project Title:
FP7-60061-EEB-ICT-2011.6.5	SmartKYE	Smart Grid Key Neighbourhood Indicator Cockpit

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type*-Security*:
M10 (August 2013)	M10 (August 2013)	R-PU

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

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

Responsible:	Organisation:	Contributing WP:
José Javier García	BDigital	WP2

Authors (organization):

José Javier García (BDigital), Stamatis Karnouskos (SAP), Dejan Ilic (SAP), Wei Cheng (SAP), Rafael Peris (ETRA I+D), Lola Alacreu (ETRA I+D), Robert Sauter (UDE), Aris Dimeas (ICCS), Eleana Hatzoplaki (ICCS), Sergi Saldaña (TFLEX), Lucas Pons (ETRA I+D), Diego García-Casarrubios (ETRA I+D).

Abstract:

This document presents efforts carried out towards defining the SmartKYE reference architecture. Background information, definition process, methodology, detailed concepts and definition of services as well as examples of the API are presented. This document is intended to be the starting point for all development work that depends on the common architecture.

Keywords:

SOA, Energy Management, Web Services, Architecture

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose and Scope of the Document

The purpose of this document is to present a complete view of the SmartKYE architecture and energy services design process as well as the results obtained. In order to properly design SmartKYE architecture, it was first required to commonly share an understanding of the goals and scope of the architecture, as stated by the several use cases and requirements defined in Deliverable D1.1. Secondly, it was required to present and follow the use of well-known international standards and industry best practices that ensure a proper coordination of the different stakeholders during the design phase but also guarantee the quality of the resulting architecture and energy services. Hence, the purpose of this document is to present the results of all this process as well as settle the basis for the development work packages, which should be responsible for the SmartKYE development based on the architecture and energy services definition presented here.

This document is intended to provide a first complete version of SmartKYE architecture and energy services provided by the platform, detailing the standards and best practices followed during the design phase, as well as providing a detailed view of the information model and platform interfaces required to achieve the main goal of designing an open energy service platform.

The SmartKYE Architecture design will be an iterative process that will allow the definition of a first complete version of the architecture by M10 of the project (which results are presented in this deliverable D2.1). This version of the architecture will be the starting point for the development work packages, triggering the Energy Management Systems (EMS) adaptations as well as the development of the Open Energy Service Platform (OESP) and business and monitoring and control cockpits. Based on the feedback received from development WP, a second and final version of the architecture and energy services provided by SmartKYE platform will be completed and released in the deliverable document D2.2 by M24.

In summary, the scope of this deliverable is detailed as follows:

- Follow and ensure fulfilment of architectural requirements specified during the first WP of the project, where use cases and architecture requirements for the SmartKYE platform were defined.
- Provide a vision of the state of the art related to energy efficiency at a neighbour-hood and citywide level, identifying potential aspects of interest for SmartKYE plat-form development.
- Provide a complete vision and reference document of the SmartKYE open energy service platform and energy services provided.

1.2 Energy efficient neighbourhood – State of the art review

Future energy systems should be planned and operated as a holistic multi-energy system, taking into account all forms of energy. The energy systems to be operated at the neighbourhood or citywide level should take into consideration not only many energy production systems (centralised and decentralised), energy storage or energy distribution networks, but also business objectives and end-users needs and behaviour. In this sense, information and communication technologies (ICT) enable better management of the entire energy system at the neighbourhood level via optimisation and control algorithms, which improve economic aspects as well environmental impact, while providing better analysis tools for authorities and decision takers.

Thus, sharing a common vision and architecting such systems plays a key role in the design of future energy systems, as well as to guarantee interoperability and faster market adoption. So that, in this section we briefly review the vision of ICT enabled energy efficient neighbourhoods, as defined within the European project ICT4E2B [1] and IREEN [2], being the topics of latter closer to SmartKYE project. IREEN is an EC's 7th Framework Programme Coordination and Support Action project which aim is to extend the notion of energy positive buildings, districts and neighbourhoods. ICT4E2B and IREEN have already defined challenges, aspects to consider and potential directions with respect to the energy efficient buildings in the context of smart grid cities, many of which have already been considered when designing this architecture.

1.2.1 Principles and vision of ICT enabled energy efficient neighbourhoods

The energy efficiency of neighbourhoods is maximised by operating and planning the entire neighbourhood system as an optimal whole. This requires a holistic approach, including the entire energy chain. The vision is to plan and operate neighbourhoods as energy efficiently as possible in each local surrounding and circumstances. The key principles for achieving energy efficient neighbourhood are:

- Efficient energy production, distribution and storage
- Increased energy efficiency via better energy managemet
- ICT solutions as an enabler for better integration/interaction of energy stakeholders

ICT solutions certainly play a key role in the integration of the myriad of heterogeneous energy sub-systems of a neighbourhood into a holistically optimised and efficiently operated energy system. In this sense, within IREEN project it has been proposed a common vision for planning and management of neighbourhoods. The vision is for the wide availability of relevant data and ICT tools to support holistic decision making in neighbourhoods, which is in the end what is pursued with the development of SmartKYE platform.

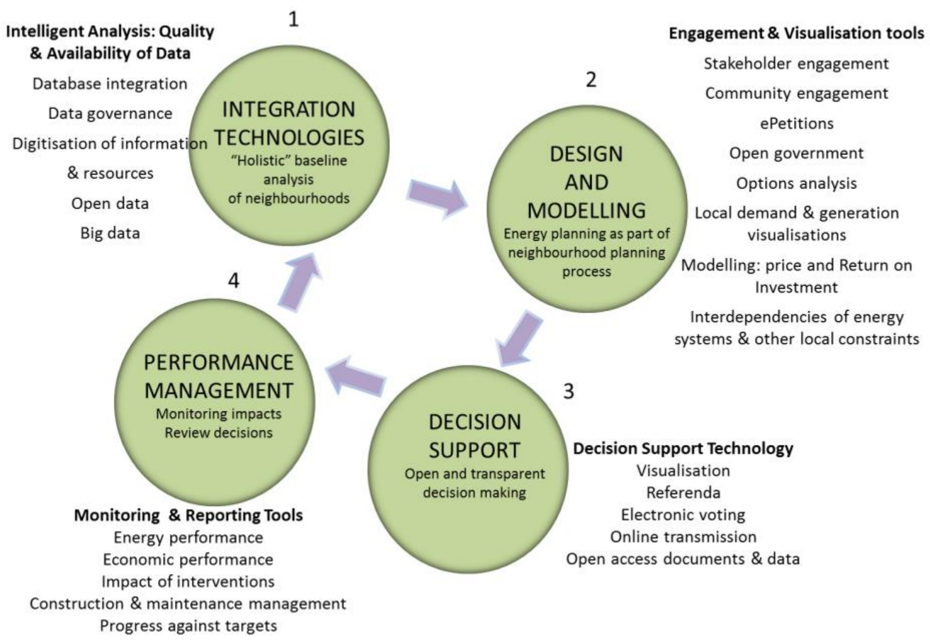

1
Intelligent Analysis: Quality & Availability of Data
Database integration
Data governance
Digitisation of information & resources
Open data
Big data
INTEGRATION TECHNOLOGIES
"Holistic" baseline analysis of neighbourhoods
2
Engagement & Visualisation tools
Stakeholder engagement
Community engagement
ePetitions
Open government
Options analysis
Local demand & generation visualisations
Modelling: price and Return on Investment
Interdependencies of energy systems & other local constraints
DESIGN AND MODELLING
Energy planning as part of neighbourhood planning process
3
DECISION SUPPORT
Open and transparent decision making
Decision Support Technology
Visualisation
Referenda
Electronic voting
Online transmission
Open access documents & data
4
PERFORMANCE MANAGEMENT
Monitoring impacts
Review decisions
Monitoring & Reporting Tools
Energy performance
Economic performance
Impact of interventions
Construction & maintenance management
Progress against targets

Figure 1 Neighbourhood planning process (Source: IREEN project).

Citizens, businesses and public authorities co-create and manage the data and the intelligent systems that enable energy efficient living. In general, neighbourhood plans are supported by advanced information derived from live systems. When it comes to energy efficiency planning, the process involves meeting and monitoring energy efficiency targets within a neighbourhood as part of wider legal and regulatory frameworks. ICT systems enable robust planning decisions based on optimising environmental and economic performance. Figure 1 summarizes the vision proposed.

All these principles and vision have been taken into consideration while defining the SmartKYE architecture. Furthermore, this analysis has been extended with a thorough state of the art review as well as the participation in workshops [3] where possible collaboration with other European projects has been identified. Results from this analysis are presented in the following section.

1.2.2 ICT enabled platforms for energy efficient neighbourhoods

As it can be seen from the previous section, ICT plays a key role in the integration of a myriad of heterogeneous energy systems, enabling easy information exchange between these systems. However, achieving this goal at a city level is challenging task. In this context, SmartKYE as well as several other European and international projects are defining and developing technological solutions that demonstrate the feasibility and benefits of facilitating information exchange between different systems at a neighbourhood and citywide level.

A neighbourhood or citywide platform that enables better business decisions to be made by public authorities should be more than the sum of the individual subsystems (already deployed and running in isolation across neighbourhoods and cities). In this sense, there are several common objectives that are considered not only by SmartKYE project but also by several other public and private initiatives detected. These objectives are summarized in the following list:

- **Open frameworks and platforms:** Dynamic nature of many systems and activities within a city imposes challenges to the way these systems are integrated in order to provide reliable and useful information to authorities. This should require a reliable, open and adaptable framework that could suit neighbourhood and citywide changing conditions.
- **Standardization:** This is a major challenge but at the same time a enabler for the development of open platforms that engage the most stakeholders possible. At the same time this could guarantee longer terms solutions lifecycles, making them more attractive to public authorities.
- **Generic** [4]: Any citywide platform should be capable to adapt to the continuous shifting of responsibilities amongst all stakeholders in a city, changing operational disciplines, functional areas, 3rd party processes, city policies, etc. In summary, the platform should offer generic functional building blocks.
- **Scalable solutions:** As any other system, a city wide platform should be ready to adapt to the increasing amount of information generated by current systems as well as the integration of some other future systems. This is also related to requirements such as decentralized energy management, security and reliability of the platform, stability, performance, etc.

Several European projects are developing either solutions related to energy efficiency at a neighbourhood level or to specific areas such as integration of renewable and distributed energy sources or integration of electric vehicle (EV). The following table includes a list of projects that have been considered of interest for SmartKYE project. In particular, the focus of this analysis is to gather information about the scope of such initiatives as well as related topics and potential areas of collaboration with SmartKYE.

Table 1 ICT enabled energy efficient neighbourhoods – State of the art overview

Project Acronym	Project Full Name	Related topics	Project Scope/Objectives
Adapt4EE	Occupant Aware, Intelligent and Adaptive Enterprises	Take profit of the experience and results obtained for the development of buildings energy management system. This could provide valuable insights for the integration of the buildings components in each of the pilot sites.	Adapt4EE aims at augmenting the contemporary architectural envelope by incorporating business and occupancy related information thus providing a holistic approach to the design and evaluation of the energy performance of construction products at an early stage and prior to their realization. At the same time, Adapt4EE aims to deliver and validate a holistic energy performance framework that incorporates architectural metadata and environmental parameters (BIM), critical business models (BPM), treating occupants as the central reference point.
Ambassador	Autonomous Management System Developed for Building and District Levels	Take profit of the approach followed for the integration of buildings within the energy management in districts as well as the demonstrations performed in three pilot sites.	The purpose of the AMBASSADOR project is to study, develop and experiment systems and tools that will aim at optimising the energy usage in a district by managing the energy flows, predicting and mastering energy consumption and energy production. The overall goal is to define and experiment a system that optimises the cost of energy in a district. In order to achieve this, the project proposes the development of real time adaptive and predictive behavioural models of buildings and districts.
BaaS	Building as a Service	Take profit and potentially collaborate in areas such as the design of a service middleware platform to abstract the building physical devices and support high level services on the cloud. It could also be relevant to follow the approach of this project towards the definition of energy models for performance estimation and control services.	The BaaS project aims to design and develop a generic ICT-enabled to optimize energy performance in non-residential buildings in operational stage. The main objective is to provide tools that enable an integrated assessment, prediction and optimization services that guarantee harmonious and parsimonious use of available resources.
BEAMS	Buildings Energy Advance Management System	Taking profit of the advantages of the Open interoperability gateway [5], as well as main results and conclusions from this project could be interesting for SmartKYE project public buildings integration.	BEAMS strategic goal is the development of an advanced, integrated management system which enables energy efficiency in buildings and special infrastructures from a holistic perspective. The project is developing an open interoperability gateway that will allow the management of diverse, heterogeneous sources and loads, and a Facility Management Environment to support that enables facility managers to take inform decisions with regards to energy efficiency in their facilities.

BeyWatch	Building EnergY Watcher	Take into consideration the developments of this project when it comes to energy management at a neighbourhood level.	The main objective of this project is to provide an interactive energy monitoring, intelligent control and power demand balancing solution at home and neighbourhood level.
ENERSIP	Energy Saving Information Platform for Generation and Consumption Networks	Take profit of service-oriented approach followed for ENERsip platform design.	The main objective of ENERsip project is to create an adaptive, customizable and **service-oriented energy monitoring and control platform** that allows end users to optimise, in near real-time, and to save energy by remotely monitoring, controlling and coordinating power generation and consumption in **neighbourhoods with residential and commercial buildings**.
ENRIMA	Energy Efficiency and Risk Management in Public Buildings	The DSS tools proposed in this project could be valuable for the integration of buildings and public facilities within SmartKYE platform, in the same way as in BEAMS project.	The purpose of this project is to develop a decision-support system (DSS) to enable operators to control energy flows in energy-efficient buildings and areas of public use by integrating management of conflicting goals such as minimising costs, improving energy efficiency, CO_2 emissions reduction, etc. This would be achieved by providing tools to enable long-term planning aimed at increasing energy efficiency, specifically analysis of retrofits and/or expansion of on-site energy sub-systems in order to meet certain goals, improving energy efficiency and sub-system integration in line with EU targets.
FINSENY	Future Internet for Smart Energy	Take profit of potential architectures, standards or simply best practices in the scenarios of interest for SmartKYE.	In the FINSENY project, key actors from the ICT and energy sectors will team-up to identify the ICT requirements of Smart Energy Systems. This will lead to the definition of new solutions and standards, verified in a large scale pan-European Smart Energy trial. Project results will contribute to the emergence of a sustainable Smart Energy infrastructure, based on new products and services, to the benefit of all European citizens and the environment.
GeM	Green eMotion - Development of a European Framework for Electromobility	Take profit of the open service-oriented platform and marketplace design. Although similar in concept to SmartKYE platform, Green Emotion is intended mainly EV services across Europe	One of the main goals of Green eMotion (GeM) is the definition and demonstration of the European Electric Vehicle Marketplace. That means the ICT needed for electric mobility processes and services required for an European wide EV services platform. By the open architecture, standardized interfaces and business objects (e.g., identification of charging points and contracts) it is ensured that all market participants can develop and commercialize their own services on the Marketplace.
Hesmos	ICT Platform for Holistic Energy Efficiency Simulation and Lifecycle Management Of Public Use FacilitieS	Take into consideration the building models and concepts defined, particularly putting especial emphasis in the concepts proposed regarding the integration of buildings and their surrounding spaces,	The main objective of Hesmos is to provide advanced simulation capabilities to decision makers in the whole life-cycle of buildings, taking into account energy savings, investment and life-cycle costs, by integrating a Virtual Laboratory to connect CAD and eeTools (energy efficiency Tools) in order to enhance building industry actor's ee-competences. In addition, the project will provide tolls that could allow closing the gap between Building Information Modelling (BIM) and Building Automation Systems (BAS) so that decisions can be made economically (energy & cost related) in all life-cycle phases, and even integrating surrounding areas extending current BIM to eeBIM

ICT4E2B	European stakeholders' forum crossing value and innovation chains to explore needs, challenges and opportunities in further research and integration of ICT systems for Energy Efficiency in Buildings	ICT4E2B [1] provides considerations, trends and roadblocks when dealing with energy efficiency in buildings, which is taken into consideration for the SmartKYE architecture.	ICT 4 E2B Forum is aimed at bringing together all relevant stakeholders involved in **ICT systems and solutions for Energy Efficiency in Buildings**, at identifying and reviewing the needs in terms of research and systems integration as well as at accelerating implementation and take-up. ICT4E2B intends to promote, through community building activities, a better understanding, a closer dialogue and a more active cooperation between researchers, end-users/practitioners, building owners, technology-suppliers, and software developers as regards the use of ICT to support informed decision-making. The ICT4E2B Forum project aims at the following objectives: - Bring together relevant stakeholders to identify and review the needs in terms of research and systems integration - Update the REEB research roadmap - Promote the use and further development of ICT for improved energy efficiency of buildings
IntUBE	Intelligent Use of Buildings' Energy Information	Take into consideration the developments of this project when integrating buildings energy management into the SmartKYE platform.	The business models and ICT tools developed in IntUBE will enable the multi-phase, multi-role management of buildings' energy information. In the IntUBE concept, the Energy Information Service Provider (EISP) will use different kind of information to provide customers with the required knowledge for making energy-efficient decisions related to the buildings they use, own or operate.
IREEN	IREEN – ICT for Energy Efficient Neighbourhoods	Participation in workshops to present project results as well as being aligned with the vision and guidelines derived from the discussions within the project.	IREEN is an EC's 7th Framework Programme Coordination and Support Action project which aims at engaging European and other international experts and stakeholders in discussions and workshops to gather their input in order to define a **common strategy for ICT based solutions for energy efficient in Europe**. The main objectives of the project will focus on the following aspects: - Building of a knowledge community adding value to existing ones (REEB, ICT4E2B Forum, etc.) for ICT-enabled energy-positive extensive areas. - Lead a community of experts towards identifying and updating barriers and challenges towards ICT solutions for energy efficient - Review and analysis of strategic innovation agendas as developed in previous or on-going EC coordination actions (REEB, Smart Grids, REViSITE, etc.), and relevant EU initiatives. - Development of an "Innovation roadmap".

NOBEL	Neighbourhood Oriented Brokerage ELectricity and monitoring system	Take profit of the results obtained in this project with regards to the development and validation of energy monitoring and control systems at a neighbourhood level as well as a platform to ease information flow among different stakeholders [6] [7].	Within the NOBEL project it was proposed and developed an energy brokerage system [8] [9] that enables individual energy consumers to communicate their energy needs directly with both large-scale and small-scale energy producers, thereby making energy use more efficient. The main scientific and technical objectives of this project were: - Dynamically obtain and process information from current available installed equipment. - Development of a service oriented framework [10] that will allow easy flow of information among the prosumers and the enterprise systems - Development of cooperation approaches [11] for all entities involved. This assumes cooperating objects at device level, at the energy brokerage system, at service level etc. - Development of a Neighbourhood Oriented Energy Monitoring and Control System, Energy brokering, and experimental energy efficient processes e.g., with the public lighting system.
PEBBLE	Positive-Energy Buildings through Better control decisions	Take profit of the results obtained for energy management of the buildings in each of the pilots of this project.	PEBBLE covers the aspects of building description, sensor and actuator device deployment in the building, sensor data, information flow between the Building Optimization and Control system and the deployed components, as well as the interactions with the users, mainly the occupants of the building.
REViSITE	Roadmap Enabling Vision and Strategy for ICT-enabled Energy Efficiency	Being up to date with results and recommendations derived from the project.	The main goal of REViSITE is to identify cross-sectorial research priorities covering the domains of grids, manufacturing, buildings and lighting, for Europe in the area of ICT for Energy Efficiency (ICT4EE).
SEEMPUBS	Smart Energy Efficient Middleware for Public Spaces	Take profit of the middleware platform to be developed for the integrated energy management of buildings and public spaces.	SEEMPubS specifically addresses reduction in energy usage and CO_2 footprint in existing Public buildings and Spaces without significant construction works, by an intelligent ICT-based service monitoring and managing the energy consumption. Special attention will be paid to historical buildings to avoid damage by extensive retrofitting.
SEMANCO	Semantic Tools for Carbon Reduction in Urban Planning	Take profit of the ontologies and data models generated within this project, as well as strategies and methods for CO_2 reduction that could be applied within SmartKYE context.	SEMANCO's purpose is to provide a semantic tool that enable different stakeholders in an urban environment to make informed decisions about how to reduce CO_2 emissions in cities. In order to achieve these goals, the project will focus on: - Structuring energy related data held in distributed sources and diverse formats - Classifying buildings for energy analysis - Visualising urban energy consumption - Assessing different methods of reducing CO_2 emissions - Predicting future energy demand - Providing appropriate energy indicators for local authorities

SmartHouse/ SmartGrid	Smart Houses Interacting with Smart Grids to achieve next-generation energy efficiency and sustainability	The SmartHouse / SmartGrid project validated and tested how ICT-enabled collaborative technical-commercial aggregations of Smart Houses provide an essential step to achieve the needed radically higher levels of energy efficiency in Europe.	SmartHouse/SmartGrid developed a holistic concept [12] [13] for Smart Houses situated and intelligently managed within their broader environment. Intelligent networked ICT technology for collaborative technical-commercial aggregations enables Smart Houses to communicate, interact and negotiate with both customers and energy devices in the local energy grid so as to achieve maximum overall energy efficiency as a whole.
URBGrade	Decision Support Tool for Retrofitting a District, Towards the District as a Service	It will be interesting to follow and find potential collaboration opportunities with this project due to both project share a common pilot site (Barcelona), and the “District as a Service” concept would be a concept for which SmartKYE platform would contribute.	The URBGrade project designs, develops and validates a Platform for Decision Support that will allow the city authorities and utilities to promote and choose the correct actions to upgrade a district to become more energy efficient, cost effective and to increase comfort for its citizens in a District as a Service Platform approach. The main objective of the project will focus on exploring the concept of “District as a Service”, and design, development and validation a platform based on needs and experience extracted from real end users.

2 SmartKYE Architecture Definition Approach

In order to provide a detailed service oriented architecture that enables the SmartKYE vision, as well as ensure the alignment with other European initiatives and industry standards, a careful review of the state of the art was performed, and briefly summarized in section one of this document. Based on this analysis, it was selected a well-known international industry standard as a guideline for the SmastKYE architecture definition.

In particular, it was selected “The Open Group Architecture Framework (TOGAF) version 9.1” [14] and in particular its Architecture Development Framework (ADM) and reference architecture “Open Group Standard SOA Reference Architecture (RA)” [15].

Considering as starting point the standard and reference architecture previously mentioned, it was defined a methodology adapted to SmartKYE project particular characteristics and needs. This section presents a brief overview of the main concepts related to the TOGAF framework and SOA RA of relevance for SmartKYE architecture definition.

2.1 Introduction

Nowadays, information is everywhere, but getting access to the right information at the right time can be very difficult and costly. In addition, the cost and complexity of IT systems have increased over time, as the demand for more sophisticated and complex solutions has increased as well. These problems are faced across many different fields, including energy management.

Furthermore, as proprietary solutions provide less differentiation and the growth and adoption of open standard-based solutions increases, it is necessary to adopt these standards and follow best industry practices when developing new solutions. While defining SmartKYE architecture, it has been important to consider all these aspects, as well as functional and non-functional requirements defined by project stakeholders in the first work package of the project. In this sense it has been possible on one hand to capture business goals defined by different stakeholders and potential SmartKYE platform users (namely public authorities), and on the other hand to translate these needs to proper building blocks of an open platform.

The main objective has been to leverage knowledge and best practices from well-known standards in order to design an open, scalable, flexible and secure platform that provides public authorities with the right tools to efficiently manage all energy aspects of their corresponding neighbourhoods or cities.

However, industry standards such as TOGAF can be quite abstract and generic, mainly in order to provide a common framework on which many different organizations can relish in order to define solutions targeting specific needs. This could be a major drawback for adopting such standard, although with proper adaptation to particular project needs, this standard provides the right tools to manage the whole lifecycle of an enterprise architecture definition. In addition, defining any kind of system architecture is a challenging task, not only for the architects but also for the overall stakeholders involved.

Although TOGAF could be used to define many different kinds of system architectures, the main focus within SmartKYE project will be the definition of a service-oriented architecture, even more leveraging the architecture definition on well-known reference architectures such as “the Open Group SOA RA”. This reference architecture provided a blueprint on which SmartKYE systems architects based the definition of the SOA architecture targeting particular project needs. Following sections review the methodology followed as well as the general concepts regarding the SOA RA used during SmartKYE architecture definition.

2.2 Architecture definition methodology - TOGAF

TOGAF is a detailed method and a set of supporting tools for developing enterprise architectures. In general, TOGAF allows following good practices that have evolved in the work of enterprise and IT architects over many years, and in particular it will help architects to decide where and how to use SOA concepts.

TOGAF reflects the structure and content of an architecture capability within an enterprise, as shown in Figure 2. Central to TOGAF is the Architecture Development Method (ADM), which is going to be analysed in more detail in following sections of this document. The ADM breaks the complex process of architecture development into a number of simpler steps, or phases, in which the architect considers different aspects of the overall problem.

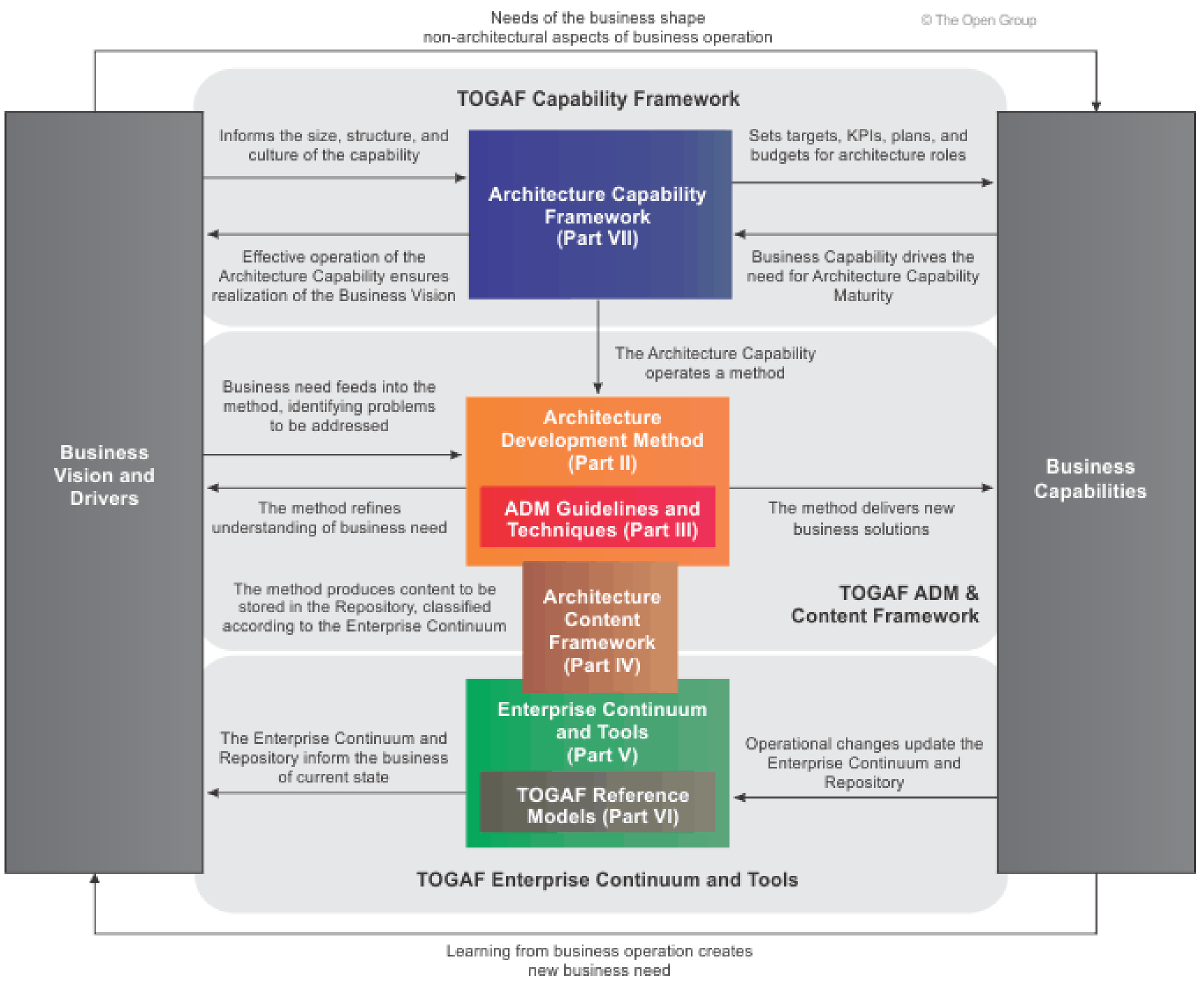

Needs of the business shape non-architectural aspects of business operation
© The Open Group
TOGAF Capability Framework
Informs the size, structure, and culture of the capability
Architecture Capability Framework (Part VII)
Sets targets, KPIs, plans, and budgets for architecture roles
Effective operation of the Architecture Capability ensures realization of the Business Vision
Business Capability drives the need for Architecture Capability Maturity
The Architecture Capability operates a method
Business need feeds into the method, identifying problems to be addressed
Architecture Development Method (Part II)
ADM Guidelines and Techniques (Part III)
The method refines understanding of business need
The method delivers new business solutions
Business Vision and Drivers
Business Capabilities
The method produces content to be stored in the Repository, classified according to the Enterprise Continuum
Architecture Content Framework (Part IV)
TOGAF ADM & Content Framework
Enterprise Continuum and Tools (Part V)
TOGAF Reference Models (Part VI)
The Enterprise Continuum and Repository inform the business of current state
Operational changes update the Enterprise Continuum and Repository
TOGAF Enterprise Continuum and Tools
Learning from business operation creates new business need

Figure 2 TOGAF Content Overview (Source: The Open Group)

TOGAF covers the development of four related types of architecture. These four types of architecture are commonly accepted as subsets of an overall enterprise architecture, all of which TOGAF is designed to support. They are shown in Table 2 [16].

Table 2 TOGAF Architecture Types Supported

Architecture Type	Description
Business Architecture	The business strategy, governance, organization, and key business processes.
Data Architecture	The structure of an organization's logical and physical data assets and data management resources.
Application Architecture	A blueprint for the individual applications to be deployed, their interactions, and their relationships to the core business processes of the organization.
Technology Architecture	The logical software and hardware capabilities that are required to support the deployment of business, data, and application services. This includes IT infrastructure, middleware, networks, communications, processing, and standards.

As it can be seen, the ADM is the major component of TOGAF and provides guidance for architects on a number of levels [16] as indicated in the following list:

- It provides a number of architecture development phases (Business Architecture, Information Systems Architectures, Technology Architecture) in a cycle, as an overall process template for architecture development activity.
- It provides a narrative of each architecture phase, describing the phase in terms of objectives, approach, inputs, steps, and outputs. The inputs and outputs sections provide a definition of the architecture content structure and deliverables (a detailed description of the phase inputs and phase outputs is given in the Architecture Content Framework).
- It provides cross-phase summaries that cover requirements management.

