Deliverable D2.2

System Requirements and Smart Objects Model

Editor:	Jorge Cuellar, SAG
Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	31 May 2014
Actual delivery date:	31 May 2014
Suggested readers:	Application Owners, Developers of Applications, Service Providers, Hardware Manufacturers, Researchers
Version:	1.0
Total number of pages:	111
Keywords:	RERUM, Internet of Things, applications, requirements, security, privacy, confidentiality, integrity, availability, authentication, authorization, accounting, hardware, software, middleware, virtualization, modelling

Abstract

This deliverable extracts requirements from the RERUM smart city use cases defined in deliverable D2.1. A thorough review of the state of the art with respect to related projects is presented, aiming to analyse the research topics under investigation and the corresponding approaches that have been followed in the literature. Based on this analysis and the previously defined threats for the use cases in focus, several requirements are elicited in terms of applications, networking, RERUM devices, middleware and virtualization, and security and privacy. The requirements will help to measure and verify the needed functionality and value for RERUM stakeholders, as well as to mitigate possible risks and threats for the envisioned RERUM use cases. Additionally, an abstract view of the so-called RERUM devices is presented, where physical things are enriched and become ICT manageable. A clear understanding of RERUM devices is the key to support "security, privacy and reliability by design" concepts. The present deliverable will serve as the basis for defining the system architecture, the selection of adequate mechanisms to fulfil the identified requirements, as well as their verification in trial scenarios.

Disclaimer

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 609094.

Impressum

Full project title	Reliable, resilient and secure IoT for smart city applications
Short project title	RERUM
Number and title of work-package	WP2 - The IoT Architectural Framework for Smart Cities
Number and title of task	T2.2 - Smart objects modelling and middleware T2.3 - User, System and Application Requirements
Document title	System Requirements and Smart Objects Model
Editor: Name, company	Jorge Cuellar, SAG
Work-package leader: Name, company	Theodore Mouroutis, CYTA
Estimation of person months (PMs) spent on the Deliverable	

Executive summary

This report elicits requirements for the Use-Cases (UCs) presented in deliverable, D2.1. In a first step, a closer look at related work is done in Section 1, where general requirements for smart cities are analysed. Afterwards, related scientific efforts and European projects are categorized accordingly. By comparing these different projects, it becomes evident that security and privacy by design, have not been the focus so far in the IoT research community.

Based on the analysis of the use cases descriptions and their threat analysis as presented in deliverable D2.1, Section 2 presents the requirements of the RERUM platform. These requirements are divided into the following categories:

System Architecture Requirements: These are the non-functional requirements that give the basic high-level view of the functionalities that the system architecture should support.

Application Requirements: These are requirements for application developers concerning the interaction with the RERUM ecosystem. Application requirements are an additional effort to support developers, as application development is not within the RERUM scope.

Networking and QoS Requirements: Every use case depends heavily on the networking capabilities of its components; thus, networking and QoS requirements affect all RERUM use cases. This section specifies which RERUM capabilities should exist for the efficient interconnection of the RERUM devices, so as to ensure the best possible implementation of the use cases.

RERUM Device Requirements: RERUM Devices are responsible for interaction with other entities in the physical world. As such, they need to fulfil a variety of requirements regarding performance, connectivity and dynamical behaviour.

Middleware and Virtualization Requirements*:* In this part we specify the requirements that are needed for the virtual representation of the heterogeneous RERUM Devices, and their inclusion into the middleware that connects them with the applications in the IoT world.

Security and Privacy Requirements*:* Security requirements address the threats and vulnerabilities identified in D2.1. In a further step, security mechanisms such as cryptographic protocols will be chosen and/or developed to fulfil these requirements. Privacy requirements address the privacy threats of D2.1, as well, but are also derived from the European Directives 95/46/EC and 2002/58/EC.

Section 3 discusses the RERUM Devices and how they will fit into a larger IoT architecture. A well understood architectural model is required to design and ultimately implement the virtual representation of the RERUM Devices from the IoT point of view. For this task, RERUM will carefully consider as basis the IoT-A architecture reference model.

List of authors

Company	Author	Contribution
Siemens AG	Jorge Cuellar Fabienne Waidelich Kai Fischer Santiago Suppan	Editor Privacy Requirements, overall sec. 2.6 Bootstrapping Requirements 2nd editor, privacy requirements, overall sec. 2 (particularly sec. 2.6.)
ATOS	Dario Ruiz Lopez Guadalupe Rodriguez Diaz Michal Mardiak	Responsible for section 'Requirements for the Secure Provision of SW and Configuration', Co-editorial to 'Miscellaneous security requirements' Contributed to sections 'Integrity, Confidentiality and Authentication Requirements' 'RD software requirements' 'Application requirements' Middleware and Virtualization Requirements (section 2.5)
UNIVBRIS	George Oikonomou	Requirements for the SW of SOs Energy-efficient cryptographic primitives and infrastructure Overall (co-)editorial for 2.4 (Smart Object Requirements)
LiU	Vangelis Angelakis: David Gundlegård: Dan Helgersson: Magnus Lundgren:	Contribution to requirements in Sections 2.2, 2.3 Contribution to requirements in Sections 2.2, 2.3 Contribution to requirements in Section 2.3 Contribution to requirements in Section 2.3
UNI PASSAU	Henrich C. Pöhls	Requirements for Security and Privacy (esp. focused on Integrity, Privacy and Authentication) Co-editorial for sect. 2.6 (Security and Privacy Requirements) Co-editorial for sect. 3 (Smart Object Model) and drawing graphical representations Internal Review of the complete Deliverable
Zolertia	Francisco Paredes Vera Marc Fàbregas Bachs	Requirements for the HW of SOs
FORTH	Elias Tragos	"Networking and QoS requirements", "system architecture requirements", RD modelling section, overall editing and Internal Reviewer

CYTA	Athanasios Lioumpas, Theodore Mouroutis	Network and application requirements
AJTGN	Xavier Reina Virgili	Analysis of use cases and Application requirements.
HER	Costis Mochianakis, Manolis Fotakis	Application and hardware requirements
SSRL	Septimiu Nechifor	RD modelling requirements, Contribution and editing to Section 3

Table of Contents

Executive summary 4
Table of Contents 7
List of Requirements 9
Abbreviations 11
Definitions 14
1 Introduction 16
1.1 The Smart City context 16
1.2 Deliverable structure and scope 17
1.3 State of the practice and related work 19
2 Requirements 22
2.1 Requirements for the System Architecture 22
2.2 Application Requirements 24
2.3 Networking and QoS Requirements 27
2.4 RERUM Device Software, Hardware and Modelling Requirements 35
2.4.1 RD Hardware Requirements 35
2.4.2 Gateway Requirements 39
2.4.3 RD Software Requirements 40
2.4.4 RD Modelling requirements 43
2.5 Middleware and Virtualisation Requirements 47
2.6 Security and Privacy Requirements 54
2.6.1 Energy-efficient cryptographic primitives 55
2.6.2 Integrity, Confidentiality and Authentication Requirements 56
2.6.3 Privacy Requirements 67
2.6.4 Bootstrapping Requirements 75
2.6.5 Requirements for the Secure Provision of SW and Configuration 77
2.6.6 Miscellaneous Security Requirements 79
3 The IoT Domain Model of RERUM 81
3.1 *RERUM Device*: The Hardware and Network view 82
3.1.1 Hardware Components of a RERUM Device 83
3.1.2 Layers of Software and Hardware on a RERUM Device 84
3.1.3 Cognitive Radio Agent on a RERUM Device 86
3.1.4 RERUM Device in the context of the RERUM IoT Model 87
3.2 Virtual RERUM Device: The IoT view 88
3.2.1 Connectivity model of VRDs 93

3.2.2 Naming and addressing models for VRDs 93
3.2.3 Functional model for the establishment and execution of VRDs federation 93
3.2.4 VRD profile model 94
3.2.5 VRD execution model 94
3.3 RERUM's IoT Model 95
3.3.1 Term: Physical Entity (PE) 95
3.3.2 Term: Virtual Entity (VE) 96
3.3.3 Term: Augmented Entity 96
3.3.4 RERUM Term: RERUM Device (RD) 97
3.3.5 RERUM Term: Mobile Phone used as a RERUM Device 98
3.3.6 RERUM Term: Resource on a RERUM Device 98
3.3.7 RERUM Term: RERUM Service offered by a RERUM Device 99
3.3.8 RERUM Term: Virtual RERUM Device (VRD) 99
3.3.9 RERUM Term: Federation of Virtual RERUM Devices (VRD-Federation) 100
3.3.10 RERUM Term: Generic Virtual RERUM Object (VRO) 100
3.3.11 RERUM Model including Normal and Administrative Users 102
4 Conclusions 105
References 107

List of Requirements

Req. 2.2-1 Remote control of RDs 25
Req. 2.2-2 Suitable Sensory data can be released to the application 25
Req. 2.2-3 Rate of data collection 26
Req. 2.2-4 Time-efficient connectivity of devices for data uploading to application 27
Req. 2.3-1 Support of a large number of attached devices/objects 29
Req. 2.3-2 Dynamic spectrum management 30
Req. 2.3-3 distributed spectrum selection 30
Req. 2.3-4 Operation in unused spectrum bands 31
Req. 2.3-5 Centralised management of constrained networks 32
Req. 2.3-6 Partitioning of the network into clusters 32
Req. 2.3-7 Multi-technology and multi-operator connectivity. 33
Req. 2.3-8 Message prioritization 33
Req. 2.3-9 Overall QoS. 34
Req. 2.3-10 Self-* mechanisms 34
Req. 2.4-1 Enclosure IP rating 35
Req. 2.4-2 Microcontroller performance 36
Req. 2.4-3 Autonomous operation, processing consumption and low-power modes 37
Req. 2.4-4 Volatile memory 37
Req. 2.4-5 Persistent storage 38
Req. 2.4-6 RD Communication interfaces 38
Req. 2.4-7 Reconfigurable network interfaces 39
Req. 2.4-8 Gateway computational performance 40
Req. 2.4-9 Gateway communication interfaces 40
Req. 2.4-10 Low energy consumption 41
Req. 2.4-11 Lightweight dynamic data compression 42
Req. 2.4-12 Over-the-Air Programming 43
Req. 2.4-13 RERUM to apps interface 44
Req. 2.4-14 RERUM to RD interface 45
Req. 2.4-15 Difference between Federated RD and RD 45
Req. 2.4-16 RD as a service provider 46
Req. 2.4-17 RD state/stateless capability 46
Req. 2.4-18 RD to RD communication in federations 47
Req. 2.5-1 Virtualisation of RDs and PEs 49
Req. 2.5-2 Discovery of GVOs and management of the life-cycle of RDs and PEs 49

Req. 2.5-3 Filtering and Decision making 51
Req. 2.5-4 Interconnectivity of VRDs through the MW 51
Req. 2.5-5 Monitoring and traceability by the middleware 52
Req. 2.5-6 Support and partial enforcement of QoS Policies 53
Req. 2.5-7 Support of accounting and SLAs 54
Req. 2.6-1 Energy-efficient cryptographic primitives 56
Req. 2.6-2 Integrity protection of SL-I data in transit 58
Req. 2.6-3 Integrity protection of SL-I data at rest 59
Req. 2.6-4 Authorised modification of integrity protected data 60
Req. 2.6-5 Detection of authorised modification of integrity protected data 62
Req. 2.6-6 Confidentiality protection of SL-C data at rest 62
Req. 2.6-7 Confidentiality protection of personal data in transit 63
Req. 2.6-8 Device authentication 64
Req. 2.6-9 User authentication 65
Req. 2.6-10 Attribute-based access control 67
Req. 2.6-11 User Consent and choice 69
Req. 2.6-12 Purpose legitimacy and specification 70
Req. 2.6-13 Collection limitation 71
Req. 2.6-14 Data minimisation 72
Req. 2.6-15 Accuracy and quality 73
Req. 2.6-16 Notice and access 73
Req. 2.6-17 Individual participation and access 74
Req. 2.6-18 Accountability 75
Req. 2.6-19 Secure bootstrapping of operational cryptographic credentials 75
Req. 2.6-20 Availability of initial credentials 76
Req. 2.6-21 Support of different operational credentials types 76
Req. 2.6-22 Avoidance of manual interactions during credential bootstrapping 77
Req. 2.6-23 Update of operational credentials 77
Req. 2.6-24 Find deployable software to RERUM devices 78
Req. 2.6-25 Object configuration isolated per application 79
Req. 2.6-26 Secure design and implementation of RERUM components 79
Req. 2.6-27 Reputation mechanism for reliability, availability and trustworthiness 80

Abbreviations

3G	Third Generation
AAA	Authentication, Authorization, Accounting
ACE	Authentication and Authorization for Constrained Environments
A-DATA	Actuator Data
AE	Augmented Entity
AES	Advanced Encryption Standard
AI	Artificial Intelligence
AP	Access Point
API	Application Programming Interface
ARM	Architecture Reference Model
BCMP	Network of queues first studied by Baskett, Chandy, Muntz, Palacios
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
BS	Base Staion
CIA	Confidentiality, Integrity, Availability
COAP	Constrained Application Protocol
CPU	Central Processing Unit
CR	Cognitive Radio
CRC	Cyclic Redundancy Code
CS	Compressive Sensing
CSV	Comma Separated Value
CVO	Composite Virtual Objects
DICE	DTLS In Constrained Environments
DoS	Denial of Service
DRAM	Dynamic RAM
DSA	Dynamic Spectrum Access
DSL	Digital subscriber line
DSP	Digital Signal Processing
ECC	Elliptic Curve Cryptography
EEPROM	Electrically Erasable Programmable Read-Only Memory
ELF	Executable and Linkable Format
ETA	Estimated Time of Arrival
FLASH	A type of non-volatile computer memory
GB	Giga Byte (1 GB = 1024 MB)
GIS	Geo Information System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GVO	Generic Virtual Object (see Definitions)
H/W	Hardware
HEM	Home Energy Management

HW	Hardware
I2C	Inter-Integrated Circuit
ICT	Information and communications technology
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	International Protection
IPSec	Internet Protocol Security
ISM	Industrial, Scientific and Medical
ISO	International Standards Organization
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
LoWPAN	Low power Wireless Personal Area Network
LTE	Long Term Evolution
M2M	Machine to Machine
MAC	Medium Access Control
MB	Mega Byte (1 MB = 1024 Byte)
MCU	Microcontroller Unit
MHz	Megahertz
MMC	Multimedia Card
MQTT	MQ Telemetry Transport
MW	Middleware
NAND	Negated AND or NOT AND Flash Storage
NFR	Non Functional Requirements
NOR	NOR Flash Memory is a type of non-volatile computer memory
OAP	Over Air Programming
OS	Operating System
PDUs	Power Distribution Unit
PE	Physical Entity
PET	Privacy Enhancing Technologies
PHY	Physical
PM	Power Mode
QoS	Quality of Service
RAM	Random Access Memory
RD	RERUM Device
REST	Representational state transfer
RF	Radio Frequency
RFC	Request for comment
RFID	Radio-Frequency Identification
RSS	Received signal strength
RWO	Real World Object
S/W	Software

SD card	Secure Digital card
S-DATA	Sensor Data
SDR	Software Defined Radio
SL	Security Level
SLA	Service Level Agreement
SO	Smart Object
SOAP	Simple Object Access Protocol
SPI	Serial Peripheral Interface Bus
SSD	Solid-State Drive
SSL	Secure Socket Layer
SSN	Semantic Sensor Network
STRIDE	STRIDE Threat Model
SW	Software
UART	Universal Asynchronous Receiver / Transmitter
UAV	Unmanned Aerial Vehicle
UCs	Use Cases
UML	Unified Modelling Language
VE	Virtual Entity
VO	Virtual Object
VRD	Virtual RERUM Device
VRO	Generic Virtual RERUM Object
VSO	Virtual Smart Object
W3C	World Wide Web Consortium
WiFi	Wi-Fi Wireless LAN standard
WiMax	Worldwide Interoperability for Microwave Access
WP3	Work Package 3
WP4	Work Package 4
xDSL	Digital Subscriber Line
XML	Extensible Markup Language
YAFFS	Yet Another Flash File System
YAFFS2	Yet Another Flash File System Version 2

Definitions

Please refer to RERUM deliverable D2.1 for more definitions, as it introduces the definitions continuously used throughout the project.

Term	Definition	Source
Acting element	An (embedded) device that has the capability to affect the condition of a Physical Entity, (like changing its state or moving it) by acting upon an electrical signal	RERUM/ IOT-A part of actuator [56]
Actuator	A smart device that includes one or several acting elements and receives (IT-based) information (command – CDATA) translating it to electrical signal for the acting elements. An actuator can also include a sensor so that there is knowledge on the Physical Entity it acts upon, in order to translate correctly the command into the electrical signal.	RERUM/IOT-A [56]
Application server	The point responsible for the end-user services (e.g., automation services, energy management, etc.)	RERUM/IOT-A [56]
Device	It can be a single or a combination of the following elements: • Sensors, which provide information about the Physical Entity • Tags, which are used to identify Physical Entities • Actuators, which can modify the physical state of a Physical Entity	IoT-A [56]
Entity	Something that exists by itself: something that is separate from other things. Could be physical, virtual, functional, etc.	Merriam-Webster dictionary [1]
Gateway	Network node equipped for interfacing with another network that uses different protocols.	Federal Standard 1037C [78]
Generic Virtual RERUM Object (VRO)	This is a software artefact with certain properties shared between both virtualizations found in RERUM. Virtual Entities and Virtual RERUM Devices are both representations in the digital world and hence are both members of the Generic Virtual RERUM Object (VRO), which share properties like, that • they allow to be discovered, • they allow to be addressed, and they allow to be interacted with in a standardized manner.	RERUM
Physical entity (PE)	• A discrete, identifiable part of the physical environment which is of interest to the user for the completion of his goal. Physical Entities can be almost any object or environment.	Merriam-Webster dictionary [1] / IOT-A [56]

RERUM Device (RD) or RERUM Smart Object	A RERUM Device (RD) is a piece of hardware and software (incl. the Operating System) that is equipped with intelligence. It has one or more Resources that the RERUM Device is able to either fill with interpreted and pre-processed sensory data or able to read and interpret the commands that are given. The RERUM Device has some Sensing, Tag or Acting elements directly attached to it.	RERUM
Resources	Resources are software components that provide some functionality. When associated with a Physical Entity, they either provide some information about or allow changing some aspects in the digital or physical world pertaining to one or more Physical Entities. In general, they are typically sensor Resources that provide sensing data or actuator Resources, e.g. a machine controller that effects some actuation in the physical world.	IOT-A [56] On-device Resources
Sensing element	An (embedded) device that perceives certain characteristics of the real-world environment (Physical Entities), translating a change into an electrical signal.	RERUM
Sensor	• A smart device that includes one or several sensing elements and is able to translate the electrical signal of the sensing elements to some type of information (digital representation) with specific value and semantic.	IoT-A [56]
Service provider	A company that provides IoT services (e.g., home energy management services).	RERUM
Smart Object	See RERUM Device	RERUM
Virtual Entity (VE)	The digital synchronized representation of a Physical Entity.	IoT-A [56]
Virtual RERUM Device (VRD) or Virtual RERUM Smart Object (SO)	A Virtual RERUM Device (RD) is a digital representation of a RERUM Device. The same one physical RERUM Device at one time is represented by one Virtual RERUM Device. This is a software artefact, like a Virtual Entity (VE), but represents a RERUM Device (RD).	
Virtual RERUM Devices Federation	Several Virtual RERUM Devices are called a Federation if they cooperate to offer an IoT-service for a Virtual Entity (VE). The logic necessary to orchestrate the service is associated to the Virtual Entity that offers the service.	
User	An human or software that interacts with a system for transferring information.	Based on IoT-A [56]

1 Introduction

This report elicits requirements for the Use-Cases (UCs) presented in deliverable D2.1. In a first step, the reader is introduced into the context of smart cities. After a short introduction to the structure of this deliverable, general key enablers for smart cities are analysed. The relation between scientific efforts and European projects follows. Here, the state of practice is lined out, upon which RERUM will build. By comparing the different projects, it also becomes evident privacy and security by-design has not been the focus so far.

Section 2 presents the requirements for RERUM's platform and system. The requirements relate to the smart city key enablers and the use case analysis of deliverable D2.1.

Section 3 discusses the RERUM Devices and how they will fit into a larger IoT architecture. A well understood architectural model is required to design and ultimately implement the virtual representation of the RERUM devices from the IoT point of view. For this task, RERUM will carefully consider as basis the IoT-A architecture reference model [56].

1.1 The Smart City context

Cities are now the core elements of human civilization. Urban population is expected to grow by an estimated 2.3 billion in the next 40 years, having **almost 70% of world's population living in cities by 2050**. This rapid growth of cities aggravates many challenges associated with living in urban environments, such as public safety, transportation management, waste disposal, noise, air, and water pollution [38]. Meanwhile economy issues like current Eurozone crisis significantly affect cities, hampering their vision to provide prosperity and improved quality of life to their citizens. In this context, **Smart Cities** provide Information and Communication Technologies (ICT)-enabled services and applications to citizens, companies and authorities, **driving competitiveness, sustainability and improving quality of life**. Smart cities are also linked with sustainability and energy efficiency. In the current form, cities are already characterized by the potential of collecting a wealth of data, through a number of different sensing infrastructures. More or less this data is partly used for different, often isolated, application purposes. Nevertheless, beside the single domain/single owner applications and the associated sensing infrastructures that particular cities may have, the trend will continue to be towards an ever increasing amount of data [32] that will further enrich the overall "city sensing" capabilities. Input will come from connected objects (sensors) as well as from "social sensing" applications. In other words, data sources can come from: (i) the Internet of Things (IoT) that becomes more and more established; (ii) the increasing sophisticated features of the smart end-user devices, in conjunction with the powerful established social network applications.

As effect of the dramatic changes and availability of Information and Communication Technologies, an increasing number of cities have started to introduce new enabled services [38] with the objective of addressing sustainability as well as improving the operational efficiency of services and infrastructure. However, a convincing use of smart city applications is hindered by various issues, such as the difficulty of integrating heterogeneous data sources and the challenge of extracting up-to-date information in real-time and especially the difficulty to offer solid reputation and safe use of for those technologies to largest majority of users. Today the challenges are often addressed by application specific solutions, resulting in silo architectures like: public transportation, lighting, water management, public safety, energy management.

However, a challenge in the smart city approach is integration across different application domains, as well as the engagement of different city departments, city-contracted entrepreneurs and individual enterprises providing services. Today large amounts of valuable data and sensor information remain unused or are limited to specific application domains due to the large number of specific technologies and formats (traffic information, parking spaces, bus timetables, waiting times at events, event calendars, environment sensors for pollution or weather warnings, GIS databases

etc.). By consequence, the alignment of data from various sources is typically done by human expert leaving the process of validation slow and error prone. At the same time, it is difficult to control the way data are safely distributed and used. Taking also into account the dynamic character of urban data (diversity of data streams, mobility of data sources, quality and availability), and geographical overlapping of businesses, human actors and machine driven processes, we can conclude that reliability and safety of Smart City services is under the stress due to the fact that many threats are not considered by design, leaving conceptual and implementation holes.

1.2 Deliverable structure and scope

The main scope of this report is to extract the requirements for RERUM's overall system, smart objects, security and privacy, as well as to give recommendations to application developers. It also includes the design of the smart objects model. The report starts with Section 1.3, which includes a brief review of the state of the art, aiming to analyse recent research topics and the corresponding approaches that have been followed. The requirements for RERUM ecosystem are elicited in Section 2. Keeping the smart city enablers in mind and the use case analysis of D2.1, Section 2 presents the requirements for RERUM's platform and system. The requirement categories are divided into the following categories:

System Architecture Requirements: These are the non-functional requirements that give the basic high-level view of the functionalities that the system architecture should support.

Application Requirements: These are requirements for application developers on how to interact with the RERUM ecosystem. Application requirements are an additional effort to support developers, as application development is not within the RERUM scope.

Networking and QoS Requirements: Every use case depends heavily on the networking capabilities of its components; thus, networking and QoS requirements affect all RERUM use cases. This section specifies which capabilities that RERUM should be present for interconnecting to ensure the best possible implementation of the use cases.

RERUM Device Requirements: RERUM devices are responsible for interaction with the entities of the physical world. As such, they need to fulfil a variety of requirement's regarding performance, connectivity and dynamical behaviour.

Middleware and Virtualization Requirements*:* In this part we specify the requirements that are needed for the virtual representation of the heterogeneous RERUM devices and their inclusion into the middleware that connects them with the applications in the IoT world.

Security and Privacy Requirements*:* Security requirements address the threats and vulnerabilities identified in D2.1. In a further step, security mechanisms such as cryptographic protocols will be chosen and/or developed to fulfil these requirements. Privacy requirements address the privacy threats of D2.1, as well, but are also derived from the European Directives 95/46/EC and 2002/58/EC.

RERUM's focus on the provision of security and privacy by design is unique. This is reflected in Section 2.6, the most extensive requirements section in this document. Section 3 introduces the RERUM Smart Object model, addressing the object's virtualization and the way it fits into a bigger IoT model. This section introduces the basis for coming deliverables, which will further detail RERUM's system architecture. The structure of this deliverable and the relations between the separate sections are depicted in Figure 1.

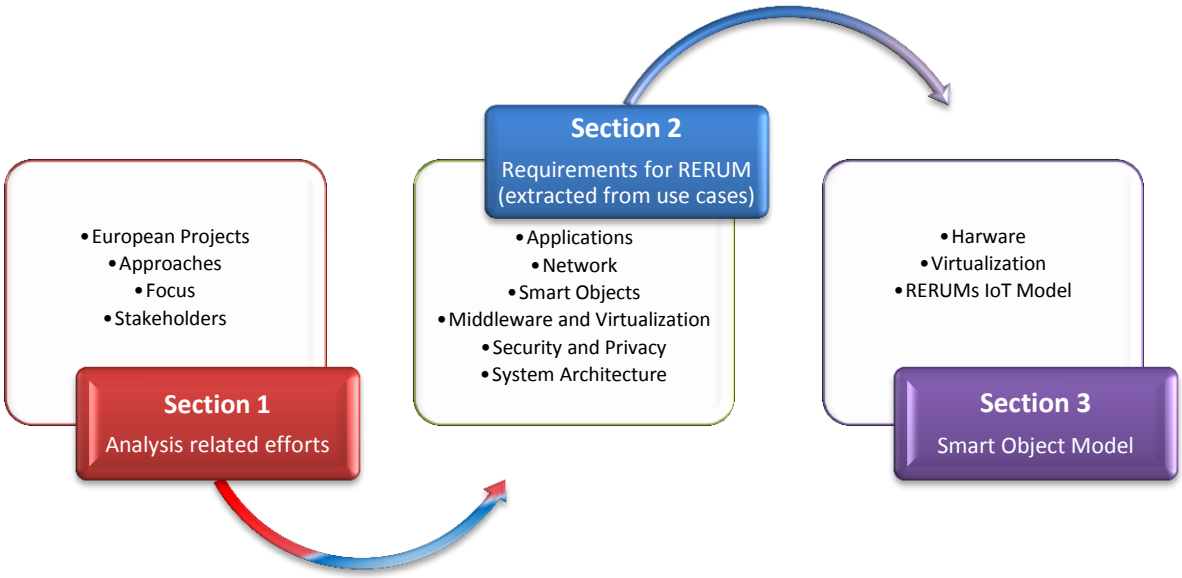


Figure 1 - The structure of this report

The main output of this deliverable (related to Task2.2 and 2.3), is the definition of the addressable requirements for designing the overall RERUM system and its architecture. These requirements will also be used as a basis for driving the work in the technical work packages (WP3 and WP4). The Smart Object model will be used in the system architecture, detailed in the next deliverable (D2.3). These relationships are depicted in Figure 2.

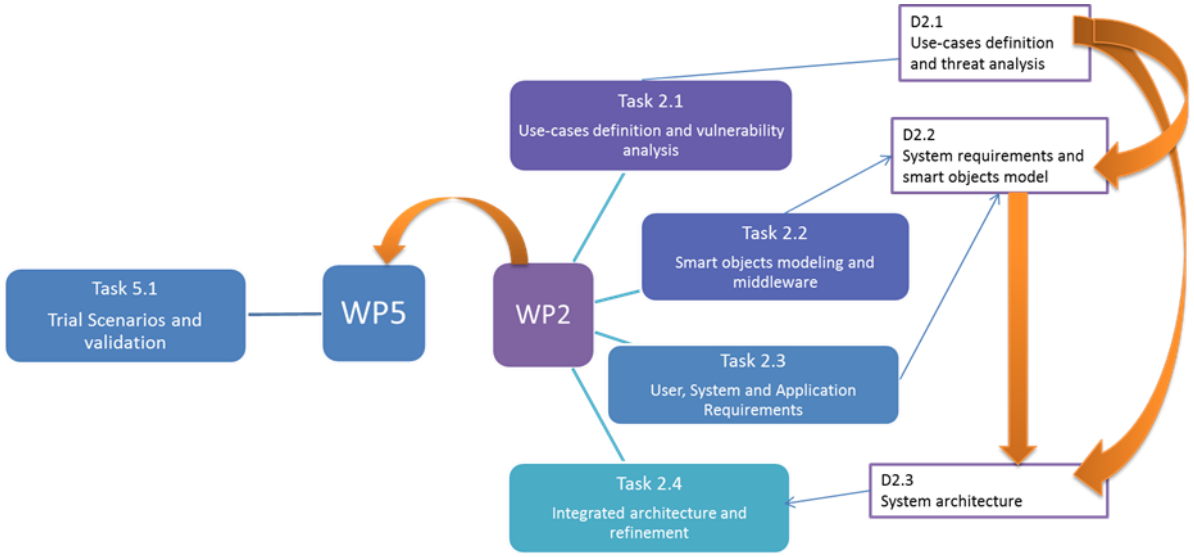


Figure 2 - Relationship between D2.2 and other tasks/deliverables in RERUM

1.3 State of the practice and related work

Smart Cities represent a concept, a vision more than a clearly defined set of technologies. The claimed smartness is usually justified implementing automation and IT systems, many times decoupled, and forming completely closed systems due to little or no interest to public awareness of relevant data. An additional argument for reluctant stakeholders is the assumed need to control implemented systems. This assertion is true for critical infrastructures (e.g. traffic control) but there are plenty of cases where open data and open programming interfaces may stimulate innovation and rapid development of localised, tailored solutions. As in the Frost&Sulivan study [39] regarding challenging megatrends, a number of 5 out of eight areas, shown in Figure 3 should be fulfilled.