2.2.1 Architecture Development Method (ADM)

The architecture Development Method (ADM) is a step-by-step approach to developing an enterprise architecture. ADM describes how to derive an organization-specific enterprise architecture that addresses business requirements. In order to achieve these goals, architects must take into account the following key points about the ADM:

- The ADM is iterative, over the whole process, between phases, and within phases. For each iteration of the ADM, a fresh decision must be taken as to:
  - The breadth of coverage of the enterprise to be defined
  - The level of detail to be defined
  - The extent of the time period aimed at, including the number and extent of any inter mediate time periods
  - The architectural assets to be leveraged
- These decisions should be based on a practical assessment of resource and competence availability, and the value that can realistically be expected to accrue to the enterprise from the chosen scope of the architecture work.
- The ADM should be a generic method in order to to be used by enterprises in a wide variety of different geographies and applied in different vertical sectors/industry types. As such, it may be, but does not necessarily have to be, tailored to specific needs. .

As previously mentioned, the TOGAF Architecture Development Method (ADM) breaks the complex process of architecture development into a number of simpler steps, or phases, in which the architect considers different aspects of the overall problem. These phases are listed below and shown in Figure 3:

- Preliminary Phase
- Phase A: The Architecture Vision
- Phase B: The Business Architecture
- Phase C: The Information Systems Architectures (Applications and Data)
- Phase D: The Technology Architecture
- Phase E: Opportunities and Solutions
- Phase F: Migration Planning
- Phase G: Implementation Governance
- Phase H: Architecture Change Management

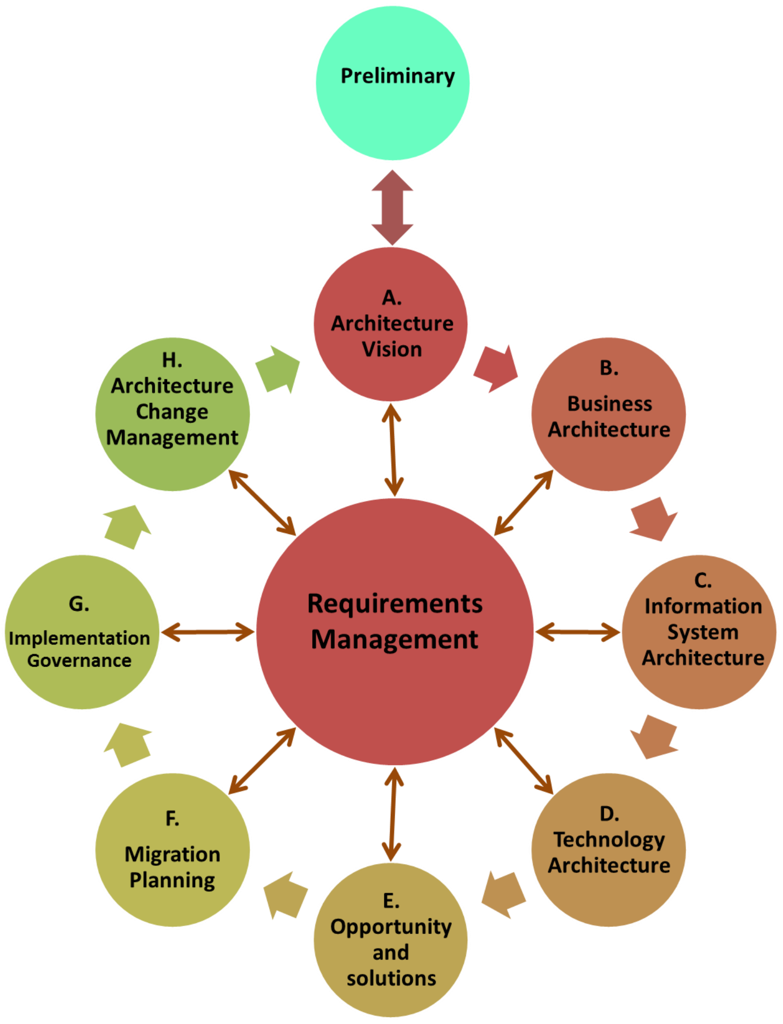

Preliminary
A. Architecture Vision
B. Business Architecture
C. Information System Architecture
D. Technology Architecture
E. Opportunity and solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

Figure 3 ADM phases (Source: The Open Group)

TOGAF provides for incremental architecture development. Each cycle through Phases A to H creates an increment to the enterprise architecture. The cycles might overlap, being Phases A to F of each new cycle being carried out in parallel with Phase G: Implementation Governance (of the previous cycle). Depending on the scope of the architecture to be developed, architects could focus more on some specific phases while leaving others for subsequent cycles of the architecture design.

Following, we provide a brief description of each phase of the ADM putting special emphasis in those phases and aspects that the architect should consider in particular when looking to apply the principle of service-orientation [17]. In addition, it will be indicated where applicable, those phases in which the SmartKYE architecture definition work focused on.

2.2.1.1 Preliminary phase

The TOGAF Preliminary Phase is about defining —where, what, why, who, and how an enterprise architecture is developed. It does the preparation and establishes the architecture framework needed for new enterprise architecture work. Within the scope of SOA architectures, the Preliminary Phase is where the architects adopt the principle of service-orientation. Within the scope of SmartKYE, this phase already started during the first work package of the project, where a common vision across the different project partners was achieved, defining specific roles and responsibilities within the project as well as use-cases and requirements, including architectural requirements.

2.2.1.2 Phase A – Architecture vision

This phase is concerned with establishing the architecture project and obtaining approval to proceed. This phase captures the scope of the architectural initiative, which depends on the nature of the enterprise and the level of detail of implementation specification. It creates a compelling vision of what the organization will have at end-of-job, and finally it identifies the key stakeholders, concerns, and business requirements.

Although this is particularly relevant in big organization or enterprise environments, within the scope of SmartKYE project it was also relevant to consider some aspects proposed for this phase. As in the previous phase, the specific tasks already specified for this phase were already started during the first work package of the project. However, within the scope of WP2, a thorough review of the results from WP1 was performed, extending those results by performing several workshops as well as consortium teleconferences that led to a common understanding and agreement among all project partners and stakeholders. The main results of this phase will be further explained along the remaining of this document.

2.2.1.3 Phase B – Business Architecture

The Business Architecture aligns the enterprise's business processes, people, operations, and projects with its overall strategy, providing a foundation on which to build the Information Systems Architectures and the Technology Architecture. This is the **first of the three TOGAF phases that produce detailed architecture descriptions** that will be the basis for the first SmartKYE architecture release, defined and presented along this document.

The different activities performed by all project partners were oriented to further define and narrow not only the scope of the project, but also the scope of the architecture. This helped to define specific business functionalities or capabilities of SmartKYE platform, and in particular of each of its components. The results of this phase were the starting point for the work and results in phases C and D of the ADM.

2.2.1.4 Phase C – Information Systems Architecture

According to the TOGAF ADM, the objectives of Phase C are to define the major types and sources of data necessary to support the business, and to define the major kinds of application system necessary to process the data and support the business. This was a major endeavour within the project since one of the strategic goals of SmartKYE is the design, development and validation of an open energy services platform.

Furthermore, this phase was developed within the scope of SmartKYE project taking into consideration SOA principles, since in the end SmartKYE architecture will be based in a set of loosely-coupled services which as a whole will become the open energy services platform envisioned.

2.2.1.5 Phase D – Technology Architecture

This phase seeks to map application components defined in the Applications Architecture phase into a set of technology components, which represent software and hardware components, available from the market or configured within the organization into technology platforms. For SOA, this means defining the software and hardware infrastructure needed to support the portfolio of services.

Phase D is the last of the three TOGAF phases (Phase B, C, & D) that produce detailed architecture descriptions required to complete the first release of the SmartKYE architecture. The starting point for the models that the architect develops in this phase is the set of key business requirements identified in Phase A plus the detailed and elaborated business requirements identified in Phase B and the information systems requirements identified in Phase C.

Although some relevant aspects of this phase have been discussed within the project consortium, the main results for this phase will be produced in work package WP3 of SmartKYE project, where the open energy services platform will be further refine and implemented.

The remaining phases within the TOGAF ADM will be briefly reviewed along the following section. However, no major efforts were dedicated to them within the scope of SmartKYE project.

2.2.1.6 Phase E – Opportunities and solutions

This phase identifies delivery vehicles (projects, programs, or portfolios) that effectively deliver the Target Architecture defined in previous phases. It reviews the target business objectives and capabilities, consolidates the gaps from Phases B to D, and organizes groups of building blocks to address these capabilities. It then generates an outline Implementation and Migration Strategy.

The identification of service and solution portfolios is a key task for SOA. The questions of what service and solution portfolios the enterprise will have, and how they will be managed, should be considered in this phase. A delivery option that should be considered particularly for SOA is the use of services provided by external companies, as opposed to the development of services in-house or the acquisition of software products that perform the services.

2.2.1.7 Phase F – Migration planning

This phase results in a detailed plan, produced in cooperation with departments responsible for concerned enterprise activities (such as the PMO, Operations, Sales and Production, Delivery, etc.), for the implementation of the architecture.

The implementation governance model is reviewed in Phase F in order to ensure that it is in place before the next phase – Implementation Governance – commences. SOA requires particular governance rules and procedures. The governance and support strategy is reviewed in the Preliminary Phase. If it needs to be updated for SOA, then this should be done before implementation starts. The architect should check in Phase F that the governance model fits for SOA, and ensure that it has been updated if necessary before proceeding to Phase G.

2.2.1.8 Phase G – Implementation governance

This phase involves participation of architects in implementation governance, to improve the quality of the implementations generally and in particular to ensure conformance with the architecture.

The activities performed in the Implementation Governance phase will depend in part on the decisions taken on the level of detail of implementation specification when the architect team scoped the architecture development in Phase A.

Although within SmartKYE some governance aspects were analysed, particularly from a technical perspective, no major efforts were devoted to this phase the development of WP2. In any case, the scope of this phase is of particular relevance when deploying the system commercially. In this sense, some of the issues related to this phase will be analysed during a whole work package related to exploitation plans of the project.

2.2.1.9 Phase H – Architecture Change Management

This phase is concerned with reviewing and updating the architecture and the architecture process itself. This includes assessing the performance of the architecture and making recommendations for change. SmartKYE project already adopted this approach when proposing two releases of the architecture, the first one presented in this document, and the second one at the end of the project, when particular feedback and insights gathered from the deployment and test phases will be taken into account to further refine the SmartKYE architecture.

2.2.2 Adapting the ADM to SmartKYE project

The TOGAF ADM is a generic method that defines a recommended sequence for the various phases and steps involved in developing an architecture. Although the ADM has been designed to deal with most systems and organization requirements, the scope has to be determined in each particular case by the architects or stakeholders involved in the architecture definition process.

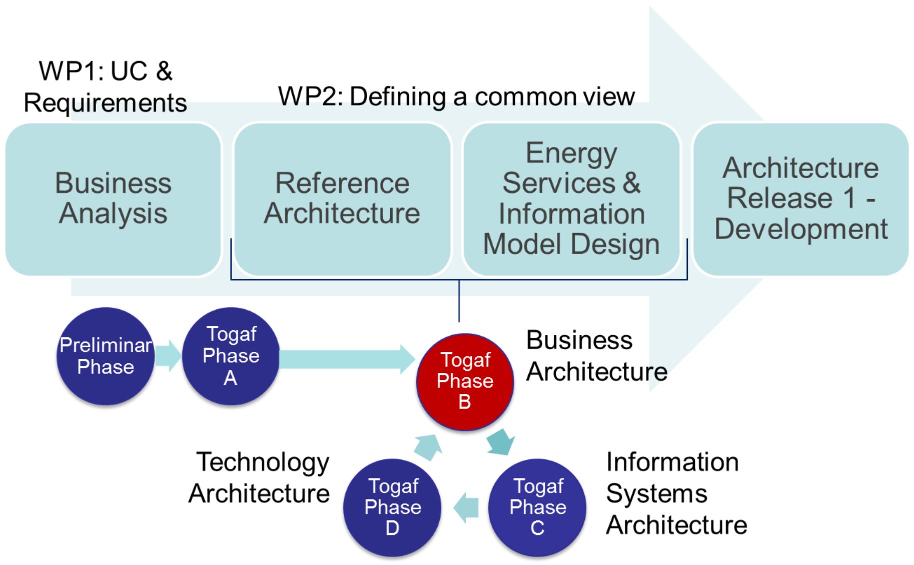


Figure 4 TOGAF ADM – SmartKYE adaptation

In addition, the iterative and flexible nature of the ADM allows architects to define the depth and breadth of the results obtained in each iteration of the process, as well as to tailor the ADM process itself to the particular circumstances and needs of an enterprise or project. As a result, a SmartKYE-specific ADM was proposed and used within the scope of WP2 activities. The process is illustrated in the following Figure:

As it is illustrated in Figure 4, the TOGAF ADM has been adapted, on the one hand to align some phases to the current project work packages, and on the other hand to focus project resources to those particular phases that would finally enable the definition and release of a first complete version of SmartKYE architecture, namely phases B, C and D.

Preliminary and Phase A of the ADM were developed in WP2 in parallel with activities in WP1 of the project. Once all project stakeholders were aligned, there were clearly defined the scope of the project as well as the scope of SmartKYE architecture. This was the basis for the SmartKYE architecture definition activities carried out within WP2 following the guidelines of TOGAF ADM Phases B, C and D.

The activities performed within Phases B, C and D follow also an iterative approach. In general, once a common agreement and concrete results were produced for a particular phase, project partners moved to the next phase, where additional aspects of the architecture were further defined. In some particular cases, some results from previous phases were further reviewed in order to provide an incremental update of the architecture.

As a result of the whole process, a first release of the SmartKYE architecture has been defined and presented in this document.

2.3 The Open Group SOA reference Architecture

TOGAF ADM provides the guidelines required to effectively define enterprise architectures, but in any case it is an architecture reference in itself. In order to assist architects in this task, a myriad of overlapping technical products and standard can be found, most of them developed by international organizations such as OASIS, OMG, and The Open Group.

In terms of SOA specifications, the “Navigating the SOA Open Standards Landscape Around Architecture” joint White Paper from OASIS, OMG, and The Open Group [18], provides an indeed analysis of the current SOA landscape scenario. This document explains and positions standards for SOA reference models, ontologies, reference architectures, maturity models, modelling languages, and standards work. The SmartKYE SOA RA selection was based on this analysis.

Within this context, The Open Group SOA Reference Architecture was selected due to it provides a layered architecture from a consumer and provider perspective with cross-cutting concerns describing those Architecture Building Blocks (ABBs) and principles that support the realizations of SOA. In addition, the combination of this SOA RA together with the TOGAF ADM provided the best tools set to enable a feasible SmartKYE architecture definition.

In summary, the Open Group SOA Reference Architecture Technical Standard provides some guidelines and options for making architectural, design, and implementation decisions in the implementation of solutions. The goal of this SOA RA is to provide a blueprint for creating or evaluating architectures, including templates and guidelines for enterprise and solution architects as well as software engineering roles within the software development life-cycle. As in the case of the TOGAF ADM, the guidelines and specifications provided by the SOA RA were closely followed within the context of SmartKYE project, although they are adapted or extended according to particular project needs.

Before presenting the SmartKYE architecture designed based on the concepts provided by the Open Group SOA RA, it is worth first to review the basic concepts provided in this technical standard.

2.3.1 The Open Group SOA Reference Architecture - Overview

The TOGAF SOA reference Architecture is a metamodel that defines a number of layers, building blocks, architectural and design decisions, patterns, options and the separation of concerns needed to design and evaluate an architecture. The RA is helpful for system architects to identify, specify and implement services, components and flows of an end-to-end solution that collectively support business processes or simply the achievement of an architecture business goals (in the context of a service oriented architecture).

Additionally, the SOA RA also acts as a communication vehicle for organizations to provide a high-level specification of what SOA components are and how to pick specific solutions, and a mechanism to align technology with business requirements.

The SOA RA is composed by a collection of layers, as illustrated in the Figure 5, where the logical solution view of the TOGAF SOA RA is shown:

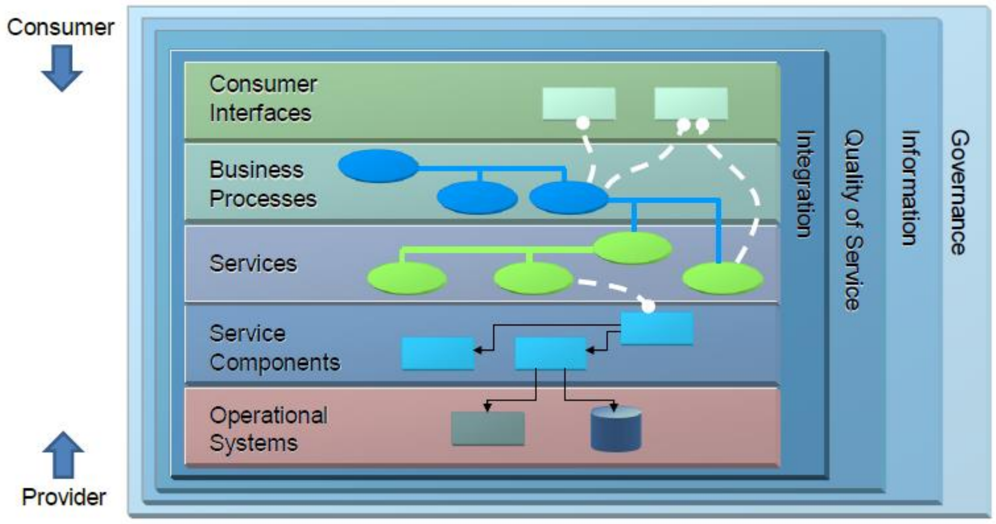


Figure 5 The Open Group SOA RA Logical Solution View (Source: The Open Group)

Each layer of the SOA RA provides a set of “capabilities” that are realized by a set of ABBs. The ABB or group of ABBs may be implemented in a target implementation platform or product by multiple vendors. This approach allows users of the SOA RA to look for a set of capabilities and assess or create an architecture that realizes those capabilities using a set of logical building blocks without tying those building blocks to a specific vendor implementation.

2.3.1.1 The Open Group SOA RA basic concepts

In order to fully take profit of all concepts and fundamentals presented by the SOA RA, it is important to understand the following concepts extensively used across the standard:

- **ABB (Architecture Building Block):** A constituent of the architecture model that describes a single logical aspect of the overall model. Each layer can be thought to contain a set of ABBs that define the key responsibilities of that layer. In addition, ABBs are connected to one another across layers and thus provide a natural definition of the association between layers. The particular connection between ABBs that recur consistently in order to solve certain classes of problems can be thought of as patterns of ABBs. Each ABB resides in a layer, supports capabilities, and has responsibilities. It contains attributes, dependencies, constraints, and relationships with other ABBs in the same layer or different layer.
- **Layer:** An abstraction of a grouping of a cohesive set of ABBs, architectural decisions, interactions among ABBs, and interactions among layers, that support a set of related capabilities.

- **Capability:** An ability that an organization, person, or system possesses to deliver a product or service. A capability represents a requirement or category of requirements that fulfil a strongly cohesive set of needs. This cohesive set of needs or functionality is summarized by name given to the capability.
- **Options:** A collection of possible choices available in each layer that impact other artefacts of a layer. Options are the basis for architectural decisions within and between layers, and have concrete standards, protocols, and potentially solutions associated with them. An example of an option would be choosing SOAP or REST-style SOA services since they are both viable options. The selected option leads to an architectural decision.
- **Architectural Decision:** A decision derived from the options. The architectural decision is driven by architectural requirements, and involves governance rules and standards, ABBs, KPIs, and Non-Functional Requirements (NFRs) to decide on standards and protocols to realize an instance of a particular logical ABB.
- **Interaction Pattern:** An abstraction of the various relationships between ABBs. This includes diagrams, patterns, pattern languages, and interaction protocols.
- **KPI (Key Performance Indicator):** A KPI may act as input to an architectural decision.
- **Enabling Technology:** A technical realization or instance of ABBs in a specific layer. Examples are web services or REST.
- **Information Model**: A structural model of the information associated with ABBs including information exchange between layers and external services. The information model includes the metadata about the information being exchanged.
- **Solution Building Block:** A runtime realization or instance of ABBs in a specific layer. A candidate physical solution for an ABB; e.g., a Commercial Off-The-Shelf (COTS) package such as a particular application server.

2.3.1.2 The Open Group SOA RA layers

As shown in Figure 5, the SOA Reference Architecture (SOA RA) has nine layers representing nine key clusters of considerations and responsibilities that typically emerge in the process of designing an SOA solution or defining an enterprise architecture. Each layer proposed in the standard is designed to correspond, reinforce and facilitate the realization of each of the various perspectives of SOA business value.

From the logical solution point of view of the SOA RA, the relevant layers for SmartKYE architecture definition are described in the following list. Three of the layers address the implementation and interface with a service (the Operational Systems Layer, the Service Component Layer, and the Services Layer). Three of them support the consumption of services (the Business Process Layer, the Consumer Layer, and the Integration Layer, although business process layer might be optional). Finally, four of them support cross-cutting concerns of a more supporting nature (the Information Layer, the Quality of Service Layer, the Integration Layer, and the Governance Layer).

A brief explanation of the main aspects of this layer is provided in the following list:

- **Consumer layer:** This layer provides access to services. It allows consumers to consume services either through a GUI or and API-like connection to the services. The consumer layer is responsible for invoking a service end-point. It accesses the services via the integration layer (or directly if integration layer is not developed in an architecture).
- **Integration layer:** It is a layer of choice through which services are invoked.
- **Service Component layer:** This layer contains software components, each of which provides the implementation or realization for services and their operations. It provides the ability to support the exposure of a service in a standards-compliant

manner (supporting interoperability) as well as exposing the service via an integration stack from the underlying platform in which the service functionality resides (aka within the Operational Systems Layer). Note that the protocol (SOAP/REST/J2EE, etc.) is not prescribed but determined by the associated architectural decision. The main capabilities this layer should support are:

   - Service Realization and Implementation: This category of capabilities supports the realization of the services.
   - Service Publication and Exposure: This category of capabilities supports service exposure and service contract publication (publish the service contract/descriptions in a standards-compliant, interoperable manner).
   - Service Deployment, invocation and binding

- **Service layer:** The Services Layer consists of all the services defined within the SOA. This layer can be thought of as containing the service descriptions for business capabilities and services together with their IT manifestation during design time, as well as service contract and descriptions that will be used at runtime. Since the definition of this layer is strongly related to TOGAF Phase B methodology, following sections will focus mostly on this layer as well as exploring in more details other layers relevant for the SmartKYE core services/components definition.
- **Governance layer:** It is the layer responsible for the services provisioning (among other tasks). Service provisioning means doing all tasks required to make services available to consumers to be able to invoke. For instance, provisioning can include updating the services registry, which contains the service metadata that consumers need to find, bind and invoke service. Logically, it is in this layer where services registry and repository are located, enabling in this way to manage, monitor, and administer the services registry and/or repository.
- **Operational system layer:** It provides the actual runtime for all layers of the architecture.

While using the SOA RA as a blueprint for the SmartKYE architecture definition, it will become clearer how the different components proposed in the standard interact among each other, and how the proposed SOA RA is adapted or even extended to accomplish the particular goals of SmartKYE project.

3 SmartKYE Architecture Vision

One of the SmartKYE project strategic goals is the design, development and validation of an open energy services platform that would empower decision makers at a neighbourhood or city level to take actions with the goal of energy efficiency and CO_2 reductions. In order to achieve this goal, SmartKYE platform should rely on different software components provided by different stakeholders and on valuable energy-related data coming from a myriad of heterogeneous systems in order to finally opening up the access to this information in a reliable and secure way.

The previous goal and its derived requirements should be reflected in a system architecture designed to achieve integration and interoperability between the different components of the architecture, potentially scattered across several different domains managed by different organizations. Service-oriented Architecture (SOA) principles followed as the basis of SmartKYE platform provide the design principles that help to address these issues. However, prior to actually designing the architecture, this section presents the common vision and architecture objectives agreed among the different stakeholders as well as the principles guiding its design. Finally, several aspects influencing the SmartKYE architecture such as stakeholders involved or systems to be integrated are presented.

3.1 Introduction

This chapter lays out the definition for important aspects driving the design and development of SmartKYE architecture. In this sense, it is important to review the main stakeholders involved in the SmartKYE architecture design, development and use, the main architectural requirements identified and finally the energy production and consumption systems that could be integrated within the platform. Due to the open nature of SmartKYE platform, previous aspects do not pretend to be a closed list of elements influencing the SmartKYE platform design, but some key elements driving it. The final result should be an open platform that many different stakeholders, even those not mentioned in this document can take profit of.

Finally, at the end of this sections are summarized the main principles driving the SmartKYE architecture design and the high-level architecture vision and objective. All these aspects influenced in some way or another some architectural decisions taken, so that it will be important to keep them in mind when reviewing technical aspects of the SmartKYE architecture in subsequent chapters of this document.

3.1.1 Stakeholders

SmartKYE platform will eliminate barriers for many stakeholders that would be able to access the platform, either to offer valuable energy-related data, or use it in new and innovative ways that assure an efficient use of energy resources and a reduction in CO_2 emissions in a neighbourhood or a city.

A thorough stakeholder analysis was performed in work package WP1 of SmartKYE project, an important step towards the definition on valuable use-cases and requirements driving the design and development of the platform. As indicated in the deliverable “D1.2 Use case specification and scenario analysis”, the European Technology Platform “SmartGrids Strategic Research Agenda 2035” document classifies the most important non-research stakeholders involved, as shown in the list below, where it is assumed certain new roles towards the year 2035. The future role of each stakeholder is, however, subject to research itself.

- **Municipalities:** This stakeholder is the main target user of SmartKYE platform. It comprises the administrative persons responsible for the various technical and business aspects of the municipality. This includes also planning, monitoring and management activities including the operation of all the different systems within a municipality (e.g., public buildings, public lighting, EV infrastructure, etc.). From the point of view of SmartKYE project, this stakeholder is seen as a human operator interacting with SmartKYE system through the two cockpits e.g., the BC as well as the MCC. In addition, municipalities could be in charge of managing SmartKYE platform, although this could also be done by other stakeholders.
- **Energy Management Systems (EMS):** These systems, deployed in the municipality or neighbourhood, are the actors that can be monitored and controlled by SmartKYE system. There can be many different energy systems that are already in place in a municipality (or would be in place in the short term) e.g., public buildings, public lighting, EV, wind farms, etc. From the point of view of SmartKYE project, specific stakeholders related to EMS systems can be:
    - Consumers or end-user: End-users of electricity e.g., public buildings, electric vehicles and public lighting systems, and potentially consumers with the additional role of self-provided (owned) electricity generation and/or storage.
    - Distributed Generators: Small- and medium-scale generation of mainly renewable based electricity either for third party consumers or for own consumption.

Finally, there are some additional stakeholders related to SmartKYE platform that could be considered as third-party stakeholders or actors that could take profit out of SmartKYE platform capabilities. A brief description of these stakeholders and their main objective within the scope of SmartKYE is provided below:

- **Electric Vehicle users:** For those scenarios where the integration of EV (hybrid or full electric vehicle) will be possible, an EV user will be considered as an actor that can interact with SmartKYE system. For example, EV user might interact with SmartKYE system in order to provide particular requirements regarding his EV charging process management (by SmartKYE). The users will be required to interface mobility needs with quality and security of supply needs of the electricity system.
- **Energy Retailers:** Selling energy and other (related) services and products to consumers. Retailers will develop consumer oriented programs and offerings.
- **Energy Service Companies (ESCOs):** Provision of a broad range of comprehensive energy solutions, including designs and implementation of energy savings projects, energy conservation, energy infrastructure outsourcing, power generation and energy supply and risk management.
- **Storage Providers:** Delivery of storage products and services, including their maintenance and operation thereby shifting electricity and energy consumption in time either for third parties or own purposes.
- **Ancillary Service Providers:** Provision of services such as Power Balancing, Voltage Profile Support and Blackstart
- **Distribution System Operators (DSOs):** Provision of services for secure, efficient and sustainable operation of electricity distribution systems. Legal obligation of a high quality, secure planning, operation and maintenance of the distribution grid.
- **Weather information/forecast providers**: This actor represents another external system interacting with SmartKYE system in order to provide updated weather information and/or weather forecast that can be used internally by SmartKYE to perform energy management.

- **Data processing service providers:** Provision of data processing services respecting consumer privacy.

3.1.2 Architectural requirements - Overview

Besides analysing SmartKYE stakeholders, work package WP1 also produced a comprehensive set of requirements for the whole SmartKYE platform components, analysed and presented in the deliverable “D1.1 Requirements Specification”.

Architectural requirements are an important element of the whole set of requirements provided in D1.1 document. Since these requirements have been one of the starting points for the work performed in work package WP2, they are briefly summarized in the following table.

It has been included an initial indication of the SmartKYE components where specific building blocks will provide the requirement capabilities to accomplish the specific requirements. In some cases, an architectural requirement can be accomplish through the interaction among many different architectural building blocks.

Table 3 D1.1 SmartKYE architectural requirements

Req. ID	Description	Architecture Building Blocks	Comments
ARC_001	SmartKYE reference architecture should be based on standards for SOA reference architectures	all	Covered
ARC_002	SMARTKYE reference architecture should support data exchange with AMR Systems	OESP	Applicable open interfaces defined
ARC_003	SMARTKYE reference architecture should support data exchange with DMS (Distribution Management Systems)	OESP	Applicable open interfaces defined
ARC_004	Support of DPWS (Devices Profile for Web Services)	EMS	Support not needed at system level; potential integration implications at EMS level to be investigated
ARC_005	The whole system should have the capability to receive measurements with different time intervals (1min, 5min, 10min, 15min, 30min, etc.)	EMS, OESP	The platform APIs enable the capability to consider/adapt to different requirements
ARC_006	Support of Standard Protocols	all	Considered web service interaction among all layers
ARC_007	SMARTKYE reference architecture should support data exchange with Enterprise System	OESP	Open interfaces defined
ARC_008	All critical systems should support redundancy	OESP	Distributed system approach with some explicit redundancy e.g. data owned by EMS is also stored

			in OESP
ARC_009	All systems should handle the quality indexes of tags	all	Integrated in open interfaces
ARC_010	The architecture should be expandable	all	Extensibility considered in the design
ARC_011	System Maintenance should be easy	OESP/ EMS	Considered for basic functionalities and integration
ARC_012	All Applications should be synchronized	all	Implicitly covered by time sync protocols
ARC_013	The system should support summer time changes	all	Implicitly covered
ARC_014	The Graphical User Interface (GUI) should be adapted easily to future system changes	BC, MCC	GUIs defined in a mash-up form, easily adaptable
ARC_015	The System should have emergency routines	OESP, MCC	Interactions among the EMS/OESP are possible via the service APIs
ARC_016	The system SHOULD allow defining calendars	MCC	Addressed in the MCC GUI
ARC_017	The system SHOULD allow having more than one situation detected as active at the same time	OESP, MCC	Integrated via grouping at the service API
ARC_018	Communication of energy related information and events at a resolution of at least 15 mins.	all	Addressed and defined in APIs
ARC_020	Quality of Information delivered by OESP services	OESP	Addressed and defined in APIs
ARC_021	Secure interactions among the SmartKYE parts	all	All communication will be done securely over *https.* Security service in place.
ARC_022	Sanity checks on data communicated should be done by all systems (EMS, OESP, MCC, BC ...)	EMS, OESP	Semantics defined in the service APIs with checks at EMS/ OESP level
ARC_023	The municipality should define which high level objectives it has in a measurable form and in relation to the KPIs in D1.2	MCC, BC	Integrated in the service API

3.2 Architectural drivers – Energy Management Systems (EMS) integration

Besides all those aspects already presented in this section, it was important within work package WP2 to analyse in detail the characteristics and data provided by the EMS systems that will be integrated in SmartKYE platform (in each pilot site within the context of the project). This task will establish the basis for the adaptation of each EMS system, helping in the definition of the open interfaces and information models required to achieve the SmartKYE open energy service platform objectives.

Within the context of SmartKYE project, two pilot sites will be available for demonstration of the capabilities of SmartKYE platform. In each of these pilot sites different EMS systems will be integrated. In general, the following EMS systems have been considered in SmartKYE, although this is not a definitive list of systems supported by SmartKYE, since the open nature of this platform would enable the integration of many other EMS systems.

- Wind power plants
- District public facilities (public buildings)
- Public lighting system (PLS)
- Electric vehicle infrastructure

In the next sections, a brief description of each of these EMS systems is provided.

3.2.1 Wind Power Plants

Wind power plants EMS will be integrated in both pilot sites in SmartKYE project (Barcelona and Crete), although the characteristics of these systems in each case will be different. In the annex section of this document detailed information regarding the data available for each for these systems is provided.

3.2.2 Public buildings

Besides power production systems, several different power consumption systems will be integrated in SmartKYE platform, mainly public buildings from both pilot sites. As it could be expected the characteristics of both systems are different. The main data available from these systems is summarized and presented in the annex section of this document.

3.2.3 Public lighting system

The public lighting system EMS integrated within SmartKYE will be able to provide data about the following elements:

- Physical elements:
  - Segment Controller: Equipment that physically controls lighting lines.
  - Light Controller: Equipment that controls a point of light.
- Logical elements:
  - Point of light
  - Group of points of light: Entity that controls various points of light regardless of its physical connection.

A detailed list of the data and services available for each of the elements listed above is presented in the annex section of this document.

3.2.4 Electric vehicle infrastructure

Similar to the case of public lighting systems, when it comes to EV EMS, it will be able to provide data about the following elements:

- Physical elements:
  - EV charging station plug
- Logical elements:
  - Charging station (set of plugs)

Data available for each of these elements is detailed in the annex section of this document.

3.3 SmartKYE architecture vision

In general, an architecture vision is created early on in a project lifecycle in order to provide a high-level, aspirational view of the end architecture of the solution to be implemented. An architecture vision serves also to reach an agreement among different stakeholders about what the desired outcome should be for the architecture. SmartKYE architecture vision has been developed through extensive consultation and discussions among the different project stakeholders as well as considering potential end users of the platform.

The vision behind SmartKYE platform, and concretely through its architecture design is to enable improvements in the availability and quality of energy-related data that enable decision takers in a municipality or even at city-wide level, to take informed decisions that lead to a more efficient use of energy resources, contributing in this sense to a reduction in CO_2 emissions.

By using SOA concepts to develop a federated platform, the basic idea is providing the common infrastructure to enable many different stakeholders to interact in a transparent way, exchanging valuable energy that could lead to the development of innovative energy efficiency strategies (by public administrations) or even new solutions and services (by private organizations) in a neighbourhood or city. This vision is better illustrated in the following Figure:

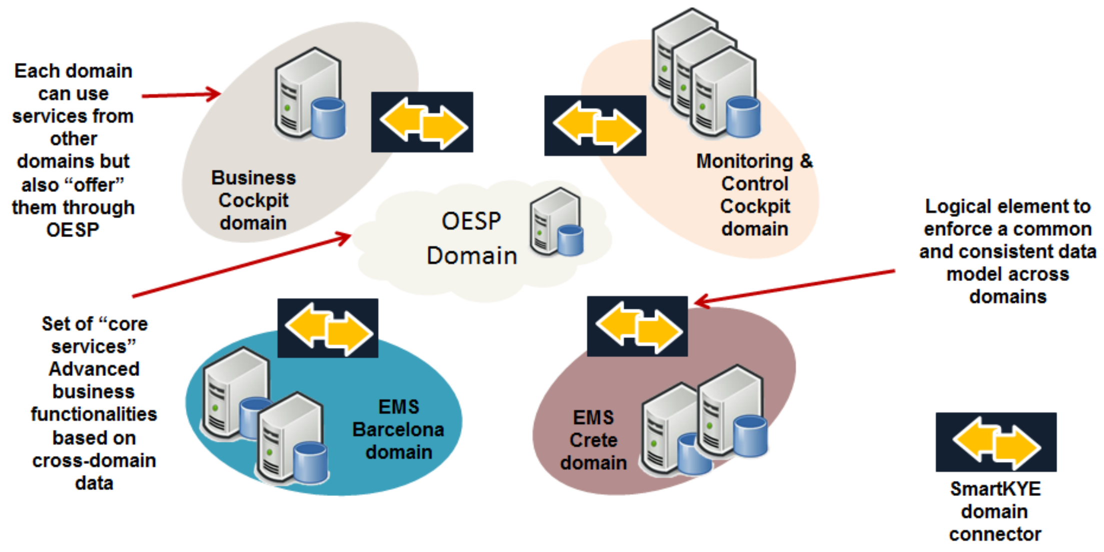

Each domain can use services from other domains but also "offer" them through OESP
Business Cockpit domain
Monitoring & Control Cockpit domain
OESP Domain
Logical element to enforce a common and consistent data model across domains
Set of "core services" Advanced business functionalities based on cross-domain data
EMS Barcelona domain
EMS Crete domain
SmartKYE domain connector

Figure 6 Example of SmartKYE independent system interaction view via OESP

As it can be seen, SmartKYE architecture vision is based on the concept of a federated platform, were several components of the architecture could be scattered across different domains potentially managed by different stakeholders, although providing common functionalities used by other components of the architecture. It is also relevant to highlight that besides following a federated approach, SmartKYE platform still provide core functionalities through the Open Energy Service Platform (OESP) component, a core element whose main objective is to guarantee the operation of the overall platform. Third parties will be able to access SmartKYE platform through a "SmartKYE domain connector" Software component, whose main function is to adapt existing or legacy systems in a particular domain, to the common and open SmartKYE interfaces. From an implementation perspective, this component will be based on the SmartKYE service interfaces (e.g., the OESP services and their integration to the other systems e.g. EMS, MCC, BC) explained in subsequent chapters of this document.

In summary, SmartKYE vision is to create an open energy service platform that will provide the right tools to empower decision takers in public administrations to monitor and manage in a more efficient way the energy resources available in municipalities and even in our cities.

3.4 SmartKYE architectural principles

Architecture principles define the underlying general rules and guidelines for designing the architecture. These principles describe the approach or strategy chosen to solve certain non-functional requirements as qualities or constraints. In this sense, the TOGAF ADM makes the step of establishing architecture principles fairly straightforward, which are defined as part of the Preliminary Phase, and finally agreed from a governance perspective, and used to guide and direct the organization on all future architectural decisions.

The TOGAF ADM provides a set of architecture principles, arranged in the following in five principal categories [19]:

- Business Principles
- Data Principles
- Application Principles

3.4.1 Business Principles

Principle:	*Service orientation*
Statement:	The architecture will be based on the design of integration services which mirror real world business activities

Principle:	*Minimize costs*
Statement:	The main objective will be to minimize application development costs by taking profit of developments already available

Principle:	*Maintenance requirements*
Statement:	Services must be coherent and somewhat independent components of the overall system. This principle guarantees lower maintenance costs

Principle:	*Single responsibility principle*
Statement:	Each component or module should be responsible for only a specific feature or functionality, or aggregation of cohesive functionality

Principle:	*Reusability*
Statement:	No duplicate functionality. There should be only one component providing a specific functionality. This functionality should not be duplicated in any other component as far as possible. This makes components cohesive and makes it easier to optimize the components if a specific feature or functionality changes. Components and subsystems should be suitable for use in other applications and in other scenarios. Reusability minimizes the duplication of components and also the implementation time. Minimizing the number of applications used to perform similar services reduces confusion and improves productivity

Principle:	*Interoperability*
Statement:	All subsystems should have the ability to operate successfully by communicating and exchanging information with other external subsystems written and run by external parties

Principle:	*Semantic interoperability*
Statement:	This is concerned to ensure that the precise meaning of exchanged information is understandable by any person or application receiving the data. The architecture must allow different subsystems to effectively exchange data, combine it with other information resources, and subsequently process it in a meaningful manner. To achieve this, agreement is required on the context and precise meaning of the exchanged data

Principle:	*Available Anytime from Anywhere*
Statement:	Access must be available to those entitled to it, in a timely manner regardless of where they are or what time it is.

Principle:	*Sharing of Information*
Statement:	SmartKYE platform will facilitate sharing and use of knowledge (energy-related) throughout stakeholders.

3.4.2 Data principles

Principle:	*Data is shared*
Statement:	Data is shared across different services. Data must be available to those services that require these data to perform their duties. It is less costly to maintain timely, accurate data in a single application, and then share it, than it is to maintain duplicative data in multiple applications

Principle:	*Data is accessible / Boundaryless Information Flow*
Statement:	Required data is accessible for users (services) to perform their functions. Accessibility involves the ease with which users obtain information. Access to data does not constitute understanding of the data. According to the Open Group, Boundaryless Information Flow concept should strive to eliminate stovepipes by promoting an architecture that allows the sharing of data across systems.

Principle:	*Common vocabulary and data definitions*
Statement:	The data that will be used in the development of applications must have a common definition to enable sharing of data. A common vocabulary will facilitate communications and enable dialog to be effective.

Principle:	*Data security*
Statement:	Data should be protected from unauthorized use and disclosure. Data owners will take care of safeguarding the privacy of data according to existing laws and regulations of national security.

3.4.3 Application Principles

Principle:	*Technology independence*
Statement:	Front-end applications are platform independent and therefore can operate on a variety of technology platforms. Independence of applications from the underlying technology allows applications to be widely used and increase user acceptance. The intent of this principle is to ensure that application front-end is not dependent on specific hardware and operating systems software

Principle:	*Promote usability*
Statement:	An easy to manage GUI that integrates information coming from different sources showing useful recommendations and offering services from different parties could allow users to save time and money and increase his/her satisfaction.

Principle:	*Ease-of-use*
Statement:	Applications should be intuitive and easy to use. The underlying technology must be transparent. The more a user has to understand the underlying technology, the less productive that user is. Training will be kept to a minimum in order that the risk of using the system improperly is low.

4 SmartKYE Reference Architecture Concept

Following the SmartKYE architecture vision presented in previous section, and taking into consideration the SOA Reference Architecture provided by The Open Group, a SmartKYE RA was defined. Due to the architecture definition is a dynamic and iterative process, within SmartKYE project two main releases of the architecture has been defined. The results obtained for the first version of SmartKYE RA is presented along this section. The second and final release of the SmartKYE architecture will be delivered by the end of the project and presented in a second project deliverable "Reference Architecture and Energy Service Specification – Final Version".

4.1 Introduction

SmartKYE SOA RA will be described in detail from different perspectives. Along this chapter is presented a conceptual view, where different building blocks of the architecture are identified and their corresponding capabilities or functionalities are defined. The concepts presented will follow The Open Group SOA RA, so that the building blocks identified will be mapped to the different logical layers of SOA RA.

In order to ease the reading and understanding of the concepts presented in this section, Figure 7 presents a relation between all these concepts. Capabilities as defined by The Open Group represent "an ability that an organization or system possesses" (or will possess), and their main advantage while designing an architecture is that capabilities allow architects and stakeholders involved to focus the process on the "what" rather than on the "how". Capabilities are then expressed in functional and non-functional needs or requirements that guide and constraint the architecture.

The layers in the SOA RA provide a convenient means of consolidating and categorizing the various capabilities and building blocks that are required to implement a the SOA architecture designed.

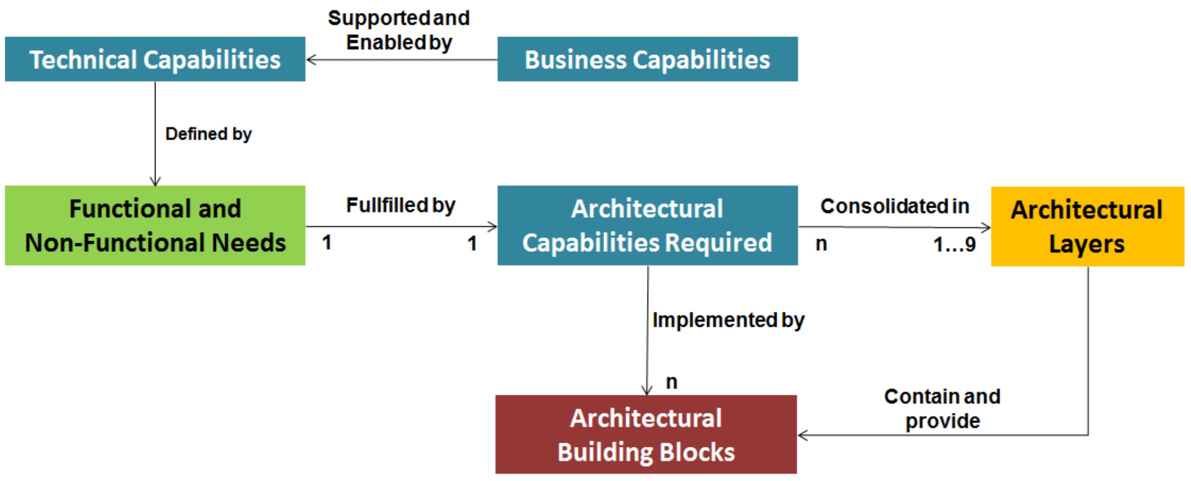


Figure 7 Relationships among Requirements, Capabilities, Building Blocks, and Layers (Source: The Open Group)

From a logical perspective, the building blocks in each layer will interact among each other in order to provide any given business capability. A typical interaction flow between the different layers in the SOA RA provided by The Open Group is presented in Figure 8. Not all layers are required in the development of an architecture e.g., Integration layer could interact with the Business Process layer or directly with the Services layer if the former one is not supported (as it is the case in SmartKYE project).

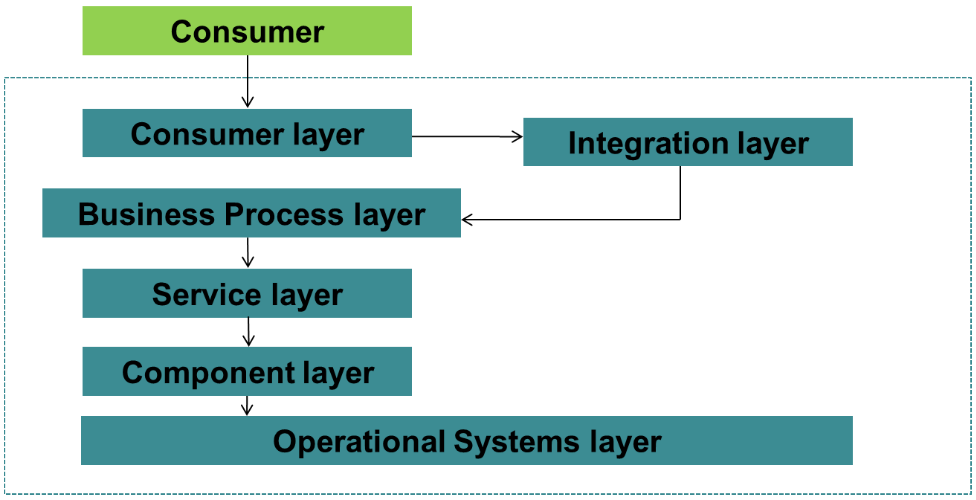


Figure 8 Typical Interactions among the Layers of the SOA RA (Source: The Open Group)

Following section will describe in greater detail SmartKYE architecture layers, and derived capabilities and building blocks. Along the analysis of the capabilities of each layer of the architecture, it is assigned a number to every single capability to be supported by each layer. For the sake of simplicity, following sections list only those capabilities supported by the corresponding layer within the context of SmartKYE project. Each layer can provide or support more capabilities, which can be review in the The Open Group SOA RA document [15].

Finally, in subsequent chapters of this document the SmartKYE architecture vision is complemented by providing detailed explanations on the information model and energy services interfaces. Overall, these components will be the starting point for the development of the architecture.

4.2 SmartKYE Reference Architecture – Horizontal or Functional Layers

Figure 5 illustrates the logical view of The Open Group SOA RA, where there are five horizontal layers, which are more related to the functionality of the architecture, and four crosscutting or vertical layers more related to supportive functionalities.

This section presents the analysis performed for those horizontal layers of relevance for SmartKYE project. In particular, consumer layer will be briefly reviewed while the main focus will be put in the service layer. Lower layers such as Service Components and Operational Systems layers will not be covered since WP3 of the project fully is focused on implementation issues of the architecture.

For each particular layer it is presented a list of capabilities organized in different categories as it is stated in the standard. Along the analysis presented for each layer, only those categories and corresponding capabilities of relevance for SmartKYE will be presented. Numbers associated to each capability correspond to the number of that capability as stated in The Open Group SOA RA.

4.2.1 SmartKYE Consumer Layer

The consumer layer is the point where consumers interact with the architecture. It enables an SOA to support a client-independent, channel-agnostic set of functionality, which is separately consumed and rendered through one or more channels (client platforms and devices). Thus, it is the point of entry for interactive consumers (humans and other applications/systems) and services from external sources such as Business-to-Business (B2B) scenarios. In the case of SmartKYE project, such capabilities will be offered by the different cockpits envisioned where applicable.

Consumer layer capabilities:

There are multiple categories of capabilities that the consumer layer needs to support. The following list summarizes these capabilities.

Table 4 Consumer Layer Capabilities

Consumer Services.
2. Ability to support consumer interaction and integration; e.g.,, the ability to capture the input from the user (consumer) of the SOA and provide the response to the consumer
Presentation Services
3. Ability to support the creation of a presentation view by the composition of a number of atomic components
4. Ability to configure information which will support specific capabilities associated with ensuring consistency (similar to a style guide)
5. Ability to provide navigation logic and flow for the processing of consumer interactions (presentation control)
6. Ability to provide the Consumer layer with the ability to support customer-specific information (enabled by the Information Layer) and personalization and customer-specific preferences to be used by the presentation controller for navigation and content presentation purposes
Backend Integration
8. Ability to mediate services from other SOA layers such as the Business Process Layer and the Integration Layer into the Consumer Layer; it provides the ability to integrate the underlying SOA into the Consumer Layer
9. Ability to support the translation of input data/content from a format supported by the user of the SOA to a format required by the other layers of the SOA and to convert content returned from them into a user acceptable response format
Caching and Streaming Content
10. Ability includes the handling of streaming content
11. Ability to cache interaction data to improve performance and quality
Security and Privacy
12. Ability to provide access to authentication/authorization capabilities (enabled through policies) to be used by the presentation controller to allow/prevent what content can be presented to the consumer
13. Ability to filter to control access to the underlying SOA
14. Ability to monitor the usage of the Consumer Layer components

Information Access
8. Ability to access data and metadata through the Information Layer

Consumer layer capabilities and building blocks mapping:

The mapping between the capabilities previously listed and the corresponding building blocks of SmartKYE architecture is presented in the Annex section of this document.

4.2.2 SmartKYE Service Layer

The Services Layer consists of all the services defined within the SOA. The Services Layer is the layer of the SOA, which **describes functional capabilities of the services** in the SOA. The Services Layer introduces the notion of **services, which are well-defined interfaces for a capability** of the architecture.

This layer primarily provides support for services, from a design-time perspective. It defines runtime capabilities for service deployment, but the runtime instantiation of the building blocks enabling these capabilities are housed in the Operational Systems Layer. It also provides the service contract elements that can be created at design time to support subsequent runtime requirements.

Service specifications provide consumers with sufficient detail to locate and invoke the business functions exposed by a provider of the service. Ideally, this is done in a platform-independent manner. The main responsibilities of the Services Layer include:

- To identify and define services
- To provide a container which houses the services
- To provide a registry that virtualizes runtime service access
- To provide a repository to house and maintain service design-time information

Service layer capabilities:

There are multiple categories of capabilities that the services layer needs to support. The following list and table summarizes these capabilities:

Table 5 Service Layer Capabilities

Service Definition
1. Ability to define services in terms of service descriptions/contracts
Service Runtime Enablement
3. Ability to enable the service container and the service registry to manage the storage and invocation of different services with minimal impact to users of the SOA
4. Ability to interact with other layers within the SOA RA, particularly the Integration Layer
5. Ability to define the binding to the Service Component that implements a given service
6. Ability to support the hosting of services
Access Control
12. Ability to support the integration of the security access control descriptions for services with the runtime elements of the Governance and Quality of Service Layers of the SOA RA

Service layer capabilities and building blocks mapping:

The mapping between the capabilities previously listed and the corresponding building blocks of SmartKYE architecture is presented in the Annex section of this document.

The core elements in this section are the service definition capability (capability #1 in Table 5) and the corresponding service building block. Although represented only as one building block. This will be translated as several different SmartKYE services, which actually constitute the interface of the platform. The specification of these services is further explained in following chapters of this document.

4.3 SmartKYE Reference Architecture – Cross-cutting or Supportive Layers

This section focus on the analysis of the SOA RA crosscutting layers responsible for supporting other layers in the architecture (horizontal layers), and in consequence certain capabilities required for the overall architecture.

4.3.1 SmartKYE Integration Layer

The integration layer is a key enabler for an SOA as it provides the capability to mediate, which includes, transformation, routing, and protocol conversion to transport service requests from the service requester to the correct service provider.

This layer enables the service consumer/requester to connect to the correct service provider through the introduction of a reliable set of capabilities. The integration can start with modest point-to-point capabilities for tightly-coupled end-points and cover the spectrum to a set of much more intelligent routing, protocol conversion, and other transformation mechanisms often described as, but not limited to, an Enterprise Service Bus (ESB). WSDL specifies a binding, which implies location where a service is provided, and is one of the mechanisms to define a service contract. An ESB, on the other hand, provides a location-independent mechanism for integration, and service substitution or virtualization.

Integration layer capabilities:

There are multiple categories of capabilities that the integration layer needs to support in the SOA RA. The following list and table summarizes these capabilities:

Table 6 Integration Layer Capabilities

Communication, Service Interaction, and Integration
1. Ability to take a service call and messages to the end-point; e.g.,, to enable a service consumer to connect/interact with service providers
2. Ability to handle service request and service response
3. Ability to support communication through a variety of protocols
4. Ability to support variety of messaging styles such as one-way, pub-sub, request-response
5. Ability to route messages to the correct service provider
6. Ability to transform protocol formats; e.g., from SOAP/HTTP to SOAP/Message Queue or SOAP/JMS
7. Ability to link a variety of systems that do not directly support service-style interactions so that a variety of services can be offered in a heterogeneous environment
Quality of Service
13. Ability to handle transactions from the other layers, especially when a statically

composed service invokes a service chain 14. Ability to handle exceptions raised in the process of service invocation and message passing
Security
15. Ability to authenticate/authorize for service invocation and message routing

Integration layer capabilities and building blocks mapping:

The mapping between the capabilities previously listed and the corresponding building blocks of SmartKYE architecture is presented in the Annex section of this document.

4.3.2 SmartKYE Quality of Service layer

The key responsibilities of the quality of service layer include:

- Monitoring and management both at the business level (e.g.,, business processes), and at the IT systems level for the security, health, and wellbeing of IT systems, services, applications, networks, storage, and compute servers
- Monitoring and enforcement of a multitude of policies and corresponding business rules including business-level policies, security policies, access privileges, data access policies, etc.

QoS layer capabilities:

There are multiple categories of capabilities that the QoS layer needs to support. Although The Open Group provides an extensive list of QoS capabilities (more than 80) that any SOA could support, within the context of SmartKYE project only a few of them will be consider and finally implemented. In case the platform moves to a commercial stage, additional QoS building blocks might be required to be added to the architecture.

The following list summarizes the QoS layer capabilities considered in SmartKYE:

Table 7 QoS Layer Capabilities

Security Management:
This category of capabilities provides the ability to manage roles and identities, access rights and entitlements, protect unstructured and structured data from unauthorized access and data loss. Within this category, the main capabilities considered are:
9. Ability to ensure appropriate authentication based on proper roles
10. Ability to ensure appropriate authorization based on proper roles
11. Ability to ensure appropriate encryption of messages
13. Ability to assure that access to resources has been given to the right identities, at the right time, for the right purpose
15. Ability to protect unstructured and structured data from unauthorized access and data loss, according to the nature and business value of information
17. Ability to address how software, systems, and services are designed, developed, tested, operated, and maintained throughout the software lifecycle including the use of technology as well as processes and procedures which are followed during all aspects of software development and deployment
21. Ability to provide and enforce policies for access control

QoS layer capabilities and building blocks mapping:

The mapping between the capabilities previously listed and the corresponding building blocks of SmartKYE architecture is presented in the Annex section of this document.

4.3.3 SmartKYE Information Layer

The information layer is responsible for manifesting a unified representation of the information aspect of an organization as provided by its IT services, applications, and systems enabling business needs and processes and aligned with the business vocabulary.

In particular, this layer can be thought of as supporting multiple categories of capabilities of the SOA RA:

- **Ability to support information services capability, critical to support a shared, common and consistent expression of data**
- Ability to integrate information across the enterprise in order to enable information services capability
- Ability to define metadata that is used across the SOA RA and in particular the metadata that is shared across the layers
- **Ability to secure and protect information**
- Ability to support business activity monitoring and critical to the usage of the SOA RA and its realization

In particular, an **information virtualization and information service capability typically involves the ability to retrieve data from different sources, transform it into a common format, and expose it to consumers** using different protocols and formats.

Information layer capabilities:

Since this layer is relevant for SmartKYE platform design, the following list provides a detail of the main characteristics of each category of capabilities information layer should support. We have to point out that these are general guidelines we consider but the degree to which these will be reflected and/or implemented may differ. These categories are:

- **Information Services:** This category of capabilities addresses the support of information services. Information services provide a uniform way of representing, accessing, maintaining, managing, analysing, and integrating data and content across heterogeneous information sources. There are primarily two approaches to achieve that. First approach focuses on building a single view of business-critical data for customers, products, location, and others delivered in context; e.g.,, single view of enterprise (MDM) approach. The second approach focuses on integrating the appropriate information in a timely and consistent manner, analysing and attempting to improve the quality of data, and ensuring consistency and integrity of business-critical data and facts across the enterprise. This approach is known as the Information as a Service (IaaS) approach.
- **Information Integration:** This category of capabilities addresses the support of information integration and enables capabilities for information services.
- **Basic Information Management:** This category of capabilities addresses basic information management concerns such as metadata and unstructured data management.
- **Information Security and Protection:** This category of capabilities addresses the support of information security and protection concerns.
- **Business Analytics:** This category of capabilities addresses the support of

business analytics and business activity monitoring. It enables organizations to leverage information to better understand and optimize business performance. It supports entry points of reporting to deep analytics and visualization, planning, aligned strategic metrics, role-based visibility, search-based access and dynamic drill-through, and alert and detect in-time actions.

- **Information Definition and Modelling:** This category of capabilities defines fundamental constructs of SOA information and events.
- **Information Repository:** This category of capabilities addresses support of the information repository in order to store data such as metadata, master data, analytical data, operational data, and unstructured data.

There are multiple categories of capabilities that this layer needs to support in the SOA RA. The following list and table summarizes these capabilities categorized according to the different categories previously listed:

Table 8 Information Layer Capabilities

Information Services
1. Ability to expose data as services, to add/remove/manipulate data entries in different services or service components, and to disable some data from outside access
3. Ability to handle representing data from various data sources in a unified data format; ability to transform and map data from one format to another and align data from different resources
5. Ability to manage the hierarchy and relationship among data
Information Integration
9. Ability to perform Extract-Transform-Load (ETL capabilities) data from one source to other; ability to extract relevant information from sources, transform the information into the appropriate integrated form, and load the information into the target repository
11. Ability to virtualize data representing actual data from the actual data repositories of various types, such as a DB2 database in the Operational Systems Layer, or an Excel file
12. Ability to handle data transformation (including transformation of data types and contents) and to aggregate data from multiple data sources
15. Ability to cache data in support of the data virtualization/information services capability
Information Security and Protection
19. Ability to handle access privileges of various participants to data
20. Ability to control access on individual data items
Business Analytics
24. Ability to visualize interactively the results from business analytics and data analysis

Integration layer capabilities and building blocks mapping:

The mapping between the capabilities previously listed and the corresponding building blocks of SmartKYE architecture is presented in the Annex section of this document.

4.3.4 SmartKYE Governance layer

SOA governance ensures that the services and SOA solutions within an organization are adhering to the defined policies, guidelines, and standards that are defined as a function of the objectives, strategies, and regulations applied in the organization and that the SOA solutions are providing the desired business value.

The governance layer includes both SOA governance (governance of processes for policy definition, management, and enforcement) as well as service governance (service lifecycle). This covers the entire lifecycle of the services and SOA solutions (e.g.,, both design and runtime) as well as the portfolio management of both the services and SOA solutions managing all aspects of services and SOA solutions such as Service-Level Agreement (SLA), capacity and performance, security and monitoring).

Governance layer capabilities:

There are multiple categories of capabilities that the layer needs to support in the SOA RA. The following table summarizes these capabilities:

Table 9 Governance Layer Capabilities

SOA Metadata Storage and Management
8. Ability to support the capture of service-related information at design time, and its dissemination to the other layers in the SOA in a standards-compliant interoperable manner
9. Ability to support the storage and dissemination of information supporting these capabilities: • Service contract definition (e.g., WSDL) • Service policy management • Service version information management • Service dependencies (e.g., the ability to integrate with a CMDB tool) • Service management descriptions • Canonical form and domain model specification for integration with the Information Layer
13. Ability to advertise for services and metadata about services
14. Ability to find or query for services and metadata about services
Management
31. Ability to do change control to implement and maintain governance

Governance layer capabilities and building blocks mapping:

The mapping between the capabilities previously listed and the corresponding building blocks of SmartKYE architecture is presented in the Annex section of this document.