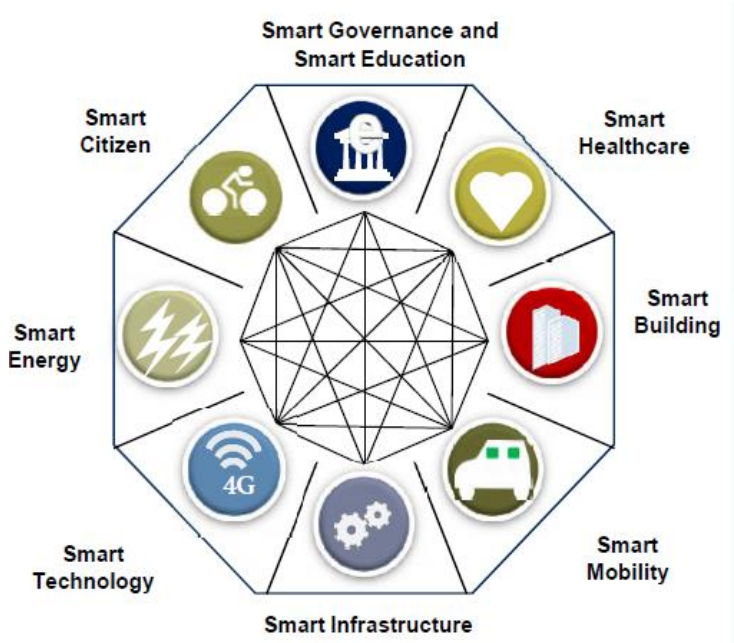


Figure 3 – Smart City Megatrends

Each of the listed "smart" areas represents a technological or application-related aspect that is relevant for the quality of living and service. Each one of them asks for active or passive involvement of large classes of city stakeholders: city management (as policy makers), citizens (as main beneficiaries of smart city services and businesses relevant on two aspects: as utilities providers (such energy or transport providers) or localized providers in the cities (shops, public events organizers, theatres).

Current efforts on smart cities aim to fulfil some broad targets such as sustainability, or generation of opportunities. Due to fast changes on the functional changes of city districts/areas, the planning is placed under increased stress for reliability and efficiency on a long term. Therefore, Smart City initiatives require serious planning capable of evaluating most of the constraints and responding properly to them on a medium/long term.

The current state-of-the-art contains a wide variety of technologies and results, ranging from the efforts of public authorities (such as transportation maps with real time track of buses, open data platform available in cities as Aarhus or Berlin) to the work done in private enabling platforms as is the case of Xively (initial Pachube) platform, which enable individuals to publish as web services home devices. While the domains of Smart Energy Grids, Smart Roads, Smart Building can be considered as closed families in a vertical view of value creation, the IoT appears as a basic technology to provide horizontal interoperability and heterogeneity solution via virtualisation of resources and data sources behind uniform interfaces (such as REST interfaces, for example). The

largest impact of IoT enablement is that data sensed in a certain part of a proprietary infrastructure may become available for processing and decision support in multiple contexts and multiple applications.

Smart Cities have provided multiple practical experiences across Europe, some as a result of research projects, e.g., the SmartSantander[1] platform, while others are public initiatives driven by local governments. We will list some examples next:

Smart City initiative	Specific things addressed
Berlin Open Data	www.berlin.de Berlin.de is the online portal to the city of Berlin, Germany. The site offers information and news about government activities, arts and entertainment, tourism, economy, traffic, weather, and more. Berlin Open Data (daten.berlin.de) is the data portal for the records of the Berlin public administration. Available records include: labour market; education; geography and urban planning; sports and recreation; elections; economy. A single, RESTful API is available to access data from both. Responses are formatted in XML, JSON, CSV, KML, and more.
Smart Aarhus	www.smartaarhus.eu Aarhus is the second largest city of Denmark is a flagship for Scandinavian way for thinking in Smart Cities area. Key aspect is the highlight on publishing as Open Data both streams and static data regarding public services, environmental data, business relevant data, everything support by a coherent wireless networks infrastructure.
Smart City Vienna	smartcity.wien.at/site/en/ City of Vienna drives an active programme regarding reshaping for smartness, enablement and interactivity. Vienna has set itself the task of consistently and continuously modernising the city in order to reduce energy consumption and emissions significantly without having to forego any aspects of consumption or mobility. The Smart City Wien completes its projects using the latest technology, in compliance with high ecological standards, in a socially responsible way and with the greatest possible involvement of all citizens.

[1] www.smartsantander.eu/

As mentioned above, there is a number of EU funded, open source and commercial initiatives opening IoT towards Smart Cities implementations.

From RERUM's point of view we consider the following to be very influential:

Project name	Funding authority	Description
Smart Santander www.smartsantander.eu	EC (FP7)	Reference deployment of a massive volume of sensors in the city of Santander (Spain). Collected streams are published via REST API.
IOT-A www.iot-a.eu	EC (FP7)	Reference architectural perspective for IoT EC funded projects. Offer an aligned methodology to design and implement IoT projects
ALMANAC www.almanac-project.eu	EC(FP7)	Peer projects for RERUM targeting various approaches regarding the Smart and Reliable IoT for Smart Cities solutions.
CityPulse www.ict-citypulse.eu		
COSMOS www.iot-cosmos.eu		
SMARTIE www.smartie-project.eu		
VITAL Vital-project.eu		
SOCIOTAL Societal.eu		

2 Requirements

The definition of the requirements is an essential task, since it determines functional needs of the RERUM ecosystem. This section derives several types of requirements from the use cases that were described and analysed in RERUM Deliverable D2.1. Some of these requirements provide value and utility in a more generic way and are applicable to all use cases, while others are more specific and use-case dependent. The identified requirements are presented in the following sections, categorized as follows: System Architecture Requirements (2.1), Application (2.2), Networking and Quality of Service (2.3), RERUM Devices (2.4), Middleware and Virtualization (2.5), and Security and Privacy (2.6).

2.1 Requirements for the System Architecture

In the subsections 2.2 – 2.6 the functional requirements for the system will be analytically presented. These requirements are related to functional components that help define specific behaviour of system functions, i.e. bootstrapping, authentication, network interconnectivity, cryptography etc. In this section, the non-functional requirements for the system architecture will be presented. The **Non-Functional Requirements (NFRs)** are requirements that specify criteria to judge the operation of the system, rather than specific mechanisms [25]. The non-functional requirements are the constraints, the goals, and the overall attributes of the system; these cannot be easily measured in terms of technical work that is needed to address them, so they are described in a textual format below and each of them are numbered as “NFR-#X”.

The RERUM system is built upon the basic requirement for **security-by-design (NFR-#1)**. This means that the system will incorporate from its design process mechanisms that will enable the secure interconnectivity of devices, the secure bootstrapping and authentication, as well as the secure exchange of information throughout the whole lifecycle, starting from the capturing of sensing data from the sensors, and securing the whole route of the data going up to the applications [24]. The system will be designed securely by assessing the vulnerabilities and threats of Smart City applications, and it will incorporate both proactive and reactive mechanisms. The proactive mechanisms will ensure that no attacks will have an impact on the system, and that the reactive mechanisms will ensure that the impact of an attack will be minimized via the rapid identification of the attack and the launch of appropriate mitigation strategies.

Privacy-by-design (NFR-#2) is another important system requirement [24]. Personal user data will be protected throughout the whole system, starting from the originating device (i.e. the user smartphone, or the device that monitors the user). In this respect, several privacy-preserving mechanisms (i.e. compressive sensing, anonymisation and other Privacy Enhancing Technologies) will be implemented as software on the RERUM devices, such that a sufficient level of privacy is ensured, taking into account the requirements of the application. These application requirements regarding the user’s privacy are crucial for limiting the usage but also already the collection of data and especially RERUM needs to avoid disclosure of information to other applications.

Additionally, another fundamental requirement for RERUM is the **trustworthiness (NFR-#3)** of the system and its devices. Depending on the application’s requirements, only trusted devices should be able to exchange information, and the devices should send data to only trusted applications/users. In order to avoid malicious or misbehaving devices to affect either the decisions of the whole system or the applications, the devices should be able to identify the trustworthiness of the neighbouring devices [73]. This applies to both the control and the data plane. In order to do so, mechanisms for measuring the trustworthiness of each device (in terms of reputation) and efficient reputation management mechanisms will be the core of the system architecture.

The architecture should also ensure the **reliability (NFR-#4)** of the system with respect to system failures and the **resilience (NFR-#5)** of the system with respect to the ability to recover from faults and failures. The system should also increase its **robustness (NFR-#6)** to faults and attacks [75]. The RERUM system should be designed to be reliable and provide reliable services to the users. In this respect, the system architecture will ensure a minimum number of failures of the system, while it will minimize the mean time between failures. Respective parameters should be included in the monitoring functions of the smart devices and should be related to the self-healing mechanisms. On the other hand, when a failure happens (i.e. a local failure), the system should be able to identify it fast, to find the origin and the cause of the failure, and to recover fast, avoiding an escalation of the problem that may result to a global failure of the overall system. The system should always be able to recover fast from problems, attacks, malfunctions or other degraded states in order to be able to support the requirements for availability, high performance and QoS. System malfunctions and security attacks may result to degraded system performance, unavailability of RDs and information. The system should be designed with respective components and mechanisms to ensure that any faults will not affect its overall performance and operation. For example, anti-jamming, or spectrum-handover mechanisms ensure the robustness of the system against jamming.

Directly related to the above is the architecture requirement for **high availability (NFR-#7)** of the system [75]. Due to the requirement to support various applications, including some that are related to emergencies and alarms, the system architecture, the network and the components should always be available. For example, monitoring a human's health needs to always give access to the patient's data. Comfort quality monitoring should also be able to always access the measurements to be able to identify hazards, like smoke in case of a fire, in the home/business environment. In this respect, the system architecture should be designed in such a way that the measurements from the devices will always be available to the applications upon their request.

The availability of the system depends on the load of the centralized entities that take the decisions. The larger the number of the interconnected devices the higher is the load of the centralized entities. To address this issue, the system should be designed to be scalable (**NFR-#8: scalability**) enough support the interconnectivity of a large number of RDs without degrading its performance. Since centralised systems are seldom scalable, distributed and self-*mechanisms should be included in the system, which could also include hybrid architectures with local and global controllers/decision making elements (cluster heads). The implemented mechanisms should ensure low communication overhead, ensuring fast decision making and high performance.

The distributed or clustered approaches impose as a non-functional requirement for the system architecture the ability to support **multi-level data processing (NFR-#9)**. Depending on the application and the security/privacy requirements the data may need to be aggregated/processed/fused at different system levels, i.e. at device level, at the gateways, at intermediate nodes (cluster heads or neighbouring nodes), etc. For example, Compressive Sensing [89] aggregates/fuses/encrypts data at intermediate nodes, but this may not be efficient for some types of private data. Another example is related with the requirement to transmit aggregated data of user speeds in the traffic monitoring use cases for privacy preservation. Similarly, fusion/encryption at device level may waste resources/energy.

Most devices used in IoT applications are power constrained since they usually operate on batteries. This means that the additional mechanisms that RERUM will implement and embed on the devices should comply with the requirement for **energy efficiency (NFR-#10) [19].** In this respect, the developed mechanisms should be lightweight minimizing the energy consumption of the devices: unused devices should enter sleep mode; trade-offs for energy efficiency should be considered; when battery is low, only the most urgent and important mechanisms should be operating. This way, the system and the measurement availability in the long term will be ensured.

Another non-functional requirement that relates to trade-offs between security/privacy and energy efficiency/system performance is the requirement for **different privacy/security levels (NFR-#11).** The system should be able to support different levels of security and privacy, depending on trade-offs with performance, energy efficiency, the usage of data in internal/external applications, etc. Strong privacy and security mechanisms may require a lot of resources that are not available to all types of RDs. On the other hand, lightweight mechanisms may not fully protect the system and the data. The architecture should support both approaches to ensure that the needed level of security/privacy can be maintained.

IoT devices can be mobile i.e. traversing around a city area, which may result in them being disconnected from the system in areas that have no network coverage. For addressing this problem, the system shall be able to support **opportunistic communications (NFR-#12)** in order to guarantee data availability [16]. When Internet connectivity is not available (i.e. due to mobility, interference, etc.) opportunistic communication technologies are able to avoid loss of data. For example, efficient store-and-forward techniques will be able to transmit the data whenever Internet connectivity is back alive.

Finally, a key requirement for the system architecture is the alignment with the architectural reference models of other IERC projects and especially IoT-A. IoT-A has developed a set of reference architectures to be used for developing IoT applications. RERUM's architecture will be **aligned with the IoT-A [55] view (NFR-#13)** of the architecture of Internet of Things, in order to be able to utilize the excellent results of this and other projects and avoid re-inventing a new architectural model from the scratch. A first attempt to address this requirement is being presented in the next section (Section 3) where the model of IoT-A is considered as a basis for the definition of the RERUM RD model.

2.2 Application Requirements

Application requirements determine the interaction between the end-user and the RERUM ecosystem. The end-user could be a human or an external application. This section captures those application requirements, which are related to the specific characteristics and needs of each use case.

For the Home Energy Management use case (UC-I1), the remote control and monitoring of RERUM devices (RDs) is an essential requirement that enhances the end user service experience. The remote control should be provided through user-friendly interfaces (e.g., via web browsers) in a secure manner. Regarding the security, there may be different levels of security, depending on the criticality of the interaction between the end user and the application in terms of commands and queries.

Title	Remote control of RDs
Priority	High
Requirement Level	REQUIRED
Description	The end user is able to remotely control, monitor and program RDs. For example, the end user should be able to securely log-in via a web browser to the Home Energy Management application and perform specific operations, such as: • Wake up RDs • Send commands to RDs which include actuators and sensors • Query RDs for information. • Configure RDs The operations may be characterised by hierarchical security, in the

	sense that more critical operations (e.g., configuration and actuation) may require additional authentication than i.e. monitoring.
Rationale	This requirement is necessary for enabling the remote control of RDs. It is useful for operation and maintenance purposes.
Pros and Cons	Pros: Remote control provides added value to smart city applications and more services including more transparency to the end-user. Cons: Adds the need to secure remote access.
Depends on	Network connectivity of the devices.
Competes with	Nothing

Req. 2.2-1 Remote control of RDs

RERUM's basic functional goal is to enable an application to use RERUM technology to receive information from the sensing elements or to send commands to actuating elements. For example, in the Smart Transportation use case (UC-O1), the application must be able to read and process the information provided by various types of sensors such as accelerometers or positioning devices. In detail, it needs the following information: GPS coordinates, Speed, acceleration vectors and also the wireless connection data (RSS values of sensed access points/base stations). This information is necessary for calculating the speed and position of vehicles. Without the access to this information, the application would not properly work. In any case, the transmission of these data to the application server must comply with the RERUM privacy and security goals.

Title	**Suitable Sensory data can be released to the application**
Priority	High
Requirement Level	REQUIRED
Description	If a device is equipped with sensing elements that allow capturing of specific sensory data required by the application, the RERUM system shall allow the device to release only those specific and suitable data to the application if no other policy for security or privacy are violated by this release.
Rationale	To let the application acquire the needed sensory data and only these
Pros and Cons	Pro: Direct access to sensory information would give high data quality. In the UC-O1 this would allow an accurate positioning of user device and offer the potential to provide an application with an automated mode for travel inference; Con: Direct access to sensory information would pose a high risk of user exposure
Depends on	- application's implementation and data quality needs (out of RERUM scope) - policy (possibly depends on the individual citizen)
Competes with	Nothing

Req. 2.2-2 Suitable Sensory data can be released to the application

Another critical application-related functional requirement that RERUM must meet is that needed sensory data are provided to the application at the required sensing rate. Using again as an example the Smart Transportation Use Case (UC-O1), related applications would benefit if the rate, meaning the frequency of getting new readings, at which the sensed data are collected and also transmitted is sufficient to meet the application's demands. If a device sends location information once every hour, then the resulting traffic estimation would be insufficient and unrealistic. Similarly, if the device sends location information every millisecond, then the application will not gain in accuracy, but the device will waste significant energy resources and the network will get congested due to the excess number of packets transmitted by the devices. From this example, one can easily assume that the rate under which the devices sense the environment should be adjustable in order to meet the application demands. Furthermore, the applications should be able to know this sensing rate, in order to be able to produce accurate results. For example, if the device senses the location every one minute and the application assumes that the sensing rate is every 2 minutes, then the extracted speed of the device would not be correct. Moreover, the gateways or the application should be able to change the sensing rate of the devices (if they are authorised to do so), to meet specific application or energy efficiency requirements.

Hence, an adjustable sensing rate that can be controlled by the application would be beneficial for each application to manage individually the trade-off between the number of transmissions and sensing data accuracy. Also from a privacy point of view an adjustable sensing rate might be beneficiary.

Title	**Adjustable rate of data collection**
Priority	High
Requirement Level	REQUIRED
Description	Allow adjusting the rate at which data are sensed, and also control at which rate the sensed data are transmitted.
Rationale	Collecting data can be adjusted to the specific needs of the application. By this, the application can reduce the currently sensed data quality and increase energy efficiency and save network resources.
Pros and Cons	Pro: can also reduce the transmitted information of users, and save energy and network resources Con: can increase the overhead and coordination between the applications and the deployed devices
Depends on	- application specifics (out of RERUM scope) - current network utilization
Competes with	Nothing

Req. 2.2-3 Rate of data collection

Another application requirements would be how timely would be the transmission of the sensory information. Again the UC-O1 for smart transportation applications is a good example, since location information is time sensitive in the sense that the collected information may become obsolete and hence useless for the end-users (e.g., traffic estimation).). To this end, the device participating in a time-sensitive traffic application should be able to estimate whether the acquired sensory data

should be sent or not to the application server, depending on their validity. For example, out-dated data should not be transmitted by the device, which also results in energy savings. Furthermore, the application might require the RERUM system to allow for certain sensory information to be transmitted within a given time-constrained to the application.

Title	**Time-efficient connectivity of devices for data uploading to application**
Priority	High
Requirement Level	REQUIRED
Description	If the collected data are allowed to be transmitted then it needs to be able to reach the application server within a time constraint, otherwise it fails to serve the applications needs.
Rationale	For example in UC-O1, the requirement for real time traffic estimation requires each device to upload sensory data to the application server that are not older than i.e. 10min.
Pros and Cons	Pro: Ensures low local (device) storage requirements when not in connection, since out-dated data will be deleted from the local device. Con:
Depends on	RERUM-external service requirements Network requirements
Competes with	Nothing

Req. 2.2-4 Time-efficient connectivity of devices for data uploading to application

2.3 Networking and QoS Requirements

The deployment of IoT ecosystems poses several challenges; a major one is the requirement for the installation of RDs and other devices, particularly in use cases that involve massive deployments of sensors (e.g., environmental monitoring, smart transportation). Several factors must be taken into consideration during the installation of the RDs, such as the physical protection, the interference from other networks and sources, as well as the possible power sources [94]. Furthermore, the connectivity between devices and the Internet or between different devices is a major challenge, taking into account the expected high number of interconnected devices [88]. Prolonged lifetime of IoT networks is also a major challenge, since the massive deployments of devices raises several difficulties if battery replacements are required [90]. Another challenge is the need to support and satisfy the requirements of a large number of different applications and services, which makes it difficult to design a unified system [20]. Low integration costs and market attractiveness is expected to play a key role in the deployment of the IoT networks. The integration of the IoT networks should go through the enhancement of today's communications networks, inheriting crucial benefits from these systems (e.g. ubiquity), while keeping the installation and updates convenient and cost-effective. On top of all of these challenges, the lack of common standards for a unified IoT architecture makes things even more difficult [79]. Besides the network and hardware challenges, IoT applications also raise several challenges for software developers.
In what follows, the RERUM requirements are discussed as they have been extracted based on the use cases defined in RERUM Deliverable D2.1.

2.3.1 Network and service operator requirements

The basic IoT communication channel model of the IoT-A project [55] has been adopted, which includes several constrained or unconstrained networks, the Internet, and telecommunication access networks; see **Figure 4**. As by the definition of IoT-A, the unconstrained networks are characterised by high-speed communication links (e.g., in the order of Mbps), while the link-level transfer latencies are small due to network-level congestion and not due to physical layer limitations. Examples of such networks may include 802.11-based technology or wired technologies, such as Ethernet. On the other hand, constrained networks provide relatively low transfer rates (e.g., less than 1 Mbps), as offered by, e.g., IEEE 802.15.4. These networks are also characterised by high latencies, due to the involved low-bit rate physical layer, or the power-saving policies (e.g., sleep-wake mechanisms for energy-efficiency reasons). The following sub-section describes the requirements from the point of view of the communication network (access, constrained and unconstrained networks), and network service providers. The access network facilitates the interconnectivity of the devices. Examples of such access networks include wireless ones, employing for example GPRS, LTE, WiMax, or wired ones, using for example xDSL technology. The gateway provides the interconnectivity between the access networks and the constrained/unconstrained networks.

In RERUM use cases, as in most IoT deployments, a large number of interconnected RDs are expected. These objects may be directly connected to an access network (e.g., a cellular network), although the most common case will be a connection to the network through a gateway, which aggregates, processes, and retransmits data (i.e., traffic aggregation [58]) through the access network, in an attempt to reduce the traffic congestion. Nevertheless, even in that case, the number of network gateways that will require access to the network is expected to be relatively high compared to the cases of conventional communication scenarios (e.g., human-to-human communications). In this context, the networks that will provide such interconnectivity will have to support a large number of attached devices/objects.

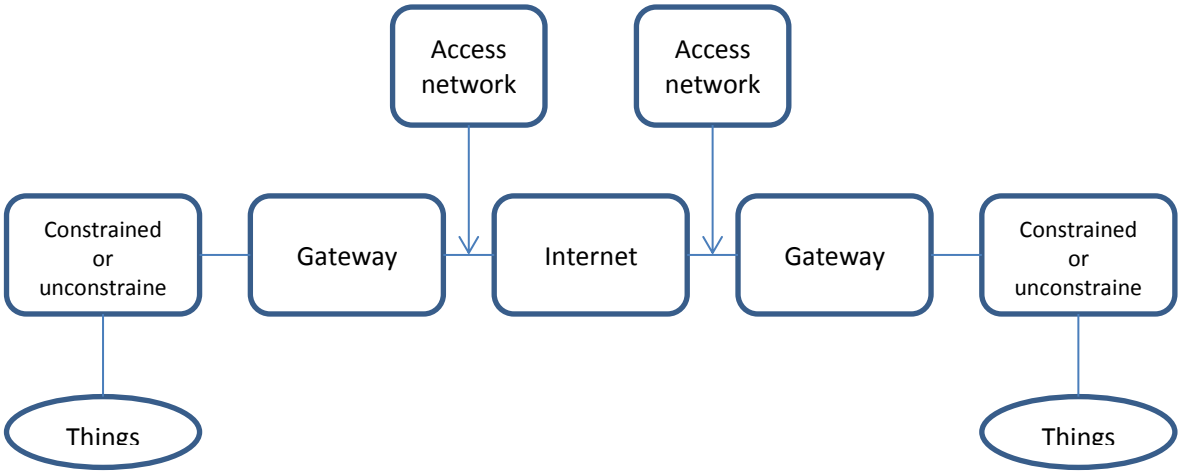


Figure 4 - Communication channel model [55]

Title	Support a large number of attached devices/objects
Priority	High
Requirement Level	REQUIRED
Description	Smart city applications are expected to allow the attachment of a very large number of RERUM Devices (RDs) to the network. RDs may be connected either directly or via Gateways/Aggregators using various technologies.
Rationale	Conventional networks are designed for human-to-human

	communications, where the number of connected devices is comparable to the number of customers. With Smart cities applications, the number of devices (RDs) that must be attached/connected to the network is expected to increase dramatically.
Pros and Cons	Pro: Networks prepared for supporting a large number of attached devices will result in low probability for traffic congestion. Con: Nevertheless, this comes at the cost of extra complexity.
Depends on	Dynamic spectrum management
Competes with	Nothing

Req. 2.3-1 Support of a large number of attached devices/objects

One of the most important networking aspects is the ability to manage the network traffic produced by the RDs and, to accordingly manage the spectrum resources in order to ensure smooth and reliable operation of the network. Spectrum management [13], [10] includes a set of mechanisms that are dealing with the efficient usage of the wireless spectrum and it includes spectrum sensing, spectrum assignment, spectrum handover, spectrum sharing. The current implementations of smart object devices, like sensors and actuators, use wireless interfaces to interconnect, whereas the most frequently found technology is IEEE802.15.4 [21], with IEEE802.11 [60] also used in various deployments. These technologies utilize the unlicensed Industrial, Scientific and Medical (ISM) spectrum bands that can be easily overcrowded, resulting in high interference, which known to have severe impact on the performance of wireless networks [83]. To mitigate the interference issue, Cognitive Radio technology, utilizing the concept of Dynamic Spectrum Access (DSA) has been proposed. Managing the spectrum access in a dynamic and opportunistic way is a flexible way of identifying and accessing the unused spectrum bands. DSA also contributes to the efficient interconnectivity of a large number of devices since it minimizes interference between neighbouring wireless links.

Title	**Dynamic spectrum management**
Priority	High
Requirement Level	REQUIRED
Description	Due to the large number of attached/connected RDs spectrum utilization must be optimized, such that: • RDs and conventional mobile devices can co-exist, in the case of licensed spectrum utilization • the number of attached/connected devices is maximized In this sense, spectrum should be dynamically allocated to devices. This could be achieved either automatically (e.g., via Cognitive Radio mechanisms) or via the network operator, which may be able to make decisions on the resources utilization and how they will be shared among devices. This requirement applies to the wireless devices within the constrained/unconstrained networks.
Rationale	Random RD deployments may cause network malfunctions, e.g., due to radio interference and resources re-use.
Pros and Cons	Pros: Congestion avoidance during high traffic situations. Cons: complexity of dynamic spectrum management mechanisms. In distributed approaches, trustworthiness of RDs should be ensured.

Depends on	Hardware capabilities of the devices; the support for the large number of attached devices
Competes with	Nothing

Req. 2.3-2 Dynamic spectrum management

There are two basic variants on selecting and assigning spectrum in wireless devices: centralised and distributed [82] approaches. Depending on the deployment of the RDs, their capabilities and the requirement for the spectrum utilization, the most appropriate approach should be selected. For example, distributed frequency selection can be utilized in order to optimise the spectrum usage in small deployments, because it makes better use of the unutilised frequencies. On the other hand, centralised frequency selection can be optimal in terms of fairness and can be utilized in large deployments. Centralised frequency selection includes the scenario of clustering, where the frequency selection is split into groups of network devices, and at each group there is a cluster head that decides for the frequency selection in that group (in coordination with other cluster heads in the same area) [82], [11].

Title	**Distributed and centralised frequency selection**
Priority	Medium
Requirement Level	RECOMMENDED
Description	The RDs should be able to decide upon the spectrum/channel they will choose in an either distributed or centralised way depending on the situation.
Rationale	When two RDs need to exchange messages in an ad-hoc manner, they should be able to decide by themselves which frequency/spectrum they will use. On the other hand, in case of a large number of RDs in a deployment area, centralised frequency selection and clustering may result in a more efficient frequency usage.
Pros and Cons	Pros: Efficient frequency utilisation Cons: in centralised approaches there might cause unnecessary signalling traffic in the network.
Depends on	Nothing
Competes with	Nothing

Req. 2.3-3 distributed spectrum selection

As mentioned above, interference in the ISM bands can severely affect the performance of a wireless network. Other, unused spectrum bands may be better suited for connecting RERUM devices. This may mean that the devices should be able to operate also in licensed bands, but with the ultimate constraint that they must not interfere with the operation of licensed users [14].

Title	Operation in unused licensed spectrum bands
Priority	Medium
Requirement Level	RECOMMENDED
Description	The RDs should have the capability to operate freely in both licensed and unlicensed spectrum bands, given that they comply with the interference requirements for each case.
Rationale	In order to avoid interference from the RDs to the other wireless devices and vice versa, the RDs should be able to operate in a large spectrum area or in various spectrum bands. To achieve this, efficient spectrum sensing and allocation techniques must be used.
Pros and Cons	Pros: Optimised wireless operation, interference avoidance, energy efficiency (not always) Cons: requires advanced hardware capabilities
Depends on	Dynamic spectrum management and hardware capabilities
Competes with	Nothing

Req. 2.3-4 Operation in unused spectrum bands

The smooth and normal operation of the network usually depends on the capability to operate and manage the devices that are attached to the network in an efficient and orchestrated way [97]. This usually involves functions such as network visualization, topology view, signalling tracing, performance measurements and statistics. The constrained networks consist of devices that have limited resources and thus they may not have the capabilities to run complex algorithms for achieving advanced security, privacy, management, network connectivity, etc. Thus, in those cases, centralised network management should be considered.

Title	Centralised management of constrained networks
Priority	High
Requirement Level	REQUIRED
Description	The management and maintenance of devices within the constrained networks by a network administrator should be possible. Here, the administrator refers to the entity (e.g., end-user, service provider, etc.) that has the rights to monitor and configure the devices. For example, the service provider could update the firmware of the devices.
Rationale	The administrator of the constrained/unconstrained network may be able to manage the devices attached to the network in order to ensure the smooth operation of the network and the services to the end-user.
Pros and Cons	Pros: Easier troubleshooting. Cons: Centralised management usually results in higher backhaul bandwidth requirements.
Depends on	Nothing

Competes with	Nothing

Req. 2.3-5 Centralised management of constrained networks

To support the interconnectivity of a large number of devices from the network point of view, and to perform a more efficient network management and monitoring of the devices, the partitioning of the network into clusters will be analysed as described in [11]. This can be also applied in the scenario of the constrained networks. Since most smart object devices are resource-constrained, the partitioning into clusters will increase network's flexibility.

Title	**Partitioning of the network into clusters**
Priority	MEDIUM
Requirement Level	RECOMMENDED
Description	The network should be able to support the grouping of devices into clusters.
Rationale	In Smart City applications, the number of RDs that must be attached/connected to the network is expected to increase dramatically in the near future. In order to be able to provide the required QoS, the network should be able to support the interconnectivity of a large number of devices, without affecting the performance. This will be facilitated by partitioning the network into clusters.
Pros and Cons	Pros: Networks prepared for supporting a large number of attached devices will result in low probability for traffic congestion. Cons: Nevertheless, this comes at the cost of extra complexity and higher energy consumption of the cluster heads.
Depends on	The existence of devices that have more resources (battery, computational, etc.) that can play the role of cluster head
Competes with	Nothing

Req. 2.3-6 Partitioning of the network into clusters

The connectivity between the RERUM devices and the Internet via the access network should be flexible. This means that the devices could be able to choose different network access technologies (e.g., WiFi, or cellular) depending on the available resources and energy requirements.

Title	**Multi-technology and multi-operator connectivity**
Priority	Low
Requirement Level	OPTIONAL
Description	The devices should allow their connectivity with various technologies and network operators.
Rationale	The devices may have the option to select any available network operator and network technology for connecting to the internet, considering the available networks in the area and the available hardware on the RD.
Pros and Cons	Pros: allow a nearly ubiquitous connectivity

	Cons: increased hardware complexity
Depends on	Hardware capabilities of the devices (i.e. existence of multiple network interfaces)
Competes with	Nothing

Req. 2.3-7 Multi-technology and multi-operator connectivity.

The data that are exchanged between the smart objects may be related to different applications. In order to be able to support the required QoS of each application, a mechanism for message prioritisation should be included in the network devices [96]. That way, messages with higher priority than others (i.e. emergency messages, or real-time messages) should be entered either in the first place of the transmission queue, or in a higher priority transmission queue.

Title	**Message prioritisation**
Priority	High
Requirement Level	REQUIRED
Description	A proper prioritisation mechanism is needed for classifying messages' criticality and enabling prioritization and network congestion management.
Rationale	The notifications are essential for the proper interaction between the RDs and the end-user. The on-time delivery of emergency notifications is critical. For example, in fires/hazardous situations, the users should be immediately informed.
Pros and Cons	Pros: Enables emergency services and alarms. Cons: Requires advanced scheduling algorithm
Depends on	QoS of applications
Competes with	Nothing

Req. 2.3-8 Message prioritization

QoS allows proper service prioritisation and device communication, scheduling mechanisms, and session continuity [15]. Critical services have certain QoS requirements (depending on the application they support) for data rate, delay, jitter, etc. and proper mechanisms should ensure that these requirements are met.

Title	**Overall QoS**
Priority	High
Requirement Level	REQUIRED
Description	The system should be able to meet the various QoS requirements of the applications.
Rationale	Applications have different requirements in terms of QoS (i.e.

	throughput, delay, BER, etc.) and the system should be able to consider these requirements in order to prioritise the resources in favour of critical applications or high-QoS requirements.
Pros and Cons	Pros: Ensures system reliability, Cons: System complexity
Depends on	Application requirements
Competes with	Nothing

Req. 2.3-9 Overall QoS.

For optimising the system deployment and maintenance costs and for supporting the mostly distributed nature of the system, the system components and the network functionalities should support self-* functionalities, like self-monitoring, self-configuration and self-healing [57], [26], [29]. Local monitoring and healing can also be an effective mechanism for securing the network and detecting possible compromises of devices. Furthermore, the network should be able to evaluate its status and detect changes in the topology regardless of the events that caused those changes. The network should be adaptive (e.g., in terms of functionalities, resources management) for offering the best QoS possible to the connected devices.

Title	**Self-* mechanisms**
Priority	HIGH
Requirement Level	MANDATORY
Description	All system components and network functionalities shall support self-* mechanisms
Rationale	The distributed nature of the system should be ensured with the support for self-* mechanisms, like self-configuration, self-monitoring, self-healing. RDs should be able to configure themselves or in groups without the need of centralised configuration, monitor their behaviour (or their neighbours) and trigger healing (reconfiguration) mechanisms when required. Auto-bootstrapping is also included in the self-* mechanisms.
Pros and Cons	Pros: allow fast decision making, fast configuration, minimises signalling overhead, avoids centralised “single-point-of-failure”, supports scalability. Reduction of operational and maintenance costs. Cons: not always achieved best results due to only local knowledge about system behaviour, distributed configuration is error-prone due to possibility of being affected by un-trusted neighbour RDs. Increased complexity and equipment/devices costs.
Depends on	None
Competes with	The requirement for centralised management of RDs

Req. 2.3-10 Self-* mechanisms

2.4 RERUM Device Software, Hardware and Modelling Requirements

In this section we describe the requirements for RERUM Devices and Gateways. We consider requirements in terms of RD hardware, as well as the software running on them. We first define a set of baseline requirements for the hardware of RDs. We subsequently discuss requirements for the hardware of gateway devices, which are expected to have higher capability. In the following subsection, we present requirements for RD software. Lastly we discuss requirements related to the modelling of RDs.

2.4.1 RD Hardware Requirements

At this early stage of the project, it is not feasible to quantify the exact computational requirements (e.g. processor speed or memory size) for RDs. We thus base the analysis on products currently available in the market.

The International Protection (IP) rating [17] is an index commonly used to rate the physical protection of an enclosure, or a system in general, against dust particles, water (high humidity, rain), and against accidental damage. This is very widely used in electronic device specifications. Enclosure drills for input/output cables, antennas, attachment support, lock, etc., must meet the IP rating listed in the RD's specification. The enclosures must be able to support the RD's total weight, including the weight of batteries, which often significantly affects the overall weight of the devices.

Title	Enclosure IP rating
Priority	Low
Requirement Level	RECOMMENDED
Description	The enclosure hosting a RD must have a minimum IP (International Protection) rate to protect the hardware inside from outdoor conditions.
Rationale	Devices installed outdoors will be exposed to bad weather, rain, high winds, low and high temperatures, fog, and high humidity conditions. The RD hardware will normally not be resilient to such conditions and it needs to be protected. This is not required, and even not recommended, for RDs installed indoors, since these enclosures are expensive and hard to manufacture.
Pros and Cons	Pros: Protection against damage due to weather conditions Cons: Price
Depends on	Whether the installation is outdoors or indoors.
Competes with	None

Req. 2.4-1 Enclosure IP rating

The core of a typical RD will be a microcontroller that executes the software, measures sensing element readings, controls communication interfaces and executes RERUM-related security- and privacy-enhancing mechanisms. Current state-of-the-art devices available in the market use 32-bit, 32MHz microcontrollers [81], [80]. For RDs requiring higher energy efficiency, even lower-end microcontrollers can also be considered.

Title	Microcontroller performance
Priority	High
Requirement Level	REQUIRED
Description	Microcontroller performance in terms of frequency and register width
Rationale	The MCU will need to be able to support all of RERUM's security- and privacy-enhancing mechanisms.
Pros and Cons	Pros: High performance can support complex algorithms. Cons: Higher MCU frequency often, but not necessarily, results in higher energy consumption.
Depends on	None
Competes with	None

Req. 2.4-2 Microcontroller performance

RDs are intended to exhibit low power consumption characteristics. Especially in the case of RDs powered by batteries or an energy harvesting mechanism (e.g. solar cells), low consumption can increase autonomy very significantly. The main contributors to energy consumption are:

- Micro Controller Unit (MCU) activity
- Network interface activity (idle listening, transmission, reception)
- Flash write operations

Thus, in order to reduce energy consumption, putting a device to sleep during periods of inactivity is common practice [84]. The underpinning hardware (MCU, RF transceiver and other peripherals) must be able to support this functionality. More specifically, contemporary chips can operate under multiple Power Modes (PMs). To enter a low power mode, some peripherals get gated (their power supply is turned off), and subsequently the MCU turns off. Different power modes turn off a different number of peripherals, and therefore exhibit different power consumption levels. Transitions between power modes are not instant, and the exact timing depends on the chip as well as the power mode the chip is switching to, which in turn depends on the number and type of peripherals that need to be switched off. Transitions from low power modes to normal operation are not instant either. The software running on the RD controls transitions between PMs.

Title	Autonomous operation, processing consumption and low-power modes
Priority	Medium
Requirement Level	REQUIRED
Description	The MCU and other RD peripherals must provide support for multiple power modes, with each PM exhibiting a different level of energy consumption.
Rationale	Low power modes can increase RD autonomy significantly. This is important for battery-powered RDs, as well as for RDs powered through some energy harvesting mechanism, such as solar cells.

Pros and Cons	Pros: achieve better management of the available battery on the device
Depends on	None
Competes with	None

Req. 2.4-3 Autonomous operation, processing consumption and low-power modes

RDs must have both volatile and fast RAM, as well as non-volatile, persistent storage (Flash or EEPROM). Volatile memory is used to store program memory, interrupt vectors and data at run-time. Capacity and performance characteristics of storage should be suitable for RERUM firmware and storage of configuration, data and metadata. Current state-of-the-art volatile memory chips are capable of retaining their contents even under low PMs [81], [80].

Title	**Volatile memory**
Priority	High
Requirement Level	REQUIRED
Description	RDs must be equipped with volatile memory of size, speed and retention characteristics suitable to run RERUM applications.
Rationale	Required in order to be able to store program memory, interrupt vectors, data and metadata at run-time.
Pros and Cons	Pro: Increases the size, speed and retention capability of volatile RAM Con: May increase RD's total cost
Depends on	None
Competes with	None

Req. 2.4-4 Volatile memory

Persistent storage is used in sensor nodes to store local configuration or log files. From a hardware perspective, this is normally a flash storage device, connected to the MCU over an SPI or I^2C interface. Those flash devices are typically NAND or NOR technology [65], which is characterised by block-erasure and limitations in terms of the number of program-erase cycles a sector can undergo before storage capability starts deteriorating. This phenomenon is also referred to as "flash degradation" [65]. Additionally, flash writes are considered to be an expensive operation in terms of energy consumption and it is therefore desirable to optimise their frequency. Usually, this is achieved through the use of a flash-specific file system, such as Yet Another Flash File System 2 (YAFFS2) [95]. Those file systems take these characteristics into consideration; by using buffered writes and by spreading out write operations evenly across sectors (wear levelling), energy efficiency and flash hardware lifetime can both be increased.

Title	Persistent storage
Priority	Medium
Requirement Level	RECOMMENDED
Description	Support for persistent storage devices, such as external flash.
Rationale	Required in order to be able to store configuration and log files locally on RDs
Pros and Cons	Pros: Intelligent wear levelling algorithms may increase RD life expectancy Cons: increased complexity of the storage / file system firmware
Depends on	None
Competes with	None

Req. 2.4-5 Persistent storage

Communication interfaces on RDs are the basis to connect them to the RERUM network. Currently, IEEE 802.15.4 [54] seems to be the technology of choice for wireless sensor platforms and has been adopted by ZigBee [101] and 6LoWPAN [33]. In terms of wired communication between a sensing element and the MCU within a single RD, standard specifications with high market penetration are Serial Peripheral Interface (SPI) [76], Inter-Integrated Circuit (I^2C) [67] and Universal Asynchronous Receiver / Transmitter (UART) [9].

Title	RD Communication interfaces
Priority	High
Requirement Level	REQUIRED
Description	Wired and wireless interfaces in order to support inter- and intra-RD communication
Rationale	RDs must have a communication interfaces enabling intra-RD communication (between the MCU and sensing elements) and inter-RD communication (between different RDs)
Pros and Cons	Not applicable
Depends on	None
Competes with	None

Req. 2.4-6 RD Communication interfaces

Cognitive radio and dynamic spectrum management mechanisms are rely on Software Defined Radio (SDR), which requires radio interfaces that are reconfigurable by software [37]. In this respect, radio interfaces are capable of selecting the central frequency, bandwidth, modulation and other parameters, in order to meet the requirements of a transmission.

Title	Reconfigurable network interfaces
Priority	Medium
Requirement Level	RECOMMENDED
Description	The RDs should have reconfigurable wireless network interfaces.
Rationale	In order to enable dynamic spectrum management and cognitive radio mechanisms, RDs should have reconfigurable wireless network interfaces. That way, RDs should be able to identify both the central frequency (in a specific band) and the frequency band's width needed to support their application QoS requirements. For example, when they need to send small packets they don't need to use and hereby block the complete width of a 20MHz-wide channel of 802.11.
Pros and Cons	Pros: Optimised wireless operation, interference avoidance, energy efficiency (not always) Cons: Requires advanced hardware capabilities
Depends on	None
Competes with	None

Req. 2.4-7 Reconfigurable network interfaces

2.4.2 Gateway Requirements

The "Gateway" can be considered as an advanced device, so we specify additional requirements for RDs applicable to gateways only; the hardware requirements of standard devices were described in the previous subsection.

The additional functionalities to run on the Gateway to connect the RERUM network to the Internet, (mainly to the Application Server) require that the Gateway should have higher computational capabilities than the standard RDs. For example, the Gateway may be required to maintain routing tables for the entire deployed network of RDs. It will normally be equipped with multiple network interfaces, or it may be required to act as concentrator / aggregator of user data. It is thus desirable to have a high-end device to fill the role, such as a device being capable of running embedded Linux. The characteristics of what constitutes a higher-end device are derived by the requirements in this sub-section. Higher performance requirements apply to processor speed as well as to memory performance and capacity. In terms of volatile memory, Dynamic RAM (DRAM) is recommended due to its speed characteristics, complemented by persistent memory such as an SD card, MMC card or a Solid-State Drive (SSD).

Title	Gateway computational performance
Priority	Medium
Requirement Level	REQUIRED
Description	The Gateway must have a minimum CPU clock of 500 MHz (desirable 700 MHz), minimum register length of 32 bits (desirable 64 bits), a minimum volatile memory capacity of 256 MB (desirable 512 MB) and a minimum non-volatile memory size (for firmware and storage) of 1 GB (desirable 8

	GB)
Rationale	Due to additional functionality running on Gateways, higher performance requirements are mandated.
Pros and Cons	Not applicable
Depends on	None
Competes with	None

Req. 2.4-8 Gateway computational performance

One of the Gateway's roles is to connect the deployed RDs to external networks. To achieve that, the Gateway will need multiple communication interfaces. As discussed in use-case descriptions in D2.1, it is anticipated that the Gateway will be equipped with a low-power radio transceiver (e.g. a IEEE 802.15.4 compliant one), and with a second interface to communicate with the application server. Depending on the technology used for this connectivity, the second interface can be a wired Ethernet interface or a mobile broadband GPRS/3G/LTE (4G) wireless interface. Other interfaces, such us WiFi or WiMAX, may also be considered for different scenarios, depending on deployment-specific cost and performance requirements.

Title	**Gateway communication interfaces**
Priority	Medium
Requirement Level	REQUIRED
Description	The Gateway should have at least one wireless interface to be able to communicate with the RDs (this can be a 802.15.4 or a 802.11 interface) and optionally other network interfaces to communicate with the backbone internet (this can be an Ethernet interface or a mobile interface, i.e. GSM/3G/4G).
Rationale	This extra communication interfaces will be used to communicate with the application server.
Pros and Cons	Pros: Enables the efficient interconnectivity with the RDs and the global internet.
Depends on	None
Competes with	None

Req. 2.4-9 Gateway communication interfaces

2.4.3 RD Software Requirements

Energy efficiency for RDs is dependent on multiple factors, with the exact consumption being dependent on hardware components. More importantly, some operations are very demanding irrespective of the type of hardware used. Examples of operations in this category include RF transmission, idle listening and flash write operations. Because of this, it is important for software

operating on RDs to keep the frequency and duration of those operations as low as possible, in order to optimise RD energy source lifetime [81], [80]. Microcontroller activity also attributes to energy consumption, albeit considerably less so than RF activity and flash write operations. In order to improve energy efficiency for sensor nodes, the following design aspects are taken into consideration:

- MCU wake-up / sleep cycles
- RF wake-up / sleep cycles
- Energy efficient networking protocols. In this context
- Reduced frequency of flash write operations.
- Network protocol optimization in terms of transmission frequency and packet size

Many of these mechanisms can only achieve their purpose if the hardware provides low-power modes, MCU sleep / deep sleep functionality, peripheral / clock gating.

Title	Low energy consumption
Priority	High
Requirement Level	REQUIRED
Description	Implementation of several measures to obtain an energy-efficient operation. This includes RF and CPU sleep / wake-up cycles, optimized flash write operations, optimized network operations, and/or duty cycled MAC layer algorithms, with or without time synchronisation
Rationale	Increased energy efficiency increases the lifetime for battery-powered devices
Pros and Cons	Pros: Reduced energy consumption is always desirable, irrespective of whether a device is battery-powered or not. Especially in the case of battery-powered RDs, energy efficient operation will increase deployment lifetime.
Depends on	Autonomy, processing consumption and low-power modes
Competes with	None

Req. 2.4-10 Low energy consumption

Another requirement to support low energy consumption is related to data transmission. When a device is transmitting a packet i.e. with sensor measurements it consumes a large amount of energy [85]. The smart objects are assumed to be power-constrained devices, so their software should ensure that the data transmissions are minimised. One common way for this in sensor networks is to utilize the advantages of Compressive Sensing (CS), which compresses transmitted data, incorporating also a type of encryption [89]. That way, the number of transmissions is minimised, thus preserving energy. In order to do so, the data compression method used should be lightweight, in order to avoid consuming more energy for compressing the data than for transmitting them. Finally, depending on the application requirements, the level of compression should also be dynamically adapted to ensure a specific level of data accuracy, since the reconstruction of the original data (from the compressed) may result in some error.

Title	Lightweight dynamic data compression
Priority	High
Requirement Level	REQUIRED
Description	Data transmitted by RDs to RDs or the Gateway shall be compressed in order to save energy and prolong the network's lifetime. Data's compressibility may vary depending on the measured data (e.g. temperature, humidity). Moreover, the compressibility for a specific type of data may also vary if sudden changes appear in the surroundings of the area monitored by the network (e.g. a sudden change in the measured temperature due to a fire). The compression rate shall be adapted to the data compressibility, estimated on run-time. Furthermore, the compression algorithm shall be lightweight.
Rationale	This requirement is necessary, as RDs shall save energy by primarily minimising the communication cost.
Pros and Cons	Pros: Consumed energy minimization Cons: Data compressibility estimation may not be very accurate
Depends on	Data compressibility estimation mechanism
Competes with	None

Req. 2.4-11 Lightweight dynamic data compression

In order to support loadable software modules, real-time updating of software libraries and future installation of new crypto algorithms, it is necessary for RDs to support re-programmability over the air [72]. Inability to conduct Over-the-Air Programming (OAP) means that RDs would need to be un-deployed, programmed and re-deployed, a procedure which has considerable practical difficulties due to the potentially high number of RDs and the likelihood of them being deployed in difficult to reach locations, for example on tree tops or lamp posts [72].

This requirement underpins multiple requirements discussed elsewhere in this document. In a nutshell, OAP functionality requires an implementation-dependent combination of the following components [72]:

- An application layer network protocol, used to transport the update to the RD
- Multicast support at the network layer
- Support for on-device storage of multiple firmware images, one of which will be the one being executed during the OAP process, and another one will be the incoming updated image.
- Support for a 'golden' firmware image, to be used as a fall-back in case both the running as well as the incoming image get corrupted during the OAP process
- A mechanism to verify firmware validity
- The ability to build modules for the correct RD architecture from source code. This will require a build system for the architectures of interest, as well suitable toolchains. For

instance, when building modules for RDs powered by ARM MCUs, one candidate toolchain is arm-gcc [35].

- Support for loadable modules in order to be able to do partial updates. This involves linker script parameterisation [36], firmware with capability to load and unload modules dynamically at runtime (e.g. an ELF loader), storage for symbol tables alongside a supporting API for module developers.

Title	Over-the-Air Programming (OAP)
Priority	High
Requirement Level	REQUIRED
Description	Implementation of a mechanism that will allow reprogramming RDs with new firmware over the air. This can be either a full firmware or a partial update, in the shape of loadable software libraries
Rationale	Required for software updates to e.g. fix bugs or security vulnerabilities. Note that this functionality needs to be authenticated: Only authorised entities may initiate such update. This functionality needs to be conducted in a fail-safe fashion. Possible failures during the OAP must leave the RD in an operational state, so that the update can be attempted again.
Pros and Cons	Pro: Allows S/W updates after initial deployment Con: Increased complexity of the loader
Depends on	Persistent Storage, Volatile Memory, Authentication
Competes with	None

Req. 2.4-12 Over-the-Air Programming

2.4.4 RD Modelling requirements

On the way towards a functional architecture RERUM needs to define what the relevant attributes of RD are. An RD model will formali**s**e the way how an RD instance is considered in RD, how it interacts (where interaction may be: used-by, call, called-by, part-of, etc), how an RD execute**s**, etc. Below a number of initial capabilities of RD**s** are listed. The list will be evaluated and extended in D2.3 and D2.4 based on project evolutions:

- Differentiate the align the way how apps access platform and platform access RD
- Differentiate between federation and RD
- Differentiate how RD should be called
- Differentiate between RD as service provider (in IoT world) and backend HW or SW
- Clear RD to RD communication needs
- Clear cases for stateful/stateless RD
- Naming and addressing/discovery of RD

- Requirements for things RD and virtual RD
- RD monitoring
- RD enrolment, remove, passive
- RD interactions, in/outs

As specific for an IoT approach, RERUM considers a large heterogeneity of devices and services and provides a meaning to abstract/virtualise them towards a uniform model (Virtual RERUM Device – VRD) to be used by the applications. Of course, if RERUM shall be useful for many practical cases, e.g. also for high quality video surveillance, then RERUM needs to evaluate the potential delays generated by this approach. Therefore, a single unified set of non-contradictory requirements may be difficult to achieve in terms of expected performance for a given application domain. The following requirements are detailing what is identified as basic set of needs exposed on a system and the use cases requirements. RERUM specifications benefit from the conceptual base laid by the projects IOT-A and iCORE projects regarding the investigations around the IoT ecosystem components and functional identification of sensing, actuating and functional aggregation capabilities.

In order to give a positioning of identified artefacts and their model, a RERUM instance/platform will be considered as a bundle of technical meanings capable to expose a group of “managed” RD services to applications. This concept will be further specified and developed in D2.3 and D2.4.

One first requirement is to be able to expose RD services and federations’ services to applications to use them. In order to do this the interfacing should be made following a declared interfacing protocol. The mechanism to find out this protocol should be uniform and accessible to applications.

Title	Exposure of RD to applications
Priority	High
Requirement Level	REQUIRED
Description	Each RERUM instance should expose to the outside world a single uniform way to address and use RERUM Devices and RERUM Devices federations.
Rationale	Applications should have available predictable meanings to use one or more RERUM instances and differentiate between them.
Pros and Cons	Pro: Separate instance management and middleware control. Allow multiple instances running on top of virtualized RD based on different sets of policies. Cons: Restrict the freedom of application developers to freely access device level functions for immediate optimisation.
Depends on	None
Competes with	None

Req. 2.4-13 RERUM to apps interface

A RERUM instance manages the RD through middleware functions. In order to perform this interfacing should be done based on known schema(s).

Title	RERUM to RD interface
Priority	High
Requirement Level	REQUIRED
Description	Each RD should be managed based on own protocol. RERUM instance should be able to support and declare a minimal set of supported protocols and the coverage.
Rationale	Even if the applications directly use the RERUM Devices/ federations, the middleware should be able to "talk" to each of them (e.g. RD registry)
Pros and Cons	Pro: Ease the management policies and enforce explicit procedures for RD use. Cons: Restrict the freedom of application developers to freely access device level functions for immediate optimisation.
Depends on	RERUM to apps interface
Competes with	NONE

Req. 2.4-14 RERUM to RD interface

RERUM middleware should differentiate between federated RD and simple RD. This difference should be transparent to outer applications that should search, find, bind, and use them in the same manner.

Title	Difference between Federated RD and RD
Priority	High
Requirement Level	REQUIRED
Description	Applications may select and use single RD or federations of them. Difference should be made in RD registry indicating the "lead RD" for federations.
Rationale	A federation may implement a complex functionality provided directly by RERUM instance (e.g. combination of sensing and actuation) and the communication between RDs are done inside the instance without the involvement of outer application.
Pros and Cons	Pro: Federations expose "ready-made" services to be used. Cons: Security constraints should apply to whole federated objects. Federations should refer queuing methods for actuator access and signalling toward concurrent applications.
Depends on	RERUM to RD interface
Competes with	None

Req. 2.4-15 Difference between Federated RD and RD

RERUM should align to current practices in IOT space in order to be interoperable. One common approach is to expose devices/things as services and RD should consider this.

Title	RD as service provider
Priority	High
Requirement Level	RECOMMENDED
Description	RD should expose an “IoT like” interface towards the applications (e.g. REST)
Rationale	Uniform exposure in IoT space
Pros and Cons	Pro: Uniform approach. Cons: Limits flexibility, impose virtualisation on HW RD
Depends on	RERUM to RD interface
Competes with	NONE

Req. 2.4-16 RD as a service provider

Due to the fact that RD should reflect the state of exposed function (sensing/actuating/tagging) the RD model should consider if the state should be managed (via an interface) or not.

Title	RD state/stateless capability
Priority	High
Requirement Level	RECOMMENDED
Description	A RD should be able to maintain its state and the designer must express how stateful RD should be managed (e.g. actuators)
Rationale	Enable controlled state transition on actuators.
Pros and Cons	Pro: Uniform approach. Cons: Ask for predesigned transitions, state space should be known at design time
Depends on	RERUM to RD interface
Competes with	NONE

Req. 2.4-17 RD state/stateless capability

Federations are perceived in RERUM as key instruments to expose complex services that use multiple individual RD services. In order to do this step, individuals RD should expose interfaces for sending/receiving/acknowledging commands. Usually a federation execution is controlled by middleware functionality. One relevant aspect of federations would be that near individual RD service calls it might contain additional logic (control logic, math computation) used to deliver an enhanced functionality to applications.

Title	RD to RD communication in federation
Priority	High
Requirement Level	Should
Description	RD should be able to discover and call meaningful (in a predesigned sense of concept) commands on other RD for federation implementation
Rationale	Enable federation implementation with minimal use of a centralized control
Pros and Cons	Pro: Uniform approach. Cons: Ask for predesigned federations, no discovery of RD at run time available
Depends on	None
Competes with	None

Req. 2.4-18 RD to RD communication in federations

2.5 Middleware and Virtualisation Requirements

Middleware (MW) is a SW layer between applications and RDs that supports the provision of several services. The basic functionalities of a standard middleware for IoT applications have been described in various documents in the state of the art, i.e. [99], [61], [12], [34]. Within RERUM, the MW has the following main functionalities:

- MW supports the discovery of RDs and virtual/abstract RDs, it provides registry information to the applications in order to use the services provided by the virtual/abstract RDs
- MW also supports the discovery of Virtual Entities (according to the terminology of IoT-A [55]), providing registry information to the applications, including information regarding their attributes and their associations with the resources (data) of the virtual RDs
- The RERUM MW uses the term of Generic Virtual Object (GVO) to denote the generic abstract class that corresponds to either Virtual RDs or Virtual Entities
- MW is instrumental for creating the virtual RDs and Virtual Entities
- MW has Filtering and Decision making capabilities
- The MW provides a common Interconnectivity layer, both syntactically and semantically
- Monitoring and traceability of middleware
- QoS Policies
- SLA and accounting functionality for RD's services

Virtualisation is one of the fundamental functionalities of the Middleware. The main purpose to enable virtualization in RERUM platform is to create virtual instances of RDs and Physical Entities (PEs) creating their virtual representations of VRDs and VEs respectively. The Middleware will also allow separating or combining the usage of VRDs for providing services associated with VEs and to simplify the use of services and resources provided by RDs by abstracting their characteristics and functionalities [87]. Virtualisation is also related to applications: every VRD should correspond to a known and authenticated RD. Similarly, every VE should correspond to a known PE, which is associated with one or more RDs. Middleware should allow the formation of semantically

interoperable virtual RDs that are able to provide the requested functionality associated to one or more PEs. These more complex VRDs can be formed in hierarchy which allows applications to use more sophisticated services.

The Middleware should be capable of dealing with the management of GVOs. Management of GVOs is also related to the management of their corresponding RDs and PEs and effective creation of groups/federations of RDs for providing specific functionalities related to PEs. In addition, the Middleware should also handle situations when one or multiple VRDs that are part of more complex group of VRDs become unavailable. Management functionality should be able to replace missing functionality with other VRD (if possible) or modify functionality of the group of VRDs (if it can still satisfy application's request) or dismiss the group of VRDs (if the purpose of VRDs is not valid).

The virtualization of services must provide loosely coupled services; this reduces the risk that a change made within one service will create unanticipated changes with other services.

Title	**Virtualisation of RDs and PEs**
Priority	High
Requirement Level	REQUIRED
Description	Middleware should support virtualisation of RDs and PEs. The following partial functionalities are required: • Identification of RDs and creation of Virtual RDs • Creation of more complex, hierarchical, virtual RDs • Identification of PEs (through their associated RDs) and creation of VEs • Application virtualisation • Management of GVOs • Services must be loosely coupled units of functionality that have no calls to each other
Rationale	One of the main purposes of virtualisation is to create logical instances of RDs (creating VRDs), which correspond to known authenticated RDs. Similarly, virtualisation will create logical instances of PEs (creating VEs). These VRDs will provide services to applications. However requested functionality can be more complex than functionality of one simple RD (VRD). In order to provide such a complex functionality middleware has to support virtualisation that will allow separate or combine usage of VRDs and creation of more complex, hierarchical, VRDs. Middleware should be able to manage all of these situations in a very efficient way. VEs will be associated to VRDs through services that expose resources that are related with the PE (that is the physical entity that is abstracted to that VE). VEs will also have services, allowing reading and altering the attributes of the VEs.
Pros and Cons	Pro: Virtualisation enables the homogeneous presentation of devices and physical entities to the applications, concealing their heterogeneity.

	Con: None
Depends on	None
Competes with	None

Req. 2.5-1 Virtualisation of RDs and PEs

The Middleware should support the interconnection between different RDs, virtual/abstract RDs (VRDs) and applications. However, to be able to interconnect them, MW needs to know which RDs and VRDs are available and therefore it has to discover RDs and the respective VRDs first. In a similar approach, when an application needs to access (read or change) some information regarding a Physical Entity, the MW should be able to discover the Virtual Entity that corresponds to that specific Physical Entity, as well as the corresponding VRDs (and the corresponding RDs) that are associated to that Virtual Entity. MW should allow discovering different RDs, VRDs and VEs (using, say, on a registry mechanism RDs, VRDs and VEs) based on various criteria (e.g., requested functionality, location etc.) and make their services publicly available for applications (publish service). In addition, it should ensure that applications will have access only to authenticated registered RDs and VRDs with published services. Furthermore, it should allow only authorised VRDs to access services associated to VEs, taking into account the policies for these services.