4.4 SmartKYE architecture conceptual view

The conceptual view of SmartKYE architecture lays out domains and identifies architectural building blocks within these domains. The buildings blocks derived from the analysis performed in the previous section have been mapped to specific layers of the Open Group SOA RA. The result obtained is the SmartKYE SOA Reference Architecture conceptual view presented in Figure 9:

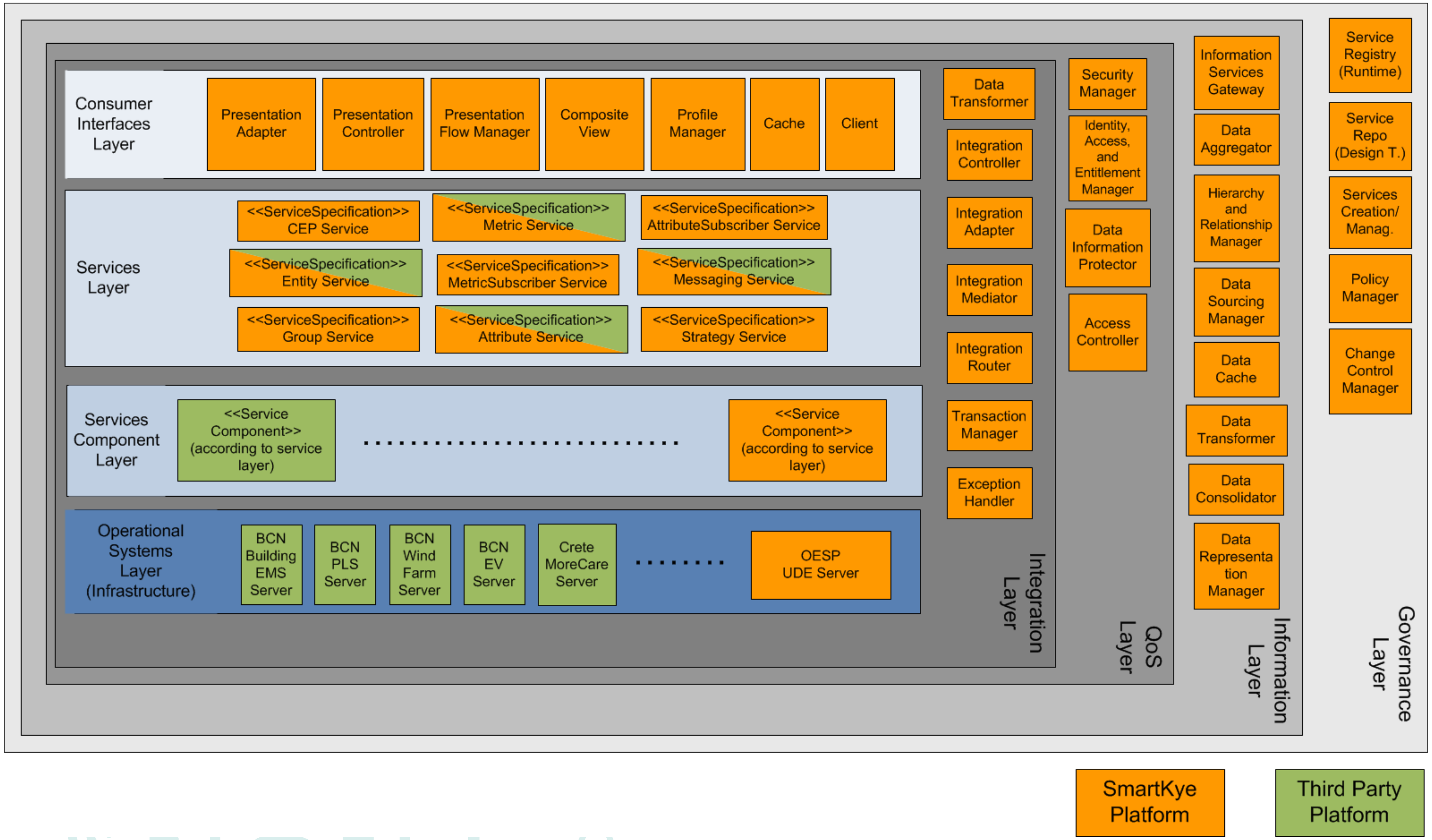

Consumer Interfaces Layer
Presentation Adapter
Presentation Controller
Presentation Flow Manager
Composite View
Profile Manager
Cache
Client
Services Layer
<<ServiceSpecification>> CEP Service
<<ServiceSpecification>> Metric Service
<<ServiceSpecification>> AttributeSubscriber Service
<<ServiceSpecification>> Entity Service
<<ServiceSpecification>> MetricSubscriber Service
<<ServiceSpecification>> Messaging Service
<<ServiceSpecification>> Group Service
<<ServiceSpecification>> Attribute Service
<<ServiceSpecification>> Strategy Service
Services Component Layer
<<Service Component>> (according to service layer)
<<Service Component>> (according to service layer)
Operational Systems Layer (Infrastructure)
BCN Building EMS Server
BCN PLS Server
BCN Wind Farm Server
BCN EV Server
Crete MoreCare Server
OESP UDE Server
Data Transformer
Integration Controller
Integration Adapter
Integration Mediator
Integration Router
Transaction Manager
Exception Handler
Integration Layer
Security Manager
Identity, Access, and Entitlement Manager
Data Information Protector
Access Controller
QoS Layer
Information Services Gateway
Data Aggregator
Hierarchy and Relationship Manager
Data Sourcing Manager
Data Cache
Data Transformer
Data Consolidator
Data Representation Manager
Information Layer
Service Registry (Runtime)
Service Repo (Design T.)
Services Creation/ Manag.
Policy Manager
Change Control Manager
Governance Layer
SmartKye Platform
Third Party Platform

Figure 9 SmartKYE SOA Reference Architecture - Conceptual view

The previous conceptual view of the SmartKYE architecture showcase the building blocks identified as core elements of the architecture. Besides mapping each building block to specific layers according to The Open Group SOA RA, Figure 9 shows two different kinds of building blocks: (i) those under SmartKYE platform domain (in orange), and (ii) those under third-parties domain (in green).

This separation becomes clearer if we analyse for the crosscutting layers. Since most of the building blocks and their corresponding capabilities will be developed and deployed in the OESP platform, the core element of SmartKYE architecture, these buildings blocks have been assigned to the SmartKYE platform category. However, in the case of services, from a conceptual perspective they have been assigned to SmartKYE platform category, although from an operational point of view these services will be developed within SmartKYE project and run in physical systems or servers operating in many different geographical locations, and possible managed by different stakeholders (what is actually what service component and operational system layers illustrate).

The idea behind this separation of building blocks is to illustrate the concept of a federated architecture previously shown in Figure 6. Figure 10 provides a different perspective of the same concept, where it is clear that there can be service requesters/providers and SmartKYE services, which are software components that run either in SmartKYE platform or in third-parties platforms, that can interact seamlessly through SmartKYE platform (OESP component).

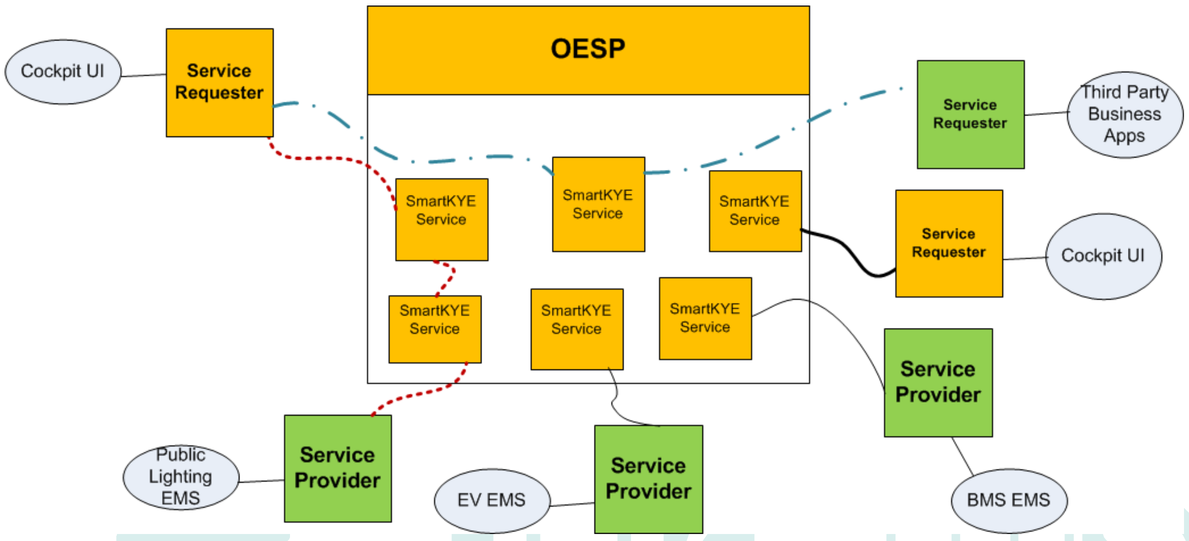


Figure 10 SmartKYE platform interaction example

5 SmartKYE Service Architecture

5.1 Service Architecture Overview

The SmartKYE service architecture is depicted in FMC notation (www.fmc-modeling.org) [20] in Figure 11. We can clearly depict the main stakeholders and parts. The users (e.g., the municipality business and technical manager) interact via the BC and MCC respectively. Both BC and MCC make service calls to the services provided by the OESP platform. The EMSs are attached to the OESP and interact with it via well-defined services on-demand. The OESP exposes several services as shown. Access to these services is allowed with the necessary valid security credentials. The OESP stores locally minimum information for its own operation as well as cache of some usually accessed data.

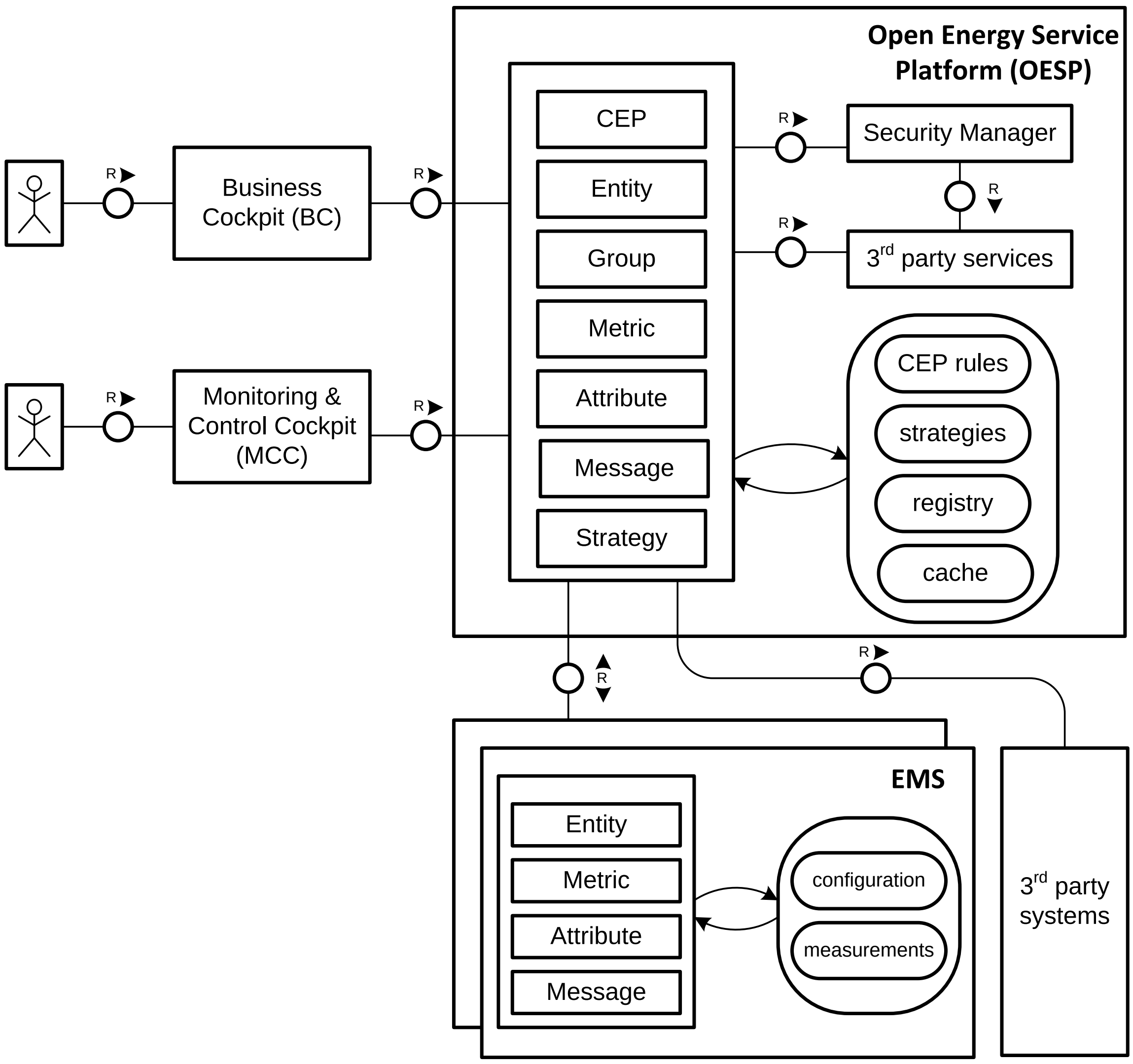


Figure 11 SmartKYE Service Architecture

We can distinguish the following services:

- **Attribute**: This provides information complementary to the Entity service dealing with internal OESP and EMS data. Attributes can be used together with the

Message Service for control/management purposes.

- **CEP**: Complex Event Processing is used for calculating KPIs from simpler metrics as well as delivering the relevant information in an event based manner.
- **Entity**: This is a general-purpose service of an entity description. It offers information about the attributes, KPIs and metrics (as defined in D1.1) being provided by an individual entity.
- **Group**: This service manifests the creation of ad-hoc groups (that may span one or more EMSs) and can be used for getting e.g. aggregated data
- **Message**: This service corresponds to the control part applied from the MCC to the various EMSs in the system. This, together with functionalities of the Attribute service, is used for control/management within SmartKYE.
- **Metric**: This service primarily provides the basic metrics as defined in D1.2 of SmartKYE project and is designed to be extendable to future requirements for the metrics. It is used also as a basis for the CEP service.
- **Security**: basic authentication and authorization operations.
- **Strategy**: Strategies can be communicated from BC to the MCC via this service which has storage and retrieval capabilities

Table 10 Service implementation availability in OESP and EMS

Service host → Service Name↓	OESP	EMS
Attribute	X	x
CEP	x	
Entity	x	x
Group	x	
Message	x	x
Metric	x	x
Security	x	
Strategy	x	

As we can see in Table 10, all of the services are implemented in OESP which acts as a glue between the EMS and the cockpits, and some of them are realized in the EMS. The *(x)* implies indirect or optional usage of the respective service, e.g., the security service is used by each request indirectly or the CEP. We have taken extra steps to ensure that a common API is used for both interactions e.g., BC/MCC with OESP and OESP with EMS for the common set of services.

Table 11 Service usage by the OESP

OESP Service → uses ↓	Attribute	CEP	Entity	Group	Message	Metric	Security	Strategy
Attribute (OESP)							(X)	
Attribute (EMS)	X						(X)	
CEP		X					(X)	
Entity (OESP)		X		X			(X)	
Entity (EMS)			X				(X)	
Group		X	X	X			(X)	
Message (OESP)							(X)	

Message (EMS)		X		(X)
Metric (OESP)	X			(X)
Metric (EMS)			X	(X)
Security				
Strategy				(X)

Table 11 provides an overview of how OESP uses the available services that may be implemented in OESP and/or EMS. Table 12 depicts the usage of services by the two cockpits.

Table 12 Service usage by the BC and MCC

Service used by → **Service Name↓**	**BC**	**MCC**
Attribute	X	x
CEP	x	x
Entity	x	x
Group	x	x
Message		x
Metric	x	x
Security	(x)	(x)
Strategy	x	x

5.2 Service Specification

5.2.1 Entity Service

Every EMS connected to the platform is defined as an entity. Furthermore, each EMS can offer its underlying entities to the service consumers, where OESP service consumers (such as BC and MCC) are able to communicate to any of the entities. In dependency of their underlying complexity, entities can be top level entities (e.g. an EMS), or children of the top level entities e.g. a single controllable light point of a public lighting system. The platform offered services are designed to deliver a service consumer to receive the top level entities and also their descendants. Once all entities can be accessed, they will be providing timeseries of predefined metrics (as defined in D1.2 of the project) to service consumers, therefore the series they provide are also reachable by the entity service.

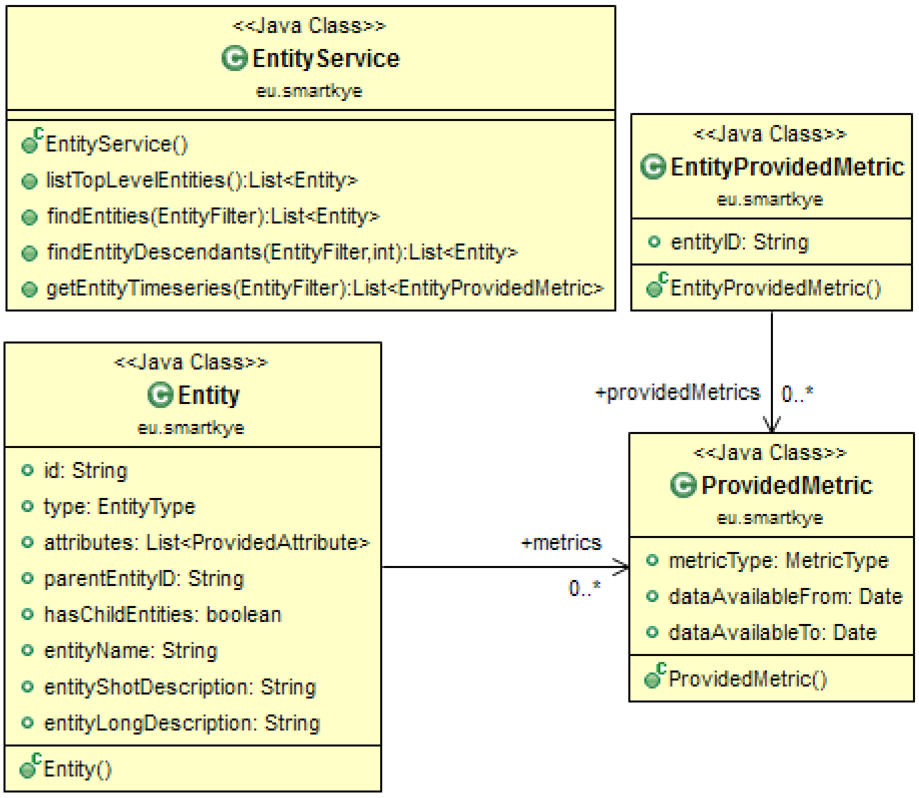


Figure 12 The Entity Service of the platform

As depicted in the Figure 12, most of the listed methods consume the basic *EntityFilter* object and return a list of the service related *Entity* objects. One of the methods is used to return parental objects and one can notice that no input parameter is required. Due to the very limited number of EMSs connected to the platform such filter is not required, while other methods use the filtering object to specify interest in description of any entity of the system. The same filtering can be applied to the metrics they provide, where the service consumer understands quality and capability of delivering certain metrics from filtered entities. The complete technical description of the depicted API is shown in section 9.7.2.

5.2.2 Group Service

Entities within the platform are accessible through top-level entities that hold the logic for services they offer. One can consider a top-level entity as one group of all of its underlying entities. However, in many SmartKYE scenarios the cockpits do not always focus to one particular entity, but rather to certain characteristics of entities. For example, one may be interested in observing only wind turbines accessible through many different top-level entities. Although this particular case may be achieved already by filtering of the entity type (with *EntityFilter* from the section 6.2), more complex grouping might be required. The grouping services are envisioned for those purposes. Once entities of interest are identified, one can connect those entities into a logical group. This service is depicted on Figure 13.

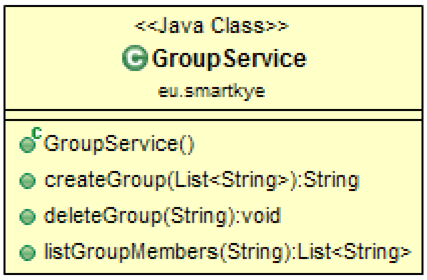


Figure 13 Services for entity grouping

The creation method uses the unique entity identification and takes a list of them for composing a group. After its creation, the method returns the group identification of the OESP. The other two methods are used for deletion of a group and listing of its members. We do not support the modification of existing groups to increase the scalability of the distributed platform: modification would require an always-consistent view of the groups of all platform subsystems. However, with the current approach a subsystem only needs to ask for a group if a request contains it and the deletion of a group can be propagated in the background, since at most a group could be used for some more time but never lead to an incorrect view. The WSDL description of the service is documented in section 9.7.3.

5.2.3 Metric Service

Experiences obtained from previous smart grid projects [10] helped unifying numerous energy services usually envisioned to be individual as a single service. This is the *MetricService* responsible for delivering advanced and expandable functionalities to the OESP platform. Once entities are reachable through the OESP, significant part of their purpose within the platform is timely delivery of the metrics they are capable, or willing, to provide. This service is responsible for delivering these metrics as timeseries, last value and even to subscribe to multiple metrics of one or more entities. In context of this project, a metric is everything that can be represented by a real number at a point of time. With this in mind, certain mathematical operations are possible for all the metrics. In fact, aggregating the same metric from numerous entities is one of the main interests of the SmartKYE cockpits. Figure 14 provides an overview of the method definitions of the service, with additional objects relevant for their functionality description.

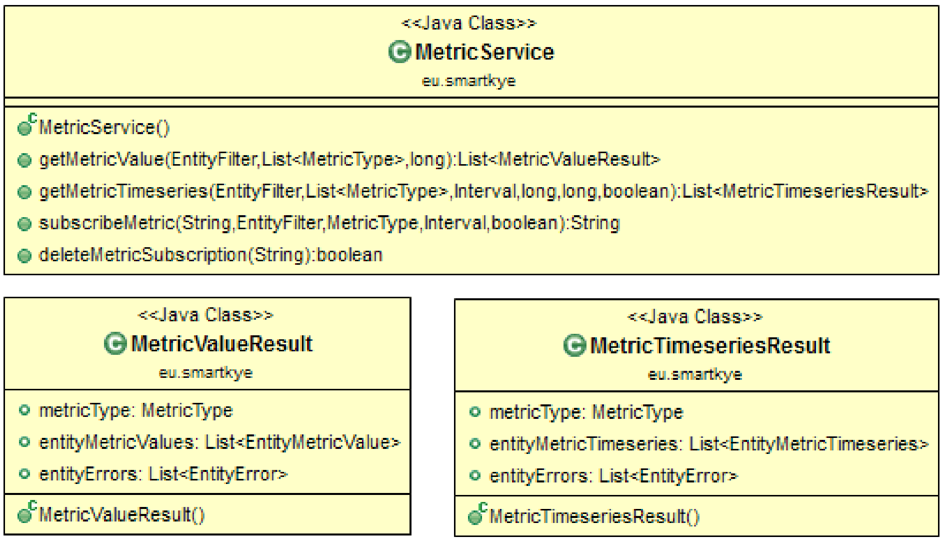
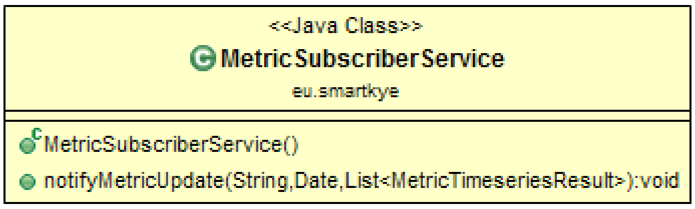

<<Java Class>>
MetricService
eu.smartkye
MetricService()
getMetricValue(EntityFilter,List<MetricType>,long):List<MetricValueResult>
getMetricTimeseries(EntityFilter,List<MetricType>,Interval,long,long,boolean):List<MetricTimeseriesResult>
subscribeMetric(String,EntityFilter,MetricType,Interval,boolean):String
deleteMetricSubscription(String):boolean
<<Java Class>>
MetricValueResult
eu.smartkye
metricType: MetricType
entityMetricValues: List<EntityMetricValue>
entityErrors: List<EntityError>
MetricValueResult()
<<Java Class>>
MetricTimeseriesResult
eu.smartkye
metricType: MetricType
entityMetricTimeseries: List<EntityMetricTimeseries>
entityErrors: List<EntityError>
MetricTimeseriesResult()

Figure 14 Metric Services for entities of the OESP

Listed methods of the *MetricService* are mostly requiring the *EntityFilter* object for specifying the entities of interest. The first two methods offer access to the metrics of a last sampled value as metric timeseries. Since the SmartKYE architecture is distributed, the methods require a timeout of the request. If numerous entities are observed, the timeout value will trigger the delivery of already obtained results. Entities with no response, or no data availability, will be listed in the *EntityError* (defined in the section 6.2) array of the response result. As depicted, results received may take one value or series of them e.g., *MetricValueResult* and *MetricTimeseriesResult* respectively.

Some more parameters for the metric timeseries method are required. As input the method also takes the *Interval* description and their count. For example, one day in hourly basis is represented with start time and its duration (of one hour) within the *Interval* object, while number of consequent intervals is a parameter of the method. Finally, the result of the response from the entities filtered can be aggregated by simply assigning the TRUE value to the aggregation parameter. As already mentioned, even if some entities will not deliver their individual metrics to the OESP; still the aggregation will be executed for available results while unavailable ones will take their place in the *EntityError* list of the *MetricTimeseriesResult* object.

The subscription to the metrics of entities stems from the near real-time cockpit functionalities, for both BC and MCC. Subscriptions are envisioned to deliver metrics from one or more entities in periodical fashion. As the last two methods of Figure 14 indicate, for every subscription one will need to provide subscriber identification (e.g., response back URL), *MetricType* and *Interval* object. The *Interval* information is later used for periodic delivery of the specified metric. Similarly to the timeseries method, the subscription responses can be signed to be aggregated if required. Finally, once a cockpit subscribes and a response back URL is provided, the implementation of a notification interface is expected on subscriber's side. Figure 15 describes the service required to be implemented on cockpit's side.



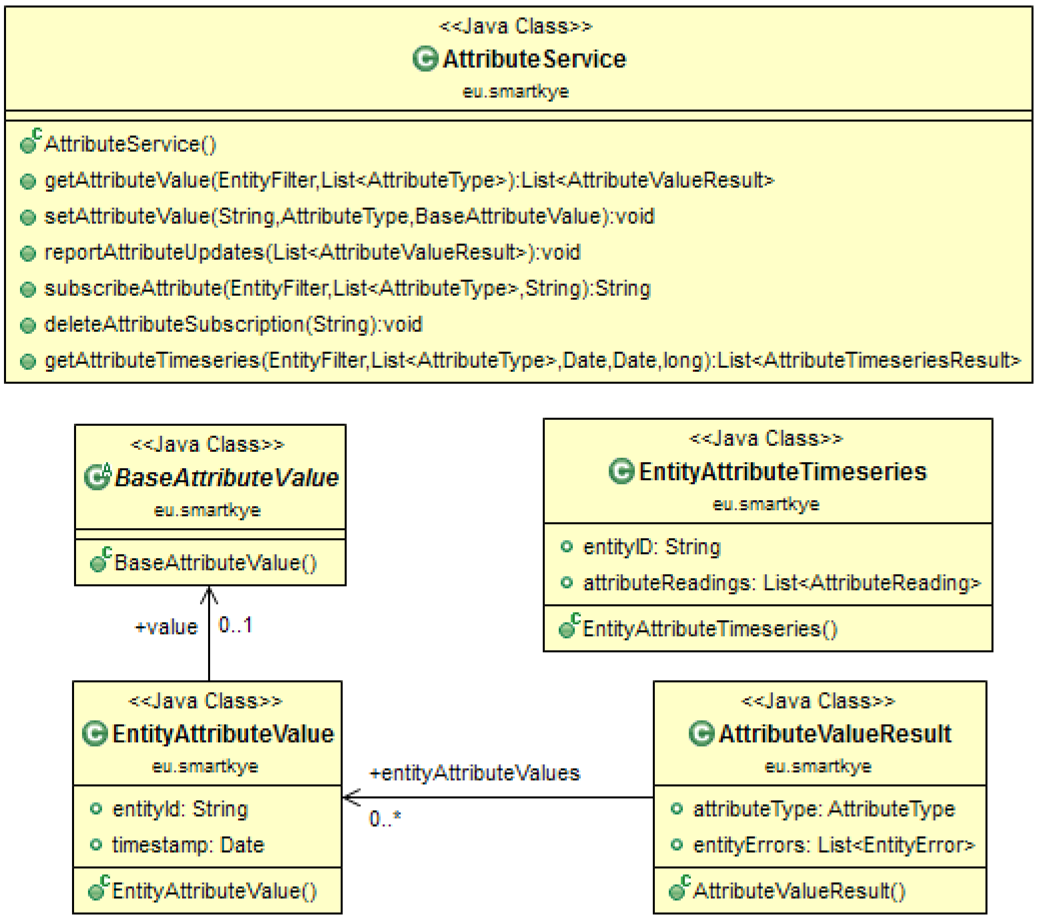
<<Java Class>>
MetricSubscriberService
eu.smartkye
MetricSubscriberService()
notifyMetricUpdate(String,Date,List<MetricTimeseriesResult>):void

Figure 15 Service for notification of periodical metrics of entities

Only one method is envisioned and requires subscription handle provided on subscription to the metric service. Once a subscription is identified, metric values and their timestamp are requested as input to the notification service. Note that the timestamp here is not the original timestamp, but rather one that is assigned to the notification interval. Both, regular and subscriber services are described in WSDL within the section 9.7.5.

5.2.4 Attribute Service

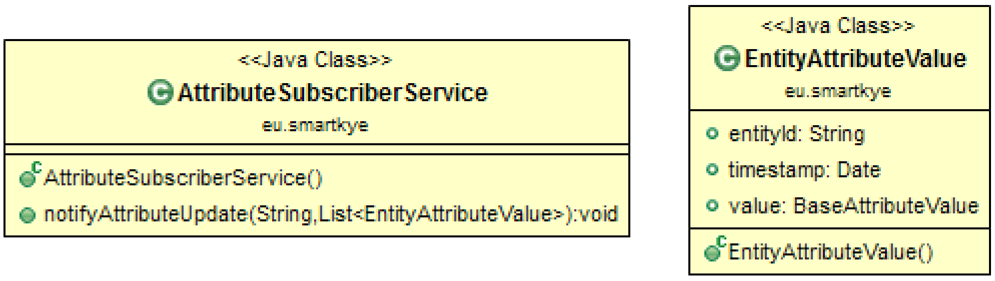
Heterogeneity of the entities within the OESP may bring extreme complexity of identifying services of mutual interest for numerous different entity types. Previous experiences from SmartGrid projects (www.ict-nobel.eu) allowed introduction of the *MetricService* for delivering huge number of functionalities from the OESP platform. Still, uniqueness of the entities involved in this project required a more abstract approach to describe capabilities and information offered by an entity. Attributes of entities therefore take shape in a global object named *BaseAttributeValue*. The values of attributes can be updated or retrieved as last assigned value or even series of changes in a timeframe. In Figure 16 one can have overview of the methods provided by the service.



<<Java Class>>
AttributeService
eu.smartkye
AttributeService()
getAttributeValue(EntityFilter,List<AttributeType>):List<AttributeValueResult>
setAttributeValue(String,AttributeType,BaseAttributeValue):void
reportAttributeUpdates(List<AttributeValueResult>):void
subscribeAttribute(EntityFilter,List<AttributeType>,String):String
deleteAttributeSubscription(String):void
getAttributeTimeseries(EntityFilter,List<AttributeType>,Date,Date,long):List<AttributeTimeseriesResult>
<<Java Class>>
BaseAttributeValue
eu.smartkye
BaseAttributeValue()
+value
0..1
<<Java Class>>
EntityAttributeTimeseries
eu.smartkye
entityID: String
attributeReadings: List<AttributeReading>
EntityAttributeTimeseries()
<<Java Class>>
EntityAttributeValue
eu.smartkye
entityId: String
timestamp: Date
EntityAttributeValue()
+entityAttributeValues
0..*
<<Java Class>>
AttributeValueResult
eu.smartkye
attributeType: AttributeType
entityErrors: List<EntityError>
AttributeValueResult()

Figure 16 Services for entity attributes

The first two methods are used for updating and retrieving values of different *AttributeType*. The previously mentioned timeseries method can be used for retrieving series of variable updates or the actual objects representing a certain attribute. The object of their abstraction is named *EntityAttributeValue* holding the timestamp of the assignment of the value, with the timestamp variable. Finally, the three subscription methods listed are similar to the functionalities of the *MetricService*, thus the *AttributeService* also accepts the subscription for provided attribute types for filtered entities. Similarly, to the MetricService, every subscriber needs to implement the notification services, here depicted in Figure 17.