These functionalities pose additional requirements in management of life-cycle of RDs and PEs. Management functionality should be responsible for life-cycle of RDs, PEs i.e. only available RDs and PEs are registered and once they become unavailable, the MW will react accordingly.

Title	**Discovery of GVOs and management of the life-cycle of RDs and PEs**
Priority	High
Requirement Level	REQUIRED
Description	Middleware should be able discover new RDs, PEs and GVOs based on various criteria (register these RDs, PEs and GVOs) and make their services publicly available for applications. In addition, middleware should support life-cycle management functionality to handle life-cycles of RDs and PEs (creation, modification or deletion).
Rationale	In order to provide applications with GVO services, RDs, PEs and their corresponding GVOs need to be known. To avoid increasing complexity of applications, this will be done by middleware (applications will not be aware of any discovery process). This functionality poses additional requirements in life-cycle management of RDs and PEs and thus the middleware should handle it.
Pros and Cons	Pro: enables the applications to access the appropriate and authenticated VRDs according to its requirements Con:
Depends on	None
Competes with	None

Req. 2.5-2 Discovery of GVOs and management of the life-cycle of RDs and PEs

Filtering and decision making is another key capability of the Middleware. It allows information filtering and decision making based on the various requirements (e.g., received data from RDs, VRDs

or applications, accessing and altering attributes of VEs), and thus decreasing the processing demands related to decision making or filtering on applications, RDs or VEs side. Filtering of information can be done via i.e. RESTful interfaces and by using Complex Event Processing.

In situations where particular functionality can be provided by more VRDs, the selection taken out of all these VRDs may have a negative impact. For example, consider having two sensors measuring temperature in the “living room" (that is the PE). The first one is older, less accurate with higher energy consumption, while the second one is newer with higher accuracy and less energy consumption. Of course, the usage of the second one will be more efficient. The MW should help the application to choose the most appropriate VRD that can provide access to the attributes of the corresponding VE (the virtual representation of the living room): Each available virtual RD can be characterised by different set of values e.g. utilities, trustworthiness/reputation values and costs. These values are weighted (positive, negative) according to the requested policies of the VE and therefore enable making the decisions in more efficient way.

Presence or absence of previously available RDs and VRDs may have impact on functionalities of existing RDs or VRDs (especially for more complex VRDs) and thus influence decision-making. Middleware should be able to create and manage the context based on various situations (i.e., creation of new VRDs and VEs, discovery of not known RDs and PEs, removal of unused VRDs and VEs) and different methods (e.g., pattern recognition, events received from other RDs etc.).

RERUM platform will facilitate the connection of many applications that will communicate with VRDs and VEs. However, this can lead to situations where different applications would like to communicate with the same VRDs for accessing information related to one VE, but the required functionalities may contradict. For example, imagine if one application requires a window to be opened (because of quality of air condition), and the other requires it to close (because of temperature). This will lead to further questions: How can both applications be satisfied? Will the quality of service be gracefully degraded, or can it be optimized in some sense?

Title	**Filtering and Decision making**
Priority	High
Requirement Level	REQUIRED
Description	MW should filter information and be able to make some decision based on received information or requested information
Rationale	RERUM platform will facilitate the connection of many different applications, VRDs and VEs, which may produce huge amount of information. Filtering and decision-making will enable exchange of necessary information according to received requests. To provide requested functionality in a most effective way (more RDs provide same functionality but with different parameters e.g. trust or cost) or in case of conflict requests, the ability to make decisions is crucial. The following partial functionalities are required: • Filtering information based on various requirements from applications • Supporting an optimized decision making • Supporting the resolution of conflicting requests
Pros and Cons	Pro: allows the provision of the exact information that is needed by the applications Con: None
Depends on	None

Competes with	None

Req. 2.5-3 Filtering and Decision making

As a SW layer between the applications and RDs, the MW needs to interconnect different applications, VRDs and VEs. The MW should provide syntactic and semantic interoperability of RDs and applications. Syntactic interoperability refers to the message and data formats, the protocols, etc., while the semantic interoperability refers to a common understanding of information sent between involved parties (the meaning of the messages exchanged) [86].

Interconnectivity can be implemented using a publisher-subscriber method: the MW publishes the services of the registered VRDs corresponding to registered VEs and the applications should subscribe in order to receive information.

Interconnection functionality poses additional requirements to management. Management of GVOs should handle situations where one RD is related (via a set of constraints) to another. Those RDs can be forming a neighbourhood or a Realm of interest that is related to a specific PE. This is what we call a "**federation of RDs"**, which is a set of RDs and services related to each other based on location, functionalities etc. and focus on providing a specific service related to one VE. Those federations are subject also to dynamic behaviour induced by the mobility of RDs (e.g., mobile sensor for sensing the gas in the air can move inside Area 21 or move to Area 39). Moreover, the management functionality of MW should also handle symbolic hierarchical organisation (e.g., room is part of the building) example: the thermometer of the meeting room of level 3 of building B ("B.3.MRoom.thermo"). Services provided by VRDs depend on RDs, so the functionality of RDs' management should cooperate with management of VRDs (part of virtualization requirement).

It is easy to find solutions for the interconnectivity that have deficits on performance, security, or scalability. In this sense, this requirement competes with requirements on performance, security, and scalability: the interconnectivity of the RDs and applications via the MW should consider those crucial requirements. The solution space to implement this requirement is constrained by them.

Title	**Interconnectivity of VRDs through the MW**
Priority	High
Requirement Level	REQUIRED
Description	The MW should provide syntactic and semantic interoperability of RDs and applications
Rationale	Different types of RDs should be interconnected with different applications which pose requirement for common interface, i.e. RDs have to provide a common interface to applications, which, in turn, may be of different types and using different invocation styles. Moreover middleware needs to support management functionality to manage and control new and/or already created interconnections.
Pros and Cons	Pro: Allows the efficient interconnectivity of heterogeneous types of VRDs over a homogeneous layer Con:
Depends on	None
Competes with	Performance Security Scalability

Req. 2.5-4 Interconnectivity of VRDs through the MW

Monitoring and traceability is important to bring all issues of RDs and their network connectivity into a clearer view. In particular, the capability to monitor middleware functionality by the middleware is important in order to ensure availability of middleware and fast recovery in case of failure; capability to monitor RDs, provided services and applications ensures that those available RDs, services and applications are requirements that directly follow from that. If some of them become unavailable affecting the provided functionalities, the middleware should be aware of it and restore its functionality triggering healing solutions. Monitoring/diagnostic solutions can identify these issues.

Title	Monitoring and traceability by the middleware
Priority	High
Requirement Level	REQUIRED
Description	MW incorporates a solution to monitor system faults, measure its performance and restoring process: • monitor functionality of MW itself • monitor functionality of RDs • monitor functionality of services • monitor functionality of application
Rationale	The middleware has to offer the possibility of monitoring messages received and messages forwarded in order to detect faults or errors and calculate and provide performance measures, and help in restoring process. In addition middleware should be also capable of monitoring functionality of RDs, services provided by these RDs to be able to provide up-to-date information and react on current situation.
Pros and Cons	Pros: Allows self-monitoring of the MW and monitoring of the RDs
Depends on	None
Competes with	None

Req. 2.5-5 Monitoring and traceability by the middleware

In order to ensure overall QoS, the middleware should be able to support and enforce QoS Policies in different areas such as: (i) transmission parameters (delay, failure of transport, discarded PDUs, etc.), (ii) network parameters in terms of bandwidth, congestions, etc., (iii) priority of services, and (iv) QoS related to RDs (ensure that RD can communicate when it needs, prioritise different RDs etc.).

Title	Support and (partial) enforcement of QoS Policies
Priority	High
Requirement Level	REQUIRED
Description	Ensure QoS in case of changes in the architecture, and allow middleware to react maintaining the requested QoS in situations like congestion, node failures, transmission failure, bandwidth control following QoS Policies are required: • QoS policy for transmission (delay, fail to transport, discarded

	PDUs etc.) • QoS policy for network in terms of bandwidth, congestions etc. • QoS policy for services (priority) • QoS policy for RDs (ensure that when RD want to communicate it can, prioritise different RDs etc.)
Rationale	In order to ensure overall QoS, middleware as a part of the RERUM platform should support and enforce QoS policy.
Pros and Cons	Pro: Ensures QoS for the applications; avoids congestions
Depends on	None
Competes with	None

Req. 2.5-6 Support and partial enforcement of QoS Policies

RDs can be owned by different users and possibly a fee can be charged providing premium services. Example: a private company moves their newest mobile gas-sensing device into Area 39; the new sensor is able to provide more accurate results than the current gas sensors in Area 39. However, this company may then decide apply some fee for this service.

In a similar concept, a RD may consume high energy and the owner may charge for providing access to it. Example: there is only one sensor for temperature that is in the building B in a meeting room. This sensor is old and has a high energy consumption. If an application needs to know the temperature in that meeting room, the owner of the sensor may request a fee, because there is an increased maintenance cost due to replacing the battery more often.

Correspondingly, some VE services may have policies requiring payment for accessing (reading/altering) their attributes. For example, consider a VE that represents a turnstile at a cinema. For passing the turnstile, the user (i.e. automatically through his phone) may have to pay the ticket price that will allow him to enter. In this respect, the VRD (phone) will pay for accessing the VE (turnstile) and altering its attribute (open/closed) from closed to open.

These payment/fees can be done in different ways, for example, pre-paid credit or daily/weekly/monthly/yearly fee. The Middleware should be able to support such as accounting and SLA functionality.

The opportunity to receive payments from utilization of RDs (VRDs) or accessing VEs and the respective income may have positive impact on the modernization of the RDs. This will also provide incentives for installing RDs in many places/areas for receiving income.

Title	Support of accounting and SLAs
Priority	High
Requirement Level	REQUIRED
Description	MW should allow providing premium services for a fee for accessing particular RDs/PEs.
Rationale	Some RDs (or PEs) can be owned by different users and they can require a fee to provide GVO services. This can be done in different ways, for example, pre-paid credit or daily/weekly/monthly/yearly fee. Middleware should support this functionality.
Pros and Cons	Pros: Provides incentives for installing RDs in many areas.
Depends on	None
Competes with	Performance (Latency Messaging can be increased)

Req. 2.5-7 Support of accounting and SLAs

2.6 Security and Privacy Requirements

This section builds upon the results of RERUM Deliverable D2.1 and extracts the requirements for enhancing the security and privacy of the IoT in Smart City application environments. These requirements are split in six main categories:

(i) Energy-efficient cryptographic primitives and infrastructure: This subsection analyses efficient cryptographic solutions, given the limited resources of many of RDs.

(ii) *Integrity, confidentiality and authentication requirements*: Since a large number of devices are expected to be located beyond the conventional boundaries of regulated entities, manufacturers will hardly be able to close security vulnerabilities with updates after the fact. This could allow access to the devices of an immense amount of users, and to the devices of a utility, if integrity requirements are not given enough attention. This is intelligible from the European Directive 2004/22/EC [30] on measuring instruments in Annex I (8.4) which states that: “measurement data, software that is critical for measurement characteristics and metro-logically important parameters stored or transmitted shall be adequately protected against accidental or intentional corruption”. This subsection analyses requirements derived from the threat analysis in RERUM Deliverable D2.1, particularly those related to the CIA-triad and AAA.

(iii) Privacy Requirements: As RERUM’s focus is the protection of user information; it must identify which data and metadata at different layers electronically capture this information. Then the data flows must be judged according to the layer(s) they are being discussed at. We assume that a data flow is always between end-points on the same layer. It might even happen that information is not flowing from one RERUM device to another, but is exchanged between different modules within a RERUM device. In another case, even the software that enables an application to run on a device can be considered as the information that needs protection.

We see a need to protect the following data:

(a) Personal data: This is data associated with the environment or the actions of a human user that inhabits or visits the “smart” environment. This covers data like the temperature of the living room (S-DATA) or the command sent to the lights (A-DATA) generated when the user triggered this by some remote control appliance or by walking into a space that was monitored for movements by a presence detector that issues the command. Such data could leak information about the humans inhabiting the monitored and controlled environment. Hence, this data needs sufficient care-taking as governed by privacy regulations and the privacy-by-design paradigm. Personal Data following

[RFC6937][50] is any information relating to an individual who can be identified, directly or indirectly. Such data must be protected according to data protection laws.

Note that this information might already be deferred by traffic analysis, i.e. by just observing traffic flows (presence, absence, amount, direction, timing, packet size, packet composition, and/or frequency), even if flows are encrypted [RFC4949][49], [RFC6937][50].

(b) Application data. This is data considered sensitive from the application's point of view. While the data that is leaked might not be personal data, its confidentiality needs protection.

An example of sensitive application data are trade secrets that are communicated, e.g. data that must be kept secret to keep a competitive advantage, e.g. stay in business. For example, a company using an application might need to protect details of internal workings or the names of customers and subcontractors. Other examples could be algorithms, encryption keys, serial numbers, or licence keys.

(iv) Bootstrapping requirements: Devices without any knowledge about the environment and their neighbouring nodes are imposed to several attacks. This section analyses the vital requirement of minimizing security attacks during the initial configuration of RERUM devices.

(v) Requirements for the secure provision of software and configuration: This section analyses the requirements for the PRRS framework, which provides dynamic security configuration of the RERUM ecosystem.

A PRRS (Platform for Runtime Reconfiguration of Security) [43] is a tool that allows finding security components according to given criteria (normally taken from a configuration store), then downloads them and prepares them for a later execution. In the concrete case of RERUM, this will usually also imply building the binary executable and deploying it in the RD. As this is strongly tied with another functionality of the project, which is the 'over the air programming', this section states only those extra functionalities required solely for a PRRS.

(vi) Miscellaneous security requirements: RERUM closely follows related work and best practices to enhance and achieve security and privacy-by-design. This section analyses related and complementary requirements, and especially the requirements related to trust mechanisms.

2.6.1 Energy-efficient cryptographic primitives

Many of the RERUM's security-and privacy-enhancing mechanisms will rely on techniques such as data encryption, hashing, and signing of data. Due to the high complexity of implementing those algorithms in software, contemporary wireless sensor platforms often have built-in cryptographic hardware acceleration. When present, those co-processors can significantly speed up crypto computations, reduce code size and increase energy efficiency. However, co-processors are tied to specific algorithms (e.g. 128-bit Advanced Encryption Standard (AES)) and they are impossible to upgrade [81], [80].

At this stage it is difficult to predict the exact cryptographic algorithms that will be used for RERUM, although AES and Elliptic Curve Cryptography (ECC) seem to be already enjoying some market penetration [81], [80], [71]. During RERUM, we will consider the option of adopting hardware acceleration and we shall evaluate the pros and cons, but it is unrealistic to require this feature for all possible alternatives. Therefore, RDs are required to provide support for cryptographic algorithms implemented in software in an energy efficient fashion and with code size and memory requirements small enough to be deployable in embedded devices. IETF's ACE [47] and DICE [48] Working Groups are currently investigating suitable algorithms and mechanisms that would fall in this category. Future upgrades of cryptographic algorithm implementations to newer versions should also be possible.

Title	Energy-efficient cryptographic primitives
Priority	High
Requirement Level	REQUIRED
Description	Energy efficient implementation of cryptographic algorithms for RDs
Rationale	Fundamental building block for many of RERUM's security- and privacy-enhancing mechanisms, such as machine-to-machine authentication, secure bootstrapping, protection of the integrity and confidentiality of data in transit as well as at rest.
Pros and Cons	Software implementation of cryptographic primitives may increase energy consumption and will have an adverse impact on RD lifetime for battery-powered RDs. Hardware implementations are faster and operate with a smaller code / RAM footprint, but are tied to specific algorithms (e.g. AES) and are impossible to upgrade.
Depends on	None
Competes with	None

Req. 2.6-1 Energy-efficient cryptographic primitives

2.6.2 Integrity, Confidentiality and Authentication Requirements

Recall from RERUM Deliverable D2.1 that there are six security properties: confidentiality, integrity, availability, authentication, authorisation, and accounting; corresponding to these six properties in the STRIDE methodology: Spoofing, Tampering (integrity threat), Repudiation, Information disclosure (confidentiality threat), Denial of Service (DoS, availability threat), and Elevation of privilege.

The first two security properties, confidentiality and integrity, apply directly to data. We will say that a piece of data that requires confidentiality has a security level "**SL-C**", data that requires integrity has "**SL-I**" and data that requires both has "**SL-CI**". We see SL-CI as *implying* both SL-C and SL-I.

Instead of saying "data with security level SL-I", we also say "data with SL-I", or more shortly "SL-I data" and similarly for SL-C and the other security levels to be defined.

Clearly, data with any of these security levels should be suitably protected, both at rest and during communication. The protection may be based on cryptography (encryption, message authentication codes, signatures, etc.) or they could be based on the detection of manipulation/errors due to redundancy/plausibility checks, or on monitoring/self-healing procedures.

There is a stronger form of confidentiality security level that we will denote by "**SL-P**" (P stands here for privacy), referring to personal data, which requires privacy protection. This will be discussed later in Section 2.6.3 level SL-C, data with the privacy level SL-P can be combined with SL-I: SL-PI is the data that requires both privacy protection, but also must be integrity protected.

So let us consider first Integrity, then Confidentiality, and finally Authentication in this section; we will consider at Privacy separately in the next section.

2.6.2.1 Integrity Protection Requirements

Whenever data is exchanged, the transmission involves sending the data over a medium (wireless or a wire), which is prone to errors or manipulation, and it often involves "network hops", e.g. other

nodes, routers, or systems, which could change the data. Usually an application requires preserving the integrity of the message exchanged.

> "Data integrity is the property that data has not been altered or destroyed in an unauthorized manner" [ISO_7498-2][52].

We must leave it to a per use-case specific vulnerability analysis to identify which data requires protection in certain applications and use cases. In RERUM Deliverable 2.1 vulnerabilities have been identified for four different use-cases. One example is a genuinely issued command sent to an actuator: It should not be modified in an unforeseen and unexpected way on its path to the intended RD, because otherwise the RD might receive an unauthorised and potentially maliciously modified command instead of the intended command and hence, the actuator might not act as intended.

The integrity protection, that RERUM requires, must allow the "detection of integrity compromises" [96], as opposed to mechanisms concerned with "prevention" [ISO_10181-6][51] of integrity breaches. The focus will be on protection against the following three violations:

> "a) unauthorized data modification;
>
> [...]
>
> c) unauthorized data deletion;
>
> d) unauthorized data insertion;" [ISO_10181-6][51]

When describing integrity in this section we assume a workflow as follows: An entity first applies some integrity protection to data by running an algorithm that creates additional data called the integrity checksum. Another entity, called the verifier, is then able to verify the checksum together with the integrity-protected data to identify if the integrity of that data is still valid or if the data has been subject to unauthorized alteration or if the data has been destroyed.

To sum up, in RERUM data integrity means that the verifier can detect that data has not been modified (alteration or insertion) or destroyed in an unauthorised manner since the integrity protection was applied. Seen from the attacker's point of view, an attacker successfully breaks data integrity when:

- the attacker modified (alteration, deletion, insertion) data in a way that was not authorised by the entity that applied the integrity protection and the verifier

and

- the verifier is not able to detect this unauthorised modification.

As discussed before, all kinds of data, depending on the use case might require integrity protection when sent over a network. With the term "data in transit" we describe the data that are being communicated over the network or reside in temporary memory for communication or processing purposes.

Title	Integrity protection of SL-I data in transit
Priority	High
Requirement Level	REQUIRED
Description	Data transmitted by RDs to RDs or the Gateway shall be protected against unauthorised undetected manipulation. This requirement only applies to data identified as SL-I, e.g., when the use-case specific vulnerability analysis identified a significant influence on the use case if an unauthorised modification would remain undetected.

Rationale	This is a mitigation against integrity related threats described in D2.1. For instance, sensor data might be manipulated by unauthorised parties during transmission. This could lead to false traffic estimation or false environmental indicators.
Pros and Cons	Pro: Allows identifying breaches of integrity, which can be used for assessing the trustworthiness. Cons: Additional overhead for generating, verifying and transporting the integrity checksum.
Depends on	none
Competes with	none

Req. 2.6-2 Integrity protection of SL-I data in transit

With the term “data at rest” we describe the data that reside on a RD storage (volatile or persistent). Data at rest can be archival or reference data that are changed rarely or never; data at rest can also be data that are subject to regular but not constant change. Examples include historical logs of previously sensed values and configuration files.

To continue the previous example, configurations stored into a smart object should be integrity protected (additionally the authenticity of the sender of that configuration needs to be verified to be a trusted source for configuration) before the smart object accepts the configuration. By marking configurations as SL-I this measure prevents an attacker to modify a configuration in the smart object without being detected. Also any updates of SW components, configuration data, files, credentials, etc. deployed in RERUM must be as originally intended, and not undetectably modified in any unauthorised way. If a certain version of a file or package is sent to a node for deployment, this package, should be integral (complete and unmodified) when it is validated by the RD for installation.

Title	**Integrity protection of SL-I data at rest**
Priority	High
Requirement Level	REQUIRED
Description	Data stored in RERUM devices shall be protected against unauthorised manipulation. This requirement only applies to data, which may have a significant influence on the functionality of the services or applications.
Rationale	Sensor data stored in RERUM devices might be manipulated by unauthorised parties. This could lead to false traffic estimation or false environmental indicators, and hence later to false decisions.
Pros and Cons	Pro: Allows to identify breaches of integrity and that will allow to use this for accessing the trustworthiness. Cons: Additional overhead for generating, verifying and transporting the integrity checksum.
Depends on	none

Competes with	none

Req. 2.6-3 Integrity protection of SL-I data at rest

RERUM will ensure data integrity even in the cases of the Privacy mechanisms that can modify some data for privacy reasons. Hence, RERUM strives to enable the originator to protect the data's authenticity and define a level of integrity protection suitable for the data at hand. Not all data needs the same level of integrity protection; especially personal data that will need to be changed by Privacy Enhancing Technologies (PETs) must be selectively treated different w.r.t. integrity protection. To give an example of the different levels of Integrity protection there is the one end of the protection that is very strict: Integrity protection to the maximum possible technical extend means that a change of a single bit is a violation. The other extreme is that you do not put data under the umbrella of integrity protection, which leaves them unprotected, that is no modification of anyone is detected. To allow a PET to apply changes, RERUM will need to introduce some weaker form of integrity protection. For example we plan to allow an authorised entity, e.g., an RD or Gateway running some PET, to change integrity protected data. Hence, RERUM needs to provide adjustable integrity mechanisms; adjustable not only in their scope or their strength of the integrity protection. With a weaker integrity protection level, RERUM can support the following process: a RD or Gateway can apply an integrity protection, but allow that if needed at a later stage, a different RD or Gateway can change certain values, e.g. to tackle now arising privacy concerns. So the applying entity can authorise a PET or a RD. The entity delegates the right to change certain data. Only if you received the right to modify by a delegation of the entity that applied the integrity protection, the PET or the entity running the PET can adjust the data. The goal is to retain some guarantees close to or a subset of the original integrity and authenticity protection.

Title	**Authorised modification of integrity protected data**
Priority	Medium
Requirement Level	RECOMMENDED
Description	Data stored and transported in RERUM devices is protected against unauthorised and undetected modification, e.g. has some form of integrity protection. However, sometimes entities other than the entity that initially applied the integrity protection, shall be able to manipulate the data in an authorised way, e.g. the gateways might aggregate (meaning data fusion) integrity protected data by RDs. To fulfil this requirement the integrity protection mechanisms must allow defining certain changes as authorised and not flag those changes as Integrity violations during verification. This kind of integrity protection shall only be applied to data that may need to be changed later.
Rationale	Sensor data stored in RERUM devices might be manipulated by authorised parties for various purposes. This authorised and tolerable change might not lead to a total loss of integrity if authorised violations are excluded. However, each tolerable violation weakens the integrity protection. Hence, the system must control who can make such modifications.

	If the verification of the integrity checksum succeeds with a positive answer, this is a cryptographic proof, verifiable also by third parties that the integrity-protected information was never subject to unauthorised modifications since the integrity protection was applied. If the integrity mechanism includes the authentication of origin, a delegated authorised modification shall not change the verifiability of the origin of the entity that applied the integrity protection first. It is up to the application to allow such an authorised modification to be detected by the verifier to enable it to influence trust and reputation calculations.
Pros and Cons	Pro: Allows delegating the right to do certain modifications to dedicated entities that carry out these so-called 'authorised modifications'; these are especially privacy preserving modifications. This means that an authorised PET will for example be able to delete names and insert pseudonyms; however, by doing this, it will not invalidate the integrity protection. Cons: More complex integrity protection and verification algorithm requires additional computation or additional space for the integrity 'checksum'.
Depends on	Integrity Protection Mechanisms mentioned in Req. 2.6-2 and 2.6-3
Competes with	none

Req. 2.6-4 Authorised modification of integrity protected data

If RERUM allows authorised modifications to happen without invalidating the integrity protection checksum, this has consequences for the integrity verification mechanism. Integrity verification classically delivers a binary result: valid or invalid.

By the previous requirement, data stored and transported in RERUM devices could be protected against unauthorised modification while tolerating authorised modifications, i.e., to allow for some entity (or entities) -other than the entity that initially applied the integrity protection- to manipulate the data in an authorised way.

Hence, when verifying the integrity of an authorised modified message, the result will indeed be "valid". This is the same result as an unchanged message. Thus, an authorised modification is undetected. However, the verifying entity might also need to detect such an authorised modification, i.e., it can be differentiated from unmodified data.

RERUM therefore foresees various levels of integrity detection: further, to the binary detection if an authorised modification has happened or not, various levels of detection can be required, e.g. knowing details like which part of the data is modified, the potential range of the change, the applied function (e.g. sum or average) of the integrity violation. This also includes the question if the detection of an authorised integrity violation would allow identifying the modifying entity, such that this entity can be held accountable for the changed value.

When verifying the Integrity of data, the verifying entity can use these details as a basis for its later assessment for decisions like trustworthiness and reliability of the data. Towards the reputation, an unchanged data means that it is reliably and will give a good reputation due to Integrity.

However, the ability to differentiate between some data that has not been modified, i.e. neither modified in an authorised way nor in an unauthorised way, and data that has been modified in an authorised way might infringe privacy. If the verifier can differentiate, than when an authorised PET changes the data, even if it would write the same values into the data, the fact that an authorised PET has potentially modified the data is not hidden. This allows the verifier to differentiate between potentially modified and never modified data. This can be simply solved in two ways, if the privacy policy would require to hide this fact, but the used integrity protection mechanism would allow detection of the three states, than the PET would just always "mark the data as authorised modification", even if it just replaces the previous original value with a new value that is equal to the original. Or if this privacy policy is known during time of integrity checksum creation, the mechanisms could be chosen such that the integrity protection checksum created does not allow such detection. In general, this "always change" approach will introduce additional costs but allows to still overcome a potential leakage of information by the integrity protection mechanism that allows this detection. In other words: to hide the fact learned by detection that the message was unchanged, the message must always be changed artificially, e.g. changed without changing the value. If for privacy it is always necessary to appear as potentially changed in an authorised way, it might be actually better to use an integrity protection scheme where authorised modifications cannot be differentiated from unchanged versions, e.g. not fulfil this requirement of "Detect an authorised integrity violation".

Title	**Detection of authorised modification of integrity protected data**
Priority	Medium
Requirement Level	RECOMMENDED
Description	Data stored and transported in RERUM devices is protected against unauthorised and undetected modification, e.g. has some form of integrity protection. However, sometimes entities other than the entity that initially applied the integrity protection may be allowed to manipulate or change the data in an authorised way. This requirement only applies to data that was integrity protected by a method that fulfils the "Authorised modification of integrity protected data" requirement and allows the verifier to detect whether or not an authorised change of the integrity protected data has taken place.
Rationale	Data stored in RERUM devices might be manipulated by authorised parties for various purposes. This authorised and tolerable change might not lead to a total loss of integrity as only pre-defined authorised violations are excluded from the forgeries. However, each tolerable violation weakens the integrity protection. Furthermore, the verifying entity might need to identify the entity (or group of entities) that was authorised to potentially modify the data. If this requirement is fulfilled, by verifying the Integrity of data, the verifying entity will be enabled to identify which strength of integrity protection was achieved, in order to know which violations might have been authorised and whether or not they could be detected. Further depending on the detectability, it will be able to detect what kind of authorised change has happened. Depending on the strength, later decisions like trustworthiness and reliability of the data can be assessed.

Pros and Cons	Pro: Allows fine-grained decisions of trust based on the authorised violations of the integrity protection that are present in the data Cons: Detecting that data has not potentially been modified may induce additional overhead when this information must be hidden for privacy protection.
Depends on	Integrity Protection Mechanisms mentioned in Req. 2.6-2 and 2.6-3 and must support subsequent authorised modifications from Req. 2.6-4
Competes with	none

Req. 2.6-5 Detection of authorised modification of integrity protected data

2.6.2.2 Confidentiality Protection Requirements

Data sensitive from an application point of view will be considered as SL-C. This means that unauthorised third parties MUST NOT be able to read these data. While the leaked data might not be personal, their confidentiality needs protection, for example, to stay in business, a company has to protect its sensitive data, i.e. internal workings, names of customers, and subcontractors. So in other words, data that is of relevance for the correct working of an application and if not protected would allow to influence, understand, or mimic the application without the need for the original, genuine involved RDs.

For ensuring the confidentiality of SL-C data, RERUM needs to implement techniques and mechanisms for the secure storage and transmission of those data.

Title	**Confidentiality protection of SL-C data at rest**
Priority	High
Requirement Level	REQUIRED
Description	Data stored in RERUM devices shall be protected against unauthorised disclosure if they include confidential data (e.g. sensor data from crowdsourcing smartphones or from energy management systems).
Rationale	There might be several reasons why data may be confidential. For example, personal data underlies data protection laws and shall be hence protected against unauthorised disclosure.
Pros and Cons	Pro: Allows prohibiting breaches of confidentiality. Cons: Potential additional overhead in communication, e.g., for handshakes and key agreement. Additional overhead in computation.
Depends on	none
Competes with	none

Req. 2.6-6 Confidentiality protection of SL-C data at rest

Again we need to separate the data at rest, e.g. residing on some storage to be stored inside the RD for later retrieval, from "data in transit" which are being communicated over the network or reside in temporary memory for communication or processing purposes. Data in transit is communicated over wired or wireless networks and such the requirement to protect the confidentiality is of high importance for RERUM.