<<Java Class>>
AttributeSubscriberService
eu.smartkye
AttributeSubscriberService()
notifyAttributeUpdate(String,List<EntityAttributeValue>):void
<<Java Class>>
EntityAttributeValue
eu.smartkye
entityId: String
timestamp: Date
value: BaseAttributeValue
EntityAttributeValue()

Figure 17 Service for notification of attribute updates

Only one method to notify an attribute update is envisioned; and it takes a list of

EntityAttributeValue objects as the input. The other parameter is the subscription handler used to identify a subscription. Note that timestamp of the updated attributes is actually the original timestamp of their change, thus one can track behaviour of any attribute over time. The complete description of the *AttributeService*(s) is included as WSDL in section 9.7.1.

5.2.5 Complex Event Processing (CEP) Service

5.2.5.1 Introduction

The complex event processing service (CEP) allows the clients of the platform to push (a part of) the processing of data in the platform. This approach fosters the sharing of common processing functionality, e.g. the calculation of KPIs as defined in the SmartKYE deliverable document D1.2, between different clients and allows the platform to distribute work among its distributed subsystems and increase the scalability of the system. This also allows sharing the handling of errors between different stakeholders (e.g., when information from some entities is missing).

For increased flexibility compared to simple query processing approaches, we choose to represent the necessary processing rules as a processing graph consisting of a set of connected processing components (processing steps).

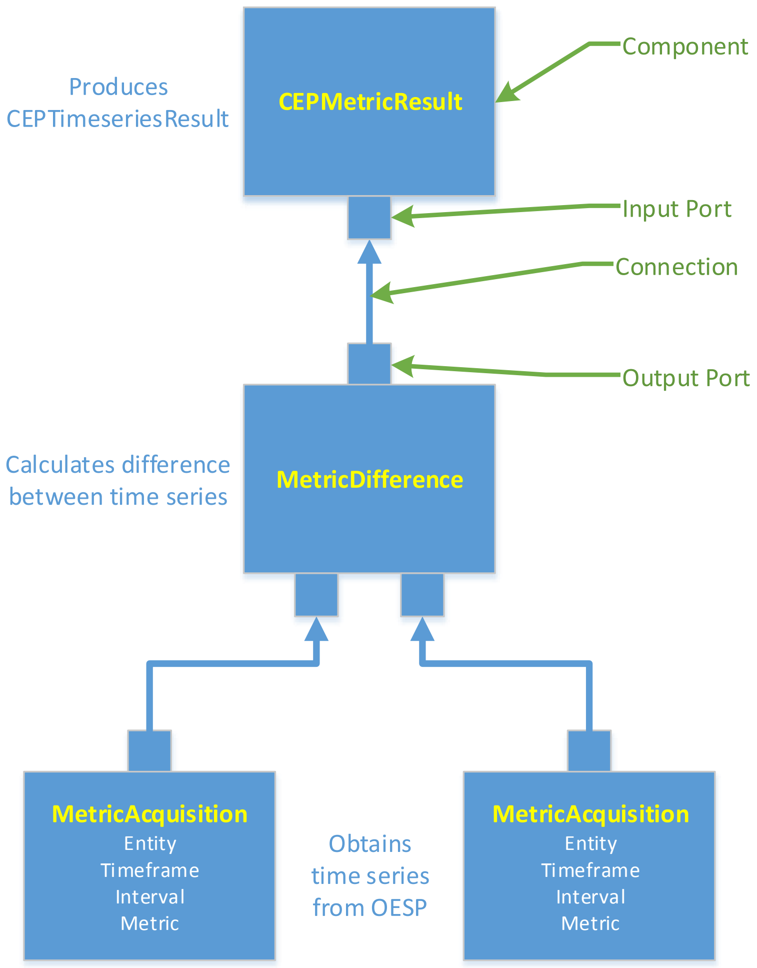


Figure 18 CEP example for computing the KPI "Difference in energy consumption"

In Figure 18, we present as a simple introductory example a potential processing graph for the calculation of KPI #5 “Difference in energy consumption” (defined in the SmartKYE deliverable document D1.2), which just compares the energy consumption of the same entity between two intervals. The example graph depicts four components: two *MetricAcquisition* components and one *MetricDifference* and *CEPMetricResult* component. Each component can have an arbitrary number of input ports where data is fed into the components and an arbitrary number of output ports where the processing results of the components are available. At the bottom of the graph, we show two input components that do not have input ports. In this case, these are components using the *MetricTimeseries* service of the OESP to request data. In the center of the graph (Figure 18) is the single processing component. In this instance, it is a (rather simple) specialized component that works on two metric time series and computes their difference. It contains two input ports, where the data from the *MetricAcquisition* components is fed in. The output of the *MetricDifference* component feed its result into the *CEPMetricResult* component at the top of the graph which is responsible for converting the data in the same output format that is provided by the *getMetricTimeseries* calls. The final concept of the CEP system shown in the graph is the list of parameters of the processing components. In this case, only the data acquisition components feature parameters that describe which data (*Timeframe, Interval, Metric*) they should obtain from where (Entity).

It is important to note that while in this case both the graph and the data processing component that computes the difference between two time series are rather simple, the system targets a wide range of possible data processing needs. On the one hand, it would for example be possible to implement a complex forecasting algorithm as a single component – with significantly more logic than provided by common query languages. On the other hand, the system allows creating much more complex graphs for example using basic pre-processing, filtering and feature recognition components with suitable parameters and connections to implement event detection systems.

In the following section we provide a more detailed description of the concepts provided by the CEP and the API for using it offered by the OESP. This description will be complemented by a more detailed specification on how to develop the processing components in D3.1.

While we have firm plans for developing several components in the project, e.g., components that interact with the OESP API and to compute the KPIs described in D1.2, is important to note that there is a strong focus on providing an extendable system that provides enough flexibility to allow implementing specialized functionality within the project and beyond.

5.2.5.2 CEP System

In this section, we describe the basic building blocks used in our complex event processing system.

Component

The component is a reusable building block that constitutes the data processing logic. Similar to other systems such as J2EE or OSGi, components can be defined at arbitrary levels of granularity. However, in contrast to these systems, they can be instantiated multiple times and they provide parameters to support different application requirements.

There are three important subclasses of components:

- Input components: the input components do not have any input data and instead are responsible for obtaining information from outside the CEP, e.g., by calling the OESP metric services provided, or generate data based on their parameters.
- Processing components: processing components have both input and output ports and are activated as soon as information for all ports is available.

- Result/Notification components: these components are responsible for providing the data to the outside of the CEP either by preparing them in an expected format or by directly invoking outside services, e.g., for push notifications.

While the granularity of the input components and the output components is usually easy to determine, there is a greater variety of possible divisions for the processing components. There is a need to weight the benefit of small components to encourage reusability with overhead involved in exchanging data between components and the complexity involved in extracting the suitable parameters of more complex systems. While our system does not impose a certain granularity, it is certainly not a goal to obtain the result of a simple formula by constructing a graph of individual simple arithmetic operators. The provided example in Figure 18 is from this perspective an unusual example. However, the need for the difference between two time series is so common, that it merits a dedicated component.

Parameter

Each component can be associated with a list of typed parameters (e.g., *String, Integer*, etc.) that allows developing generic components and fosters reusability of component implementations among different applications. In the example shown in Figure 18, the input components offer parameters to specify which data should be obtained from the OESP. However, parameters can be used by all kinds of components, e.g., a notification component could have a parameter for specifying the subscriber URL or a feature recognition component could expose threshold parameters. It is also possible (and shown in the example) to have multiple instances of a component with different parameters.

Input/Output Port

In order to facilitate the composition of multiple processing components, each component can contain strongly typed input and output ports. An output port of a component can be connected to an input port of another component with the same type.

Connection

We use connections to establish a link between an output port of one component and an input port of another component. However, to increase reusability and robustness, a component is not aware of which other components are connected to it. Instead, a component is simply informed about new data by the CEP system and provides its output data back to the CEP system. This also allows 1: n connections where one output port is connected to several input ports.

Graph and graph instance

Finally, the processing graph specifies the necessary components and their interconnections to implement a certain data processor. This is similar to implementing a class in an object-oriented programming language. The CEP system then allows instantiating such a graph which requires resolving all parameters provided by the components of the graph.

Figure 18 also exemplifies common use cases for the parameters of a processing graph. Some parameters are specified already in the graph description. In the example the Metric parameters of both acquisition components would be set to *EnergyConsumption*. Some parameters should be specified during instantiation such as the interval and timeframe parameters, and some parameters should be specified during instantiation but should be shared between several components. In the example, the *entity* parameter should be shared as the process is for computing the difference of the energy consumption of the same entity in different time intervals.

Therefore, the system design considers not only parameters of the components but parameters of the whole graph. For each component parameter, there are three possibilities:

- Resolved parameter: a parameter value is already specified in the graph
- Bound parameter: a parameter is bound to a parameter exposed by the graph and, therefore, must be specified during instantiation
- Bound parameter with default value: similar to the previous possibility the parameter is exposed but a default value is provided to make setting the parameter optional during instantiation

The binding of a component parameter to a graph parameter allows ensuring that some parameters of different components share the same value.

CEP data model and API

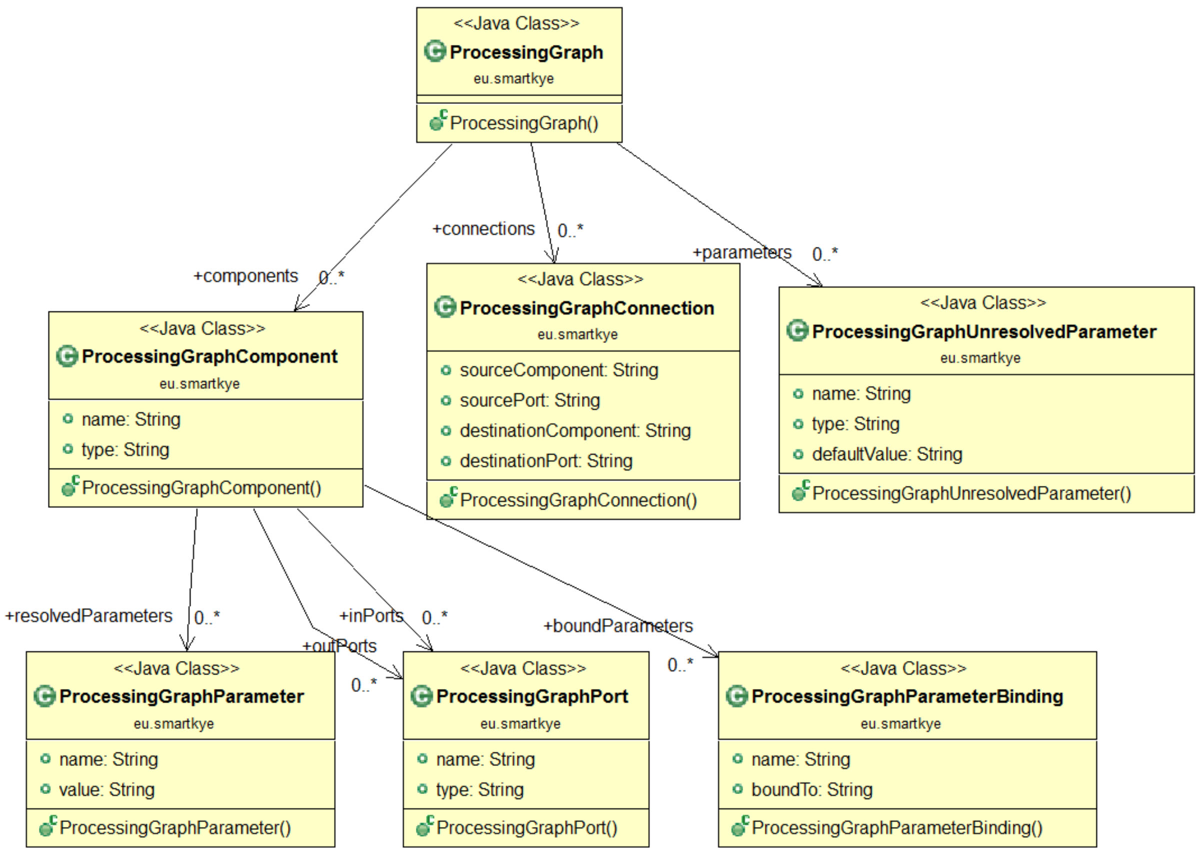
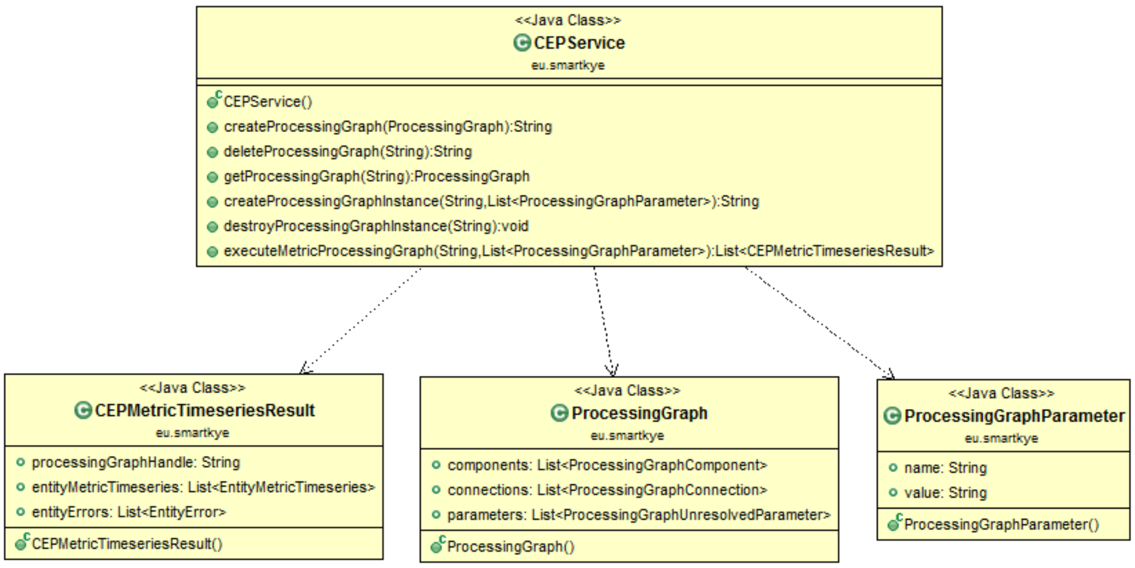


Figure 19 CEP Data Model

Since the CEP data model and its API are very tightly coupled, we show both here for easier understanding. In Figure 19, we can see the UML class diagram for the parts of a processing graph. The structure follows closely what has been described in the previous subsection regarding the graph, component, port and connection class. The distinction between resolved parameters and bound parameters for the component allows for describing the sharing of parameters as explained above.



<<Java Class>>
CEPService
eu.smartkye
CEPService()
createProcessingGraph(ProcessingGraph):String
deleteProcessingGraph(String):String
getProcessingGraph(String):ProcessingGraph
createProcessingGraphInstance(String,List<ProcessingGraphParameter>):String
destroyProcessingGraphInstance(String):void
executeMetricProcessingGraph(String,List<ProcessingGraphParameter>):List<CEPMetricTimeseriesResult>
<<Java Class>>
CEPMetricTimeseriesResult
eu.smartkye
processingGraphHandle: String
entityMetricTimeseries: List<EntityMetricTimeseries>
entityErrors: List<EntityError>
CEPMetricTimeseriesResult()
<<Java Class>>
ProcessingGraph
eu.smartkye
components: List<ProcessingGraphComponent>
connections: List<ProcessingGraphConnection>
parameters: List<ProcessingGraphUnresolvedParameter>
ProcessingGraph()
<<Java Class>>
ProcessingGraphParameter
eu.smartkye
name: String
value: String
ProcessingGraphParameter()

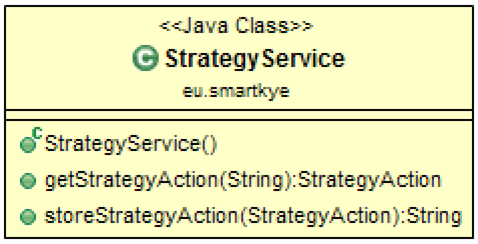
Figure 20 CEP API

Since most of the complexity of the CEP system is contained in the data model, the API (Figure 20) is intentionally kept simple. On the one hand, it provides methods for storing and retrieving processing graphs, which also allows sharing and reuse of graph definitions between several clients. On the other hand, it provides methods for creating and controlling the instances of these graphs. Since the processing of metric time series is by far the most common use case, the API contains a dedicated convenient method to handle this.

5.2.6 Strategy Service

Within the vision of the SmartKYE project, strategies are expected to be generated by the BC and communicated to the MCC in its decision process on applying energy efficient control to the underlying infrastructure. These strategies are communicated from the BC to the MCC via the Strategy service running in OESP.

The *StrategyService* provides two basic methods of Strategy, which are *getStrategyActon()* and *storeStrategyAction()* for retrieval (expected to be used by the MCC) and storage (expected to be used by the BC). Figure 21 shows the structure of *StrategyService*.



<<Java Class>>
StrategyService
eu.smartkye
StrategyService()
getStrategyAction(String):StrategyAction
storeStrategyAction(StrategyAction):String

Figure 21 Services for Strategy

The method *getStrategyAction()* gets a strategy by *StrategyActionID* and returns the results as a *StrategyAction*. The method *storeStrategyAcition()* stores a strategy. The BC and MCC are expected to exchange XML documents which have all the necessary information.

5.2.7 Message Service

Message services together with functionalities of the Attribute service are used for control/management of specific entities in the system by the MCC.

Control of specific entities within SmartKYE is enabled by the characteristics of each particular EMS. As it was explained in previous sections, from the initial analysis of each single EMS to be integrated in each of the pilot site, there are several control capabilities offered by specific EMS, e.g., setup of operation parameters or control for the public lighting EMS (see Annex section).

Thus, message services are based mainly on a “message type” (EMS dependent) and the corresponding target entity. Not all EMS will require to implement and offer these capabilities, e.g.,, a weather forecast service does not need to offer any control capabilities; it will simply provide weather data as required.

5.2.8 Security

5.2.8.1 Introduction and Rationale

Security is critical for architectures like SmartKYE and it poses several challenges:

- Deals with confidential and sensitive information
- Various stakeholders with different security needs
- Architecture supports data exchange between all stakeholders, but individual stakeholders can have different contracts providing different data or quality-of-service
- Federated system with stakeholders of different institutions using different technologies
- OESP handles data not for itself but for clients

To handle all these challenges, the security model must embrace the federated nature of the envisioned system in a similar way as the architecture. While data confidentiality can be achieved using SSL/TLS encrypted communication between all partners, authentication and authorization requires a two-level approach. On the one hand, the interactions between the OESP and the other stakeholders are secured using SSL/TLS certificates. On the other hand, the interactions of stakeholders among each other but via the OESP is handled with decentralized responsibilities where the data providing stakeholders can decide which data to provide to whom and the OESP acts as an intermediate for the clients.

The advent of cloud technologies has also led to increase interactions among these services, where e.g., an online task management service can be instructed to use an online calendar service by another provider to insert deadlines into the calendar view. To solve the problem of controlling the access of one service to the data of another as an intermediate for the owner of the data, the OAuth protocol has been developed. Since this protocol is applicable to the challenges provided by the SmartKYE vision, its standardization by the IETF and the widespread use by global service provides such as Google, we choose it as the mechanism for the SmartKYE security architecture.

5.2.8.2 The OAuth 2.0 Protocol

In the following, we provide a short introduction to the OAuth 2.0 protocol [21] as well as a mapping of the entities to the stakeholders in the SmartKYE architecture.

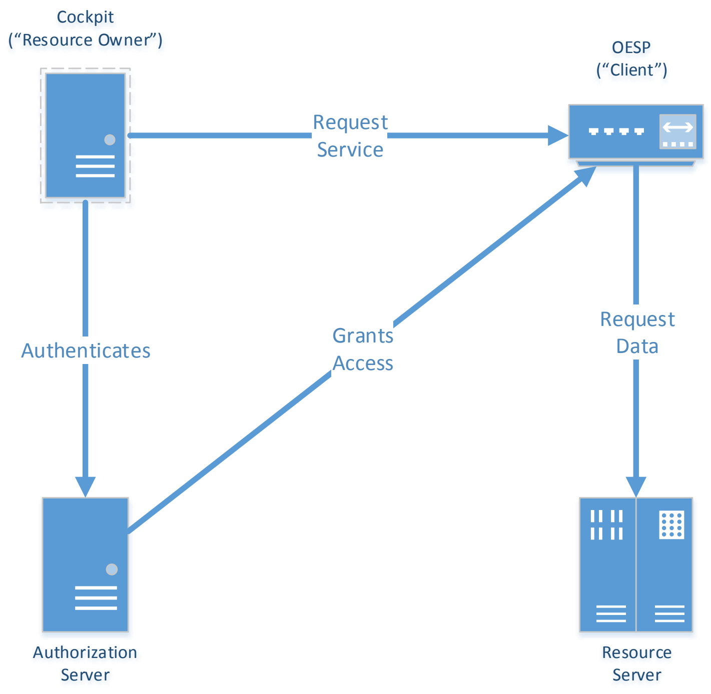


Figure 22 OAuth 2.0 High-Level Overview

In Figure 22 we show the main stakeholders and a high-level overview of their interactions. The terms used by the OAuth protocol and the common representative stakeholder of the SmartKYE architecture are represented.

It is important to highlight that at least three different institutions are present: the resource owner, for SmartKYE mostly a cockpit, the OESP and the resource server. The authorization server belongs conceptually to the institution of the resource server, but can technically be operated by a separate party. Furthermore, we want to highlight that the "resource owner" and the "resource server" are from different institutions and while the term "resource owner" is applicable to common online services providing (e.g., calendar functionality wherethe user owns its data), in terms of the SmartKYE architecture it is more apropriate to speak of a client that has a contract with a data provider.

From a high-level perspective, we have the following interactions:

- The cockpits request a service from the OESP, e.g. a time series
- The cockpit authenticates itself directly at the authorization server since the OESP only acts as a proxy for the cockpit. The OESP does not need to know the credentials since the cockpit authenticates itself directly (e.g., the password, of the cockpit for the resource server)
- The authorization servers issues an access token to the OESP on behalf of the cockpit
- The OESP uses this token to request data from the resource server, which in turn uses the token to control the access to its data

It is important to note that the token has a certain lifetime and, therefore, the authorization and token exchange does not need to occur for every service request.

The most important difference from the usual operation of the OAuth protocol is that a single request to the OESP may result in accessing multiple resource owners, which in turn will potentially require the authentication with multiple authorization servers. While this is an overhead to a centralized solution, this operation happens rarely, because, as described before, the access token can be cached by the OESP.

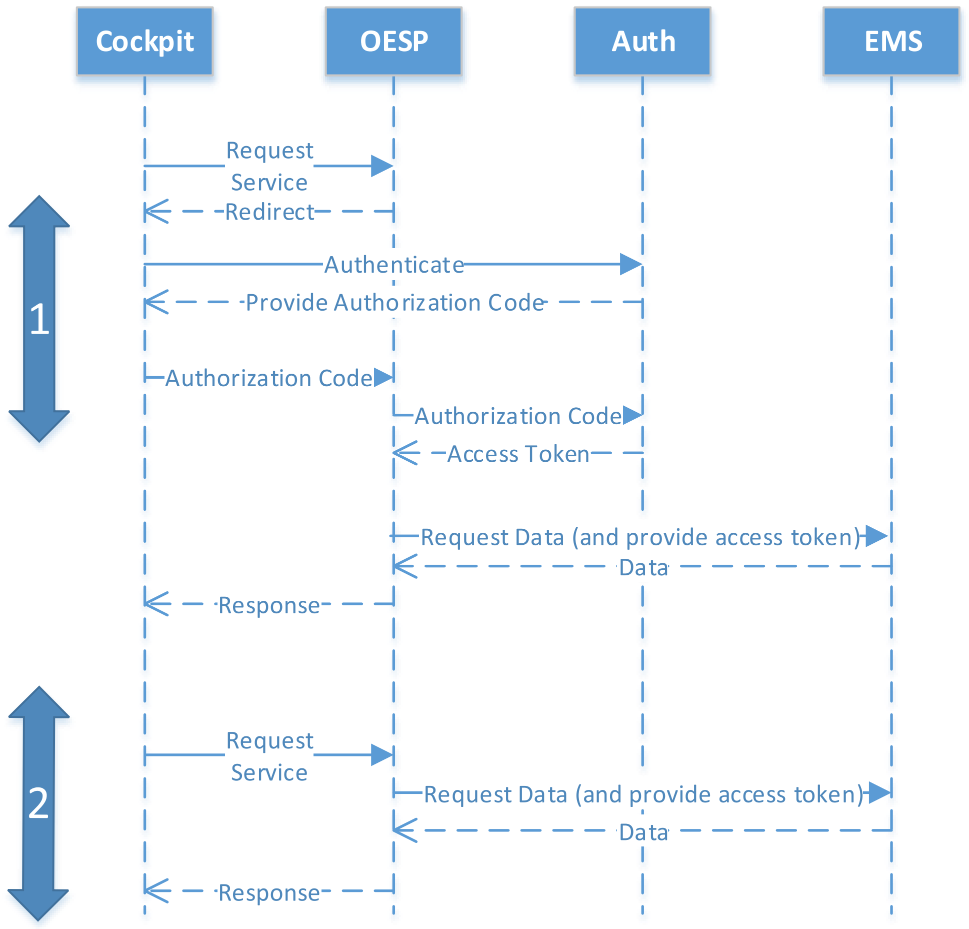


Figure 23 OAuth 2.0 Sequence Diagram

In Figure 23 we show a sequence diagram illustrating in more detail the interactions of the OAuth 2.0 protocol.

The first parts show the steps necessary when the OESP does not have an access token for the cockpit to the EMS yet and authorization is required:

1. The cockpit request a service from the OESP
2. The OESP maps the request to the correct authorization server associated with the resource server from which data is requested and redirects the cockpit to the URL of the authorization server
3. The cockpit authenticates directly with the authorization server without divulging any secret information to the OESP
4. The authorization server provides an authorization code and redirects back to the OESP
5. The cockpit provides the authorization code to the OESP
6. The OESP uses the authorization code to request an access token from the authorization server
7. The authorization server provides the access token
8. The OESP requests data from the resource server while providing the access token

9. The resource server checks the token and if successfully validated returns the data
10. The OESP potentially processes the data or combines it with other information and provides the response to the cockpit

After this process occurs once for a cockpit-EMS pair, the following requests are much simpler and are depicted in the bottom part of the figure. The only difference in the interactions compared to a system not using OAuth is that the OESP always provides the access token to the EMS when requesting data.

While the figure depicts just the simple case of one EMS, the use of multiple EMS just implies a loop of the sequence 1. And it is again important to note from a performance viewpoint, that the sequence 2 (requesting a service and receiving data) can be repeated multiple times without requiring a new authorization process.

In summary, the combination of SSL/TLS and OAuth 2.0 provides a flexible standards-based security model for the federated SmartKYE architecture offering full flexibility for the definition of access control to the stakeholders without requiring the transfer of credentials to the OESP.

6 Data Exchange Specification

6.1 Introduction

This section introduces the globally shared objects between many services of the OESP platform and also the ones related specifically to a service, which were denominated as the service-related objects. The XML schema definition (XSD) was used for their technical description and is also attached in section 9.6.

6.2 Basic Objects

6.2.1 EntityType enumeration

Integration of heterogeneous energy sub-systems of the envisioned OESP can only be efficiently done if types of its components are clearly separated. Every entity within the OESP platform must have a type, which resulted in identification of numerous types an entity can have e.g. a wind farm, a wind turbine, a point of light, a public lighting system, etc. Since this is an enumeration of many types, section 9.6 contains the XSD where currently identified types are listed.

6.2.2 MetricType enumeration

With introduction of the MetricServices a level of abstraction on the MetricType can be noted. Therefore, this type is used for identifying metric of a timeseries and list of types that can be provided by an entity. Since this is an enumeration of many types, the section 9.6 contains the XSD where currently identified types are listed. This listing is an extended version of the metrics already identified in the SmartKYE deliverable document D1.2.

6.2.3 AttributeType enumeration

Since this is an enumeration of many types, section 9.6 contains the XSD where currently identified types are listed.

6.2.4 Interval

A simple and commonly used object describes a time interval, defined only from start time and the length of the interval. However, its value grows with introduction of the *IntervalUnit* enumeration. As one can imagine, an interval value can be daily, monthly, weekly etc. which can be expressed in seconds. In contrast, month and year durations can differ from time to time, thus the *IntervalUnit* enumeration plays significant role in specifying how an interval should be processed. Figure 24 depicts the basic objects discussed.

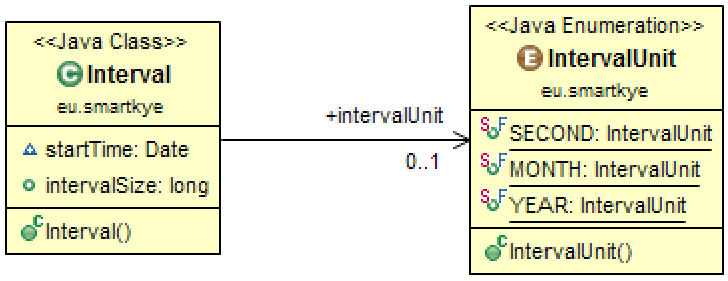


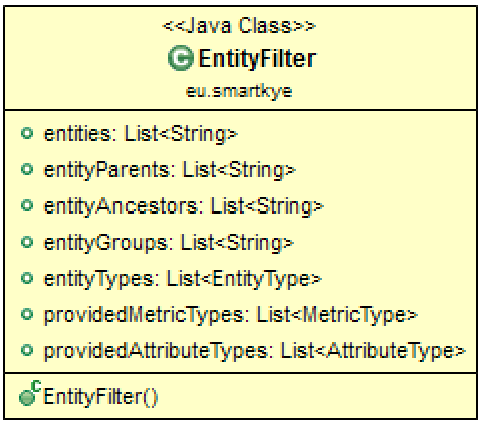
Figure 24 Globally used object for defining an interval

6.2.5 SmartKyeException

Due to the distributed nature of the system, all functions of the OESP can indicate problems using the used SmartKyeException exception type.

6.2.6 EntityFilter

Numerous entities are expected to be reachable through the OESP platform. A point where components can be filtered needs to be considered. Once entities of interest are identified, the filtering object can be passed to numerous services offered by the platform. In Figure 25 the filtering capabilities identified as needed by cockpits are listed.



<<Java Class>>
EntityFilter
eu.smartkye
entities: List<String>
entityParents: List<String>
entityAncestors: List<String>
entityGroups: List<String>
entityTypes: List<EntityType>
providedMetricTypes: List<MetricType>
providedAttributeTypes: List<AttributeType>
EntityFilter()

Figure 25 Globally used object for entity filtering

The possibilities for the selection fall into three categories:

- First, entities can be selected by their ID or their relationship (having a certain parent or ancestor) to others or being in a group.
- Second, entities can be selected based on their category (e.g., wind farm).
- Third, they can be selected based on their functionality, e.g.,, on which metrics or attributes they offer.

The complete XSD description of this object can also be found in section 9.6.

6.2.7 EntityError

Any distributed architecture, as what SmartKYE is dealing with, will have multiple points of failure. The same problem can be encountered for any request directed to the OESP platform. As no limitations are considered for the entity filtering, one needs to be able to address limitations of entities unable to deliver the data. This object is proposed to address listings of the entities failing to deliver metrics, thus the aggregation step was not 100% reliable. With that in mind, a cockpit designer can provide indications on the quality of the data delivered, quantitatively and weighted. The same problem was also identified for the entity attributes, which led to global usage of the object depicted in Figure 26.