Title	**Confidentiality protection of SL-C data in transit**
Priority	High
Requirement Level	REQUIRED
Description	Data transmitted by RDs to RDs or the Gateway shall be protected against unauthorised disclosure if they include confidential data (e.g. sensor data from crowdsourcing smartphones or from energy management systems).
Rationale	There might be several reasons why data may be confidential. For example, personal data underlies data protection laws and shall be hence protected against unauthorised disclosure.
Pros and Cons	Pro: Allows prohibiting breaches of confidentiality. Cons: Potential additional overhead in communication, e.g., for handshakes and key agreement. Additional overhead in computation
Depends on	none
Competes with	none

Req. 2.6-7 Confidentiality protection of personal data in transit

2.6.2.3 Authentication Requirements

In RFC 4949 [49] authentication is defined as: The process of verifying a claim that a system entity or system resource has a certain attribute value.

The RFC allows Authentication to be further divided into two categories:

1. Simple Authentication: An authentication process based on a password as the information needed to verify an identity claimed for an entity.
2. Strong Authentication: An authentication process based on a cryptographic security mechanism – particularly public-key certificates – to verify the identity claimed for a system entity

Within the limits of practical and energy-efficient techniques, RERUM aims to provide mechanisms that achieve the highest security level, preferably the one of strong authentication, to allow privacy and security-critical decisions to be based on a strong foundation.

In the definition of strong authentication we used the term "system entity". For RERUM the term "system entity" in general refers to the definition given in RFC 4949 [49]: An active part of a system - a person, a set of persons (e.g., some kind of organization), an automated process, or a set of processes (or subsystem) - that has a specific set of capabilities.

The mentioned capabilities mean that the entity can provide some verifiable evidence (i.e., establish the truth) of an attribute value claimed by or for this system entity.

This is the “authentication information” (again the following definition is taken from RFC 4949 [49]):

Authentication information is information used to verify an identity claimed by or for an entity. Authentication information may exist as, or be derived from, one of the following: (a) something the entity knows (e.g. password); (b) something the entity possesses (e.g.: token); (c) something the entity is.

Example: In a public key based authentication, a system entity that claims to be the one whose identifier is listed as the subject of a public-key certificate generates a verifiable proof that it is using, or is permitted and able to use, the matching private key. Using the private key to produce a digital signature by signing a challenge posed by the verifier can be a simplified example on how to achieve this. The verifier must either trust the public key of the signing entity or be able to trace the signing entities certificate via a chain of certificates towards a certificate signed with a public key the verifying entity trusts (trust-anchor).

RERUM distinguishes two different types of entities for the authentication: devices and users.

Title	Device authentication
Priority	High
Requirement Level	REQUIRED
Description	The RERUM devices shall only accept data coming from authenticated devices. This requirement applies for the communication between RDs, gateways and application servers. Gateways should be able to authenticate to the servers or access points outside of the RERUM environment with which they communicate, and vice-versa.
Rationale	This requirement derives from threat #14 of RERUM Deliverable D2.1, regarding “Device Identity spoofing”, where an attacker claims to be a genuine device. Without authentication, attackers might pose as legitimate devices and try to intercept transmitted data, which they are not authorised to receive.
Pros and Cons	Pro: Allows authenticating the entity. Cons: Potential additional overhead in communication, e.g., for handshakes. Additional overhead in computation.
Depends on	Having locally stored, or having a way to obtain, trusted authentication information for the destination device or being able to provide the destination entity with the authentication information it requests from the originating device.
Competes with	none

Req. 2.6-8 Device authentication

In the RERUM authentication process, a user is a system entity that consumes a product or service provided by the system, or that accesses and employs system resources to create a product or

service of the system (previous is taken from RFC 4949 [49]). This term refers to a living human being acting either personally or in an organisational role. However, the term also may refer to an automated process that runs on behalf of a living human being in the form of hardware, software, or firmware; to a set of such living human beings; or to a set of such processes.

Title	User authentication
Priority	High
Requirement Level	REQUIRED
Description	Users shall authenticate themselves for remote access to RERUM applications and systems. This includes: – authentication of administrators for maintenance purposes of RERUM systems – authentication of end users for the remote monitoring and control of devices and appliances in energy management scenarios
Rationale	This requirement derives from threat #13 of RERUM Deliverable D2.1, regarding "Identity Spoofing of a User with Higher Privileges". Access must be forbidden to unauthorised parties, so RERUM needs to be able to authenticate the User requesting the access.
Pros and Cons	Pro: Allows authenticating the entity. Cons: Potential additional overhead in communication, e.g., for handshakes. Additional overhead in computation.
Depends on	Having locally stored, or having a way to obtain, trusted authentication information for the end user that an entity wants to authenticate or being able to provide the other entity with the authentication information it requests from the user.
Competes with	none

Req. 2.6-9 User authentication

2.6.2.4 Access Control Requirements

One of the most central pillars of security is access control. Access control allows deciding if a subject (that is a user, an application, a process, an external partner, or any type of entity) should be allowed to access a resource. Examples for an access to a resource are: reading certain information, writing a message, or sending a command to an actuator. Intuitively, the easiest way to solve the problem is to codify who is allowed to do what on which resource in a matrix. Such matrices would hence directly relate subjects to resources and the type of access that they are allowed to execute. However, this procedure is error-prone, it lacks flexibility, and it does not scale up. The current state-of the-art is to use declarative policies, written in a high-level language that express the conditions under which the subjects can carry out a certain type of action on a certain type of resources.

The main idea, abstractly, behind attribute-based access control is to:

(a) associate subjects to attributes (i.e. roles, jurisdictions, groups, etc.),

(b) associate resources and types of actions on those resources to attributes (type of resource, location, domain of administration, ranges for allowed parameters in the invocation call, etc.) and
(c) write a set of policies that relate the attributes of the subjects to the attributes of the resources that they are allowed to access.

The environment itself may also have attributes, like the date and time, the state of a particular application or channel etc.

There are different languages, with different expressivity and complexity, to represent and evaluate those policies. One particular case is role-based access control, where roles are assigned to users and associated to permissions of actions on resources.

Several details should be mentioned in this context:

1. The system must provide rather strictly secure means to administer the access criteria used for authorisation. In general, this is not a trivial problem because it is difficult to separate the concerns of the different stakeholders, domains, and administrators.
2. An important privacy requirement is expressed as the so-called "need-to-know" principle: an entity should be able to access personal information only if this is strictly necessary for its purposes, and those purposes are part of the service provisioning that the data subject (that is, to whom the personal data relates to) has consented to. This is a requirement for access control, indeed, but usually it is not solved during the execution of the authorization procedure itself (the policy evaluator does not try to understand if the subject needs to know the data for what purpose). This is usually solved by privacy-by-design that should guarantee that the subject's attributes reflect his "need-to-know". (See Section 2.6.3).
3. The authorisation check should be timely enough and should only allow a limited set of activities. It is possible that the permissions of a given user / entity change during the execution of the system, either due to configuration updates or to changes in the user itself.
4. Since the access control functionality has to evaluate the attributes of the subjects and those attributes are in general dynamic (people change roles, permissions are delegated, permissions are withdrawn, etc.) the access control mechanism must be able to verify that the attributes of the subject are correct.

Title	Attribute-based access control
Priority	Medium
Requirement Level	RECOMMENDED
Description	The system should be able to make authorisation decisions based on any arbitrary, but predefined set of attributes (including entity attributes) related to the request
Rationale	This requirement derives from threats #15 (User Privilege Elevation) and #16 (Device Privilege Elevation) of RERUM Deliverable D2.1. Some authorisation decisions might require additional information from the entity. For instance, the public transport scenario might need to check that the user does not only belong to the role 'citizen', but has a valid driver license to make more difficult for him to take a car improperly (driver licenses are subject to be invalidated).
Pros and Cons	Pros: This cannot simply be managed by splitting the roles in as many

	combinations of roles as needed, because the number of combinations would increase exponentially with the number of attributes needed to make the decision. Another important positive aspect of attribute-based access control is that if a new attribute needs to be added to the existing criteria it would not be necessary to split the existing roles again to support the new attribute. Such an operation could not even be possible in certain scenarios due to the large overhead of updating the roles of the whole set of users at real-time. Hence, this authorisation mechanism, allows taking decisions based on mostly all possible relevant information, and scales well enough to support large and growing system architectures. Cons: There is a privacy issue with taking decisions based on user attributes, because they need to be accessed first for taking that decision. Besides, depending on the way this information is accessed, it could lead to obtain some user information that is not needed for the decision. For this reason, and to avoid colliding with the "Privacy by design" concept, user attributes should not be transmitted in the request, but be requested by the service only when it needs them.
Depends on	Authentication
Competes with	none

Req. 2.6-10 Attribute-based access control

2.6.3 Privacy Requirements

As it was discussed in RERUM Deliverable D2.1, there are many privacy issues related to the use of personal data[2]. Personal data in the use cases could be related to the physical location of a user, his consumption of energy, car's location, the ambient conditions in his living room, etc.

Recall from subsection 2.6.2 the introduced security levels SL-C, SL-I, and SL-CI for data that must be confidentiality protected, integrity protected, and both, respectively. There is a stronger form of confidentiality security level, that we will denote by "**SL-P**" (P stands here for privacy), referring to personal data, which requires privacy protection. Of course, data with SL-P have also SL-C: this simply says that personal data have to be protected against disclosure. Nevertheless, personal data have more requirements, for instance (under normal conditions) it must be deleted after a pre-defined amount of time. This security level SL-P can be combined with SL-I: SL-PI covers data that require both privacy and integrity protection.

The figure represents the *lattice* of security levels. The "bottom" term in the lattice, "*true*", is the less restrictive: all data have this property. The highest is the most restrictive: data in this category are personal data, subject to privacy rules, and also must be integrity protected. The hierarchy is based on implication: a property (or security level) is above another one in the lattice if the first one implies the second one. Thus SL-P implies SL-C because personal data must also be kept confidential.

[2] In the US, the term "personally identifiable information" is used, and the abbreviation PII is very widespread.

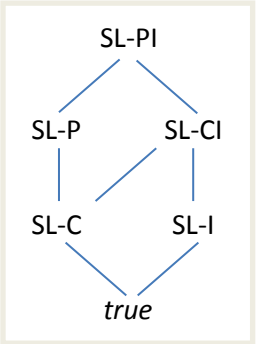


Figure 5 - **The lattice of security levels**

There are two peculiarities about this lattice:

1) If data $\mathbf{d_1}$ of a certain security level $\mathbf{SL_1}$ are combined with data $\mathbf{d_2}$ of a security level $\mathbf{SL_2}$ to form data $\mathbf{d} = F(\mathbf{d_1}, \mathbf{d_2})$, then these data have, by default, the security level $\mathbf{SL} = lub(\mathbf{SL_1}, \mathbf{SL_2})$, where lub is the "lowest upper bound". In other words, **SL** is the least restricting level that is more restrictive than both $\mathbf{SL_1}$, $\mathbf{SL_2}$; thus, information combined with other data, tends to "go up in the lattice".
2) There are exceptions to this rule:
   a) **Aggregation**: if personal data (SL-P or higher) of different users are combined to create statistical data (say, averages over relatively large number of people) the aggregated information *may be* not be SL-P. There is no clear cut rule for this, but a detailed analysis is necessary. The question that must be considered is: if the aggregated information is provided to an external recipient, how much does this recipient learn about a particular subject upon reading the aggregated information? In general, this is not only true for SL-P but also for SL-C: combination/aggregation/abstraction of confidential information *may be* not confidential. But the situation here is much simpler: the "data owner" of the combination of confidential (but not private) data can decide what the applicable confidentiality policy is.
   b) **Aging**. Confidential data which is not personal *may become* non-confidential after some time. Again, this is a policy regarding the data owner.
   c) **Copying**. If data with SL-I, which are to be integrity protected, are copied, and the copy is not used for critical purposes, this copy *in general* is not SL-I anymore.

These privacy definitions will help us to verify appropriate mechanisms for addressing privacy protection, derived from two essential European directives.

First, the European Directive 95/46/EC [31] on the protection of individuals with regard to the processing of personal data, and on the free movement of such data (referred to below as "the European Directive"), aims to protect the rights and freedoms of persons with respect to the processing of personal data by laying down guidelines determining when this processing is lawful. The guidelines are stated in articles 6 to 26.

Second, the European Directive 2002/58/EC [30] concerning the processing of personal data and the protection of privacy in the electronic communications sector Directive on privacy and electronic communications, which defines its guidelines in articles 1 to 21.

As described in RERUM Deliverable D2.1, all RERUM use cases process personal data, thus making necessary to follow the guidelines stated in the directives. These guidelines can be subsumed in eight specific, verifiable requirements. We will adopt the terminology proposed in ISO 29100 [8] to name these requirements.

When personal data are collected, generated, stored or processed otherwise, a preceding common consent between the data subject and the processing party is needed:

Title	**Consent and choice**
Priority	High
Requirement Level	REQUIRED
Description	The requirement of user consent can be exemplified with several of RERUM's use cases. Before downloading / installing the RERUM app for smart transportation on smartphones of crowdsourcing volunteers, the smartphone owner shall give his consent before sensor data (e.g. motion, environmental, and position data) are collected for real-time traffic estimation purposes. (Alternatively, this could be achieved by a paper consent form, but it would be rather unusual for app users.) In the home energy management use case, the energy supplier must ask the user for his user consent regarding metering data. The same applies for third party service providers in the comfort quality use case, which offer value added services based on U-DATA. The European Directive 2002/58/EC [30] defines user consent and choice in article 6 (3) as *"For […]the provision of value added services, the provider of a publicly available electronic communications service may process […] [personal data] to the extent and for the duration necessary for such services or marketing, if the subscriber or user to whom the data relate has given his/her consent. Users or subscribers shall be given the possibility to withdraw their consent for the processing of traffic data at any time."*
Rationale	According to directives 2002/58/EC (see above) and 95/46/EC [31] (art. 6 1(b)), personal data shall be collected and processed only for the specified, explicit and legitimate purposes. This implies that a data subject and the processing party have to agree on a common consent to process personal data that can disclose the subject's physical, physiological, mental, economic, cultural or social identity.
Pros and Cons	Pros: Application gets authorisation to access private user data, ensuring compliance with EU directives. Cons: If users do not give consent then the app will not produce any results.
Depends on	Nothing
Competes with	Nothing

Req. 2.6-11 User Consent and choice

A common consent can only be reached if both parties understand which data are going to be processed, for which purpose and in which way. This means that the collection, generation or storage of personal data has to be specific in the sense of how the data will be processed; it perquisites the

definition on a purpose to describe why it is need and this purpose needs to hold at all times, whenever the personal data is processed:

Title	**Purpose legitimacy and specification**
Priority	High
Requirement Level	REQUIRED
Description	Purpose legitimacy and specification is best described in the European Directive 95/46/EC [31] (art. 6 1(b)) as follows: "[personal data collection must be] *for specified, explicit and legitimate purposes and not further processed in a way incompatible with those purposes."* For example, purpose legitimacy and specification is needed per default on the RERUM app for smart transportation. The collected data shall be developed uniquely for traffic analysis. Individual evaluation or marketing affiliation of the data has to be ruled out by design.
Rationale	According to the European Directive personal data shall be collected and processed only for the specified, explicit and legitimate purposes (see above). The demand for purpose legitimacy ensures that data is only processed, if needed for a specific purpose. It helps system designers to consider how their processes are going to work and what (private) information they need.
Pros and Cons	Considering purpose legitimacy in the initial design of a system also benefits several other requirements: - Collection limitation is reduced if a system is designed to collect only the information it needs for specific processing. - Data minimisation in databases and the system as a whole reduces managing costs and possible consequences of data breaches.
Depends on	Nothing
Competes with	Nothing

Req. 2.6-12 Purpose legitimacy and specification

If the two requirements above are fulfilled, where a data subject gives his consent on the processing of data, there shall be no excessive collection of it. The collection must be fair in relation to the purposes defined:

Title	**Collection limitation**
Priority	High
Requirement Level	REQUIRED
Description	Defines the limitation of personal data collection efforts.
Rationale	The European directive 95/46/EC [31] on Protection of Personal Data states that the collection of personal data must be *"adequate, relevant and not excessive in relation to the purposes for which they are collected*

	and/or further processed" (art.6(c)). De-anonymisation is often achieved by combining several sources of personal data. Collection limitation helps to mitigate this problem by reducing the possible sources from which personal data can be linked together.
Pros and Cons	Pros: collection limitation will ensure a long-term compliance with EU directives. Cons: Requires constant effort to ensure that no party collects data excessively. It may be difficult to ensure an adequate collection, as different parties may define "adequate" differently, in case data minimisation is not implemented by design.
Depends on	Nothing
Competes with	Nothing

Req. 2.6-13 Collection limitation

The European Directive 95/46/EC [31] demands that "personal data shall not be processed at all, except when certain conditions are met" (article 6 (I) (b)). The requirement not to process data at all, can be very beneficial for system or architectural engineers, as, e.g., mechanisms for user consent agreement, management of personal data by the data subject and others, loose complexity can be omitted entirely. The requirement to avoid any personal data or to minimise their occurrence is one central step towards privacy by design:

Title	**Data minimisation**
Priority	High
Requirement Level	REQUIRED
Description	Data minimisation is closely related to collection limitation and purpose legitimacy. Data minimisation goes a step further; if possible, it aims at the complete renunciation of the collection of personal data. Only data which is absolutely necessary for the RERUM use cases shall be collected and processed. If data exists, it shall also be retained for as long as is necessary to fulfil that purpose and not longer. Per default, the RERUM app for smart transportation shall not collect sensor data from smartphones of crowdsourcing volunteers. After each new smartphone restart, the user could be asked if he wants to activate data collection for real-time traffic estimation purposes. Per default, RDs mounted on vehicles shall not collect data andshall be switched off. If technically possible, the driver could be "asked" when he starts the car if he wants to activate data collection for real-time traffic estimation purposes.
Rationale	This requirement complies with the European Directive 95/46/EC [31]

	article 6 (I) (b) stating: *"Personal data should not be processed at all, except when certain conditions are met*" and, if personal data is required, it shall be: "*not excessive in relation to the purposes for which […] [it is] collected and/or further processed."* Data minimisation heavily supports privacy-by-design, as it is designed to avoid the collection, generation and storage of personal data. RERUM will avoid the collection, generation and storage of unnecessary personal data by using adequate anonymisation and pseudonymisation solutions whenever possible.
Pros and Cons	Pros: Data minimisation benefits several other requirements. Assuming that data minimisation is fully achieved by anonymisation techniques and no personal data is required in the system at all, then: - Data collection procedures are not required - User consent agreements are not required - Accuracy and quality procedures are not required - Accountability responsibilities are not required - Intervention mechanisms for the data subjects to manage their data are is not required - Bandwidth is possibly reduced - Managing costs for storage is reduced
Depends on	Nothing
Competes with	Nothing

Req. 2.6-14 Data minimisation

If it is unavoidable to process personal data, as the data subjects have to be identified, then the data must be as accurate as possible and as up-to-date as possible. This will benefit processing and protect the data subject:

Title	**Accuracy and quality**
Priority	High
Requirement Level	REQUIRED
Description	This requirement ensures that data are accurate and up-to-date.
Rationale	The European Directive 95/46/EC [31] demands in article 6 (b) for personal data to be: "*accurate and, where necessary, kept up to date; every reasonable step must be taken to ensure that data which are inaccurate or incomplete, having regard to the purposes for which they were collected or for which they are further processed, are erased or rectified.*" If personal data is required for providing a service, then it is evident to a certain *quality* of personal data, to provide the best result possible. Personal data must also be *accurate* and not be misleading or defaming in regard to a person's physical, physiological, mental, economic, cultural

	or social identity.
Pros and Cons	None
Depends on	Nothing
Competes with	Nothing

Req. 2.6-15 Accuracy and quality

The European Directive 95/46/EC [31] demands in Article 12 the right of access. In RERUM, this will be defined two fold. First, the requirement for notice and access is defined:

Title	**Notice and Access**
Priority	High
Requirement Level	REQUIRED
Description	Personal data policies shall be clear and understandable, give proper notices which personal data are being in which way, and provide information on how to access and review personal data.
Rationale	From the European Directive 95/46/EC article 12 (a): *"communication to him [i.e., the data subject] in an intelligible form of the data undergoing processing and of any available information as to their source, [and the] knowledge of the logic involved in any automatic processing of data concerning him at least in the case of the automated decisions[...]".*
Pros and Cons	None
Depends on	Nothing
Competes with	Nothing

Req. 2.6-16 Notice and access

Second, the right of access demands mechanisms for interaction of data subjects. This is defined in RERUM as the requirement for individual participation and transparency:

Title	**Individual participation and transparency**
Priority	High
Requirement Level	REQUIRED
Description	If technically possible, RDs shall have a functionality, which enables the user to activate or deactivate temporarily the collection of sensor data. If data collection cannot be blocked, because a particular third party device has no collection opt-out functionality, then RERUM will enable user participation by allowing the user to block the transmission of the collected data to the gateways. If this is also not possible, there shall be a way for the user to notify the gateways to ignore / delete data originating

	from his RD. Transparency requires mechanisms to help a data subject understand where and what data are processed, and to verify that his consent remains valid for the specified data. Individual participation and transparency is for example needed in the smart transportation use case. Driven from this use case, its data collection application shall have a functionality which enables the crowdsourcing volunteers to deactivate temporarily the collection of sensor data (e.g. motion, environmental, and position data). (To deactivate data collection definitely, the app might be simply uninstalled and thus does not require a specific functionality.) Data collection shall occur in a transparent way, e.g.: – The RERUM app for smart transportation might display a notification of data collection for real-time traffic estimation purposes on the crowdsourcing volunteer's smartphone. This might take the form of an icon in the status bar. – A notification of data collection for real-time traffic estimation purposes might be displayed within the vehicle so that the driver cannot oversee it. For example, GPS-enabled devices may have a red notification light during recording. Alternatively, this might take the form of a label stuck inside the vehicle.
Rationale	According to the European Directive 95/46/EC [31] on Protection of Personal Data (art.12 (b)), data subjects have the right to block the processing of their personal data. This includes data collection. Transparency is not mentioned explicitly in the European Directive, but for a data subject it might be impossible to understand which data is processed where. Thus, the requirement for individual participation is extended with the requirement for *transparency*.
Pros and Cons	None
Depends on	Nothing
Competes with	Nothing

Req. 2.6-17 Individual participation and access

Based on the European directives[30][31], data subjects shall be able to hold personal data processing parties accountable for adhering the seven requirements discussed above. Also, in case of breaches, the data subject can demand an adequate compensation from the processing entity:

Title	**Accountability**
Priority	High
Requirement Level	REQUIRED
Description	Describes the need to specify entities, which, in case of privacy breaches, are to be held responsible.

Rationale	According to the European Directive 95/46/EC on Protection of Personal Data (art.23), data subjects have the right to receive compensation from a processing party in case of data breaches.
Pros and Cons	None
Depends on	Nothing
Competes with	Nothing

Req. 2.6-18 Accountability

2.6.4 Bootstrapping Requirements

Many devices used in IoT applications are "off-the-shelf" devices that can be bought in normal retail stores (i.e. smart meters, or advanced metering Infrastructure devices and other RDs). For this reason, it is necessary to register them in the application that they will support, create proper security associations (secret keys, secure channels, trust relationships) to other devices. and to the servers of the service providers, and to establish the ownership of the devices by the correct individual.

Title	**Secure bootstrapping of operational cryptographic credentials**
Priority	High
Requirement Level	REQUIRED
Description	Operational cryptographic credentials shall be bootstrapped in a secure way on RDs while connecting to the RERUM network.
Rationale	RERUM Devices need their own individual set of operational credentials. As these kinds of credentials cannot be configured by the manufacturer of the RDs, the credentials need to be bootstrapped in the context of RERUM.
Pros and Cons	Security services developed in RERUM e.g., to protect the integrity or confidentiality of data will rely on cryptographic primitives or protocols. These security services require operational cryptographic credentials on the RDs for successfully applying the primitives or protocols.
Depends on	May depend on the availability of initial credentials
Competes with	

Req. 2.6-19 Secure bootstrapping of operational cryptographic credentials

Bootstrapping of credentials on devices is a critical process and vulnerable to attacks. Bootstrapping can be sufficiently protected against attackers if a number of credentials have been pre-stored in the devices.

Title	**Availability of initial credentials**
Priority	High
Requirement Level	RECOMMENDED
Description	Initial or pre-established credentials e.g. installed by the device manufacturer should be available on RD before connecting to the

	network of a Smart City application.
Rationale	Initial credentials are required to increase the security level of the bootstrapping of operational credentials. Initial credentials are not applied directly for the security services developed in RERUM, but are applied only to secure the bootstrapping process.
Pros and Cons	Initial credentials allow bootstrapping of operational credentials in an automatic manner without manual interactions. On the other hand, manufacturing processes need to be adapted to deploy these kinds of credentials already during production of devices. If no initial credentials are available, bootstrapping procedures rely on a separate pre-establishment phase or a pairing process between devices requiring manual administrative actions.
Depends on	
Competes with	

Req. 2.6-20 Availability of initial credentials

Title	**Support of different operational credentials types**
Priority	High
Requirement Level	REQUIRED
Description	The bootstrapping process shall support configuration of asymmetric and symmetric cryptographic credentials.
Rationale	The RERUM security architecture needs to address several security requirements as described in the sections above like authentication, integrity, confidentiality, authorization or privacy requirements. The RERUM security services addressing these requirements will rely on different cryptographic mechanisms for which a single credential probably is not applicable, so that different types of credentials need to be supported.
Pros and Cons	
Depends on	Secure bootstrapping of operational cryptographic credentials
Competes with	

Req. 2.6-21 Support of different operational credentials types

Bootstrapping of credentials typically requires some kind of manual interactions of the person who is installing or plugging a device into a network e.g., pairing of Bluetooth devices which requires entering or comparing a Pin or manually accepting a device.

Title	**Reduction of manual interactions during credential bootstrapping**
Priority	High
Requirement Level	RECOMMENDED
Description	The bootstrapping process of operational credentials should reduce to a minimum the manual interactions at the RD by administrators or users.
Rationale	Installation and commissioning of RDs in RERUM will be typically not performed by individuals with a security background. To avoid weak or wrong configurations and to allow faster installation, the bootstrapping process should not require many interactions and should not require a strong security background.
Pros and Cons	Reduction of manual interactions allows faster and automated

	bootstrapping processes. If the need for manual interactions is completely avoided, unattended bootstrapping procedures are possible. On the other hand, unattended bootstrapping would require some kind of initial / pre-configured credentials.
Depends on	Availability of initial credentials
Competes with	

Req. 2.6-22 Avoidance of manual interactions during credential bootstrapping

Besides considering the complexity and the costs involved in this bootstrapping of trust and keys, it is important to consider subsequent key management operations like key update, revocation, and recovery mechanisms — recovery not only from malicious attacks but also from unintentional failures.

Title	**Update of operational credentials**
Priority	High
Requirement Level	RECOMMENDED
Description	During the lifetime of a RD, operational credentials should be updatable.
Rationale	Security guidelines recommend a limited lifetime of credentials. In case of compromise of credentials, the misuse is limited to the credential's lifetime.
Pros and Cons	Credentials' frequent update is a recommended practice that reduces the potential attack's consequences on compromised credentials. However, this has the drawback that a suitable key management infrastructure has to be established which may conflict with the dynamic behaviour of IoT scenarios.
Depends on	Secure bootstrapping of operational cryptographic credentials
Competes with	

Req. 2.6-23 Update of operational credentials

2.6.5 Requirements for the Secure Provision of SW and Configuration

In order to minimise the overhead of maintaining a framework for device adaptation on application demands, especially in terms of security, RERUM envisions integrating a tool for the management (e.g. change and adapt) of the security software of the system in a dynamic fashion. This shall best cover all necessary tasks for locating and installing the suitable software components. In order to do so, this security configuration tool must:

1. seek in accessible repositories (e.g. marketplaces) for software modules that fit the criteria stated by the user and download it;
2. invoke the over-the-air building service on the affected software to obtain the binary executables of the modules (when applicable) and
3. invoke the over-the-air deployment service of the affected devices deploying the correspondent binary executables on them

This provisioning provides high-quality services and a good experience to the end-users. The vision of IoT relies on the provisioning of real-world services, which are provided by heterogeneous objects that are directly related to the physical world [27]. Besides the conventional service provisioning, the on demand service provisioning is gaining particular interest, which enables the "injection" of new services as soon as they are required [101].

On the other hand, providing a way to find various types of components according to some criteria implies that the requested component must contain metadata specified in the criteria. Hence, it is likely that it is necessary to create a software repository that wraps the provided components with these metadata, unless they already have them, which is quite unlikely.

Title	Find deployable software to RERUM devices
Priority	Low
Requirement Level	RECOMMENDED
Description	The RERUM devices (e.g., gateways) should have the capability to be provisioned with new or updated software modules, after their field deployment.
Rationale	The devices should be able to search for existing solutions in repositories or marketplaces (and optionally if needed build them). This provides an easy access to the possibility of upgrading or initially installing a software module on a large number of different devices.
Pros and Cons	Pros: dynamic security reconfiguration Cons: Need to create a server that provides the metadata of the components to be found
Depends on	Over-the-air programming
Competes with	Nothing

Req. 2.6-24 Find deployable software to RERUM devices

If a RD is serving different applications, each application must be configured independently. Furthermore, a configuration server of a given application must not be able to alter the configuration of another application in that same RD (unless the other application is configured to accept the same configuration server). Even in the case that different applications in the same RD are configured to accept the same configuration server, these applications must be configured separately.

For example, if two applications A and B shared the same store, it could be possible for the administrator of application A to change some configuration parameters so it also affects application B. In practice, this means that the administrator of each application gets his privileges raised to also become administrator of the other application for that application, which implies that the administrator of application A can modify, either maliciously or accidentally, the configuration of application B.

Title	Object configuration isolated per application
Priority	High
Requirement Level	REQUIRED
Description	The system must guarantee that application configuration data cannot be mixed with each other, resulting in an improper elevation of user privileges
Rationale	It is conceivable that a single smart object serves several applications for

	different purposes. It should be impossible for the administrator of a given application to alter the configuration of the rest of the services. This requirement also impedes the administrator of an application to alter the configuration of other services involuntarily or for services that he is not allowed to modify. Helps detecting successful fake configurations
Pros and Cons	Pros: Capability of hosting several applications in the same RD without them interfering with each other. Cons: It might be more complex to implement a global management service that managed all the applications at once.
Depends on	Attribute-based Access Control
Competes with	None

Req. 2.6-25 Object configuration isolated per application

2.6.6 Miscellaneous Security Requirements

RERUMs methods for designing and building protocols, architectures and implementations are adhering or following existing best practices and known security frameworks. For example, RERUM applications and components shall be implemented according to secure coding guidelines and state-of-the-art security technologies (e.g. security protocols like SSL, IPSec) shall be used whenever possible (e.g. for secure wireless connections).