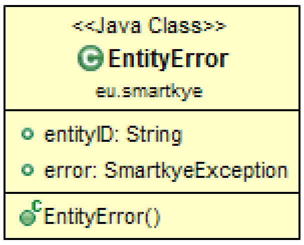


Figure 26 Listing of entities failing to deliver services

6.3 Service-related Objects

6.3.1 Attribute

Objects that are specifically related to the *AttributeService* and *AttributeServiceSubscriber* are only value-oriented. The initial point of the attributes exchanged is the *BaseAttributeValue,* which is used as abstraction object for attributes of different complexity levels. Apart from the *EntityFilter*, *AttributeType* and *EntityError* object (discussed in section 6.2), Figure 27 depicts composition and relations of the objects exchanged.

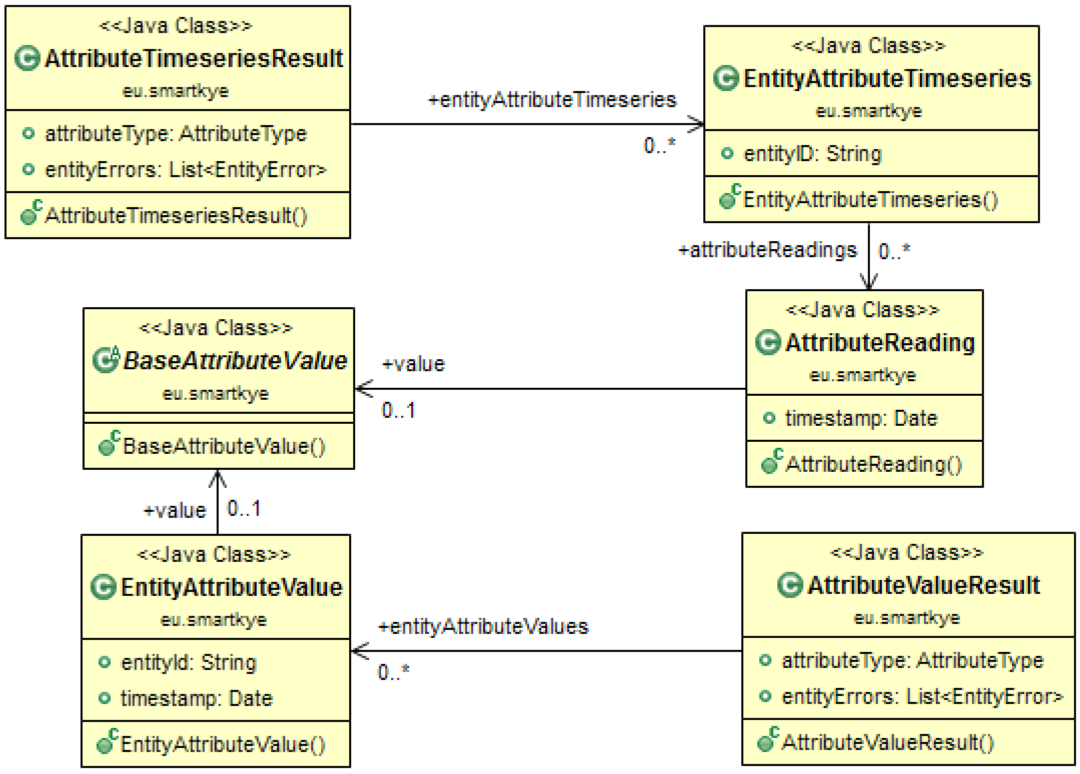


Figure 27 Attribute service related objects

Although all the objects exchanged appear to be simple, (as mentioned) their complexity lies within the extension of the *BaseAttributeValue* object. The lower left depicted object in Figure 27 holds information of single values (or last updated value) and its occurrence for an entity. The *AttributeValueResult* object holds numerous *EntityAttributeValue* objects, as a single request to the platform can be made for multiple entities. Those who fail to deliver their value, for any limitation of delivery, are listed within the *EntityError* list.

Timeseries of attributes are here treated differently, manly due to the structure of the response. The *EntityAttributeTimeseries* holds values listed for a single *AttributeType* and identifies the *entity* that holds them. Further on, many different individual results are listed within the *AttributeTimeseriesResult* object, where actual *AttributeType* is noted. All entities that haven't delivered their attributes will be listed in the *EntityError* list.

6.3.2 Entity

All entities of the system (independently if it's a top level or child entity) are described by the *Entity* object. The object holds a detailed description of the entity and the required structure triggered creation of few additional objects to complete their description. Figure 28 depicts the main object and related ones to this class.

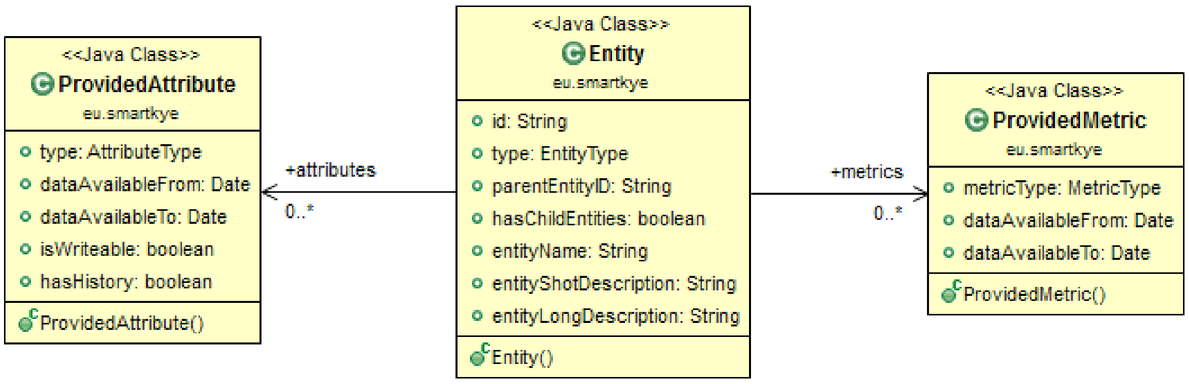


Figure 28 Entity service related objects

As shown, all the entities have a unique ID, name and the basic object *EntityType* (described in section 6.2). Textual description is proposed as short and long description offered to the application developers to decide which type of description is required. If the application allows investigation of an object, the main object holds indication for both directions in hierarchy. The parent entity is described by its unique identification, while the *hasChildEntities* Boolean value is used to indicate if an entity holds children. If the variable is set to true, one may further investigate underlying entities by obtaining their listing through the entity services described in the section 5.2.1.

The other two objects of Figure 28 are introduced due to the limited availability of certain attributes and metrics. Attributes provided by an entity may be time-limited and if their timeseries are requested from the Attribute services (described in the section 5.2.4), where certain entities may not be able to deliver values in certain time frames. As some attributes are writable, an indicator of acceptable modification (for a particular entity) is indicated by the *isWritable* variable. If modifications of attribute's value are registered by an entity, *hasHistory* indicates their existence if only the last attribute value is fetched.

Similar approach is applied to the *ProvidedMetric* object. However, since the metric is a sampled (double) value in point of time, it is considered not to be modifiable and therefore the object holds only the information of timeframe where data is existent.

6.3.3 Message

The message object holds a general description of a message used by MCC to trigger control commands in an EMS. In the Annex section of this document is included a first version of the message schema to be used in the first release of the architecture. The main attributes of a message are basically a messageType and the target entity of the message. Any further detail can be reviewed in the message XML schema included in the Annex section.

6.3.4 Metric

Similarly to the *AttributeService* related objects, the metric services can be called to obtain the last recorded value, or a timeseries of them. Identification of the *MetricType* and *Interval* of interest are basic objects used for service invocations, while *EntityError* is used in their response. All these objects are defined in the section 6.2, thus in this section service related objects will be described. Figure 29 depicts composition of those objects.

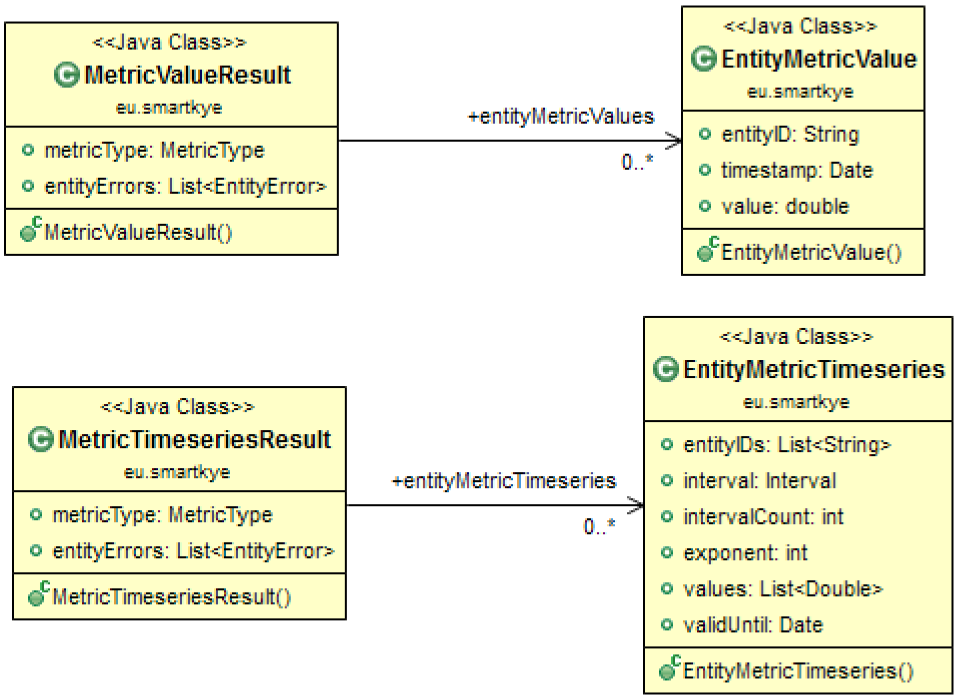


Figure 29 Objects related to the Metric services

On the right side of the figure, one can see the objects holding a single value and entire timeseries of Double data type. As mentioned, s single value is used for retrieving a last sample of the entity providing the metric, thus obvious simplicity is noticeable. The Timeseries object holds more complexity and the following listing describes the meaning of the variables:

- **entityID**: List of entities if aggregation of the metrics is requested
- **interval**: Initial interval of the timeseries (from the section 6.2)
- **intervalCount**: Number of consecutive intervals – always greater than 0 (one indicates only initial interval)
- **exponent**: Metrics can be delivered in different exponents from different entities, e.g., kWh and MWh, thus platform will use exponent variable to execute proper aggregation of provided metrics
- **values**: Values of the intervals
- **validUntil**: if an entity indicates reusability of the provided timeseries, the platform can be enhanced to caching due the performance requirement

Finally, both object from the right side of the figure, are returned within the result objects. Two resulting objects, from the left side of the figure, will hold the *MetricType* of the timeseries and the entities that failed to respond in the *EntityError* listing.

6.3.5 Strategy

The StrategyService provides function of get and store for strategy. The results of strategy are described as a *StrategyAction* object

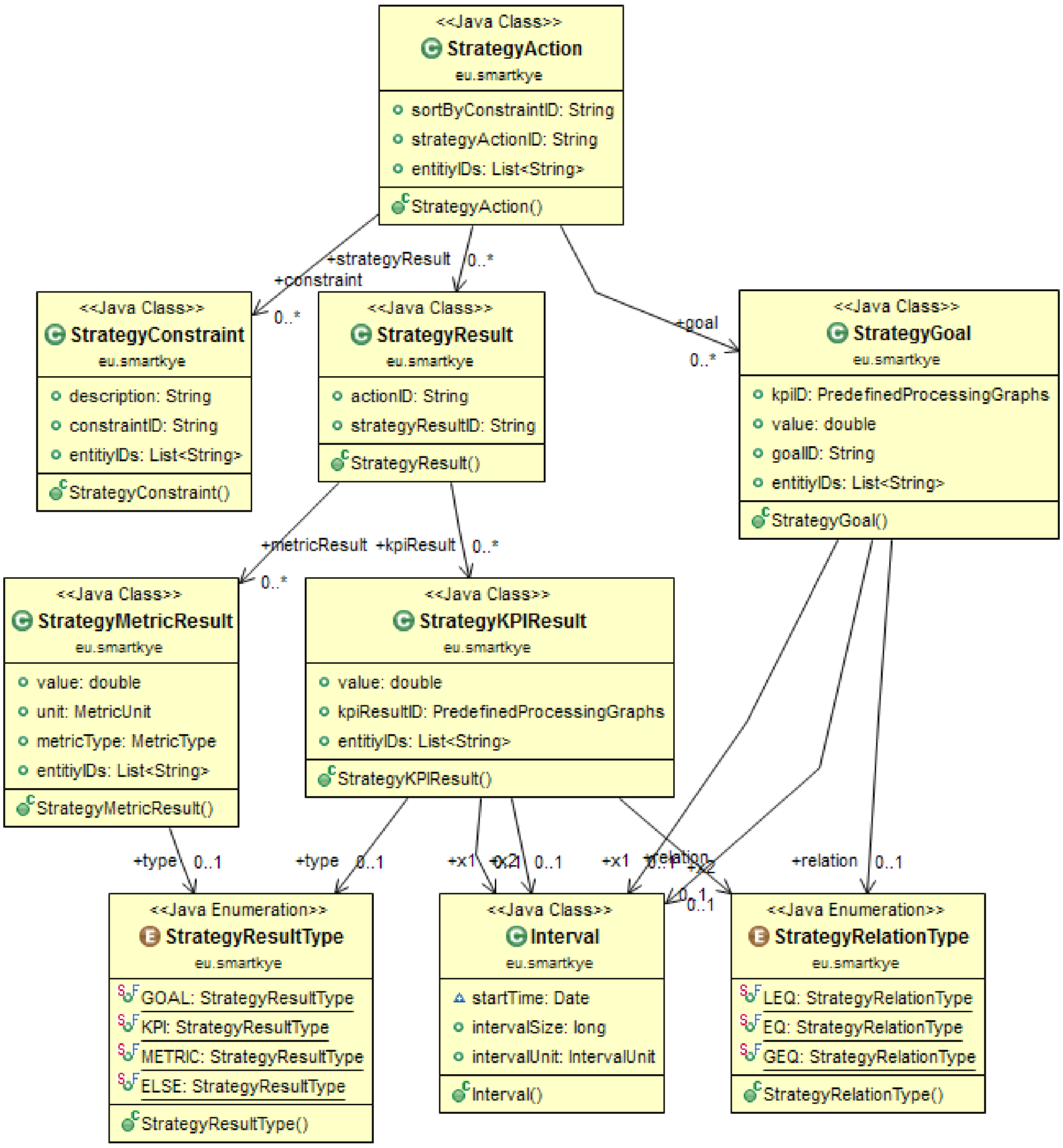


Figure 30 View of Strategy Service

Figure 30 shows the overview of StrategyAction. An object of StrategyAction contains one request of municipality and its results. Following listing describes meaning of the variables:

- **strategyActionID**: unique ID of a *StrategyAction*
- **entityIDs:** List of entities involved in this *StrategyAction*
- **sortByConstraintID**: the KPI or metric, by which the results should be sorted

StrategyAction also contains three lists: *StrategyGoal*, *StrategyConstraint* and *StrategyResult*. *StrategyGoal* and *StrategyConstraint* describe objectives and constraints of municipality, meanwhile *StrategyResult* is the output.

StrategyGoals are the high level objectives of the municipality, which can be selected from the KPIs (as defined in D1.2). For example: Increase energy consumption by 10 %. Multiple objectives are also allowed, e.g. increasing energy production by 10%, while in parallel the cost stays the same. The StrategyGoal must be selected from KPIs and consists of:

- **goalD**: unique ID of a *StrategyGoal*
- **kpiID**: ID of KPI from D1.2

- **x1, x2**: two time windows in one KPI
- **relation**: relation of time series of x1, x2
- **value**: value of the KPI
- **entityIDs:** List of entities involved in this *StrategyGoal*

StrategyConstraints are the conditions in a scenario, which should also be satisfied. Constraints can be chosen both from KPIs and Metrics.

StrategyResults are output of a request. The Strategy Service may provide several strategies for one scenario, which means a *StrategyAction* could contain more than one *StrategyResults*. A *StrategyResult* has one *StrategyResultID* to be identified and actionID to show its StrategyAction. It contains two lists, *StrategyKPIResult* and *StrategyMetricResult*, which are the results of KPI and Metric.

StrategyKPIResult consists of:

- **kpiResultID**: ID of KPI from D1.2
- **x1, x2**: two time windows in one KPI
- **relation**: relation of time series of x1, x2
- **value**: value of the KPI
- **entityIDs:** List of entities which involved in this *StrategyKPIResult*

StrategyMetricResult consists of:

- **metricType**: metric type from D1.2
- **value**: value of the metric
- **unit:** MetricUnit used by the metric
- **entityIDs:** List of entities which involved in this *StrategyMetricResult*

We have to point out that this is work in progress and only contains the draft view of the objects to be exchanged. These will be analysed and specified in more detail in other deliverables such as D5.1.

7 Conclusions

In line with the 2020 objective of achieving a significant reduction of energy consumption and CO_2 emissions in Europe, the SmartKYE project is proposing an open service platform to ease the integration of heterogeneous energy consuming and generating systems at neighbourhood and city level. By doing so, fine-grained information can be acquired by the platform and this has the potential to assist decision takers to take informed decisions. This could be achieved via sophisticated tools that will allow a holistic view of city-wide energy aspects while considering the municipality's multi-angled goals e.g., towards energy efficiency, CO_2 emissions reduction, budget compliance, operational effectiveness etc.

This document presents a service architecture that enables the multiple heterogeneous systems existing nowadays and in future smart cities to interact. The latest include EMSs for smart buildings, public lighting systems, Electric Vehicles infrastructure, energy production facilities, etc. By providing an integrated view on the smart grid city infrastructure, potential energy inefficiencies could be identified and addressed in compliance with the smart city's goals. The latter could result in better and more sustainable energy usage.

Easy to access and utilisation of energy-related data from powerful and intuitive user applications (or cockpits) is expected to be an enabler of better energy decision taking processes in neighbourhoods and cities. ICT technologies and standardization initiatives play a key role; therefore the SmartKYE architecture proposed in this document has been defined considering the latest trends in architecture definition methodologies, Service Oriented Architecture (SOA) reference architectures and technologies.

In order to achieve this objective, along this document was presented the first release of the SmartKYE reference architecture. This is the basic element to trigger all development activities required not only to develop the core elements of the platform, but also those elements required to support the integration of the different systems (EMS) in each of the demonstration pilot sites in the project, namely Barcelona and Crete.

Although only a limited number of systems will be used for proof of concept purposes, the main objective of SmartKYE is the development of an open platform that would ease the integration of any kind of system in a neighbourhood or city. In this sense, the services and data model envisioned to cover the SmartKYE platform requirements (and presented in this document) reach certain levels of abstraction to simplify development of the consuming applications.

Finally, it should be clear that the architecture proposed along this document is a first version, as changes might be mandatory during the implementation of the functionalities described here. In particular, the insights and knowledge obtained from the interaction with the stakeholders during realisation and testing, as well as the planned trials performed in order to validate the approach, will enable the release of a second and final version of the SmartKYE architecture by the end of the project, which will integrate hands-on experiences.

8 References and Acronyms

8.1 Acronyms

Acronyms List	
ABB	Architecture Building Block
AC	Alternating current
ADF	Architecture Development Framework
ADM	Architecture Development Method
API	Application Programming Interface
B2B	Business-to-Business
BAS	Building Automation Systems
BC	Business Cockpit
BIM	Building Information Modelling
BMS	Building Management System
BO&C	Building Optimization and Control
CEP	Complex Event Processing
COTS	Commercial Off-The-Shelf
DC	Direct current
DER	Distributed Energy Resources
DM	Dissemination Manager
DR	Demand Response
DSOs	Distribution System Operator
EC	European Commission
EGS	EMS of Generator System
EISP	Energy Information Service Provider
EMS	Energy Management Systems
EPL	EMS of Public Lighting
ESB	Enterprise Service Bus
ESCOs	Energy Service Company
ETL	Extract-Transform-Load
ETS	EMS of Traffic System
EU	European Union
EV	Electric vehicle
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HV	High voltage
HVAC	Heating ventilation and air conditioning
IaaS	Information as a Service

ICT	Information and communication technologies
ICT4EE	ICT for Energy Efficiency
IREEN	The ICT Roadmap for Energy-Efficient Neighbourhoods
IT	Information Technologies
J2EE	Java 2 Platform Enterprise Edition
JMS	Java Message Service
KPI	Key Performance Indicator
MCC	Monitoring and Control Cockpit
MMI	Man Machine Interface
MUN	Municipality
MV	Medium voltage
NFRs	Non-Functional Requirements
NOBEL	Neighbourhood Oriented Brokerage ELectricity and monitoring system
OASIS	Organization for the Advancement of Structured Information Standards
OESP	Open Energy Service Platform
OMG	Object Management Group
PHEV	Plug-in Hybrid Electrical Vehicles
PLS	Public lighting system
PPP	Public-Private Partnership
PV	Photovoltaic
QoS	Quality of Service
RA	Reference Architecture
RES	Renewable Sources
REST	Representational State Transfer
SCADA	Supervisory Control and Data Acquisition System
SLA	Service-Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TOGAF	The Open Group Architecture Framework
TS	Time Series
TS Data	Time Series Data
UC	Use Case
URL	Uniform resource locator
WP	Work packages
WSDL	Web Services Description Language

8.2 References

[1] S. Carosio, M. Hannus, C. Mastrodonato, E. Delponte, A. Cavallaro, F. Cricchio, S. Karnouskos, J. Pereira-Carlos, C. B. Rodriguez, O. Nilsson, I. Seppä Pinto, T. Sasin, J. Zach, H. van Beurden and T. Anderson, ICT Roadmap for Energy-Efficient Buildings -- Research and Actions, 2013.

[2] IREEN Consortium, “Deliverable 3.3.1 Strategy for European-scale innovation and take-up: Drivers, Policies, Diagnostics & Gaps, Value chain & partnerships,” June 2013.

[3] SEMANCO - Semantic tools for carbon reduction in urban planning, “Workshop in Barcelona. Analyzing and visualizing energy related data in our buildings, towns and cities.,” 2013. [Online]. Available: http://semanco-visualization-workshop.blogspot.com.es/.

[4] Oracle Public Sector, “Oracle’s Smart City Platform - Creating a Citywide Nervous System,” 2012.

[5] BEAMS - Buildings Energy Advanced Management, “D2.2.1 Reference architecture and principles,” 2012.

[6] A. Marqués, S. Karnouskos, P. J. Marrón, R. Sauter, E. Bekiaris and E. Kesidou, “NOBEL- A Neighborhood Oriented Brokerage ELectricity and Monitoring System,” in *1st International ICST Conference on E-Energy*, Athens, Greece, 14-15 October 2010.

[7] S. Karnouskos, M. Serrano, A. Marqués and P. J. Marrón, “Prosumer interactions for Efficient Energy Management in SmartGrid neighborhoods,” in *2nd Workshop on eeBuildings Data Models, CIB Conference W078-W012, European Commission*, Sophia Antipolis, France, 26-28 Oct 2011.

[8] P. Goncalves Da Silva, S. Karnouskos and D. Ilic, “Evaluation of the Scalability of an Energy Market for Smart Grid,” in *IEEE International Conference on Industrial Informatics (INDIN)*, Bochum, Germany, 29-31 Jul 2013.

[9] D. Ilic, P. Goncalves Da Silva, S. Karnouskos and M. Griesemer, “An energy market for trading electricity in smart grid neighbourhoods,” in *IEEE International Conference on Digital Ecosystem Technologies -- Complex Environment Engineering (IEEE DEST-CEE)*, Campione d'Italia, Italy, Jun 2012.

[10] S. Karnouskos, P. Goncalves Da Silva and D. Ilic, “Energy Services for the Smart Grid City,” in *IEEE International Conference on Digital Ecosystem Technologies -- Complex Environment Engineering (IEEE DEST-CEE)*, Campione d'Italia, Italy, Jun 2012.

[11] S. Karnouskos, “Communityware SmartGrid,” in *21st International Conference and Exhibition on Electricity Distribution (CIRED 2011)*, Frankfurt, Germany, 6-9 Jun 2011.

[12] S. Karnouskos, A. Weidlich, J. Ringelstein, A. Dimeas, K. Kok, C. Warmer, P. Selzam, S. Drenkard, N. Hatziargyriou and V. Lioliou, “Monitoring and Control for Energy Efficiency in the Smart House,” in *1st International ICST Conference on E-Energy*, Athens Greece, 14-15 Oct 2010.

[13] K. Kok, S. Karnouskos, J. Ringelstein, A. Dimeas, A. Weidlich, C. Warmer, S. Drenkard, N. Hatziargyriou and V. Lioliou, “Field-testing smart houses for a smart grid,” in *21st International Conference and Exhibition on Electricity Distribution (CIRED 2011)*, Frankfurt, Germany, 6-9 Jun 2011.

[14] The Open Group, The Open Group Architecture Framework (TOGAF) Version 9, 2009.

[15] The Open Group, SOA Reference Architecture, 2011.

[16] A. Josey, “TOGAF® Version 9.1 Enterprise Edition - An introduction,” 2011.

[17] The Open Group, “Using TOGAF to Define and Govern Service-Oriented Architectures,” 2011.

[18] OASIS, OMG, and The Open Group, “Navigating the SOA Open Standards Landscape Around Architecture, Joint White Paper from OASIS, OMG, and The Open Group,” 2009.

[19] D. Hawley, “The Open Group Architecture Principles,” The Open Group Internal Architecture Board, 2008.

[20] FMC Modelling, “FMC Modelling,” [Online]. Available: www.fmc-modeling.org.

[21] Internet Engineering Task Force (IETF), *The OAuth 2.0 Authorization Framework,* 2012.

[22] IREEN Consortium, “Deliverable 2.2.1: Report on state-of-the-art (International) ICT-based innovation projects,” December 2012.

[23] S. Karnouskos, P. Goncalves Da Silva and D. Ilic, “Developing a Web Application for Monitoring and Management of Smart Grid Neighborhoods,” in *IEEE 11th International Conference on Industrial Informatics (INDIN*, Bochum, Germany, 2013.

9 Annex

9.1 SmartKYE pilot sites EMS – PV/Wind Power EMS

9.1.1 PV/Wind Power Plant - Crete

The wind power plant EMS system already running in Crete (called More Care EMS) is attached to a local SCADA system (or DMS system), thus it has two major types of data that will be available for the SmartKYE platform. The two main types of data are:

- Data coming from the SCADA
- Results of EMS algorithms

The following additional data will be also provided also by the More Care EMS:

- Weather Forecasts
- Specific Crete public buildings

Every minute More Care receives via FTP the records from the main SCADA of Crete. The type of data coming from the SCADA is summarized in the following table

Table 13 PV/Wind power plant data available - Crete

	Type	Online Data	Historical Data
Substations (HV/MV)	Active Power	1min	Hourly data
	Reactive Power	1min	Hourly data
	Current	1min	Hourly data
	Voltage	1min	Hourly data
Wind Farms	Active Power	1min	Hourly data
	Reactive Power	1min	Hourly data
	Wind Speed	1min	Hourly data
	Wind Direction	1min	Hourly data
	Curtailment set-point	Event	Event
Power Station	Active Power	1min	Hourly data
	Active Power (Net)*	1min	Hourly data
	Reactive Power	1min	Hourly data
	Current	1min	Hourly data
	Voltage	1min	Hourly data
PV	Active Power	5min	Hourly data
	Reactive Power	5min	Hourly data
General	Frequency	5min	Hourly data
	Total Load	5min	Hourly data
	Total Production	5min	Hourly data
	Total Load per	5min	Hourly data

	region		
	Total RES production	5min	Hourly data
	Penetration	5min	Hourly data

*Net production is the power injected into the grid. Each generator provides power also to auxiliary systems in the station.

More Care receives also weather forecasts from the SKIRON system. The weather forecasts data include:

Table 14 PV/Wind power plant data available (weather data) - Crete

Type	Interval	Horizon	Values
Temperature	1h	5d	Schedule of 96 values
Pressure	1h	5d	Schedule of 96 values
Wind Speed	1h	5d	Schedule of 96 values
Wind Direction	1h	5d	Schedule of 96 values
Cloud Index	1h	5d	Schedule of 96 values
Irradiation	1h	5d	Schedule of 96 values
Humidity	1h	5d	Schedule of 96 values

Finally, the MORE CARE system aims to assist the operators of island systems by proposing optimal operating scenarios for the various power units, as well as the various actions needed to avoid dangerous situations, which might result from a poor prediction of load or weather or pre-selected disturbances. The insurance of increased security and reliability of the system will allow maximization of renewable penetration. As previously indicated, the additional set of data provided by this EMS in Crete is detailed in the following table:

Table 15 PV/Wind power plant data available (algorithms results) - Crete

Type	Interval	Horizon	Values
Short Term Load Forecast	20min	8h	Schedule of 24 values
Economic Dispatch	20min	Next time step	
Short Term Wind Forecast	20min	8h	Schedule of 24 values
Short Term Unit Commitment	20min	8h	Schedule of 24 values
Stochastic Load Flow	20min	Next time step	
Long Term Load Forecast	1h	24/48h	Schedule of 24/48 values
Long Term Wind Forecast	1h	24/48h	Schedule of 24/48 values
Long Term Unit Commitment	1h	24/48h	Schedule of 24/48 values

9.1.2 PV/Wind Power Plant - Barcelona

The PV/Wind EMS available in Barcelona is simpler than the one previously described, although it still provides some valuable information for SmartKYE project. The system is based on the following components:

- Weather station: Provides information on wind speed.
- Windmill tachometer provides information on turbine speed revolution.
- Inverter provides information on:
    - DC voltage: voltage from the windmill to the inverter.
    - DC current.
    - AC power: power delivered to the grid.

All this information is transmitted to a sheeva plug (embedded PC), which is responsible for sending the data to a web platform every minute. The following data (and services) provided by this EMS will be available in SmartKYE project:

Physical elements:

1. Power inverters:
    a. Information:
        - Id.
        - Operational status
        - Configuration parameters.
        - Voltage DC side.
        - Current DC side.
        - Power DC side.
        - Network frequency
        - Injected power.
        - Current at each phase.
        - Daily production.
        - Total production.
        - Alarms.

    b. Services:
        - Set configuration parameters.
2. Weather station:
    a. Information:
        - Temperature.
        - Pressure.
        - Wind speed.
        - Maximum recent wind speed.
        - Average wind speed.
        - Humidity.

Logical Elements

1. Maximum speed control.
    a. Information:
        - Windmill rpm.
        - Configuration parameters.

    b. Services:
        - Set configuration parameters.
        - Stops wind turbine.
2. Voltage control.
    b. Information:
        - Windmill voltage output.

- Configuration parameters.

c. Services:

- Set configuration parameters.
- Stops wind turbine.