Title	**Secure design and implementation of RERUM components**
Priority	High
Requirement Level	RECOMMENDED
Description	Secure coding guidelines should be considered both (i) in the design of RERUM's system according to the best practices of architecture design and (ii) in the development and implementation of the RERUM software.
Rationale	Attackers are likely to exploit existing implementation bugs and design flaws.
Pros and Cons	Pro: Less security bugs from design-flaws or implementation errors. Better re-usable and understandable program code. Cons: More work to implement and design.
Depends on	none
Competes with	none

Req. 2.6-26 Secure design and implementation of RERUM components

Regardless of the algorithm for evaluating the reputation of a node, the RERUM system has to support the configuration of parameters of this algorithm without the need to update the binary

executables. The updates and the initial configuration of the reputation mechanisms must follow the same requirements as any other configuration of the system.

However the algorithm that evaluates the reputation is likely to be dependent on a set of values / weights that need to be tuned based on the experience during system execution. These values should be stored externally, making feasible their modification without the need to upgrade the whole system. It is even possible that the algorithm itself is stored externally as a configurable model that can be updated without modifying the rest of the system. The reputation mechanisms can be attacked as any other security mechanism of the system through its configuration. Hence, the configuration must follow the same protection criteria as the rest of the configurations of the system.

For these reasons, RERUM needs to protect the reputation configuration and the messages regarding these mechanisms.

Title	**Reputation mechanism for reliability, availability and trustworthiness**
Priority	Low
Requirement Level	RECOMMENDED
Description	The system should include a mechanism for measuring the reputation of a RD in terms of its reliability, availability, and trustworthiness. Reliability refers to whether the information provided by the RD is reliable or not. Availability refers to whether the smart object is working or not. Trustworthiness refers to whether this object has been used for attacking the system or if it has been compromised.
Rationale	Even if the RD does not seem to fail, it might malfunction. A reputation mechanism that detected this, could inform an administrator to take an immediate action.
Pros and Cons	Pros: Useful for administering the system Cons: It may be difficult to evaluate the reliability of a RD, and it could require to compare its measures not only against its previous ones, but also against the rest of the objects Need to store a history of the actions or performance indicators of the devices in the system
Depends on	Nothing
Competes with	Nothing

Req. 2.6-27 Reputation mechanism for reliability, availability and trustworthiness

3 The IoT Domain Model of RERUM

While the IoT-A ARM model [55] discusses in detail "things" (called "physical entities") and their vitalization, the devices play there a secondary role and require no virtualization. The goal of RERUM is to provide methods and tools for the enforcement and support of security and privacy-by-design. While network security (encryption, for instance) is related to *devices*, which communicate with each other, privacy is directly related to *the physical entities*, with which humans interact. In other words, the devices must be equipped with cryptography keys, etc, for the purposes of communication, while the "things" must be related to the privacy policies of the corresponding data subjects. This dichotomy is the basis for choosing an IoT model in which both *devices* and *physical entities* play a central conceptual role.

The main focus of RERUM is related to the lower layers, closer to devices. We tackle issues of equipping devices with the proper SW, configuration, security parameters, routing information, etc., on the one hand and of keeping track of their state, workload, current capabilities, performance, etc., on the other hand. The management of devices and protocols can be addressed using the principles of service-oriented computing, like loose coupling and encapsulation, achieving a significant flexibility of the RERUM architecture. For the purposes of interacting with devices, it is convenient to have also IoT representations of them, as we have for physical entities. But let us consider first the physical device (with sensing and/or actuating capabilities) together with the software necessary for our purposes, as discussed below.

This section describes the abstract view of the so called RERUM Device (RD) that RERUM uses to bring together the low level view on hardware and network layers and the IoT paradigm. The hardware and network layers of RERUM Devices take care of acquiring initially the sensory data from the physical environment or receiving the commands that instruct actuating in the physical environment (e.g. a motor that opens a window). RERUM uses the security requirements and in particular the privacy policies for the physical objects that humans interact with, e.g. the lights switch in a room in a building. Here the *switch*, *room,* and *building* are concepts that RERUM, in accordance with IoT-A [45], calls *Physical Entities (PE)*. One goal of RERUM is to embed key functionality for supporting the "security, privacy and reliability by design" concepts on or very close to the hardware (and software on it) that directly interacts or monitors the Physical Entity. That way security and privacy can be safeguarded very close to the physical entity under consideration. This hardware and software is what RERUM calls a RERUM Device, or a Smart Object. For clarity, to avoid confusion with the use of the term Smart Object in other projects or efforts, and to stress the device-oriented point of view, we introduce the term *RERUM Device (RD)*, which we will prefer over Smart Object (SO)[3]. This RERUM Device will be virtualized using the Middleware functionalities, so that it can be easily discovered, managed and access from the IoT applications.

For example, a RERUM Device needs functionality, that

- allows the network to interconnect a large number of heterogeneous RERUM Devices;
- allows the management and control of the underlying hardware functionality to achieve a reliable, self-managed, robust and context-aware communication network with minimized energy consumption;

[3] Here, we have to notice that in the previous deliverable D2.1, the term "RERUM Device" was not used because it was mainly focused on the general description of the Use-Cases under RERUM consideration, without discussing the specific implementations of the RERUM system. From this deliverable and now on, the term RERUM Device will substitute what was called in D2.1 a "Smart Object".

- enables the network to transport sensory data and actuating commands in a secure, trustworthy and privacy preserving manner.

Please note that this section will establish a first version of the RERUM's high-level and low-level views of the capabilities and abstract functionalities that a RERUM Device shall offer to the different layers, together with a harmonized terminology that will be reflected in the RERUM architecture (to be analysed in future deliverables). We offer a discussion on how RERUM's view relates to other existing models and terminology in Section 3.3. Over the coming months, this model will be subjected to constant scrutiny when the project advances. In line with RERUM's work plan, we foresee to be able to extend or even refine the RD model given in this deliverable, when we build the RERUM architecture upon it. The reader is advised to read the RERUM Deliverable D2.3, which will formulate an updated version of this RD model. The planned date of delivery D2.3 is Month 12, that is, the end of August. 2014.

With this in mind, this section will be split into three parts:

Part 3.1 gives a brief description of the RERUM Device (RD) from the point of view of the hardware functionalities and the communication layers that the software and the hardware radio interface will offer. Furthermore, a first description of the CR-agent that will allow the RD to have cognitive radio capabilities is also presented. These describe the lower (below IoT) layers of a RERUM Device.

Part 3.2 discusses the virtualization of the RERUM Device as an IoT concept: the Virtual RERUM Device (VRD), which provides useful functionality for enabling the higher-levels of IoT, in a service-based view. The goal is to provide abstract functionalities of a RERUM Device for the upper layers of the IoT domain. Especially, those functionalities are independent of any proprietary protocols or constraints. This is mainly how IoT applications will be seeing the RERUM Devices.

Part 3.3 provides an integrated view of RERUM's terminology, including the RERUM terms VRD and RD, in the IoT world. We also present the RERUM model using the *IoT-A reference architecture model*.

3.1 *RERUM Device*: The Hardware and Network view

This section presents the hardware and network view of the RERUM Device

Conceptually, we start with a list of hardware building blocks, like processing units for which core functionalities can be separated from each other. The following building blocks describe the low-level functionalities of the hardware and networking layers of a RERUM Device. In other words, they describe the *RERUM Device*'s technical capabilities available to RERUM's higher level, where RERUM builds from this a "truly smart device".

3.1.1 Hardware Components of a RERUM Device

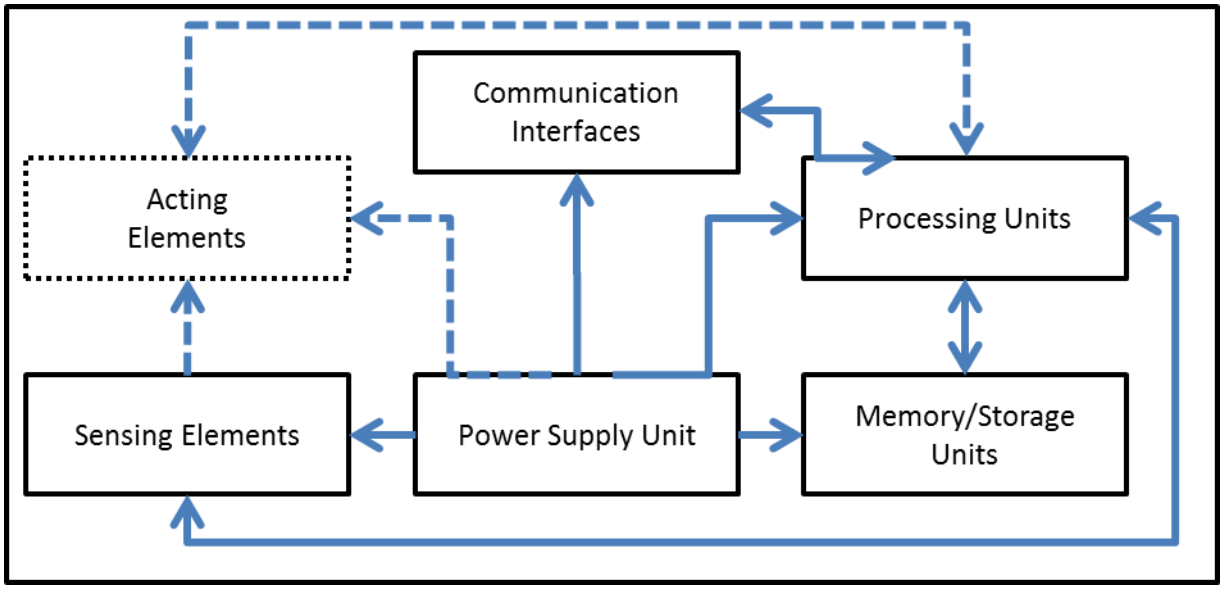


Figure 6 - RERUM Device hardware components

Figure 6 shows the basic components of the hardware of a RERUM device, including necessary components to support an operating system and to communicate with other devices.

The simplest form of a RERUM Device includes the following components:

- Power Supply Unit: consumes power from either small sized batteries or from a USB port and supplies all other components with power.
- Processing unit: manages all the computation of the device, being the main unit that executes commands and instructions. It is connected with all other components, as it is the main one that controls the operation of the others.
- Communication interfaces: these are the interfaces that allow communication of the device with other devices. Mainly they are wireless (IEEE 802.15.4 or IEEE 802.11), USB and other connectivity interfaces.
- Sensing elements: one device can have one or multiple sensing elements that measure properties of one or more Physical Entities of the real world. They also convert the analogue signals to digital, feeding them into the processing unit.
- Memory/storage unit: this unit allows the temporary (RAM) or permanent (e.g. flash) storage of data that can be either sensing/actuating data or software code, i.e. OS.
- Acting elements: these are optional components, altering the properties of a Physical Entity. They convert digital information from the processing unit into electrical signals to affect the attributes of a Physical Entity.

The current generation of IoT devices has bidirectional wireless communication and sensors that provide real-time data such as temperature, pressure, vibrations, or energy measurement. The **RERUM Device (RD)** can be considered as a small computer with communication interfaces, sensors, and (optionally) actuators.

This generic hardware specification of the RD allows for RERUM Devices to be implemented on a wide range of physical devices including smartphones and the Zolertia Z1. Most of the devices for RERUM Devices are characterized by low computational capacities and small available memories, while mobile devices also have rather tight power consumption requirements.

3.1.2 Layers of Software and Hardware on a RERUM Device

The Devices for Smart Object should be able to handle an operating system and they must be compliant with the networking protocol chosen to communicate with the other RERUM Devices and, through Gateways, to other RERUM architecture components.

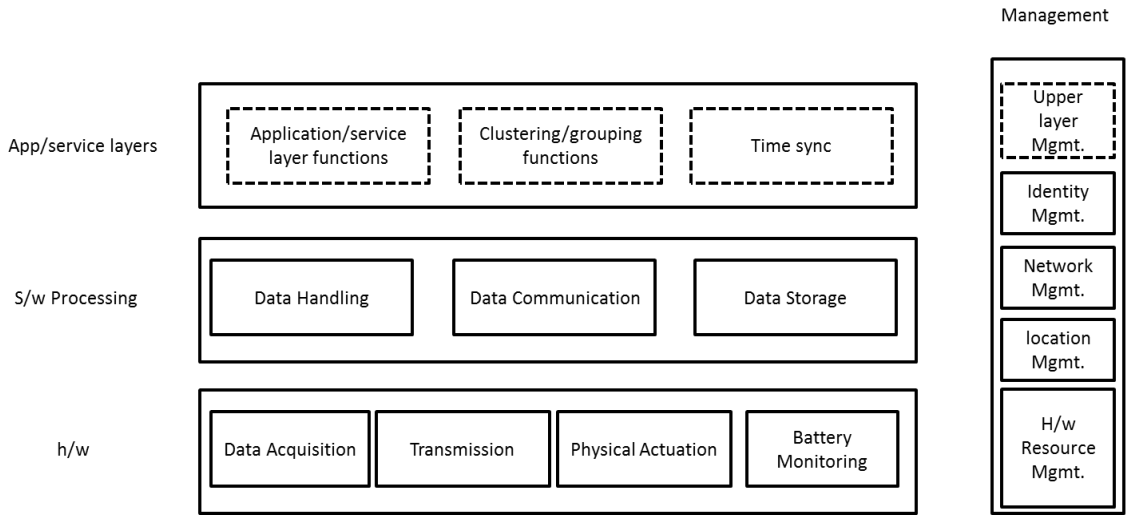


Figure 7 - RERUM Device lower layer functionalities

Figure 7 above presents the low level functionalities of a RERUM device, starting from the hardware and the software processing levels. It includes the enabling components of the upper level functionalities that will be defined in the deliverable for the RERUM architecture (D2.3). Furthermore, the SW on the device includes a set of cross-layer management mechanisms. The focus of this subsection is on the lower layer functionalities of the RERUM Device, while the upper layer functionalities will be defined in detail in the architectural Deliverable D2.3. However, in terms of completeness, in the figure above a few indicative functionalities are depicted, i.e. time synchronization of the device, the various functions that enable the device to provide efficient services, as well as all the functions that enable the device to be grouped into clusters.

The low level components of the RERUM device are:

1. Hardware (h/w)

- Data acquisition: The activation of a sensor to produce raw sensed data according to its formatting specifications, i.e. access raw measurements of the temperature of a room.
- Physical Actuation: The activation of an actuator to perform a single task, i.e. sending an electrical signal to close the door of the room.
- Battery monitoring: The process of supervising the battery (if the device is battery operated) to assess the remaining charge level.
- Reception/Transmission: The MAC/PHY activation of a single interface to transmit/receive data to/from another device within the network.

2. Software (s/w) Processing.

- Data handling: The transformation of acquired data to a form that complies with a set of predefined policies regarding e.g. formatting, compressing, or security. This software is also responsible for transforming the raw sensing data to a comprehensible value, i.e. transforms the raw binary temperature data to a value with a meaning (e.g. 10 degrees Celsius) and for

transforming the actuating command data to the digital signal to be sent to the actuator to act according to the command.

- Data storage: The storage/retrieval processes of raw or handled data on the device for running stand-alone services, local profiling, or data batching and optimization of communication
- Data communication: The set of processes needed for the correct, secure, and efficient forwarding or receiving or data. This may include the implementation of opportunistic routing or of Cognitive Radio-inspired scheduling policies.

3. Management

The RERUM device includes a set of functionalities (which may be optional) related to the management of resources of the device. The basic low-layer management functionalities are depicted in **Figure 8** (including, for completeness, the box for upper layer management that will be defined in the next deliverable):

- Identity management: This module is responsible for providing a unique identifier (Device ID) to a RERUM device, such that it can be searchable and discovered by the other nodes/devices of the system. The module is further responsible for low-level verification of credentials at lower communication layers, identities or authorization of other nodes, in particular of the RERUM administration entities.
- Network management: The module includes all network layer-related management functionalities, i.e. Cognitive Radio mechanisms (defined below), routing, opportunistic mechanisms, etc.
- Location management: This module is responsible for the management of location information of the device and it may be connected to a location sensor (i.e. GPS).
- H/W resource management: This module includes several functionalities that are depicted in the figure below:
    - Communication interfaces management: It manages the communication interfaces according to the upper layer commands, i.e. for radio interfaces it can set the frequency, bandwidth, modulation requested, etc.
    - Energy management: It manages the energy consumption of the device, applying sleep and wakeup methods
    - Storage management: It manages the storage of the data on the device and applies store and forward mechanisms in the case of Communication based on opportunistic routing.
    - CPU scheduling: It schedules the allocation of the CPU resources to processes.

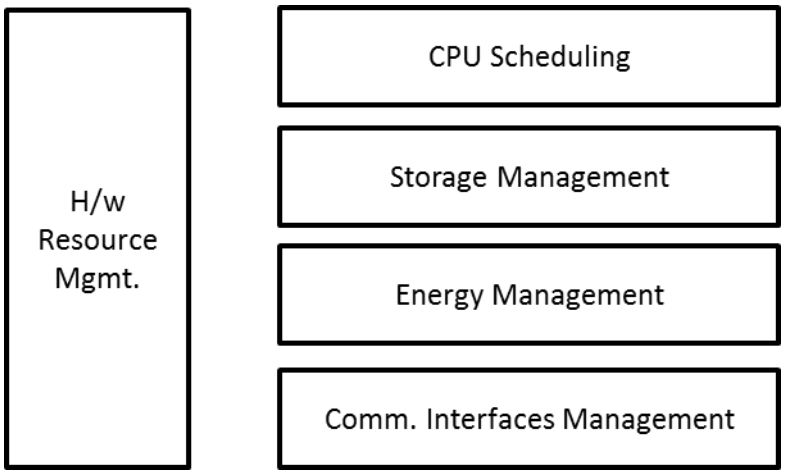


Figure 8 - RERUM Device H/W resource management functionalities

3.1.3 Cognitive Radio Agent on a RERUM Device

The efficient communications at the lower layers are crucial for the interconnectivity at the IoT layer. If the hardware devices cannot be connected to the network, then they are not able to support any type of services (they won't be able to send sensing measurements or get actuating commands by the applications). One key advantage of the RERUM Device comparing with the state of the art sensor platforms is the capability for enabling Cognitive Radio (CR) inspired communications. Cognitive Radio (CR), proposed by J. Mitola in [59], has emerged the last decade as a promising technology able to exploit the unused portions of spectrum in an opportunistic manner. CR was first introduced for opportunistic radio access in niche applications, such as wireless microphones, later adopted on traditional ad-hoc networks [45] for improving spectrum access, lately the interest in being moved to sensor networks and the smart grid as presented in [68].

For efficiently interconnecting wirelessly millions or even billions of smart objects and sensor platforms several issues have been identified [69]: (i) hardware heterogeneity, (ii) different communication protocols, (iii) different communication technologies, (iv) interference in unlicensed frequencies used by sensor networks, (v) single-radio devices, (vi) limited energy resources, (vii) no IoT-tailored QoS requirements.

IoT communications are based on CR-inspired devices can adequately address most of the above mentioned issues. This is achieved due to the intelligence that is induced in the devices and their ability to adapt to environmental conditions. The basic idea of a CR-inspired RERUM Device is that it can exploit the unused spectrum bands opportunistically, but taking only in an energy efficient way. To equip the RD with CR capabilities, we need to include in the device a **CR-agent** that handles the CR functionalities. The proposed internal structure of the CR-agent is shown in **Figure 9**.

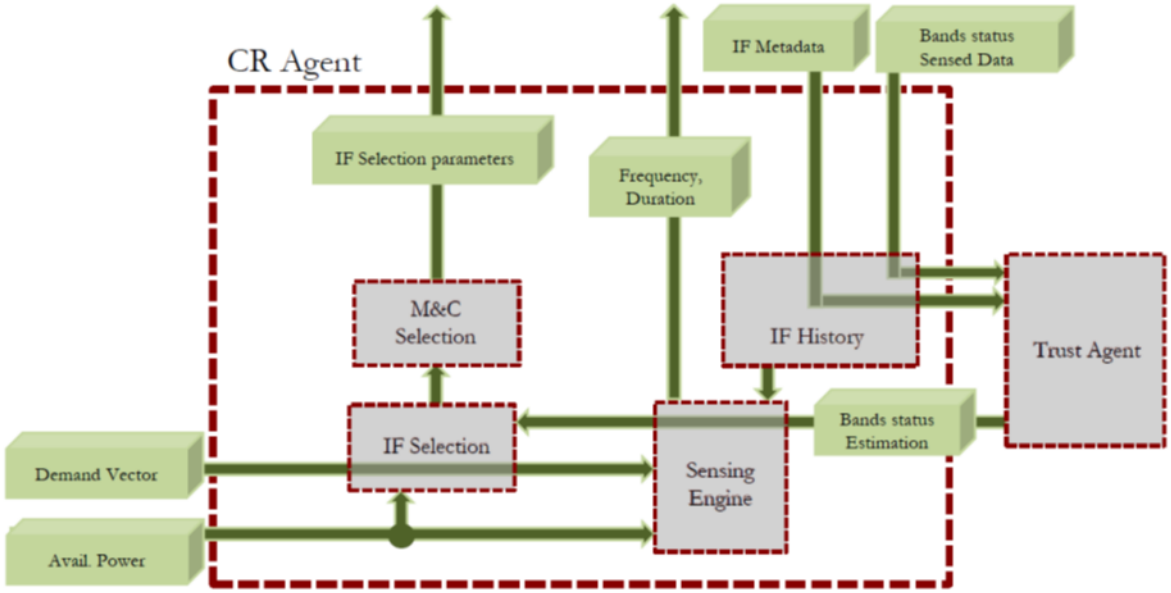


Figure 9 - The CR-agent of the RD

The internal modules of the CR-agent are those depicted with dotted lines in the figure:

- **IF History:** a database with historical information about spectrum utilization in various frequencies.
- **Sensing Engine**: the module that handles the radio interfaces and is responsible for sensing the wireless spectrum using a specific spectrum sensing technique, i.e. energy detection [23].
- **Trust Agent**: a module for evaluation of trust when collaborative spectrum sensing is used [98]. In this technique the RDs are exchanging sensing measurements in order to take a decision about which frequency to use. In this case, the RDs should know which measurements to trust. This Trust Agent does not have to be a specific module for the CR-agent, but could also be a general software module within the RD related to trust and reputation management.
- **IF Selection**: the module performing the selection of interfaces and spectrum.
- **M&C Selection:** the module selecting modulation and coding scheme to be used by the interfaces.

These modules have been identified as required modules for enabling the CR capabilities of the RD and will be defined in detail (together with the implementation of the respective CR mechanisms) in deliverable D4.1 (to be delivered end of February 2015).

3.1.4 RERUM Device in the context of the RERUM IoT Model

The definition of the RERUM Device is inspired by the definition of IoT-A's Device. In IoT-A [22], a "device" can be one of three types: sensors, tags and actuators. Furthermore, a device can also be an aggregation of several devices of different types. In RERUM, the RD is based on the definition of IoT-A, but is assumed to be more "intelligent", to address the target of the project to "embed intelligence on the devices". This type of intelligence comes from the embedded mechanisms for security, privacy and reliability and the ability of the RD to identify which mechanisms it will use for each type of application it supports. More detail on this subject will be given in the architectural deliverable D2.3.

Figure 10 shows the RD in the context of the RERUM IoT model. As depicted, a RERUM Device encapsulates the hardware and software and emits what IoT-A calls "On-Device Resources" that can be exposed to higher IoT-A Services. The RD has embedded one or more sensors, tags and actuators that use electrical signals to either sense the environment or act upon it. These analogue signals are transformed to digital through the use of Digital Signal Processing (DSP) techniques that, i.e. convert the electrical signal of a sensing element to raw data and then to a corresponding format that is readable by a machine or a human being. Then, these formatted data are being passed through the RERUM software (i.e. being encrypted, compressed, modified, secured, etc.) and are exposed to the service world as Resources.

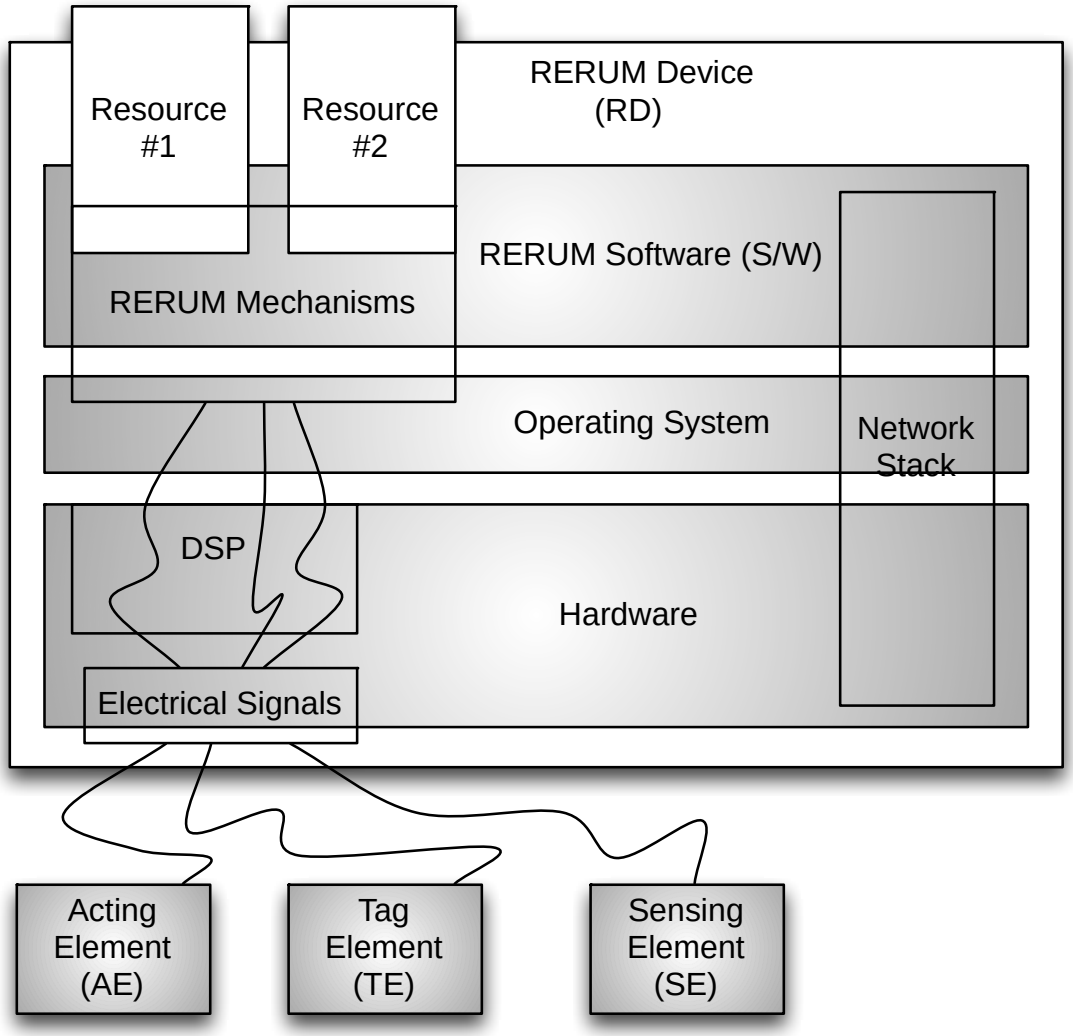


Figure 10 - The RD in the context of the RERUM IoT model

3.2 Virtual RERUM Device: The IoT view

Based on the requirement for an IoT virtualization of RERUM Device, this section discusses some basic aspects of the Virtual RERUM Device (VRD). Due to the vast work in other IoT projects regarding the implementation of Smart Object models, RERUM will avoid duplicating the work by creating yet another new Smart Object model, but rather RERUM will reuse existing models and concepts in standards and other IoT related projects, such as IOT-A [92], FI-WARE [40], iCORE [93], BUTLER [41], IOT.est [91]. From the beginning it is important mentioning that we are looking at the IoT view of

existing Smart Objects models with the intention to develop RERUM's IoT view such that it can be compatible with on-going efforts of other projects and to visibly support the objectives of RERUM.

There is a gap between the raw data generated by the sensing elements in the devices, and the need for services to become consumed by high-level applications (as it is also visible in **Figure 10**). It is not the purpose of RERUM to help bridging this gap, but to reuse or interface with existing solutions. For example, RERUM will not work on DSP techniques for transforming the raw analogue signal in digital signal. For this, a system of layers is usually required to provide meaning to the data and solve complex queries: one layer provides the raw data, a second one applies the data processing to meet the required specifications or requirements of services, and further layers provide context or enrich the data with further information, like some type of domain knowledge, statistics, forecasts, etc.

The main reference work for VRD Ontology is the semantic web [86] and the W3C SSN [42] that provide an integrated view regarding the "observer", its running environment, and the observed data.

IoT-A gives a detailed description of the IoT concepts related to the process of monitoring or sensing data by devices (proposing here the Resource and Service concepts) and data (via event processing) and their meaningful association to the IoT representations of "things" (the Virtual Entities), which then provide services to the applications, see [55].

In a follow up approach, iCORE [93] investigates how data and services on the IoT layer ("virtualized" devices) could be aggregated on demand to support decision procedures or Artificial Intelligence (AI) in response to explicit applications service requests (similar to a generic VRD federation in RERUM). iCORE's approach on virtualising the devices (called ICT Objects) is close to the concept of VRD in RERUM. In iCORE an ICT object is represented by a Virtual Object (VO) in the IoT world. The ICT Objects can be associated to non-ICT Objects (which are the Physical Entities of RERUM and IoT-A). VOs are described by a VO template, which is created based on the key functionality of the Objects and the involved Real World Object(s) (RWO). VOs enable access to the real world objects, help interfacing them (after abstraction) to the services and therefore help in complex functionality to be abstracted through a uniform way of representing data. Similar to the concept of "VRD federation" is the concept of Composite Virtual Objects (CVOs) in iCORE. CVOs comprise of one or more VOs and some functionality on top of the VOs. CVOs exploit the functions provided by VOs. A CVO is a mash-up of VOs that renders services in accordance with user/stakeholder perspectives and application requirements. Sometimes, a CVO adding functional processing can be mashed-up to one or more CVOs [28].