9.2 SmartKYE pilot sites EMS – Public buildings

9.2.1 Public buildings EMS - Crete

In the case of Crete, energy consumption data from several public buildings will be provided to SmartKYE platform by the More Care EMS system. The data available is summarized in the following table:

Table 16 Public buildings EMS - Crete

Type	Interval
Total Consumption	1m
Consumption per feeder or device	1m
Voltage	1m
Frequency	1m

9.2.2 Public buildings EMS - Barcelona

In the case of Barcelona, data from a building and an office area from this building will be provided to SmartKYE platform. The data available is summarized in the following table:

Table 17 Public buildings EMS - Crete

Type	Interval
Building data:	
Solar radiation	10min
Outside temperature	10min
Building HVAC main valves	10min
Office area:	
Office global energy consumption (disaggregated in lighting and power plugs consumption)	10min
Ambient parameters (temperature, brightness and humidity)	10min
Disaggregated power consumption of electronic equipment	10min
Energy price	10min
Energy mix	10min

9.3 SmartKYE pilot sites EMS – Public Lighting System

For the public lighting EMS to be integrated with the SmartKYE platform, the following data and services will be provided:

1. **Segment Controller**
   a. **Information**
      - Configuration parameters.
      - Communication Status.
      - Operational status.
      - Line Activation status.
      - Communication status with Flow Regulator (RF).
      - RF operation status.
      - List of LPs and LPCs.
      - Operating status of the LPCs.
      - SC alarms.
      - RF alarms.
      - Electrical parameters (V, I, cos φ).
      - Total power and power for each line.
      - Maximum power.
      - Relative power (Total Power / Maximum Power).
      - Power consumption (Active and reactive energy).

   b. **Services:**
      - Set Configuration Parameters.
      - Activation of the lines.

2. **Point of light Controller**
   a. **Information**
      - Configuration parameters.
      - Communication Status.
      - Operational status.
      - Control Mode.
      - Activation Status.
      - LPC alarms.
      - Electrical parameters (V, I, cos φ).
      - Power.
      - Power consumption (Active and reactive energy).

   b. **Services:**
      - Set Control mode.
      - Activation.

3. **Points of light**
   a. **Information**
      - Identifier.
      - Short Description.
      - V.
      - Power.

- ♦ V Evolution.
- ♦ Power Evolution.
- ♦ Consumption Evolution.
- ♦ Operational status.
- ♦ Control Mode.
- ♦ Alarms.
- ♦ Historical status.
- ♦ Alarm History.
- ♦ Activation Status.
- ♦ Actual Desired Lighting Level
- ♦ Desired Lighting Level.
- ♦ Current Lighting level.
- ♦ Activation Hours.
- ♦ Number of activations.

b. Services:

- ♦ Set Control mode.
- ♦ Enable Lighting Level.

4. Group of Points of light

a. Information

- ♦ Identifier.
- ♦ Short Description.
- ♦ Long description.
- ♦ List of LPs.
- ♦ V mean.
- ♦ Power.
- ♦ Maximum power.
- ♦ Relative power.
- ♦ V mean Evolution.
- ♦ Power Evolution.
- ♦ Consumption Evolution.
- ♦ LPCs operational status.
- ♦ Geographic Polygon surfaces.
- ♦ Activation Status.
- ♦ Control Mode.
- ♦ Actual Desired Lighting Level.
- ♦ Desired Lighting Level.
- ♦ Current Lighting Level.

b. Services:

- ♦ Set Control mode.
- ♦ Enable Lighting Level.

9.4 SmartKYE pilot sites EMS – Electric Vehicle Infrastructure

In the case of the EV EMS, the following information will be available:

- **Information**

  - Power consumption of the latest 15’ period
  - User Id of the owner of the car

- **Services:**
  - Retrieve current data for plug X
  - Retrieve historical data of plug X from date D1 to date D2

9.5 SmartKYE architecture building blocks and capabilities mapping

The following table provides the result of the analysis performed following the Open Group SOA RA. In this table the SmartKYE architecture building blocks and capabilities are mapped.

Table 18 SmartKYE SOA RA – Capabilities and ABB mapping

SOA RA Layer	Capability category	Building Block Name	Supported Capabilities	Comments
Service Layer	Service definition	Service	1	This ABB represents a published service that offers certain functionalities that business performs to achieve a business outcome or a milestone. Typically, a service is published to the Service Repository ABB in the Governance Layer during design time for search and re-use and the Service Registry ABB in the Governance Layer during runtime for service virtualization. A service is typically represented in a standard description language (e.g., WSDL) describing its accessible interfaces (e.g., function or method signatures).
	Service Runtime Enablement	Service container	3	
			5	
			6	
		Governance Layer: Service registry	3	The Service Registry ABB in the Governance Layer supports the storage of and access to bindings at runtime to services hosted in the Service Container/Gateway ABB. It manages service versioning allowing the appropriate service to be picked.
	Access control	QoS Layer: Policy Manager	12	
	Consumer services	Client (channel)	2	This ABB interacts with the Presentation Controller ABB to use the underlying services, integrating the consumer of services supported by the SOA RA. It is the element in the SOA which the consumer interacts with. As such, it provides the point interaction for the consumer. The key responsibilities include dealing with the nature of interaction that the client has with the consumer.

Consumer Layer	Presentation services	Presentation adapter	4	This ABB is responsible for integrating the client with the rest of the Consumer Layer.
		Presentation controller	3	This ABB is responsible for handling the orchestration, decomposition, and composition of the view rendered by the client.
			6	
			7	
		Presentation flow manager	5	This ABB is responsible for supporting navigation and control flow in the Consumer Layer. It is an important part in the assemblage of a view component to send back and render in the client.
		Composite View	3	This ABB is responsible for assembling data received from various services and creates a composite view which is orchestrated and then passed on to the client for rendering.
		Consumer/User Profile Manager	4	This ABB is responsible for supporting the personalization of the interface and the presentation to a particular consumer's wants and needs. It will be used both by the Client ABB and Presentation Controller ABB. It can be used for controlling individual consumer features or the creation of profiles based on roles.
			6	I would support this in smart key to enable/restrict different profiles to access specific services
		Personalization Manager	6	
	Backend Integration	Integration Layer: Integration controller	8	
		Integration Layer: Data trans-	9	

		former		
	Caching and Streaming Content	Cache	10	
			11	
	Security and Privacy	Integration Layer:	12	12 deals with providing authentication/authorization capabilities, while 13 with filtering to control access and 14 to monitor the usage of consumer layer components.
		Policy manager	13, 14	
Integration Layer	Communication, Service Interaction and Integration	Integration Controller/ Integration Gateway	1	This ABB serves as an entry point to this layer. This ABB is thus responsible for interfacing with other ABBs in this layer and managing the interaction flow among the ABBs in this layer.
			2	
			5	
		Adapter	1, 2, 3, 6, 7	This ABB is responsible for the interfacing/connectivity of SOA RA layers of a solution to external systems and components and taking a call (message) to the endpoint.
		Mediator	1-7	This ABB is responsible for handling the service request/response interaction. It also supports the transformation between message formats, conversion of protocols, and routing of service call/messages to the service provider. It uses the Data Transformer ABB and optionally the Semantic Transformer ABB for the transformations
		Router	5	This ABB is responsible to route messages between service consumer/requestor and service provider including those based on both content-based routing, straight through message passing
	Quality of Service	Transaction Manager	13	This ABB manages transactions and encapsulates transaction handling
		Exception Handler	14	This ABB is responsible for handling system exceptions raised during service invocation and message passing. System exceptions are caused due to software or hardware errors

	Security	Quality of Service Layer: Access Controller	15	
QoS Layer	Security Management	Security Manager	9	OAuth + SSL
			10	
			11	
		Identity, Access, and Entitlement Manager	13	
		Data and Information Protector	15	
		Access Controller	21	
	Information Service	Information Services Gateway	1	
		Data Aggregator	3	This ABB is responsible for efficiently joining information – for example, structured and unstructured data – from multiple sources without creating data redundancy to help form a unified data view/model. This could be required for example for some services in the OESP.
		Hierarchy and Relationship Manager	5	This ABB is responsible for managing the data hierarchies, groupings, relationships such as parent-child relationships, and relationships between enterprise data.
		Data Representation Manager	11	This ABB is responsible for handling representation of data from various data sources in a unified data format and for creation of unified views of data. In other words, this ABB intends to hide various data sources

Information Layer			and present data in uniform formats to other ABBs.
	Data Sourcing Manager	11	This ABB is responsible for enabling access to different data sources using different protocols. It provides unified access to data in files, databases, etc. It uses an Adapter ABB from the Integration Layer to provide the ability to integrate with data sources in different solution platforms (external data sources). Examples may be relational sources (e.g., DB2, Oracle, or SQL Server databases), other structured data (e.g., Excel .CSV, web service request responses in XML format, and hierarchical stores on mainframes such as IMS), as well as unstructured data stores (such as images and documents). It manages interactions with the data sources in the Solution Platform and other SOA RA layers, but it is not responsible for addressing data and protocol transformation.
	Data Cache	15	This ABB is responsible for the caching of data in support of the data virtualization/information services capability. It enables addressing variations in temporal availability of data as well as improvement of performance. The variance in temporal availability of data is an issue associated with different data sources having different schedules for data being available; for example, one data source could be a time-based file feed, the other a mainframe batch program, and the third a real-time relational database. In such a scenario, for the consistent update and availability of data, it is useful to be able to cache it in some form.
	Integration Layer: Data Transformer	12	

		Data Consolidator	9	This ABB is responsible for extracting relevant information from sources, transforming the information into the appropriate integrated form, and loading the information into the target repository. This ABB supports Extract-Transform-Load (ETL) from one or more source systems into a target system.
	Information Security and Protection	Quality of Service Layer: Access Controller	19	
		Quality of Service Layer: Data-Driven Access Controller	20	
		Analytics Visualization Engine	24	
Governance Layer	SOA Metadata Storage and Management	Service Repository	8	This ABB integrates with the Service Performance Manager ABB to support the runtime information collection and storage in order for users to evaluate service performance. It acts as a design-time repository to store and locate metadata about services, including descriptions of the service contract, information about the QoS policies and security, versioning information, and runtime information such as end-points.
			9	
			13	
			14	

		Service Registry	13-14	The ABB is responsible for allowing advertising and discovery of available services and supports the runtime binding of services and service virtualization. Think of this ABB as a runtime service repository. Services are available to the solution as well as governance processes. Advertisement of services should be governed. It contains metadata about services, including descriptions of the service contract, information about the QoS policies and security, versioning information, and runtime information such as end-points. This ABB may leverage the design-time Service Repository ABB to fetch metadata about services to fulfil the runtime needs of SOA such as dynamic/runtime binding and service virtualization. Standards for registries include UDDI.
	Management	Change Control Manager	31	

9.6 Exchanged Objects (XSD)

This section holds the XML Schema Definitions of the objects consumed by the energy services

9.6.1 Basic Objects

entityType – types of entities existent within the OESP platform

```
<xs:simpleType name="entityType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="BUILDING"/>
      <xs:enumeration value="BUILDING_EMS"/>
      <xs:enumeration value="BUILDING_HVAC_CONTROL"/>
      <xs:enumeration value="BUILDING_LIGHTING_CONTROL"/>
      <xs:enumeration value="BUILDING_OFFICE"/>
      <xs:enumeration value="ELMETER"/>
      <xs:enumeration value="EV_CHARGING_STATION"/>
      <xs:enumeration value="EV_CHARGING_STATION_CONTROL_ROOM"/>
      <xs:enumeration value="POINT_OF_LIGHT"/>
      <xs:enumeration value="PUBLIC_LIGHTING_CABINET"/>
      <xs:enumeration value="PUBLIC_LIGHTING_CONTROL_ROOM"/>
      <xs:enumeration value="PV_INVERTER"/>
      <xs:enumeration value="PV_PANEL"/>
      <xs:enumeration value="PV_POWER_PLANT"/>
      <xs:enumeration value="SEGMENT_CONTROLLER"/>
      <xs:enumeration value="UNSPECIFIED"/>
      <xs:enumeration value="WINDFARM"/>
      <xs:enumeration value="WINDMILL"/>
    </xs:restriction>
  </xs:simpleType>
```

metricType

```
<xs:simpleType name="metricType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="ENTITY_COUNT"/>
      <xs:enumeration value="ENTITIES_USED"/>
      <xs:enumeration value="ENERGY_CONSUMPTION"/>
      <xs:enumeration value="ENERGY_PRODUCTION"/>
      <xs:enumeration value="POTENTIAL_ENERGY_PRODUCTION"/>
      <xs:enumeration value="ENERGY_CONSUMPTION_FORECAST"/>
      <xs:enumeration value="ENERGY_PRODUCTION_FORECAST"/>
      <xs:enumeration value="ENERGY_MANAGED"/>
      <xs:enumeration value="ENERGY_MANAGED_REQUESTED"/>
      <xs:enumeration value="ENERGY_PRODUCTION_CURTAILMENT_FORECAST"/>
      <xs:enumeration value="ENERGY_PRODUCTION_CURTAILMENT"/>
      <xs:enumeration value="ENERGY_CONVERSION_EFFICIENCY"/>
      <xs:enumeration value="STATE_OF_CHARGE"/>
      <xs:enumeration value="POTENTIAL_CAPACITY_OF_POWER_PRODUCTION"/>
      <xs:enumeration value="STORAGE_CAPACITY"/>
      <xs:enumeration value="MAX_PEAK_POWER_DEMAND"/>
      <xs:enumeration value="ACTIVE_POWER"/>
      <xs:enumeration value="REACTIVE_POWER"/>
      <xs:enumeration value="VOLTAGE"/>
      <xs:enumeration value="CURRENT"/>
      <xs:enumeration value="POWER_FACTOR"/>
      <xs:enumeration value="AC_FREQUENCY"/>
      <xs:enumeration value="OPERATIONAL_COST"/>
      <xs:enumeration value="ENERGY_PRICE_PER_KWH"/>
      <xs:enumeration value="ENERGY_COST"/>
      <xs:enumeration value="CO2_EMISSIONS"/>
      <xs:enumeration value="TEMPERATURE"/>
      <xs:enumeration value="ATHMOSPHERIC_PRESSURE"/>
      <xs:enumeration value="WIND_SPEED"/>
      <xs:enumeration value="WIND_DIRECTION"/>
      <xs:enumeration value="CLOUD_INDEX"/>
      <xs:enumeration value="HUMIDITY"/>
      <xs:enumeration value="ENERGY"/>
      <xs:enumeration value="LOAD"/>
      <xs:enumeration value="PRODUCTION"/>
      <xs:enumeration value="PENETRATION"/>
      <xs:enumeration value="ECONOMIC_DISPATCH"/>
    </xs:restriction>
  </xs:simpleType>
```

predefinedProcessingGraphs – default KPIs offered by the OESP platform

```
<xs:simpleType name="predefinedProcessingGraphs">
    <xs:restriction base="xs:string">
      <xs:enumeration value="ASSET_PENETRATION_COMPARISON"/>
      <xs:enumeration value="ENERGY_MANAGED_ACHIEVED"/>
      <xs:enumeration value="ENERGY_PRODUCTION_COMPARISON"/>
      <xs:enumeration value="ENERGY_PENETRATION_COMPARISON"/>
      <xs:enumeration value="ENERGY_PRODUCTION_CURTAILMENT"/>
      <xs:enumeration value="ENERGY_PRODUCTION_PENETRATION"/>
      <xs:enumeration value="ENERGY_PRODUCTION_RATIO"/>
      <xs:enumeration value="ENERGY_EXCHANGE"/>
      <xs:enumeration value="ENERGY_SURPLUS"/>
      <xs:enumeration value="FORECAST_ENERGY_CONSUMPTION_ACCURACY"/>
      <xs:enumeration value="FORECAST_ENERGY_PRODUCTION_ACCURACY"/>
      <xs:enumeration value="INVESTMENT_RATION_FROM_AVAILABLE_INVESTMENTS"/>
      <xs:enumeration value="POTENTIAL_ENERGY_FLEXIBILITY"/>
      <xs:enumeration value="POTENTIAL_ENERGY_PRODUCTION_FROM_MAX"/>
      <xs:enumeration value="ASSET_COUNT_DIFFERENCE"/>
      <xs:enumeration value="ASSET_PENETRATION_CHANGE"/>
      <xs:enumeration value="CONSUMPTION_FORECAST_ACCURACY_CHANGE"/>
      <xs:enumeration value="CURTAILED_ENERGY_DIFFERENCE"/>
      <xs:enumeration value="EMISSIONS_CHANGE"/>
      <xs:enumeration value="EMISSIONS_DIFFERENCE"/>
      <xs:enumeration value="ENERGY_CONSUMPTION_CHANGE"/>
      <xs:enumeration value="ENERGY_CONSUMPTION_DIFFERENCE"/>
      <xs:enumeration value="ENERGY_CONSUMPTION_FLEXIBILITY_CHANGE"/>
      <xs:enumeration value="ENERGY_COST_CHANGE"/>
      <xs:enumeration value="ENERGY_COST_DIFFERENCE"/>
      <xs:enumeration value="ENERGY_EXCHANGED_DIFFERENCE"/>
      <xs:enumeration value="ENERGY_MANAGEMENT_ACHIEVEMENT_CHANGE"/>
      <xs:enumeration value="ENERGY_MANAGEMENT_ACHIEVEMENT_DIFFERENCE"/>
      <xs:enumeration value="ENERGY_MANAGEMENT_DIFFERENCE"/>
      <xs:enumeration value="ENERGY_PRODUCTION_CHANGE"/>
      <xs:enumeration value="ENERGY_PRODUCTION_DIFFERENCE"/>
      <xs:enumeration value="ENERGY_PRODUCTION_RATIO_DIFFERENCE"/>
      <xs:enumeration value="ENTITY_USAGE_DIFFERENCE"/>
      <xs:enumeration value="GOAL_ACHIEVEMENT_CHANGE"/>
      <xs:enumeration value="INVESTMENT_CHANGE"/>
      <xs:enumeration value="OPERATIONAL_COST_CHANGE"/>
      <xs:enumeration value="OVERALL_STATE_OF_CHARGE_CHANGE"/>
      <xs:enumeration value="POTENTIAL_ENERGY_FLEXIBILITY_CHANGE"/>
      <xs:enumeration value="POTENTIAL_ENERGY_PRODUCTION_CHANGE"/>
      <xs:enumeration value="PRODUCTION_FORECAST_ACCURACY_CHANGE"/>
      <xs:enumeration value="PRODUCTION_FROM_MAXIMUM_POTENTIAL_CHANGE"/>
      <xs:enumeration value="PRODUCTION_PENETRATION_CHANGE"/>
      <xs:enumeration value="TOTAL_ENERGY_PREDICTABILITY_CHANGE"/>
      <xs:enumeration value="WEIGHTED_ENERGY_EFFICIENCY_DIFFERENCE"/>
      <xs:enumeration value="WEIGHTED_ENERGY_PRICE_DIFFERENCE"/>
    </xs:restriction>
  </xs:simpleType>
```

attributeType – envisioned attributes for the known SmartKYE entities

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

```
    <xs:restriction base="xs:string">
      <xs:enumeration value="GEO_LOCATION"/>
      <xs:enumeration value="POSTAL_ADDRESS"/>
      <xs:enumeration value="CONNECTED_ON_GRID"/>
      <xs:enumeration value="ELECTRICAL_MEASURES"/>
      <xs:enumeration value="CONNECTED_TO_PHASE"/>
      <xs:enumeration value="SWITCHED"/>
      <xs:enumeration value="DIMMING_VALUE"/>
      <xs:enumeration value="MEASUREMENT_DEVICE"/>
      <xs:enumeration value="WORKING_MODE"/>
      <xs:enumeration value="BUILDING_PROPERTIES"/>
    </xs:restriction>
  </xs:simpleType>
```

entityFilter – OESP filter for entities of interest

```
<xs:complexType name="entityFilter">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entities"
nillable="true" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityParents"
nillable="true" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityAncestors"
nillable="true" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityGroups"
nillable="true" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityTypes"
nillable="true" type="tns:entityType"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="providedMetricTypes"
nillable="true" type="tns:metricType"/>
      <xs:element maxOccurs="unbounded" minOccurs="0"
name="providedAttributeTypes" nillable="true" type="tns:attributeType"/>
    </xs:sequence>
  </xs:complexType>
```

entityError

```
  <xs:complexType name="entityError">
    <xs:sequence>
      <xs:element minOccurs="0" name="entityID" type="xs:string"/>
      <xs:element minOccurs="0" name="error" type="tns:smartkyeException"/>
    </xs:sequence>
  </xs:complexType>
```

intervalUnit

```
  <xs:complexType name="intervalUnit">
    <xs:restriction base="xs:string">
      <xs:enumeration value="SECOND"/>
      <xs:enumeration value="MONTH"/>
      <xs:enumeration value="YEAR"/>
    </xs:restriction>
  </xs:complexType>
```

smartkyeException

```
  <xs:complexType name="smartkyeException">
    <xs:complexContent>
      <xs:extension base="tns:exception">
        <xs:sequence>
          <xs:element minOccurs="0" name="description" type="xs:string"/>
          <xs:element minOccurs="0" name="type" type="tns:smartkyeExceptionType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

smartkyeExceptionType

```
<xs:simpleType name="smartkyeExceptionType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="BASE"/>
    </xs:restriction>
  </xs:simpleType>
```

9.6.2 Service Related Objects

9.6.2.1 Entity

```
  <xs:complexType name="entityProvidedMetric">
    <xs:sequence>
      <xs:element minOccurs="0" name="entityID" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="providedMetrics"
nillable="true" type="tns:providedMetric"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="providedMetric">
    <xs:sequence>
      <xs:element minOccurs="0" name="metricType" type="tns:metricType"/>
      <xs:element minOccurs="0" name="dataAvailableFrom" type="xs:dateTime"/>
      <xs:element minOccurs="0" name="dataAvailableTo" type="xs:dateTime"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="entity">
    <xs:sequence>
      <xs:element minOccurs="0" name="id" type="xs:string"/>
      <xs:element minOccurs="0" name="type" type="tns:entityType"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="metrics"
nillable="true" type="tns:providedMetric"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="attributes"
nillable="true" type="tns:providedAttribute"/>
      <xs:element minOccurs="0" name="parentEntityID" type="xs:string"/>
      <xs:element name="hasChildEntities" type="xs:boolean"/>
      <xs:element minOccurs="0" name="entityName" type="xs:string"/>
      <xs:element minOccurs="0" name="entityShotDescription" type="xs:string"/>
      <xs:element minOccurs="0" name="entityLongDescription" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="providedAttribute">
    <xs:sequence>
      <xs:element minOccurs="0" name="type" type="tns:attributeType"/>
      <xs:element minOccurs="0" name="dataAvailableFrom" type="xs:dateTime"/>
      <xs:element minOccurs="0" name="dataAvailableTo" type="xs:dateTime"/>
      <xs:element name="isWriteable" type="xs:boolean"/>
      <xs:element name="hasHistory" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
```

9.6.2.2 Metric

entityMetricValue

```
<xs:complexType name="entityMetricValue">
    <xs:sequence>
      <xs:element minOccurs="0" name="entityID" type="xs:string"/>
      <xs:element minOccurs="0" name="timestamp" type="xs:dateTime"/>
      <xs:element name="value" type="xs:double"/>
    </xs:sequence>
  </xs:complexType>
```

metricValueResult

```
<xs:complexType name="metricValueResult">
    <xs:sequence>
      <xs:element minOccurs="0" name="metricType" type="tns:metricType"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityMetricValues"
nillable="true" type="tns:entityMetricValue"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityErrors"
nillable="true" type="tns:entityError"/>
    </xs:sequence>
  </xs:complexType>
```

entityMetricTimeseries

```
  <xs:complexType name="entityMetricTimeseries">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityIDs"
nillable="true" type="xs:string"/>
      <xs:element minOccurs="0" name="interval" type="tns:interval"/>
      <xs:element name="intervalCount" type="xs:int"/>
      <xs:element name="exponent" type="xs:int"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="values"
nillable="true" type="xs:double"/>
      <xs:element minOccurs="0" name="validUntil" type="xs:dateTime"/>
    </xs:sequence>
  </xs:complexType>
```

metricTimeseriesResult

```
<xs:complexType name="metricTimeseriesResult">
    <xs:sequence>
      <xs:element minOccurs="0" name="metricType" type="tns:metricType"/>
      <xs:element maxOccurs="unbounded" minOccurs="0"
name="entityMetricTimeseries" nillable="true" type="tns:entityMetricTimeseries"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityErrors"
nillable="true" type="tns:entityError"/>
    </xs:sequence>
  </xs:complexType>
```

9.6.2.3 Attribute

entityAttributeValue

```
  <xs:complexType name="entityAttributeValue">
    <xs:sequence>
      <xs:element minOccurs="0" name="entityId" type="xs:string"/>
      <xs:element minOccurs="0" name="timestamp" type="xs:dateTime"/>
      <xs:element minOccurs="0" name="value" type="tns:baseAttributeValue"/>
    </xs:sequence>
  </xs:complexType>
```

attributeValueResult – Resulting value for attribute timeseries

```
<xs:complexType name="attributeValueResult">
    <xs:sequence>
      <xs:element minOccurs="0" name="attributeType" type="tns:attributeType"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityAttributeValues"
nillable="true" type="tns:entityAttributeValue"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityErrors"
nillable="true" type="tns:entityError"/>
    </xs:sequence>
  </xs:complexType>
```

attributeReading

```
  <xs:complexType name="attributeReading">
    <xs:sequence>
      <xs:element minOccurs="0" name="timestamp" type="xs:dateTime"/>
      <xs:element minOccurs="0" name="value" type="tns:baseAttributeValue"/>
    </xs:sequence>
  </xs:complexType>
```

entityAttributeTimeseries

```
<xs:complexType name="entityAttributeTimeseries">
    <xs:sequence>
      <xs:element minOccurs="0" name="entityID" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="attributeReadings"
nillable="true" type="tns:attributeReading"/>
    </xs:sequence>
  </xs:complexType>
```

attributeTimeseriesResult

```
  <xs:complexType name="attributeTimeseriesResult">
    <xs:sequence>
      <xs:element minOccurs="0" name="attributeType" type="tns:attributeType"/>
      <xs:element maxOccurs="unbounded" minOccurs="0"
name="entityAttributeTimeseries" nillable="true"
type="tns:entityAttributeTimeseries"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityErrors"
nillable="true" type="tns:entityError"/>
    </xs:sequence>
  </xs:complexType>
```

9.6.2.4 CEP

processingGraphParameter

```
<xs:complexType name="processingGraphParameter">
    <xs:sequence>
      <xs:element minOccurs="0" name="name" type="xs:string"/>
      <xs:element minOccurs="0" name="value" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

cepMetricTimeseriesResult

```
<xs:complexType name="cepMetricTimeseriesResult">
    <xs:sequence>
      <xs:element minOccurs="0" name="processingGraphHandle" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0"
name="entityMetricTimeseries" nillable="true" type="tns:entityMetricTimeseries"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityErrors"
nillable="true" type="tns:entityError"/>
    </xs:sequence>
  </xs:complexType>
```

entityMetricTimeseries

```
<xs:complexType name="entityMetricTimeseries">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entityIDs"
nillable="true" type="xs:string"/>
      <xs:element minOccurs="0" name="interval" type="tns:interval"/>
      <xs:element name="intervalCount" type="xs:int"/>
      <xs:element name="exponent" type="xs:int"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="values"
nillable="true" type="xs:double"/>
      <xs:element minOccurs="0" name="validUntil" type="xs:dateTime"/>
    </xs:sequence>
  </xs:complexType>
```

processingGraph

```
  <xs:complexType name="processingGraph">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="components"
nillable="true" type="tns:processingGraphComponent"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="connections"
nillable="true" type="tns:processingGraphConnection"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="parameters"
nillable="true" type="tns:processingGraphUnresolvedParameter"/>
    </xs:sequence>
  </xs:complexType>
```

processingGraphComponent

```
<xs:complexType name="processingGraphComponent">
    <xs:sequence>
      <xs:element minOccurs="0" name="name" type="xs:string"/>
      <xs:element minOccurs="0" name="type" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="inPorts"
nillable="true" type="tns:processingGraphPort"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="outPorts"
nillable="true" type="tns:processingGraphPort"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="resolvedParameters"
nillable="true" type="tns:processingGraphParameter"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="boundParameters"
nillable="true" type="tns:processingGraphParameterBinding"/>
    </xs:sequence>
  </xs:complexType>
```

processingGraphPort

```
<xs:complexType name="processingGraphPort">
    <xs:sequence>
      <xs:element minOccurs="0" name="name" type="xs:string"/>
      <xs:element minOccurs="0" name="type" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

processingGraphParameterBinding

```
<xs:complexType name="processingGraphParameterBinding">
    <xs:sequence>
      <xs:element minOccurs="0" name="name" type="xs:string"/>
      <xs:element minOccurs="0" name="boundTo" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

processingGraphUnresolvedParameter

```
  <xs:complexType name="processingGraphUnresolvedParameter">
    <xs:sequence>
      <xs:element minOccurs="0" name="name" type="xs:string"/>
      <xs:element minOccurs="0" name="type" type="xs:string"/>
      <xs:element minOccurs="0" name="defaultValue" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

processingGraphConnection

```
  <xs:complexType name="processingGraphConnection">
    <xs:sequence>
      <xs:element minOccurs="0" name="sourceComponent" type="xs:string"/>
      <xs:element minOccurs="0" name="sourcePort" type="xs:string"/>
      <xs:element minOccurs="0" name="destinationComponent" type="xs:string"/>
      <xs:element minOccurs="0" name="destinationPort" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

9.6.3 Message

message

```
<xs:complexType name=" message ">
    <xs:sequence>
      <xs:element minOccurs="0" name="type" type="tns: messageType "/>
       <xs:element minOccurs="0" name="content" type="xs: anyType "/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name=" entityWithMessage ">
    <xs:sequence>
      <xs:element minOccurs="0" name=" entityID " type="xs: string "/>
       <xs:element minOccurs="0" name="message" type="tns: message "/>
    </xs:sequence>
</xs:complexType>
```

9.6.4 Strategy

StrategyAction

```
  <xs:complexType name="strategyAction">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="goal" nillable="true"
type="tns:strategyGoal"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="constraint"
nillable="true" type="tns:strategyConstraint"/>
      <xs:element minOccurs="0" name="sortByConstraintID" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="strategyResult"
nillable="true" type="tns:strategyResult"/>
      <xs:element minOccurs="0" name="strategyActionID" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entitiyIDs"
nillable="true" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

StrategyGoal

```
  <xs:complexType name="strategyGoal">
    <xs:sequence>
      <xs:element minOccurs="0" name="kpiID"
type="tns:predefinedProcessingGraphs"/>
      <xs:element minOccurs="0" name="x1" type="tns: interval"/>
      <xs:element minOccurs="0" name="x2" type="tns: interval"/>
      <xs:element minOccurs="0" name="relation" type="tns:strategyRelationType"/>
      <xs:element name="value" type="xs:double"/>
      <xs:element minOccurs="0" name="goalID" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entitiyIDs"
nillable="true" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

StrategyConstraint

```
  <xs:complexType name="strategyConstraint">
    <xs:sequence>
      <xs:element minOccurs="0" name="description" type="xs:string"/>
      <xs:element minOccurs="0" name="constraintID" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entitiyIDs"
nillable="true" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

StrategyResult

```
  <xs:complexType name="strategyResult">
    <xs:sequence>
      <xs:element minOccurs="0" name="actionID" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="kpiResult"
nillable="true" type="tns:strategyKPIResult"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="metricResult"
nillable="true" type="tns:strategyMetricResult"/>
      <xs:element minOccurs="0" name="strategyResultID" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