RERUM proposes that a Virtual RERUM Device provides a service or a bundle of services, offering to client applications and other VRDs a standardized (not proprietary) interface for the services and standardized formatted data, within a certain visibility space. Notice that the reason why a VRD offers such an interface to other VRDs is to facilitate the creation of federations at a device-level (which we have called Smart Object federations until now). Based on initial RERUM conceptual plan, RERUM Devices should also facilitate the design, implementation and instantiation of federations of such RERUM Devices. Federations are constrained groups of RDs realizing together a service towards another RD or a client application. An example of a VRD federation is when the mobile phone of a person interacts with the device that controls the light bulb in a room and the device that controls the door of the room, so that when the person reaches near the door, then the door open automatically and the light bulb is switched on. Here, all three RERUM Devices are cooperating to provide a service to the user ("to enter a lighted room when the user approaches within i.e. 2 meters").

Working towards the IoT/IT enablement of the real world we can consider that the Virtual RERUM Device (VRD) is the virtual representation of an intelligent, connected object or device that is associated with some observation or actuation processes of a real world object. By *virtual* we mean that the VRD is a software artefact, just as a Virtual Entity (VE), see [55].

VRDs help in accessing the real world objects and in interfacing them to applications or users. Virtual RERUM Devices can be seen as providing a very basic set of functionalities representing the actual functions of associated real world things.

Internally, a VRD contains **two major parts**: a **VRD front-end**, which interfaces to applications and other VRDs, and a **VRD functional back-end**, which interfaces to the RD. The VRD front-end includes general commands for all similar type of web connected RERUM Devices. In this way the VRD becomes an IoT concept: a VRD is exposed to the IoT world as data source and control point for the corresponding peer real world process/device. On the other hand, the VRD back-end may contain vendor-specific instructions or constraints. It is up to the device vendor to decide which information to expose from the devices and provide according the low-level, perhaps proprietary, interfaces to access this information (i.e. the DSP mechanisms described in **Figure 10**). The communication patterns are very heterogeneous because the ICT objects from various vendors can be very different.

In this respect, the RERUM perspective introduces three basic principles for dealing and transforming Virtual RERUM Devices into more usable and service bound entities, these are:

***Grouping in Federations*:** The functionalities offered by several Virtual RERUM Devices can be grouped, represented, coordinated, and controlled by other virtualized entities. In addition, these functionalities can be constrained associations, i.e., the functionalities of different Virtual RERUM Devices can be integrated in order to make them generic and easier usable for applications, or to create new functionalities based on the integration of the functions of the single Virtual RERUM Devices.

***Abstraction from the real world*:** The functionalities of a Virtual RERUM Device can be generalized and abstracted to make them adapt to a large set of situations and requirements.

***Functional Enrichment*:** the VRD is an IoT entity, which can provide the functions offered by sensing and actuating elements in both a non-proprietary and a service-oriented way. Additionally, the VRD can provide new functionalities related with the context via “context awareness” [63]. The VRD can also “enrich” the data with further information, i.e. with domain knowledge, statistics, a forecast, etc. That way, the object can have awareness of the situations it is involved in and decide itself about its actions and the type of services it will expose.

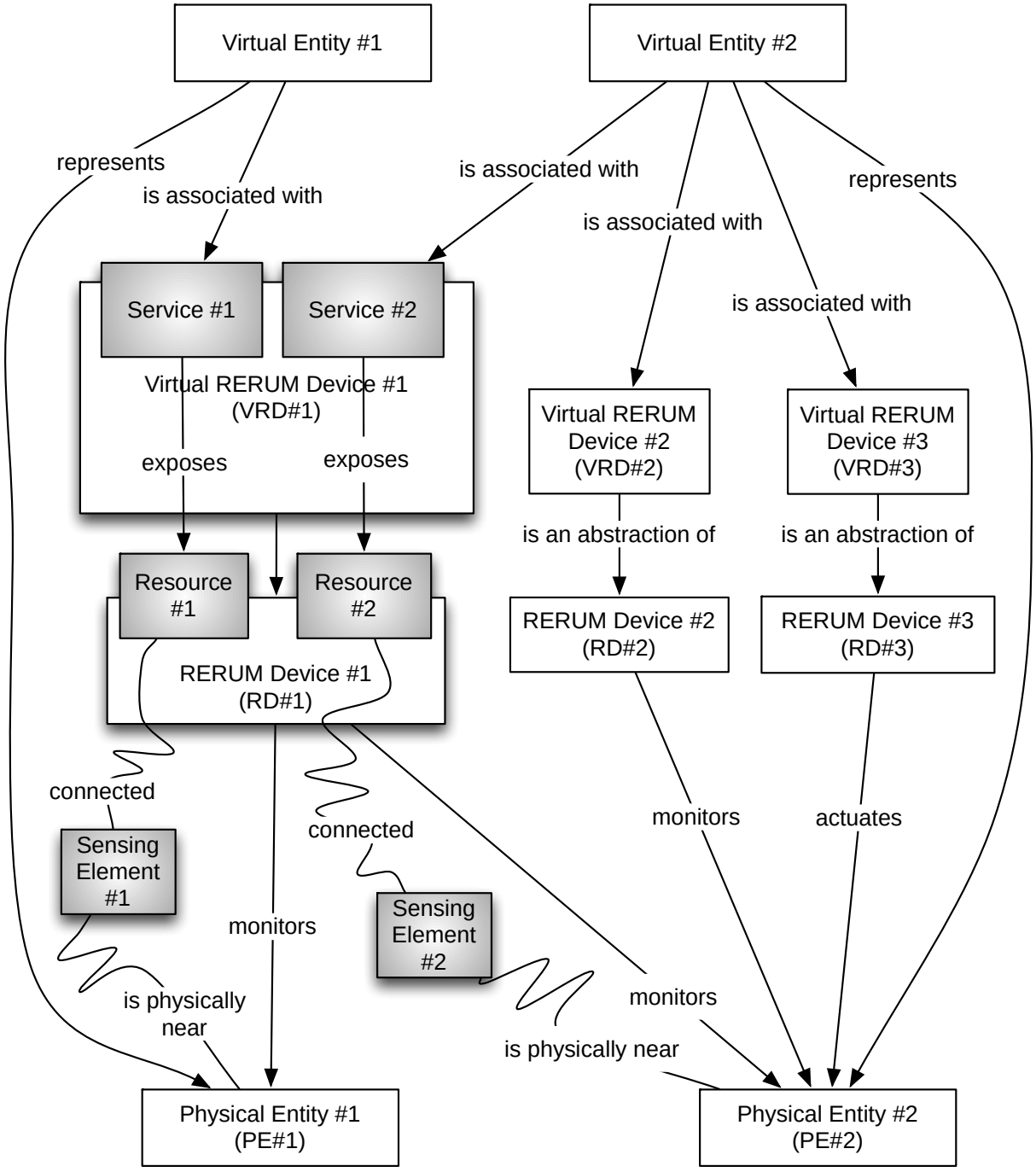


Figure 11 - Schematic representation of the relation between the different entities

Figure 11 shows, in a particular but exemplary case, the relation between the different entities discussed so far: the PE and VE, (that are expressed in a similar way like in IoT-A) and the RD, VRD from RERUM:

- The RERUM Devices are monitoring the Physical Entities. One RD can monitor multiple PEs using the same or different sensing/actuating elements. Multiple RDs can monitor the same PE. For example image a physical dual light switch that is on the wall next to a door and that would allow to operate the light in the room in front of the door and the room behind the door. Or a hardware that has two thermal sensing elements one that measures the inside temperature and one on a longer wire going to the outside of the window that measures the outside temperature. In **Figure 11** you see that:

  - RD#1 monitors PE#1 (using the Sensing Element #1) and PE#2 (using the Sensing Element #2).
  - RD#2 monitors PE#2
  - RD#3 actuates on PE#2
- The RDs can have multiple Resources on them. The Resources are the software that translate the sensing data and the actuating commands in a uniform way to be used by the services.
  - RD#1 has Resource#1 (that translates the data that come from Sensing Element #1) and Resource#2 (that translates the data that come from Sensing Element #2)
  - RD#2 and RD#3 have similar Resources
- Each RD is virtualized in the higher layers as a VRD.
- Each RD Resource is exposed in the VRD as a Service, thus VRD#1 has two services exposing the two Resources of RD#1. A request for access of the Service#1 of VRD#1 can also be denoted as a call to "VRD1.S1".
- Each VRD is associated with one (or multiple) VEs that are abstractions of the PEs that the RDs (corresponding to the VRD) are monitoring. Thus, VRD#1 is associated to VE#1 and VE#2, because RD#1 monitors both PE#1 and PE#2.

Figure 11 shows also the way RERUM will facilitate Resources to retain a one-to-one relation between the VRD and the physical hardware of a RERUM Device that has the sensing elements attached to it. Assume that PE#1 is the 'outside' and SensingElement#1 is measuring the temperature and PE#2 is the 'room' and the temperature is an attribute of the 'room' and SensingElement #2 measures it. In **Figure 11** both sensing elements are connected to one hardware device RD#1. From a low-level hardware point of view the two sensing elements will be connected at different hardware pins and be able to read out individually.

The software running on the RD will be able to process them, also differently if necessary, and store the interpreted readings of the electrical signals into two different Resources. These two Resources are then exposed also as two individual Services. These two Services constitute an integral part of the VRD's view. Hence the VRD#1 exposes the Resources as two different Services #1 and #2.

Each of these services can then be associated individually to the corresponding Virtual Entity. So that the sensing element's readings that monitor an attribute of the PE#1 are taken into account for the VE#1 that represents the Physical Entity.

Hence **Figure 11** shows also how the VRD in connection with the Resources of RERUM Devices facilitates a loose coupling of the hard and software that monitors the PE and the VE that represent the PE by means of attached RERUM Devices.

Here is an initial list of features to be provided for the VRD IoT enablement, and will be detailed in D2.3 and D2.4:

- Addressability and discoverability
- Relevant attachment/sensing/action for a real world thing
- Expose a functional description of mapped "thing" (including in/outs, restrictions regarding functional limits, associated computation)
- Restrictions (SLA: response delay, maximum users, messages, ETA etc.)
- Run in a known and identifiable container who is managing the state (if is relevant) and the lifecycle of processing, such the case of RD implemented as Event Processing Networks [77]

- Context awareness/situation model – each Virtual Smart Object is available to one or more applications and each application might ask for additional restrictions or observations when the RD is to be used
- Federation model – what are the features to be added in the RD model to enable its use in a federation
- Reference a memory (log) of own data and actions
- Compute/process data capability

Some further aspects of the VRD are discussed in the following subsections.

3.2.1 Connectivity model of VRDs

Connectivity between the RERUM Devices happens not only at the hardware level, but also at the higher levels. For a physical RD to become a VRD its network level address for communication needs to be abstracted from and it needs to become addressable as a VRD. In order for those Virtual RERUM Devices to become interconnected on the level of Virtual Objects their respective underlying network technologies are hidden, but the RERUM Services they are going to offer are exposed. In order to facilitate the consumption of those services, each Virtual RERUM Device should expose how other can communicate with it. These advertised means of communication with a VRD are recorded by the middleware probably in registries or related repositories. This allows the interaction with client applications, with intra-domain VRDs for federations, or possible with servers or gateways. The RERUM communication model will be probably based on simply structured formatted text or on semantic descriptions, which will specify the grounding, that is, the description of the protocol to be provided to the callers. Current practice can be Web Services based, likely REST*, COAP or MQTT. The choice on an exposure mechanism may depend on the use case and the data to be sensed or actuated. In short, RD connectivity will be based on service descriptions with standard grounding representation.

IoT services exposed by IoT resources may be constrained by limitations on computation capabilities and often operate in dynamic physical environments. Compared to the Web services on a back-end server, they are clearly less reliable and stable; their logic is usually much simpler and their output usually represents observation and measurement of features of interest of physical entities. Despite these characteristics, in a service oriented IoT they need to participate in service composition ([18]) and the issues on effective service adaptation and compensation mechanisms become prominent.

RERUM will consider re-using part of IOT.est IoT Services ontology as described on D3.1 and D3.2 of IOT.est, which is one of the most complete ontologies for services.

3.2.2 Naming and addressing models for VRDs

In order to be uniquely identified within its own domain (where a domain is the name space addressable in a certain context) a Virtual RERUM Device needs to have a unique ID, a name, and associated metadata (to be detailed in D2.3 and D2.4). A VRD should host at least the ID; all other data may possibly be hosted and used in the middleware or a gateway. The naming and addressing schemes for the VRDs will be developed inside the middleware and detailed in Deliverables D2.3 and D2.4.

3.2.3 Functional model for the establishment and execution of VRDs federation

A VRD can be used alone or associated in federations. The “functional model” associates the available functions of a VRD (like available sensing or actuation) and may be enriched with further information. Functional models are used by applications or other VRDs to understand the use

restrictions or pre-conditions, that is, the context related with where and when they can use a VRD. The functional model should provide the condition for federation definition, instantiation and execution, expressed as time, geo-spatial and functional constraints. In particular, federations, as detailed in the requirements section (2.4.4) represent coherent constrained groups of VRD that are combining their functionalities for providing a specific service related with changing/monitoring the conditions of a PE. With each VRD exposing a number of services, the RD federation can also be assumed as a composition of VRD services for accessing/changing an attribute of a VE.

The functional constraints of the federation are logical formulas defined using a few relevant dimensions, like the ordering of actions or service calls ("VRD1.S1 is called only after successful completion of VRD2.S2 "), the geo-spatial distribution of the devices ("VRD1.S1 is called only if VRD2.P2 = 3m range"), or the result of other activities ("VRD1.S1 will not be executed if VRD2.S2 is not true", or "VRD2.S2 is called after VRD3.S3 returns success"). In this example, VRD2.P2 denotes the relative position of the VRD2 with regards to the VRD1.

3.2.4 VRD profile model

The VRD profile Model should provide the technical means to express the geo-spatial, time, function and order dependencies or constraints to be managed by the middleware. There is a consistent work performed in IoT-A and IOT.est projects regarding semantic profiles of IoT services. On its forthcoming architecture model, RERUM will evaluate the experience gathered in the IoT-A IOT.est, OpenIOT, iCORE and BUTLER projects, and will and consider which results to reuse.

3.2.5 VRD execution model

Current experiences in FP7 projects use a conceptual view of IoT objects based on two complementary views: (i) a process view, where the objects expose web services, which are composed following classical methods or constructs in Business Processing (such as BPMN or BPEL) [39], and (ii) an event driven view where objects are considered consumers or producers of events. Those approaches do not contradict each other and can be used complementarily.

It is relevant for the execution model to consider the way IoT services are deployed in a specific environment, if a VRD will be used in an event driven manner or service oriented one. The Virtual RERUM Device execution model expresses those choices and binds them to concrete protocols or interfaces.

3.3 RERUM's IoT Model

We will now present the terminology for RERUM, which is based on the IoT-A conceptual model of the Internet of Things, as presented in detail in the IOT-A Reference Model chapter of the book "Enabling Things to Talk" [22]. We extend this model with device-centric concepts and use this extension to explain the basic RERUM concepts.

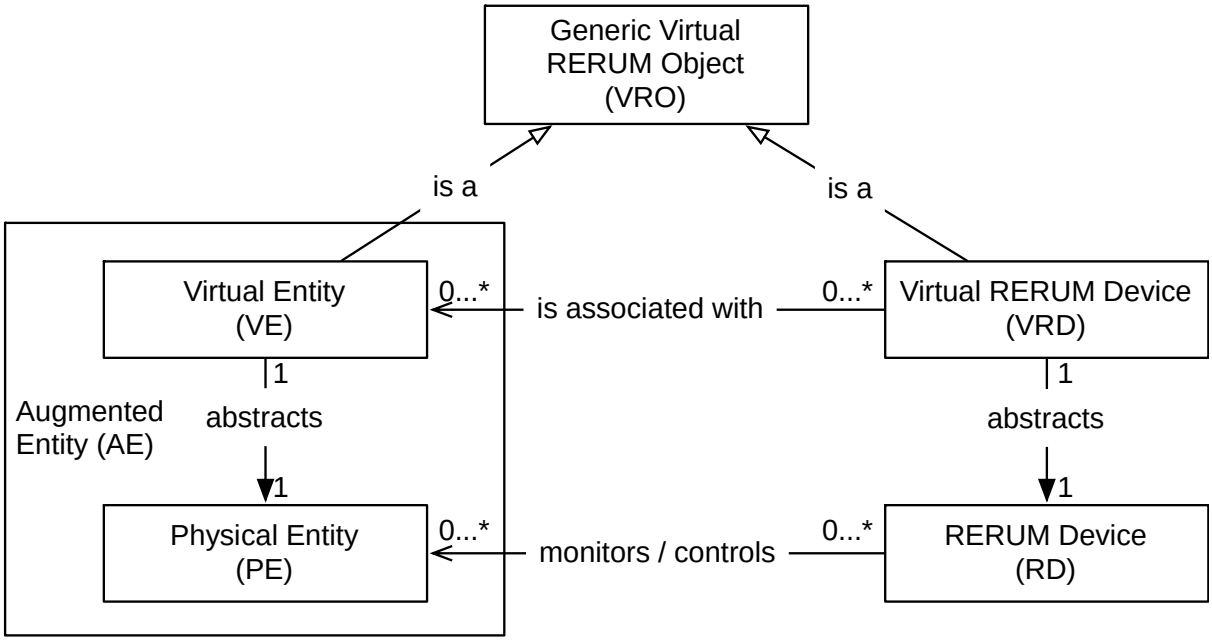


Figure 12 - Relation between RERUM Devices (RD) and Physical Entities (PE)

In **Figure 12** we show that RERUM introduces an extended and more detailed view of a device and it also considers the virtualization of devices. Note that the RERUMs understanding of PE, VE and AE is consistent with that of IoT-A. The figure uses UML-like notation and indicates the cardinalities, which are as follows:

- One RD can monitor/control none-or-many PEs.
- One VRD can be associated with one-or-many VEs.
- Only one VE represents one PE at a time.
- Only one VRD represents one RD at a time.
- One-or-many RD can be monitoring or acting on one PE.
- One-or-many VRD can be associated with one VE.

Note, that PE, VE and AE are used in the IoT-A meaning while VRD and RD are RERUM specific terms. We will provide the descriptions of the terminology next. Also note that whenever there is no need to differentiate between Virtual RERUM Devices and Virtual (Physical) Entities RERUM will use the term Generic Virtual RERUM Object (GVO). The inheritance association between both virtual representations and the GVO illustrates this. And it means that a GVO is either a VE or a VRD.

3.3.1 Term: Physical Entity (PE)

Our terminology	IoT-A correspondence	Source(s) our term is based on
Physical Entity (PE)	Physical entity	IoT-A

Description	Any physical object that is relevant from a user or application perspective. The **Physical Entity (PE)** is a discrete, identifiable part of the physical environment, which is of interest to the user for the completion of a goal. Physical Entities can be almost any object or environment; from humans or animals to cars; from store or logistics chain items to computers; from electronic appliances to closed or open environments.

In the IoT context, a Physical Entity is linked with a device, which is able to sense or manipulate its environment. In some –but not all– cases, the device is embedded onto the physical entity. For instance a camera can monitor a parking lot, which is the physical entity or a thermometer can monitor the temperature of a room. For our purposes it is very important to choose carefully what the physical entities are, in particular because this is what determines the "data subject" is in the sense of privacy, related to the information that the corresponding device collects. Thus, if a camera monitors a car, the owner of the car is the data owner and not the owner of the camera.

3.3.2 Term: Virtual Entity (VE)

Our terminology	**IoT-A correspondence**	**Source(s) our term is based on**
Virtual Entity (VE)	Virtual Entity, but RERUM assume a one-to-one relation.	IoT-A
Description	A Physical Entity (PE) is represented in the digital world via a **Virtual Entity (VE)**. Virtual Entities have two fundamental properties: 1. They are Digital Artefacts. 2. Virtual Entities are synchronised representations of a given set of aspects (or properties) of the Physical Entity Virtual Entities that are related to large Physical Entities might need to rely on several, possibly heterogeneous devices in order to provide a meaningful representation of the Physical Entity.	

The Virtual Entity is, at least, a logical container for the attributes of the PE. Some VEs are passive digital artefacts, that is they have no processing power or intelligence and thus can be implemented e.g. as a set of database entries, and some are active digital artefacts (i.e. software). In this second case, the VE may contain a "plan", a set of goals, or a business-oriented workflow program (e.g. BCMP). In the RERUM case the VE may contain security or privacy policies associated to the data and functionality of the VE.

3.3.3 Term: Augmented Entity

Our terminology	**IoT-A correspondence**	**Source(s) our term is based on**
Augmented Entity (AE)	Augmented Entity	IoT-A
Description	An Augmented Entity is the composition of one Virtual Entity and the Physical Entity it is associated.	

Example: An unmanned aerial vehicle (UAV). The body of the UAV can be considered the Physical Entity, while the UAV controller is the related Virtual Entity. Together they form the Augmented Entity.

3.3.4 RERUM Term: RERUM Device (RD)

Our terminology	IoT-A correspondence	Source(s) our term is based on
RERUM Device (RD) or RERUM Smart Object (SO)	Device plus Software, and, in particular, RERUM SW.	The concept is closely related to the "Smart Object", as discussed in Chapter 3.5 of the IERC Book [70]. It is related to IoT-A's Device and IoT-A's On-Device Resources, but RERUM additionally requires certain Software to implement RERUM's reliability, efficiency, security, and privacy mechanisms.
Description	A **RERUM Device (RD**) is a piece of hardware and software (incl. the Operating System) that is equipped with processing power and communication capabilities. In this way, a RERUM Device becomes a bi-directional communicating intelligent object, which observes its environment and is able to make decisions depending on the application it serves and based on the information extracted from the physical world. It has software (in terms of Resources) that enables the RD to provide interpreted and pre-processed sensory data or to read and interpret the commands that are given to it (when it has the role of an actuator). These Resources (corresponding to IoT-A's "on-device" resources) are presented such that they can be consumed by IoT-A's Resource Level Services. The RERUM Device has some sensing or acting elements directly attached to it. Using these, a RERUM Device is able to sense from or to act upon one or more physical entities that it will be associated with.	

Example: Zolertia's Z1 hardware, running RERUM's software and having on board a sensing device is monitoring the temperature of the air in the living room. Like usual devices, the electrical signal from an attached sensor is transformed into a temperature. Simple software would allow 'building' out of the raw electrical readings several resources, for instance one providing the current temperature (with attribute or metadata to be able to transform the reading to i.e. Celsius degrees or Fahrenheit), and a second one providing the average over a given past period of time.

Here the Z1 together with the all the software that enables those values to be collected over a network comprise the RERUM Device. The RERUM software will enable the Z1 hardware to bootstrap a network connection and will allow performing security and privacy enhancing calculations on the raw data emitted from the sensing elements, e.g. encrypt the actual temperature or compress several measurements over time in order to minimize energy consumption due to multiple transmissions.

3.3.5 RERUM Term: Mobile Phone used as a RERUM Device

Our terminology	IoT-A correspondence	Source(s) our term is based on
Mobile Phone used as a RERUM Device	none	Not based on, but probably related to the "smart mobile" from BUTLER.
Description	The mobile phone as a whole or on its own is a special type of a RERUM Device. It offers to run software (usually called an App) and has sensing elements and actuating elements. Hence, by running an application that implements the software necessary for making it a RERUM Device, called a RERUM App, the mobile phone offers the hardware and operating system and gives access to attached sensing elements to allow it being instrumented by the RERUM App as a RERUM Device.	

Example: Consider an Android smart phone of the latest technology. The smart phone has on board various sensors (examples have been given in RERUM deliverable D2.1 in the smart transportation use case). Furthermore, the smart phone has an operating system (Android) and can host specialised software. Thus, the mobile phone has all the necessary characteristics of a RERUM Device. For example, in the smart transportation use case (see deliverable D2.1) the smart phone can play the role of the sensor (sending speed measurements to the application server) and at the same time receive alarms, or traffic updates to show them to the user. That way, the smart phone plays the role of the IoT-A' Device having on board On-Device Resources, exposing Resource-level Services and hosting Active Digital Artefacts (the applications). However, exactly due to the fact that the smart phone can host software, it can host various other non-RERUM user applications, Services and Resources. Due to this high complexity of the smart phone software and operating system, we consider it as a separate type of RERUM Device.

3.3.6 RERUM Term: Resource on a RERUM Device

The IoT-A project proposed a largely agreed upon model describing the relationship between conceptual entities involved in the IoT. The introduced key concepts consider the landscape of IoT as being populated by Physical Entities, Virtual Entities, Resources and Services.

As described in Section 3.2, in terms of IoT-A concepts, a RERUM Device connects the actual electrical signals resulting from interactions in the physical real world with a Resource.

Our terminology	IoT-A correspondence	Source(s) our term is based on
Resource (on a RERUM Device)	On Device Resource	IoT-A
Description	Resources are software components that provide information about, or that enable the actuation on Physical Entities. They offer a software interface such that RERUM's Services, which reside in Virtual RERUM Entities, can consume them. Note, RERUM only considers what IoT-A calls On-Device Resources, which are on the physical Device and not so called Network Resources.	

Example: Consider a sensor platform that has on board a sensing element able to measure the temperature in a room. The sensing element (as described in Section 3.2) transmits to the sensor platform an analogue signal containing the temperature measurements. The analogue signal is transformed into digital data within the sensor platform. The Resource on the RERUM Device is the software that is able to provide the temperature measurements to the outside world. It may be a simple software code that gets the raw temperature data from the DSP and transforms it to a readable format, with some attributes (denoting i.e. the metric system of the value).

3.3.7 RERUM Term: RERUM Service offered by a RERUM Device

Our terminology	**IoT-A correspondence**	**Source(s) our term is based on**
RERUM Service	Resource-level Service	IoT-A
Description	RERUM Services expose the functionality, usually that of a RERUM Device, by accessing its hosted Resources. RERUM Services refer to a single Resource. In addition to exposing the Resource's functionality, they deal with quality aspects, such as dependability, security (e.g., access control), resilience (e.g., availability) and performance (e.g., scalability, timeliness). Note that for RERUM a Resource always resides on the RERUM device, hence a RERUM Service cannot access a network Resource, i.e. in IoT these are Resources that do not necessarily reside on a Device in the sense of the IoT Domain Model, but can also be hosted somewhere else.	

Example: Services are the mechanism that allow to the outside IoT world to access the information of a Resource on a RERUM Device. In the previous temperature measurement example, a RERUM Service can be a low-level service (either REST or SOAP based) that can be accessed by an application to get the temperature measurement from the RERUM Device. Thus, the RERUM Service is the interface of the RERUM Device (and its hosted Resource) with the IoT application world.

3.3.8 RERUM Term: Virtual RERUM Device (VRD)

Our terminology	**IoT-A correspondence**	**Source(s) our term is based on**
Virtual RERUM Device (VRD) or Virtual RERUM Smart Object (SO)	none	Related, but not based on iCORE VO or Butler's SO
Description	A Virtual RERUM Device (RD) is a virtual representation of a RERUM Device in the digital world. One RERUM Device at one time is represented by one Virtual RERUM Device. This is a software artefact, like a Virtual Entity (VE), but represents a RERUM Device (RD).	

Example:

For sake of simplicity RERUM limits itself to a one-to-one representation of a RERUM Device to a Virtual RERUM Device. This simply considers the virtual device as a virtualization of one physical hardware plus the software running on that piece of hardware; this may allow RERUM to still keep the abstraction near to one real hardware, which is helpful for privacy, as data and functionality can remain close the data subject and are not necessarily hosted centrally.

3.3.9 RERUM Term: Federation of Virtual RERUM Devices (VRD-Federation)

Our terminology	**IoT-A correspondence**	**Source(s) our term is based on**
Federation of Virtual RERUM Devices	none	Can be considered similar to the Composite Virtual Object of iCORE.
Description	A grouping of Virtual RERUM Devices is called a "VRD federation" if they cooperate to offer an IoT-service for monitoring or controlling a Physical Entity (PE). Furthermore, the RD federation can be considered as a service composition between the RERUM Services of the VRDs for accessing/changing the attributes of a VE. The logic necessary to orchestrate the service is associated to the Virtual Entity that offers the service.	

Example: Consider the case of automatic air conditioning in a room. There we have a plethora of RERUM Devices: a temperature sensor inside the room, a temperature sensor at the balcony outside the room, a window actuator and an air-condition controller. The RERUM Devices have some Resources that can provide temperature measurements inside and outside the room, "opening or closing the window" and "switching on/off the air conditioning" respectively. The respective VRDs are also exposing respective RERUM Services for accessing the aforementioned Resources of the RDs. These RDs can form a VRD Federation in order to keep the room temperature within 23 and 25 degrees. When the temperature comes over 25 degrees and the outside temperature is higher, then the air-conditioning controller switches the air-condition to "ON". If the outside temperature was low the window controller would open the window. All these functions are performed automatically with one RD directly accessing the RERUM Service of the other RDs.

3.3.10 RERUM Term: Generic Virtual RERUM Object (VRO)

Based on general requirements listed in the beginning of section 3, below are listed the characteristics considered to be relevant for all virtual instances.

Our terminology	**IoT-A correspondence**	**Source(s) our term is based on**
Generic Virtual RERUM Object (VRO)	none	new
Description	In RERUM we subsume the two different virtualizations into the class of Generic Virtual RERUM Objects (VROs): a VRO is either a Virtual Entity or a Virtual RERUM Device, as they are both representations in the digital world and hence are both members. Regarding the middleware they have common functionality or interfaces that allows them to • be discovered,	

	• be addressed, • be interacted with in a standardized manner.

The reason for this is to allow *certain* functionality or properties of the virtualized representations to be described uniformly in a way that the middleware can access consistently. Of course, there is also some functionality where Virtual RERUM Devices and Virtual Entities fundamentally differ.