StrategyKPIResult

```
  <xs:complexType name="strategyKPIResult">
    <xs:sequence>
      <xs:element minOccurs="0" name="type" type="tns:strategyResultType"/>
      <xs:element minOccurs="0" name="x1" type="tns: interval"/>
      <xs:element minOccurs="0" name="x2" type="tns: interval"/>
      <xs:element minOccurs="0" name="relation" type="tns:strategyRelationType"/>
      <xs:element name="value" type="xs:double"/>
      <xs:element minOccurs="0" name="kpiResultID"
type="tns:predefinedProcessingGraphs"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entitiyIDs"
nillable="true" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

StrategyMetricResult

```
  <xs:complexType name="strategyMetricResult">
    <xs:sequence>
      <xs:element minOccurs="0" name="type" type="tns:strategyResultType"/>
      <xs:element name="value" type="xs:double"/>
      <xs:element minOccurs="0" name="unit" type="tns:metricUnit"/>
      <xs:element minOccurs="0" name="metricType" type="tns:metricType"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="entitiyIDs"
nillable="true" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
```

StrategyRelationType

```
  <xs:simpleType name="strategyRelationType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="LEQ"/>
      <xs:enumeration value="EQ"/>
      <xs:enumeration value="GEQ"/>
    </xs:restriction>
  </xs:simpleType>
```

StrategyResultType

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

```
    <xs:restriction base="xs:string">
      <xs:enumeration value="GOAL"/>
      <xs:enumeration value="KPI"/>
      <xs:enumeration value="METRIC"/>
      <xs:enumeration value="ELSE"/>
    </xs:restriction>
  </xs:simpleType>
```

9.7 Service Interfaces (WSDL)

9.7.1 Attribute

9.7.1.1 AttributeService

```
  <xs:complexType name="subscribeAttribute">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="tns:entityFilter"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="arg1"
type="tns:attributeType"/>
      <xs:element minOccurs="0" name="arg2" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="subscribeAttributeResponse"
type="tns:subscribeAttributeResponse"/>
  <xs:complexType name="subscribeAttributeResponse">
    <xs:sequence>
      <xs:element minOccurs="0" name="return" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="setAttributeValue" type="tns:setAttributeValue"/>
  <xs:complexType name="setAttributeValue">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="xs:string"/>
      <xs:element minOccurs="0" name="arg1" type="tns:attributeType"/>
      <xs:element minOccurs="0" name="arg2" type="tns:baseAttributeValue"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="setAttributeValueResponse"
type="tns:setAttributeValueResponse"/>
  <xs:complexType name="setAttributeValueResponse">
    <xs:sequence/>
  </xs:complexType>
  <xs:element name="deleteAttributeSubscription"
type="tns:deleteAttributeSubscription"/>
  <xs:complexType name="deleteAttributeSubscription">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="deleteAttributeSubscriptionResponse"
type="tns:deleteAttributeSubscriptionResponse"/>
  <xs:complexType name="deleteAttributeSubscriptionResponse">
    <xs:sequence/>
  </xs:complexType>
  <xs:element name="getAttributeValue" type="tns:getAttributeValue"/>
  <xs:complexType name="getAttributeValue">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="tns:entityFilter"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="arg1"
type="tns:attributeType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="getAttributeValueResponse"
type="tns:getAttributeValueResponse"/>
  <xs:complexType name="getAttributeValueResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="return"
type="tns:attributeValueResult"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="reportAttributeUpdates" type="tns:reportAttributeUpdates"/>
  <xs:complexType name="reportAttributeUpdates">
```

```
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="arg0"
type="tns:attributeValueResult"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="reportAttributeUpdatesResponse"
type="tns:reportAttributeUpdatesResponse"/>
  <xs:complexType name="reportAttributeUpdatesResponse">
    <xs:sequence/>
  </xs:complexType>
  <xs:element name="getAttributeTimeseries" type="tns:getAttributeTimeseries"/>
  <xs:complexType name="getAttributeTimeseries">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="tns:entityFilter"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="arg1"
type="tns:attributeType"/>
      <xs:element minOccurs="0" name="arg2" type="xs:dateTime"/>
      <xs:element minOccurs="0" name="arg3" type="xs:dateTime"/>
      <xs:element name="arg4" type="xs:long"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="getAttributeTimeseriesResponse"
type="tns:getAttributeTimeseriesResponse"/>
  <xs:complexType name="getAttributeTimeseriesResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="return"
type="tns:attributeTimeseriesResult"/>
    </xs:sequence>
  </xs:complexType>

<wsdl:definitions name="AttributeService" targetNamespace="http://smartkye.eu/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://smartkye.eu/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/">
  <wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://smartkye.eu/" schemaLoca-
tion="AttributeService_schema1.xsd"/>
</schema>
  </wsdl:types>
  <wsdl:message name="deleteAttributeSubscriptionResponse">
    <wsdl:part name="parameters" ele-
ment="tns:deleteAttributeSubscriptionResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getAttributeValueResponse">
    <wsdl:part name="parameters" element="tns:getAttributeValueResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getAttributeValue">
    <wsdl:part name="parameters" element="tns:getAttributeValue">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getAttributeTimeseriesResponse">
    <wsdl:part name="parameters" element="tns:getAttributeTimeseriesResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="subscribeAttribute">
    <wsdl:part name="parameters" element="tns:subscribeAttribute">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="setAttributeValue">
    <wsdl:part name="parameters" element="tns:setAttributeValue">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="reportAttributeUpdatesResponse">
    <wsdl:part name="parameters" element="tns:reportAttributeUpdatesResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getAttributeTimeseries">
    <wsdl:part name="parameters" element="tns:getAttributeTimeseries">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="setAttributeValueResponse">
    <wsdl:part name="parameters" element="tns:setAttributeValueResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="SmartkyeException">
    <wsdl:part name="SmartkyeException" element="tns:SmartkyeException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="subscribeAttributeResponse">
    <wsdl:part name="parameters" element="tns:subscribeAttributeResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="deleteAttributeSubscription">
    <wsdl:part name="parameters" element="tns:deleteAttributeSubscription">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="reportAttributeUpdates">
    <wsdl:part name="parameters" element="tns:reportAttributeUpdates">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="AttributeServicePortType">
    <wsdl:operation name="subscribeAttribute">
      <wsdl:input name="subscribeAttribute" message="tns:subscribeAttribute">
    </wsdl:input>
      <wsdl:output name="subscribeAttributeResponse" mes-
sage="tns:subscribeAttributeResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="setAttributeValue">
      <wsdl:input name="setAttributeValue" message="tns:setAttributeValue">
    </wsdl:input>
      <wsdl:output name="setAttributeValueResponse" mes-
sage="tns:setAttributeValueResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="deleteAttributeSubscription">
      <wsdl:input name="deleteAttributeSubscription" mes-
sage="tns:deleteAttributeSubscription">
    </wsdl:input>
      <wsdl:output name="deleteAttributeSubscriptionResponse" mes-
sage="tns:deleteAttributeSubscriptionResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getAttributeValue">
      <wsdl:input name="getAttributeValue" message="tns:getAttributeValue">
    </wsdl:input>
      <wsdl:output name="getAttributeValueResponse" mes-
sage="tns:getAttributeValueResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="reportAttributeUpdates">
      <wsdl:input name="reportAttributeUpdates" mes-
sage="tns:reportAttributeUpdates">
    </wsdl:input>
      <wsdl:output name="reportAttributeUpdatesResponse" mes-
sage="tns:reportAttributeUpdatesResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getAttributeTimeseries">
      <wsdl:input name="getAttributeTimeseries" mes-
sage="tns:getAttributeTimeseries">
    </wsdl:input>
      <wsdl:output name="getAttributeTimeseriesResponse" mes-
sage="tns:getAttributeTimeseriesResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="AttributeServiceSoapBinding"
type="tns:AttributeServicePortType">
    <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="subscribeAttribute">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="subscribeAttribute">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="subscribeAttributeResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="deleteAttributeSubscription">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="deleteAttributeSubscription">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="deleteAttributeSubscriptionResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="setAttributeValue">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="setAttributeValue">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="setAttributeValueResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getAttributeValue">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getAttributeValue">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getAttributeValueResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="reportAttributeUpdates">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="reportAttributeUpdates">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="reportAttributeUpdatesResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getAttributeTimeseries">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getAttributeTimeseries">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getAttributeTimeseriesResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="AttributeService">
    <wsdl:port name="AttributeServicePort" bind-
ing="tns:AttributeServiceSoapBinding">
      <soap12:address location="http://localhost:9090/AttributeServicePort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

9.7.1.2 AttributeSubscriberService

```
  <xs:element name="notifyAttributeUpdate" type="tns:notifyAttributeUpdate"/>
  <xs:complexType name="notifyAttributeUpdate">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="xs:string"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="arg1"
type="tns:entityAttributeValue"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="notifyAttributeUpdateResponse"
type="tns:notifyAttributeUpdateResponse"/>
  <xs:complexType name="notifyAttributeUpdateResponse">
    <xs:sequence/>
  </xs:complexType>

<wsdl:definitions name="AttributeSubscriberService" target-
Namespace="http://smartkye.eu/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://smartkye.eu/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/">
  <wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://smartkye.eu/" schemaLoca-
tion="AttributeSubscriberService_schema1.xsd"/>
</schema>
  </wsdl:types>
  <wsdl:message name="SmartkyeException">
    <wsdl:part name="SmartkyeException" element="tns:SmartkyeException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="notifyAttributeUpdateResponse">
    <wsdl:part name="parameters" element="tns:notifyAttributeUpdateResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="notifyAttributeUpdate">
    <wsdl:part name="parameters" element="tns:notifyAttributeUpdate">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="AttributeSubscriberServicePortType">
    <wsdl:operation name="notifyAttributeUpdate">
      <wsdl:input name="notifyAttributeUpdate" mes-
sage="tns:notifyAttributeUpdate">
    </wsdl:input>
      <wsdl:output name="notifyAttributeUpdateResponse" mes-
sage="tns:notifyAttributeUpdateResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="AttributeSubscriberServiceSoapBinding"
type="tns:AttributeSubscriberServicePortType">
    <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="notifyAttributeUpdate">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="notifyAttributeUpdate">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="notifyAttributeUpdateResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="AttributeSubscriberService">
    <wsdl:port name="AttributeSubscriberServicePort" bind-
ing="tns:AttributeSubscriberServiceSoapBinding">
      <soap12:address loca-
tion="http://localhost:9090/AttributeSubscriberServicePort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

9.7.2 Entity

```
  <xs:element name="getEntityTimeseries" type="tns:getEntityTimeseries"/>
  <xs:complexType name="getEntityTimeseries">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="tns:entityFilter"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="getEntityTimeseriesResponse"
type="tns:getEntityTimeseriesResponse"/>
  <xs:complexType name="getEntityTimeseriesResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="return"
type="tns:entityProvidedMetric"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="findEntities" type="tns:findEntities"/>
  <xs:complexType name="findEntities">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="tns:entityFilter"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="findEntitiesResponse" type="tns:findEntitiesResponse"/>
  <xs:complexType name="findEntitiesResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="return"
type="tns:entity"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="findEntityDescendants" type="tns:findEntityDescendants"/>
  <xs:complexType name="findEntityDescendants">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="tns:entityFilter"/>
      <xs:element name="arg1" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="findEntityDescendantsResponse"
type="tns:findEntityDescendantsResponse"/>
  <xs:complexType name="findEntityDescendantsResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="return"
type="tns:entity"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="listTopLevelEntities" type="tns:listTopLevelEntities"/>
  <xs:complexType name="listTopLevelEntities">
    <xs:sequence/>
  </xs:complexType>
  <xs:element name="listTopLevelEntitiesResponse"
type="tns:listTopLevelEntitiesResponse"/>
  <xs:complexType name="listTopLevelEntitiesResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="return"
type="tns:entity"/>
    </xs:sequence>
  </xs:complexType>

<wsdl:definitions name="EntityService" targetNamespace="http://smartkye.eu/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://smartkye.eu/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/">
```

```
  <wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://smartkye.eu/" schemaLoca-
tion="EntityService_schema1.xsd"/>
</schema>
  </wsdl:types>
  <wsdl:message name="listTopLevelEntities">
    <wsdl:part name="parameters" element="tns:listTopLevelEntities">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listTopLevelEntitiesResponse">
    <wsdl:part name="parameters" element="tns:listTopLevelEntitiesResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getEntityTimeseriesResponse">
    <wsdl:part name="parameters" element="tns:getEntityTimeseriesResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="SmartkyeException">
    <wsdl:part name="SmartkyeException" element="tns:SmartkyeException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getEntityTimeseries">
    <wsdl:part name="parameters" element="tns:getEntityTimeseries">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="findEntities">
    <wsdl:part name="parameters" element="tns:findEntities">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="findEntitiesResponse">
    <wsdl:part name="parameters" element="tns:findEntitiesResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="findEntityDescendantsResponse">
    <wsdl:part name="parameters" element="tns:findEntityDescendantsResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="findEntityDescendants">
    <wsdl:part name="parameters" element="tns:findEntityDescendants">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="EntityServicePortType">
    <wsdl:operation name="getEntityTimeseries">
      <wsdl:input name="getEntityTimeseries" message="tns:getEntityTimeseries">
    </wsdl:input>
      <wsdl:output name="getEntityTimeseriesResponse" mes-
sage="tns:getEntityTimeseriesResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="findEntities">
      <wsdl:input name="findEntities" message="tns:findEntities">
    </wsdl:input>
      <wsdl:output name="findEntitiesResponse" message="tns:findEntitiesResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="findEntityDescendants">
      <wsdl:input name="findEntityDescendants" mes-
sage="tns:findEntityDescendants">
    </wsdl:input>
      <wsdl:output name="findEntityDescendantsResponse" mes-
sage="tns:findEntityDescendantsResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="listTopLevelEntities">
      <wsdl:input name="listTopLevelEntities" message="tns:listTopLevelEntities">
    </wsdl:input>
      <wsdl:output name="listTopLevelEntitiesResponse" mes-
sage="tns:listTopLevelEntitiesResponse">
    </wsdl:output>
```

```
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="EntityServiceSoapBinding" type="tns:EntityServicePortType">
    <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getEntityTimeseries">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getEntityTimeseries">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getEntityTimeseriesResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="findEntities">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="findEntities">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="findEntitiesResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="findEntityDescendants">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="findEntityDescendants">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="findEntityDescendantsResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="listTopLevelEntities">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="listTopLevelEntities">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="listTopLevelEntitiesResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="EntityService">
    <wsdl:port name="EntityServicePort" binding="tns:EntityServiceSoapBinding">
      <soap12:address location="http://localhost:9090/EntityServicePort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

9.7.3 Group

```
 <xs:element name="listGroupMembers" type="listGroupMembers"/>
  <xs:complexType name="listGroupMembers">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="listGroupMembersResponse" type="listGroupMembersResponse"/>
  <xs:complexType name="listGroupMembersResponse">
```

```
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="return"
type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="deleteGroup" type="deleteGroup"/>
  <xs:complexType name="deleteGroup">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="deleteGroupResponse" type="deleteGroupResponse"/>
  <xs:complexType name="deleteGroupResponse">
    <xs:sequence/>
  </xs:complexType>
  <xs:element name="createGroup" type="createGroup"/>
  <xs:complexType name="createGroup">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="arg0"
type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="createGroupResponse" type="createGroupResponse"/>
  <xs:complexType name="createGroupResponse">
    <xs:sequence>
      <xs:element minOccurs="0" name="return" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

<wsdl:definitions name="GroupService" targetNamespace="http://smartkye.eu/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://smartkye.eu/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/">
  <wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://smartkye.eu/" schemaLoca-
tion="GroupService_schema1.xsd"/>
</schema>
  </wsdl:types>
  <wsdl:message name="deleteGroup">
    <wsdl:part name="parameters" element="tns:deleteGroup">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="deleteGroupResponse">
    <wsdl:part name="parameters" element="tns:deleteGroupResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listGroupMembers">
    <wsdl:part name="parameters" element="tns:listGroupMembers">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listGroupMembersResponse">
    <wsdl:part name="parameters" element="tns:listGroupMembersResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="SmartkyeException">
    <wsdl:part name="SmartkyeException" element="tns:SmartkyeException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="createGroup">
    <wsdl:part name="parameters" element="tns:createGroup">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="createGroupResponse">
    <wsdl:part name="parameters" element="tns:createGroupResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="GroupServicePortType">
    <wsdl:operation name="listGroupMembers">
      <wsdl:input name="listGroupMembers" message="tns:listGroupMembers">
    </wsdl:input>
      <wsdl:output name="listGroupMembersResponse" mes-
sage="tns:listGroupMembersResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="deleteGroup">
      <wsdl:input name="deleteGroup" message="tns:deleteGroup">
    </wsdl:input>
      <wsdl:output name="deleteGroupResponse" message="tns:deleteGroupResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="createGroup">
      <wsdl:input name="createGroup" message="tns:createGroup">
    </wsdl:input>
      <wsdl:output name="createGroupResponse" message="tns:createGroupResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="GroupServiceSoapBinding" type="tns:GroupServicePortType">
    <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="listGroupMembers">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="listGroupMembers">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="listGroupMembersResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="createGroup">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="createGroup">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="createGroupResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="deleteGroup">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="deleteGroup">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="deleteGroupResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="GroupService">
    <wsdl:port name="GroupServicePort" binding="tns:GroupServiceSoapBinding">
      <soap12:address location="http://localhost:9090/GroupServicePort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

9.7.4 Message

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="MessageService" targetNamespace="http://smartkye.eu/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://smartkye.eu/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/">
```

```
  <wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://smartkye.eu/" schemaLoca-
tion="MessageService_schema1.xsd"/>
</schema>
  </wsdl:types>
  <wsdl:message name="SmartkyeException">
    <wsdl:part name="SmartkyeException" element="tns:SmartkyeException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="sendMessageResponse">
    <wsdl:part name="parameters" element="tns:sendMessageResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="sendMessage">
    <wsdl:part name="parameters" element="tns:sendMessage">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="MessageServicePortType">
    <wsdl:operation name="sendMessage">
      <wsdl:input name="sendMessage" message="tns:sendMessage">
    </wsdl:input>
      <wsdl:output name="sendMessageResponse" message="tns:sendMessageResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="MessageServiceSoapBinding"
type="tns:MessageServicePortType">
    <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="sendMessage">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="sendMessage">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="sendMessageResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="MessageService">
    <wsdl:port name="MessageServicePort" binding="tns:MessageServiceSoapBinding">
      <soap12:address location="http://localhost:9090/MessageServicePort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

9.7.5 Metric

9.7.5.1 MetricSubscriberService

```
  <xs:complexType name="notifyMetricUpdate">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="xs:string"/>
      <xs:element minOccurs="0" name="arg1" type="xs:dateTime"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="arg2"
type="tns:metricTimeseriesResult"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="notifyMetricUpdateResponse"
type="tns:notifyMetricUpdateResponse"/>
  <xs:complexType name="notifyMetricUpdateResponse">
    <xs:sequence/>
  </xs:complexType>

<wsdl:definitions name="MetricSubscriberService" target-
Namespace="http://smartkye.eu/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://smartkye.eu/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/">
```

```
  <wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://smartkye.eu/" schemaLoca-
tion="MetricSubscriberService_schema1.xsd"/>
</schema>
  </wsdl:types>
  <wsdl:message name="SmartkyeException">
    <wsdl:part name="SmartkyeException" element="tns:SmartkyeException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="notifyMetricUpdate">
    <wsdl:part name="parameters" element="tns:notifyMetricUpdate">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="notifyMetricUpdateResponse">
    <wsdl:part name="parameters" element="tns:notifyMetricUpdateResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="MetricSubscriberServicePortType">
    <wsdl:operation name="notifyMetricUpdate">
      <wsdl:input name="notifyMetricUpdate" message="tns:notifyMetricUpdate">
    </wsdl:input>
      <wsdl:output name="notifyMetricUpdateResponse" mes-
sage="tns:notifyMetricUpdateResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="MetricSubscriberServiceSoapBinding"
type="tns:MetricSubscriberServicePortType">
    <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="notifyMetricUpdate">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="notifyMetricUpdate">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="notifyMetricUpdateResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="MetricSubscriberService">
    <wsdl:port name="MetricSubscriberServicePort" bind-
ing="tns:MetricSubscriberServiceSoapBinding">
      <soap12:address loca-
tion="http://localhost:9090/MetricSubscriberServicePort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

9.7.5.2 MetricService

```
<xs:complexType name="subscribeMetric">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="xs:string"/>
      <xs:element minOccurs="0" name="arg1" type="tns:entityFilter"/>
      <xs:element minOccurs="0" name="arg2" type="tns:metricType"/>
      <xs:element minOccurs="0" name="arg3" type="tns:interval"/>
      <xs:element name="arg4" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="subscribeMetricResponse" type="tns:subscribeMetricResponse"/>
  <xs:complexType name="subscribeMetricResponse">
    <xs:sequence>
      <xs:element minOccurs="0" name="return" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="deleteMetricSubscription"
type="tns:deleteMetricSubscription"/>
  <xs:complexType name="deleteMetricSubscription">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="deleteMetricSubscriptionResponse"
type="tns:deleteMetricSubscriptionResponse"/>
  <xs:complexType name="deleteMetricSubscriptionResponse">
    <xs:sequence>
      <xs:element name="return" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="getMetricValue" type="tns:getMetricValue"/>
  <xs:complexType name="getMetricValue">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="tns:entityFilter"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="arg1"
type="tns:metricType"/>
      <xs:element name="arg2" type="xs:long"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="getMetricValueResponse" type="tns:getMetricValueResponse"/>
  <xs:complexType name="getMetricValueResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="return"
type="tns:metricValueResult"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="getMetricTimeseries" type="tns:getMetricTimeseries"/>
  <xs:complexType name="getMetricTimeseries">
    <xs:sequence>
      <xs:element minOccurs="0" name="arg0" type="tns:entityFilter"/>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="arg1"
type="tns:metricType"/>
      <xs:element minOccurs="0" name="arg2" type="tns:interval"/>
      <xs:element name="arg3" type="xs:long"/>
      <xs:element name="arg4" type="xs:long"/>
      <xs:element name="arg5" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="getMetricTimeseriesResponse"
type="tns:getMetricTimeseriesResponse"/>
  <xs:complexType name="getMetricTimeseriesResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="return"
type="tns:metricTimeseriesResult"/>
    </xs:sequence>
  </xs:complexType>

<wsdl:definitions name="MetricService" targetNamespace="http://smartkye.eu/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://smartkye.eu/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/">
  <wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://smartkye.eu/" schemaLoca-
tion="MetricService_schema1.xsd"/>
</schema>
  </wsdl:types>
  <wsdl:message name="getMetricTimeseriesResponse">
    <wsdl:part name="parameters" element="tns:getMetricTimeseriesResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="subscribeMetric">
    <wsdl:part name="parameters" element="tns:subscribeMetric">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="SmartkyeException">
    <wsdl:part name="SmartkyeException" element="tns:SmartkyeException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getMetricTimeseries">
    <wsdl:part name="parameters" element="tns:getMetricTimeseries">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getMetricValueResponse">
    <wsdl:part name="parameters" element="tns:getMetricValueResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="deleteMetricSubscriptionResponse">
    <wsdl:part name="parameters" element="tns:deleteMetricSubscriptionResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="deleteMetricSubscription">
    <wsdl:part name="parameters" element="tns:deleteMetricSubscription">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="subscribeMetricResponse">
    <wsdl:part name="parameters" element="tns:subscribeMetricResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getMetricValue">
    <wsdl:part name="parameters" element="tns:getMetricValue">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="MetricServicePortType">
    <wsdl:operation name="subscribeMetric">
      <wsdl:input name="subscribeMetric" message="tns:subscribeMetric">
    </wsdl:input>
      <wsdl:output name="subscribeMetricResponse" mes-
sage="tns:subscribeMetricResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="deleteMetricSubscription">
      <wsdl:input name="deleteMetricSubscription" mes-
sage="tns:deleteMetricSubscription">
    </wsdl:input>
      <wsdl:output name="deleteMetricSubscriptionResponse" mes-
sage="tns:deleteMetricSubscriptionResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getMetricValue">
      <wsdl:input name="getMetricValue" message="tns:getMetricValue">
    </wsdl:input>
      <wsdl:output name="getMetricValueResponse" mes-
sage="tns:getMetricValueResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getMetricTimeseries">
      <wsdl:input name="getMetricTimeseries" message="tns:getMetricTimeseries">
    </wsdl:input>
      <wsdl:output name="getMetricTimeseriesResponse" mes-
sage="tns:getMetricTimeseriesResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="MetricServiceSoapBinding" type="tns:MetricServicePortType">
    <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="subscribeMetric">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="subscribeMetric">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="subscribeMetricResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="deleteMetricSubscription">
```

```
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="deleteMetricSubscription">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="deleteMetricSubscriptionResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getMetricTimeseries">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getMetricTimeseries">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getMetricTimeseriesResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getMetricValue">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getMetricValue">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getMetricValueResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="MetricService">
    <wsdl:port name="MetricServicePort" binding="tns:MetricServiceSoapBinding">
      <soap12:address location="http://localhost:9090/MetricServicePort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

9.7.6 CEP

```
<wsdl:definitions name="CEPService" targetNamespace="http://smartkye.eu/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://smartkye.eu/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/">
  <wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://smartkye.eu/" schemaLoca-
tion="CEPService_schema1.xsd"/>
</schema>
  </wsdl:types>
  <wsdl:message name="createProcessingGraphInstanceResponse">
    <wsdl:part name="parameters" ele-
ment="tns:createProcessingGraphInstanceResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getProcessingGraph">
    <wsdl:part name="parameters" element="tns:getProcessingGraph">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="createProcessingGraphResponse">
    <wsdl:part name="parameters" element="tns:createProcessingGraphResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="deleteProcessingGraphResponse">
    <wsdl:part name="parameters" element="tns:deleteProcessingGraphResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="destroyProcessingGraphInstanceResponse">
    <wsdl:part name="parameters" ele-
ment="tns:destroyProcessingGraphInstanceResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="createProcessingGraph">
    <wsdl:part name="parameters" element="tns:createProcessingGraph">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getProcessingGraphResponse">
    <wsdl:part name="parameters" element="tns:getProcessingGraphResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="SmartkyeException">
    <wsdl:part name="SmartkyeException" element="tns:SmartkyeException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="executeMetricProcessingGraph">
    <wsdl:part name="parameters" element="tns:executeMetricProcessingGraph">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="destroyProcessingGraphInstance">
    <wsdl:part name="parameters" element="tns:destroyProcessingGraphInstance">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="executeMetricProcessingGraphResponse">
    <wsdl:part name="parameters" ele-
ment="tns:executeMetricProcessingGraphResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="createProcessingGraphInstance">
    <wsdl:part name="parameters" element="tns:createProcessingGraphInstance">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="deleteProcessingGraph">
    <wsdl:part name="parameters" element="tns:deleteProcessingGraph">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="CEPServicePortType">
    <wsdl:operation name="deleteProcessingGraph">
      <wsdl:input name="deleteProcessingGraph" mes-
sage="tns:deleteProcessingGraph">
    </wsdl:input>
      <wsdl:output name="deleteProcessingGraphResponse" mes-
sage="tns:deleteProcessingGraphResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="executeMetricProcessingGraph">
      <wsdl:input name="executeMetricProcessingGraph" mes-
sage="tns:executeMetricProcessingGraph">
    </wsdl:input>
      <wsdl:output name="executeMetricProcessingGraphResponse" mes-
sage="tns:executeMetricProcessingGraphResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="destroyProcessingGraphInstance">
      <wsdl:input name="destroyProcessingGraphInstance" mes-
sage="tns:destroyProcessingGraphInstance">
    </wsdl:input>
      <wsdl:output name="destroyProcessingGraphInstanceResponse" mes-
sage="tns:destroyProcessingGraphInstanceResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="createProcessingGraphInstance">
      <wsdl:input name="createProcessingGraphInstance" mes-
sage="tns:createProcessingGraphInstance">
    </wsdl:input>
      <wsdl:output name="createProcessingGraphInstanceResponse" mes-
sage="tns:createProcessingGraphInstanceResponse">
    </wsdl:output>
```

```
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="createProcessingGraph">
      <wsdl:input name="createProcessingGraph" mes-
sage="tns:createProcessingGraph">
    </wsdl:input>
      <wsdl:output name="createProcessingGraphResponse" mes-
sage="tns:createProcessingGraphResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getProcessingGraph">
      <wsdl:input name="getProcessingGraph" message="tns:getProcessingGraph">
    </wsdl:input>
      <wsdl:output name="getProcessingGraphResponse" mes-
sage="tns:getProcessingGraphResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="CEPServiceSoapBinding" type="tns:CEPServicePortType">
    <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="deleteProcessingGraph">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="deleteProcessingGraph">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="deleteProcessingGraphResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="executeMetricProcessingGraph">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="executeMetricProcessingGraph">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="executeMetricProcessingGraphResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="destroyProcessingGraphInstance">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="destroyProcessingGraphInstance">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="destroyProcessingGraphInstanceResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="createProcessingGraph">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="createProcessingGraph">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="createProcessingGraphResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="createProcessingGraphInstance">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="createProcessingGraphInstance">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="createProcessingGraphInstanceResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getProcessingGraph">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getProcessingGraph">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getProcessingGraphResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CEPService">
    <wsdl:port name="CEPServicePort" binding="tns:CEPServiceSoapBinding">
      <soap12:address location="http://localhost:9090/CEPServicePort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

9.7.7 Strategy

```
<wsdl:definitions name="StrategyService" targetNamespace="http://smartkye.eu/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://smartkye.eu/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/">
  <wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://smartkye.eu/" schemaLoca-
tion="StrategyService_schema1.xsd"/>
</schema>
  </wsdl:types>
  <wsdl:message name="getStrategyActionResponse">
    <wsdl:part name="parameters" element="tns:getStrategyActionResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getStrategyAction">
    <wsdl:part name="parameters" element="tns:getStrategyAction">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="storeStrategyAction">
    <wsdl:part name="parameters" element="tns:storeStrategyAction">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="SmartkyeException">
    <wsdl:part name="SmartkyeException" element="tns:SmartkyeException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="storeStrategyActionResponse">
    <wsdl:part name="parameters" element="tns:storeStrategyActionResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="StrategyServicePortType">
    <wsdl:operation name="storeStrategyAction">
      <wsdl:input name="storeStrategyAction" message="tns:storeStrategyAction">
    </wsdl:input>
      <wsdl:output name="storeStrategyActionResponse" mes-
sage="tns:storeStrategyActionResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getStrategyAction">
      <wsdl:input name="getStrategyAction" message="tns:getStrategyAction">
    </wsdl:input>
      <wsdl:output name="getStrategyActionResponse" mes-
sage="tns:getStrategyActionResponse">
    </wsdl:output>
      <wsdl:fault name="SmartkyeException" message="tns:SmartkyeException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="StrategyServiceSoapBinding"
type="tns:StrategyServicePortType">
    <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="storeStrategyAction">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="storeStrategyAction">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="storeStrategyActionResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getStrategyAction">
      <soap12:operation soapAction="" style="document"/>
      <wsdl:input name="getStrategyAction">
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getStrategyActionResponse">
        <soap12:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="SmartkyeException">
        <soap12:fault name="SmartkyeException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="StrategyService">
    <wsdl:port name="StrategyServicePort" bind-
ing="tns:StrategyServiceSoapBinding">
      <soap12:address location="http://localhost:9090/StrategyServicePort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```