Examples for functionality or properties that are valid for both virtualisations are:

- Association to a physical entity: The object must relate itself to a real physical entity. The object's actions will have a real world impact, either on the physical entity or the physical device.
- Connectivity: The object must be able to communicate, e.g., the status of the physical entity.
- Addressability and discoverability
- Functional description (including in/outs, restrictions on functional side, associated computation)
- Restrictions (SLA: response delay, maximum users, messages, ETA etc.)
- Reference a memory (log) of own data and actions
- Federation model and composed VEs – the features to be added in the model to enable its use in a federation compute/process capability

3.3.11 RERUM Model including Normal and Administrative Users

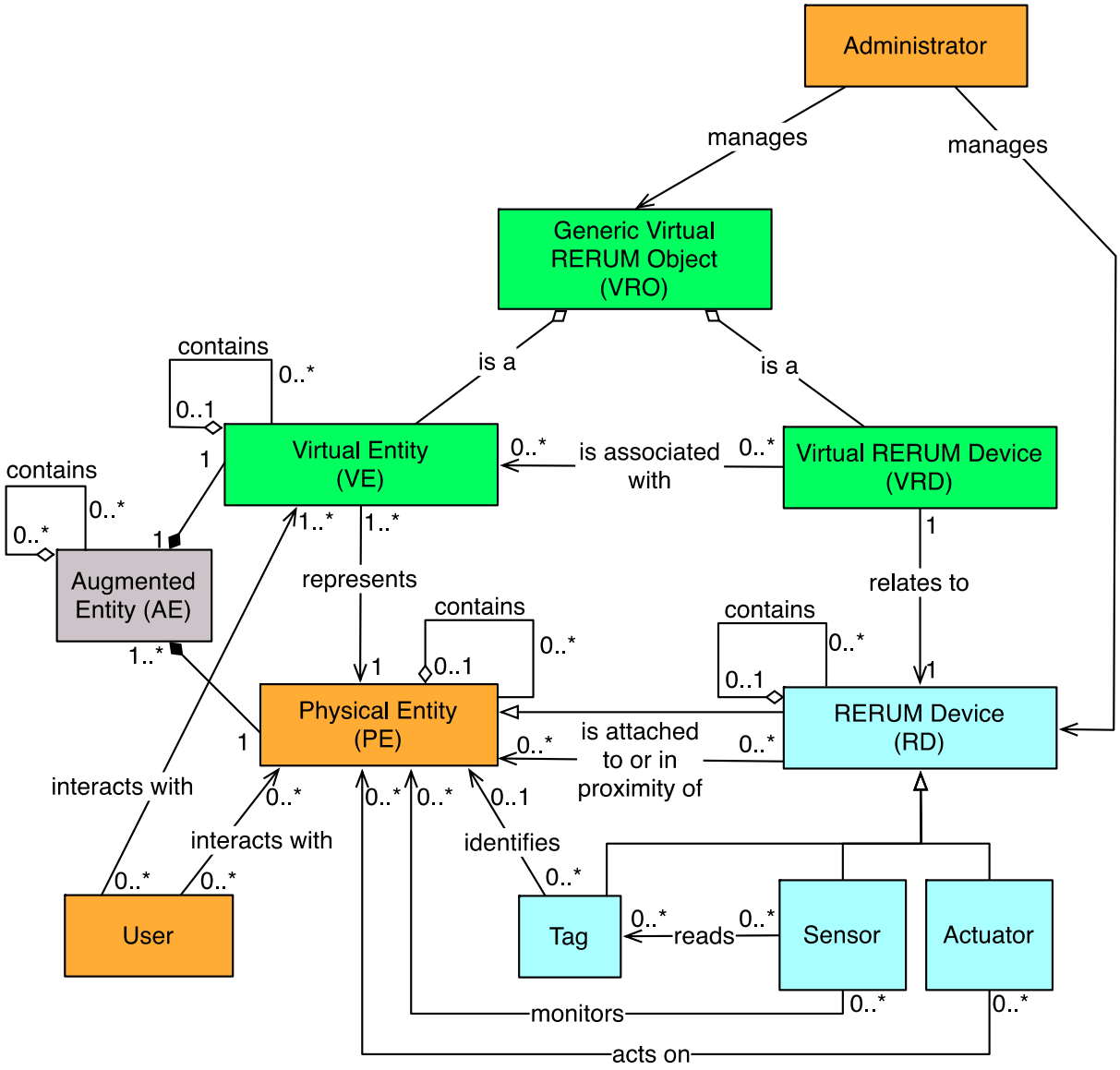
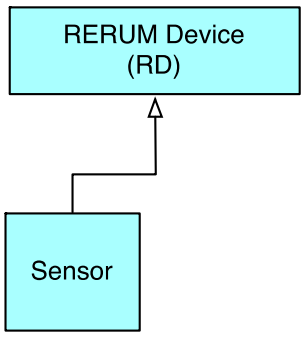


Figure 13 - UML style representation of the IoT Domain Model of RERUM

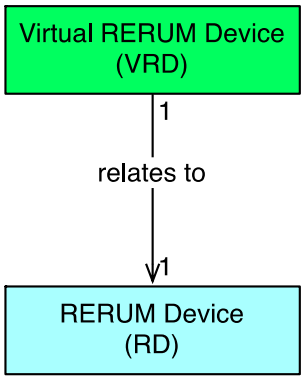
Figure 13 was adapted and extended from the IoT-A domain model [55] and depicts the interplay between all components described in the previous sections. The UML representation further displays the relationships between them. The figure is using UML in the following ways:



Hollow triangle line head:

The line that ends in a hollow triangle is used in IoT-A to depict a “is-a” relationship, meaning that a Sensor is a RERUM Device. Also meaning, that this must not be interpreted as a subclass.

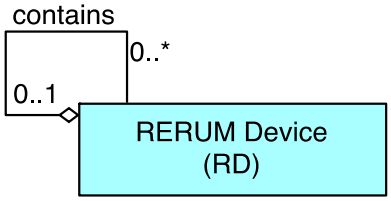
A Sensor is able to monitor characteristics of the physical entity; to do so reliably, securely, trustworthy and privacy preserving it is a RERUM Device and as mentioned in the previous subsection, the RERUM Device will use Sensing Elements to get sensory data.



Open arrow line head:

The line that ends in an open arrow and has cardinalities written at both ends depicts that the two concepts are associated. The cardinalities are used as follows: From the source of the relation read the cardinality at the target to give you the multiplicity (or singularity as in the case depicted between VRD and RD) with which the source can be in the relation with the target. The text given is for descriptive purposes. For the inverse relation the cardinality at the source is relevant.

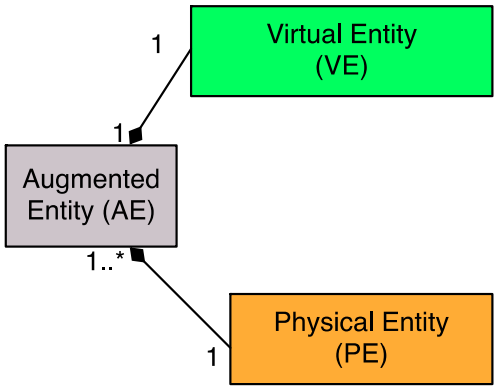
In the example to the left, this means that one VRD is always related to exactly one RD. So for the inverse relation that means that looking for a virtual entity for one given RD you will be able to find exactly one VRD.



Hollow diamond line head:

The line ending in a hollow diamond depicts an aggregation, meaning that the concept at the target of the relation can contain the concept at the source. It is used often to indicate that a concept can be an aggregation of several elements of itself.

So for example a RERUM Device can contain other RERUM Devices.



Filled diamond line head:

The line ending in a filled diamond depicts a composition, which is similar, but not equal to aggregation. The composition indicates that the composed concept only exists if the target concepts are available.

The example here is the concept of an augmented entity; the concept is at the target and we only speak of an augmented entity if we have both the PE and the related VE together.

Colours:

Note that the colour coding is as follows: virtual abstractions are green; concepts that involve software and hardware are blue; the augmented entity is grey. Finally, concepts that fit into either multiple or none of the before mentioned categories are orange.

Reading the figure is best started with the relationships surrounding the Physical Entity (PE).

A Physical Entity has relations to users that interact with the PE. RERUM Devices (RDs) actuate on or monitor the PE. VE is the virtual representation of the PE.

Physical entities can also contain other physical entities, which seems obvious in the case of an identifiable package containing other identifiable packages. This is shown in the self-containing aggregation relationship of PEs of none to many.

The relation to users interacting with the Physical Entity is shown by an association of zero to many in both directions. This means that one Physical Entity can have none or multiple Users interacting with it and also that there can be Users even if they not yet interact with physical entities. Note that users are not necessarily only humans. While the IoT-A leaves the modelling of different Users with

different roles to concrete architectures, RERUM specialises and describes Administrators as separate from Users. Hence, we model an extra explicitly mentioned Administrator. This does not restrict further modelling of different users, e.g., Adam, Alice, and Bob, within the concept of Users.

The Physical Entity, if attached to the RERUM ecosystem, is represented by at least one Virtual Entity. This is shown as a one-or-many to one association. The Virtual Entity and the Physical Entity keep a close and coherent relation, i.e., that every change in either entity must be updated in the other in a cyber or a physical form. This is again shown as a conceptual composition of both of them to form what IoT-A describes as the Augmented Entity (AE). Following IoT-A the AE is what actually enables one or more everyday objects (PEs) to become part of digital processes by composing them with the Virtual Entity that is associated to them. This relation highlights that the concepts of PE and VE belong together and keep a close and coherent relation.

A Physical Entity is monitored by sensors or changed by actuators or a sensor can identify a PE by means of a tag. A PE can exist without their relation, as shown by the associations zero to many of sensors, actuators and tags.

Sensors and actuators are RERUM Devices that have embedded sensing and acting elements. Tags can also be seen as a RERUM Devices. Note that tags in the sense of current RFIDs are special since RERUM objectives are focused on devices who own certain on device computing power; the model described aims to be relevant for sensors and actuators. For example a RERUM device that has several embedded sensing elements interprets the electrical signals and hence is a Sensor. A RERUM device can then be related to PE, by attaching it or bringing it into close proximity; by doing so the RD extends the PE and allows it to become part of the digital world, as described in IoT-A. Note that a single RERUM Device, since it can contain multiple sensing elements and acting elements, may be associated with more than one PE. This potential multiple “relation” of one RD and several PEs is then transformed into a loose coupling by that one RD becoming multiple Virtual RERUM Devices (VRDs).

RERUM Devices are represented in a digital world as Virtual RERUM Device (VRDs). As with PEs, one VRD relates to exactly one RD, as it is depicted in the figure with the one to one association between RD and VRD.

Whenever there is no need to differentiate between Virtual RERUM Devices and Virtual (Physical) Entities we will use the term Generic Virtual Object (GVO). This is illustrated by the aggregation relationship between both virtual representations and the GVO. And this means that a GVO is either a VE or a VRD.

The Administrator seen in the top of the figure is in charge of managing the configurations, interactions, and overall behaviour of RERUM Devices and their relation to other virtual objects. This is depicted by the “manage” association between him, RD and VROs. His role will be discussed in detail in the coming architectural discussion within the next deliverable.

4 Conclusions

The purpose of this deliverable is to give the basis for the design of the RERUM system architecture. This deliverable is the bridging point between the definitions of the use cases that are under consideration within RERUM, and the definition of the system architecture. Its goal is twofold: to define the detailed RERUM system requirements, and then to present a first version of the software model of the RERUM Device. RERUM has adopted a use-case oriented approach, meaning that the use-cases under consideration play a central role for the system design and implementation. The four RERUM use cases (two outdoor and two indoor) were presented in detail and analysed in terms of their vulnerabilities in Deliverable D2.1. This deliverable builds upon the results of D2.1 and analyses the use cases in terms of system requirements. However, RERUM does not aim to develop a system focused specifically on these use cases. In that case, the system would have been very specialised without taking into consideration the extreme opportunities and the plethora of applications of the IoT domain. On the contrary, RERUM aims to have a more generic approach: after analysing the specific requirements of its use cases, they are generalised to cover a broader, more vertical area of applications and the resulted requirements are presented in this deliverable.

In more detail, this deliverable presents a comprehensive list of Functional and non-Functional technical requirements that will be used for driving the design of the architectural reference model. The functional requirements are focused on the technical areas of interest of the RERUM system and are split into six major categories:

- system architecture requirements: this category includes the high-level non-functional requirements of the system, i.e. “privacy-by-design”, reliability, robustness, availability, etc.
- application requirements: this category describes the functional requirements that derive from the application that will be supported by the RERUM system.
- networking and QoS requirements: this category includes the low-layer functional networking requirements for the interconnectivity of the devices that are connected within the IoT domain
- RERUM Device Software, Hardware and Modelling requirements: this category includes the functional requirements for the devices in terms of hardware capabilities, software and their modelling in terms of abstract representation in the IoT layer.
- Middleware and Virtualization requirements: this category includes the requirements for the development of the middleware that will virtualise the devices and the physical entities, so that they can be homogeneously represented to be used by the applications
- Security and Privacy requirements: this category includes the functional requirements that have been extracted by the threat analysis presented in deliverable D2.1 and will be used for developing mechanisms for addressing the concept of security and privacy by design in the RERUM system.

At this point it has to be noted that the requirements presented in this document were not only extracted from the RERUM use cases. In order to give a more complete view of the requirements, other sources (i.e. EU projects, standardization organisations, RFCs, etc.) have been taken into consideration. Especially for adapting the RERUM system to the European laws in terms of privacy, a set of respective requirements extracted from popular privacy frameworks and directives (i.e. ISO/IEC 29100 [8]) have been described in this deliverable. This indeed shows that RERUM acknowledges the importance of privacy in the IoT world, and aims to ensure the compliance of the IoT applications it will support, with the EU laws.

The RERUM requirements analysed in Section 2 of this deliverable will be used for the development of the system architecture and for the technical work to be done in the respective work packages. A first step towards this approach is described in Section 3 by presenting a first attempt to provide a software model for the IoT-view of the RERUM Devices. This model has been split in two parts. At first, the low level model with the common software structure at the lower layers is presented. Then, follows a higher layer presentation of the components/models of the software abstraction (virtual representation) of the RERUM Device. This model of a Virtual RERUM Device (VRD) is actually a homogeneous way to represent the RERUM Devices to the applications that want to use them and access their data to facilitate the virtual representations of the Physical Entities. The VRDs will be able to communicate with each other, and the VRDs will be able to communicate with the applications in order to serve the user needs. The VRDs will also be able to form federations, which are combinations of VRDs in order to serve a specific functionality/service. In this deliverable, we also present the terminology adopted within RERUM with regards to the VRD model, after careful consideration and extensive research on the terminology of IoT-A, and other influential IoT projects. This model of the VRD will be extended and analysed in more detail in an upcoming deliverable regarding the RERUM architecture (D2.3), where we will analyse the internal components of the VRD and their interactions. If needed, a more extensive and updated version of the model will be delivered with the deliverable D2.4 presenting the basic functions of the RERUM middleware.

References

[1] Merriam-Webster Online: Dictionary and Thesaurus www.merriam-webster.com

[2] IoT-A project, D1.5 - Final Architectural Reference Model for the IoT, July, 2013.

[3] Standard, Federal. "1037C." Department of Defence Dictionary of Military and Associated Terms in support of MIL-STD-188 (1996).

[4] http://www.alcatel-lucent.com/eco/low-carbon/travel_less.html

[5] Dirks, S., Gurdgiev, C., & Keeling, M. (2010). Smarter Cities for Smarter Growth: How Cities Can Optimize Their Systems for the Talent-Based Economy. IBM Institute for Business Value

[6] Zhexuan Song; Cárdenas, A.A.; Masuoka, R., "Semantic middleware for the Internet of Things," *Internet of Things (IOT), 2010* , vol., no., pp.1,8, Nov. 29 2010-Dec. 1 2010 doi: 10.1109/IOT.2010.5678448

[7] Luis Roalter, Matthias Kranz, Andreas Möller, " A Middleware for Intelligent Environments and the Internet of Things", Ubiquitous Intelligence and Computing, Lecture Notes in Computer Science Volume 6406, 2010, pp 267-281

[8] Information Technology Task Force (ITTF)." Information technology -- Security techniques -- Privacy framework". International Standard ISO/IEC 29100. Stage: 60.60 (2011-12-05). TC/SC: ISO/IEC JTC 1/SC 2. Status: published.

[9] A. Osborne, *An Introduction to Microcomputers Volume 1: Basic Concepts*, Osborne-McGraw Hill Berkeley California USA, 1980

[10] A. Osseiran, Afif, et al. "The Foundation of the Mobile and Wireless Communications System for 2020 and Beyond: Challenges, Enablers and Technology Solutions." Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th. IEEE, 2013.

[11] Abbasi, Ameer Ahmed, and Mohamed Younis. "A survey on clustering algorithms for wireless sensor networks." *Computer communications* 30.14 (2007): 2826-2841.

[12] Aberer, Karl, and Manfred Hauswirth. "Middleware support for the" Internet of Things"." (2006).

[13] Akyildiz, I.F.; Won-Yeol Lee; Vuran, Mehmet C.; Mohanty, S., "A survey on spectrum management in cognitive radio networks," *Communications Magazine, IEEE* , vol.46, no.4, pp.40,48, April 2008

[14] Akyildiz, Ian F., et al. "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: a survey." *Computer Networks* 50.13 (2006): 2127-2159.

[15] Almalkawi, Islam T., Manel Guerrero Zapata, and Jamal N. Al-Karaki. "A cross-layer-based clustered multipath routing with QoS-aware scheduling for wireless multimedia sensor networks." *International Journal of Distributed Sensor Networks* 2012 (2012).

[16] Angelakis, Vangelis, et al. "Probabilistic Routing Schemes for Ad Hoc Opportunistic Networks." *Routing in Opportunistic Networks*. Springer New York, 2013. 209-222.

[17] ANSI/IEC 60529-2004, "Degrees of Protection Provided by Enclosures (IP Code)", International Electrotechnical Commission, Edition 2.2, 2013

[18] Ardagna, Danilo, and Barbara Pernici. "Adaptive service composition in flexible processes." *Software Engineering, IEEE Transactions on* 33.6 (2007): 369-384.

[19] Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." *Computer networks* 54.15 (2010): 2787-2805.

[20] Bandyopadhyay, Debasis, and Jaydip Sen. "Internet of things: Applications and challenges in technology and standardization." *Wireless Personal Communications* 58.1 (2011): 49-69.

[21] Baronti, Paolo, et al. "Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards." *Computer communications* 30.7 (2007): 1655-1695.

[22] Bassi, Alessandro, et al. "Enabling things to talk." Springer, 2013

[23] Cabric, Danijela, Artem Tkachenko, and Robert W. Brodersen. "Experimental study of spectrum sensing based on energy detection and network cooperation." *Proceedings of the first international workshop on Technology and policy for accessing spectrum*. ACM, 2006.

[24] Cavoukian, Ann, and Marc Chanliau. "Privacy and Security by Design: A Convergence of Paradigms." *Ontario, Canada: Office of the Privacy Commissioner (Ontario)* (2013).

[25] Chung, Lawrence, et al. "Non-functional requirements." *Software Engineering* (2000).

[26] Compton & all, Web Semantics: Science, Services and Agents on the World Wide Web, Elsevier, 2012

[27] D. Guinard, et al. "Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services." Services Computing, IEEE Transactions on 3.3 (2010): 223-235.

[28] D3.1. Virtual Object Requirements and Dependencies", iCore FP7 project deliverable D3.1,

[29] Di Pietro, Roberto, et al. "Posh: Proactive co-operative self-healing in unattended wireless sensor networks." *Reliable Distributed Systems, 2008. SRDS'08. IEEE Symposium on*. IEEE, 2008.

[30] Directive 2004/22/EC of the European Parliament and of the Council of 31 March 2004 on measuring instruments (Text with EEA relevance). http://eur-lex.europa.eu/legal-content/en/ALL/;jsessionid=F8JFTFwNvGhfJFDxpdKrl74Z0YQm2sTqlJhhqsPqy1jPXGmBSwKs!770729252?uri=CELEX:32004L0022, last accessed on 28.05.2014.

[31] Directive 95/46/EC of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data and on the free movement of such data.http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML, last accessed on 28.05.2014.

[32] Dirks, S., Gurdgiev, C., & Keeling, M. (2010). Smarter Cities for Smarter Growth: How Cities Can Optimize Their Systems for the Talent-Based Economy. IBM Institute for Business Value

[33] G. Montenegro, N. Kushalnagar, J. W. Hui, and D. E. Culler: "Transmission of IPv6 packets over IEEE 802.15.4 networks," RFC 4944, Sep. 2007.

[34] Gama, Kiev, Lionel Touseau, and Didier Donsez. "Combining heterogeneous service technologies for building an Internet of Things middleware." *Computer Communications* 35.4 (2012): 405-417.

[35] GCC ARM Embedded in Launchpad https://launchpad.net/gcc-arm-embedded

[36] GNU LD Linker Scripts: https://sourceware.org/binutils/docs-2.24/ld/Scripts.html#Scripts

[37] Haykin, Simon. "Cognitive radio: brain-empowered wireless communications." *Selected Areas in Communications, IEEE Journal on* 23.2 (2005): 201-220.

[38] http://www.alcatel-lucent.com/eco/low-carbon/travel_less.html

[39] http://www.bpmn.org/

[40] http://www.fi-ware.org/

[41] http://www.iot-butler.eu/

[42] http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/

[43] https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.Context-based_security_and_compliance#PRRS_Framework

[44] https://www.ict-rerum.eu/

[45] I. F. Akyildiz, et al., ”Spectrum management in cognitive radio ad hoc networks,” IEEE Network, vol.23, no.4, pp.6-12, July 2009.

[46] iCORE – Internet Connected Objects for Reconfigurable Ecosystems, Deliverable D2.3

[47] IETF: “Draft Charter V0.9c - Authentication and Authorization for Constrained Environment (ACE) - http://trac.tools.ietf.org/wg/core/trac/wiki/ACE_charter

[48] IETF: “DTLS In Constrained Environments (DICE)” - http://datatracker.ietf.org/wg/dice/charter/

[49] IETF: “Internet Security Glossary, Version 2” - http://www.rfc-editor.org/rfc/rfc4949.txt, last accessed on 28.05.2014.

[50] IETF: “Proportional Rate Reduction for TCP” - http://tools.ietf.org/html/rfc6937, last accessed on 28.05.2014.

[51] Information Technology Task Force (ITTF). “Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Integrity framework”. International Standard ISO/IEC 10181-6:1996. Stage: 90.93 (2006-09-20). TC/SC: ISO/IEC JTC 1. Status: published.

[52] Information Technology Task Force (ITTF). ”Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture”. International Standard ISO/IEC 7498-2:1989. Stage: 90.93 (2000-06-21). TC/SC: ISO/IEC JTC 1. Status: published.

[53] Information Technology Task Force (ITTF). ”Information technology -- Security techniques -- Privacy framework”. International Standard ISO/IEC 29100. Stage: 60.60 (2011-12-05). TC/SC: ISO/IEC JTC 1/SC 2. Status: published.

[54] Institute of Electrical and Electronics Engineers: "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)", IEEE Std 802.15.4-2006, 2006

[55] Internet of Things – Architecture (IoT-A), Deliverable D1.5 – Final architectural reference model for the IoT v3.0.

[56] IoT-A project, D1.5 - Final Architectural Reference Model for the IoT, July, 2013.

[57] Iyengar, S. Sitharama, and Richard R. Brooks, eds. *Distributed Sensor Networks: Sensor Networking and Applications*. CRC press, 2012.

[58] J. Matamoros and C. Anton-Haro. "Traffic Aggregation Techniques for Environmental Monitoring in M2M Capillary Networks." *Vehicular Technology Conference (VTC Spring)*, 2013 IEEE 77th. IEEE, 2013.

[59] J. Mitola III and G. Maguire Jr., “Cognitive radio: making software radios more personal,” IEEE Personal Communications, vol.6, no.4, 1999.

[60] L. Adams, “CAPITALIZING ON 802.11FOR SENSOR NETWORKS. Low-Power Wireless Sensor Networks”, whitepaper, GAINSPAN CORPORATION, U.S.A.

[61] Luis Roalter, Matthias Kranz, Andreas Möller, " A Middleware for Intelligent Environments and the Internet of Things", *Ubiquitous Intelligence and Computing*, Lecture Notes in Computer Science Volume 6406, 2010, pp 267-281

[62] M.H. ter Beek, A. Bucchiarone, and S. Gnesi, Formal Methods for Service Composition. *Annals of Mathematics, Computing & Teleinformatics* 1, 5 (2007), 1 - 10.

[63] Mehra, P. 2012. Context-Aware Computing: Beyond Search and Location-Based Services. *IEEE Internet Computing*, 16, 2 (March-April, 2012), 12-16

[64] Merriam-Webster Online: Dictionary and Thesaurus www.merriam-webster.com

[65] Micron: "NAND Flash 101: An Introduction to NAND Flash and How to Design It In to Your Next Product", Technical Note 29-19, 2006

[66] Misra, Sudip, and Ankur Jain. "Policy controlled self-configuration in unattended wireless sensor networks." *Journal of Network and Computer Applications* 34.5 (2011): 1530-1544.

[67] NXP Semiconductors: "I2C-bus specification and user manual", Rev. 6, 2014

[68] O. B. Akan, O. Karli, O. Ergul, "Cognitive radio sensor networks," IEEE Network, vol.23, no.4, pp.34-40, July- 2009.

[69] O. Vermesan, etal., "Internet of Things Strategic Research Roadmap", IERC Whitepaper, 2011.

[70] Ovidiu Vermesan and Peter Friess, 2013. The IERC Book. Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems, River Publishers Series In Communications, available at www.riverpublishers.com.

[71] P. Ilia, G. Oikonomou, T. Tryfonas: "Cryptographic Key Exchange in IPv6-Based Low Power, Lossy Networks", in *Proc. Workshop in Information Theory and Practice (WISTP 2013)*, ser. Lecture Notes in Computer Science, 7886, pp. 34-49, 2013

[72] P. Levis, E. Brewer, D. Culler, D. Gay, S. Madden, N. Patel, et al: The emergence of a networking primitive in wireless sensor networks. Communications of the ACM, 51(7), 2008

[73] Ranganathan, Kumar. "Trustworthy pervasive computing: The hard security problems." P*ervasive Computing and Communications Workshops*, 2004. Proceedings of the Second IEEE Annual Conference on. IEEE, 2004.

[74] S. De, et al. "Service modelling for the Internet of Things." *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, 2011.

[75] Schmidt, Heinz. "Trustworthy components—compositionality and prediction." *Journal of Systems and Software* 65.3 (2003): 215-225.

[76] Serial Peripheral Interface: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

[77] Sharon, Guy, and Opher Etzion. "Event processing network–a conceptual model." *Proceedings of VLDB, Second International Workshop on Event Driven Architecture and Event Processing Systems*. Citeseer. 2007.

[78] Standard, Federal. "1037C." Department of Defence Dictionary of Military and Associated Terms in support of MIL-STD-188 (1996).

[79] Tan, Lu, and Neng Wang. "Future internet: The internet of things." *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*. Vol. 5. IEEE, 2010.

[80] Texas Instruments: "A Powerful System-On-Chip for 2.4-GHz IEEE 802.15.4, 6LoWPAN and ZigBee Applications", CC2538 Datasheet, SWRS096A, 2013

[81] Texas Instruments: "CC2538 System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee®/ZigBee IP® Applications", Version C, SWRU319C, 2013

[82] Tragos, E., et al. "Spectrum assignment in cognitive radio networks: A comprehensive survey." (2013): 1-28.

[83] Tragos, Elias Z., et al. "The impact of interference on the performance of a multi-path metropolitan wireless mesh network." *Computers and Communications (ISCC), 2011 IEEE Symposium on*. IEEE, 2011.

[84] V. Michopoulos, L. Guan, G. Oikonomou, I. Phillips:, "DCCC6: Duty Cycle-Aware Congestion Control for 6LoWPAN Networks", in *Proc. 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, Lugano, Switzerland, pp. 278-283, 2012

[85] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen and M. Welsh, "Simulating the Power Consumption of Large-Scale Sensor Network Applications" Division of Engineering and Applied Sciences, Harvard University

[86] Veltman, Kim H. "Syntactic and semantic interoperability: new approaches to knowledge and the semantic web." *New Review of Information Networking* 7.1 (2001): 159-183.

[87] Vermesan, Ovidiu, and Peter Friess. Internet of Things-Global Technological and Societal Trends From Smart Environments and Spaces to Green ICT. River Publishers, 2011.

[88] Vermesan, Ovidiu, et al. "Internet of things strategic research roadmap." *Internet of Things-Global Technological and Societal Trends* (2011): 9-52.

[89] W. Chen and I. Wassell, "Energy efficient signal acquisition via compressive sensing in wireless sensor networks," in *Proc. of ISWPC*, 2011.

[90] Wei, Chuyuan, and Yongzhen Li. "Design of energy consumption monitoring and energy-saving management system of intelligent building based on the internet of things." *Electronics, Communications and Control (ICECC), 2011 International Conference on*. IEEE, 2011.

[91] www.ict-iotest.eu

[92] www.iot-a.eu

[93] www.iot-icore.eu

[94] Yen-Kuang Chen, "Challenges and opportunities of internet of things," *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, vol., no., pp.383,388, Jan. 30 2012-Feb. 2 2012

[95] Yet Another Flash File System 2 (YAFFS2): http://www.yaffs.net/

[96] Yigitel, M. Aykut, Ozlem Durmaz Incel, and Cem Ersoy. "QoS-aware MAC protocols for wireless sensor networks: A survey." *Computer Networks* 55.8 (2011): 1982-2004.

[97] Younis, Mohamed, et al. "Topology management techniques for tolerating node failures in wireless sensor networks: A survey." *Computer Networks* (2013).

[98] Yucek, Tevfik, and Hüseyin Arslan. "A survey of spectrum sensing algorithms for cognitive radio applications." *Communications Surveys & Tutorials, IEEE* 11.1 (2009): 116-130.

[99] Zhexuan Song; Cárdenas, A.A.; Masuoka, R., "Semantic middleware for the Internet of Things," *Internet of Things (IOT), 2010*, vol., no., pp.1,8, Nov. 29 2010-Dec. 1 2010 doi: 10.1109/IOT.2010.5678448

[100] Zhexuan Song; Cárdenas, A.A.; Masuoka, R., "Semantic middleware for the Internet of Things," *Internet of Things (IOT), 2010* , vol., no., pp.1,8, Nov. 29 2010-Dec. 1 2010 doi: 10.1109/IOT.2010.5678448

[101] ZigBee Alliance: ZigBee Specification. 2004.