Project Acronym:	**VICINITY**
Project Full Title:	**Open virtual neighbourhood network to connect intelligent buildings and smart objects**
Grant Agreement:	**688467**
Project Duration:	**48 months (01/01/2016 - 31/12/2019)**

Deliverable D1.5

VICINITY technical requirements specification

Work Package:	**WP1 – VICINITY concept Requirements, Barriers, Specification and Architecture**
Task(s):	**T1.4 – Functional & Technical Specification, Architectural design**
Lead Beneficiary:	**BVR**
Due Date:	**31 December 2017 (M12)**
Submission Date:	**31 December 2017 (M12)**
Deliverable Status:	**Draft**
Deliverable Type:	**R**
Dissemination Level:	**PU**
File Name:	**VICINITY_D1_5_Technical_requirements_specification_1.0.pdf**

European Commission
Horizon 2020
European Union funding
for Research & Innovation

IoT European Platforms Initiative

VICINITY Consortium

No	Beneficiary		Country
1.	TU Kaiserslautern (Coordinator)	UNIKL	Germany
2.	ATOS SPAIN SA	ATOS	Spain
3.	Centre for Research and Technology Hellas	CERTH	Greece
4.	Aalborg University	AAU	Denmark
5.	GORENJE GOSPODINJSKI APARATI D.D.	GRN	Slovenia
6.	Hellenic Telecommunications Organization S.A.	OTE	Greece
7.	bAvenir s.r.o.	BVR	Slovakia
8.	Climate Associates Ltd	CAL	United Kingdom
9.	InterSoft A.S.	IS	Slovakia
10.	Universidad Politécnica de Madrid	UPM	Spain
11.	Gnomon Informatics S.A.	GNOMON	Greece
12.	Tiny Mesh AS	TINYM	Norway
13.	HAFENSTROM AS	HITS	Norway
14.	Enercoutim – Associação Empresarial de Energia Solar de Alcoutim	ENERC	Portugal
15.	Municipality of Pylaia-Hortiatis	MPH	Greece

Authors List

Leading Author (Editor)			
Surname	**First Name**	**Beneficiary**	**Contact email**
Oravec	Viktor	BVR	viktor.oravec@bavenir.eu

Co-authors (in alphabetic order)				
No	**Surname**	**First Name**	**Beneficiary**	**Contact email**
1.	Sveen	Flemming	HITS	flsveen@online.no
2.	Myncasova	Aida	UNIKL	limit6715@gmail.com
3.	Tryferidis	Athanasios	CERTH	thanasic@iti.gr
4.	Paralic	Marek	IS	marek.paralic@intersoft.sk
5.	Serena	Fernando	UPM	fserena@fi.upm.es
6.	Kaggelides	Kostis	GNOMON	k.kaggelides@gnomon.com.gr
7.	Vanya	Stefan	BVR	stefan.vanya@bavenir.eu

Reviewers List

List of Reviewers (in alphabetic order)				
No	**Surname**	**First Name**	**Beneficiary**	**Contact email**
1.	Wall (Coordinator)	Nigel	CAL	nw@nigel-wall.co.uk
2.	Mach	Marian	IS	Marian.Mach@tuke.sk
3.	Margariti	Katerina	CERTH	kmargariti@iti.gr
4.	Gato	Jose	Atos	jose.gato@atos.com
5.	Vinkovic	Saso	Gorenje	Saso.Vinkovic@gorenje.com

Revision Control

Version	Date	Status	Modifications made by
0.1	6. June 2016 (M6)	Initial Draft	Oravec (BVR)
0.2	7. June 2016	Overall functional specification	Oravec (BVR)
0.3	15. June 2016	Finalization of TOC	Oravec (BVR)
0.4	25. June 2016	UC0100 Description added	Oravec (BVR)
0.5	5. July 2016	UC0200 Description added	Oravec (BVR)
0.6	13. July 2016	UC0300 Description added	Oravec (BVR)
0.7	26. July 2016	UC004 Description added	Oravec (BVR)
0.8	9. August 2016	Draft of availability requirements	Oravec (BVR)
0.9	10. October	List of use cases finalized	Oravec (BVR)
0.10	30. October	Non-functional requirements drafted	Oravec (BVR)
0.11	30. November	Non-functional requirements finalized	Oravec (BVR)
0.12	6. December 2016	Quality Check	Oravec (BVR)
0.13	20. December 2016	Consolidated QaR	Oravec (BVR)
0.14	28. December 2016	Final Draft reviewed	Oravec (BVR)
1.0	30. December 2016	Submission to the EC	Oravec (BVR)

Table of Contents

List of Tables

List of Figures

List of Definitions & Abbreviations

Abbreviation	Definition
BSD	Berkeley Software Distribution
CPU	Central Processing Unit
DNS	Domain Name Service
EC	European Commission
EU	European Union
GDPR	General data protection regulation
GPL	Public License
HTTP	Hypertext transport protocol
HW	Hardware
ICT	Information and communications technology
IoT	Internet of things
ITU	United Nations specialized agency for information and communication technologies
MIT	Massachusetts Institute of Technology
PaaS	Platform as a Service
SRC	Source code
SSH	Secure shell
TLS	Transport layer security
TM	Trademark
UC	Use case
UML	Unified Mark-up Language
URI	Uniform Resource Identifier
WP	Work package

1. Executive Summary

This document "D1.5 VICINITY Technical requirements and specification" directly addresses Objective 2.4 „VICINITY Technical requirements and solution architecture specified", in terms of specifying set of required functions and quality features that will be provided by the VICINITY interoperability platform. These key required functions and quality features are defined based on stakeholders' barriers and drivers, end-user business expectations and operational needs identified from pilot site locations.

The key VICINITY functions are:

- interoperability set-up in virtual neighbourhood with value added service and IoT device granularity enabling devices owners, service providers and integrated infrastructure operators to control access to their assets;
- connecting IoT platforms into virtual neighbourhood using VICINITY Agents (provided by VICINITY out of the box for selected IoT software platforms) or VICINITY Adapters (VICINITY Open interoperability gateway API and set of libraries will be provided for proprietary or closed IoT software platforms);
- facilitation of exchange data within the virtual neighbourhood using semantic interoperability in controlled, secure and privacy preserving way;

The VICINITY quality features focus mainly on user experience, trust, privacy, security, scalability, standardization. VICINITY functions are designed around user (even technical personnel in charge of set-up integration to VICINITY) to ensure as best as possible user experience during installation, configuration, integration of VICINITY components in her infrastructure and usage VICINITY interoperability features.

The VICINITY trust, privacy and security features are mainly build on:

- supporting of various type of verifiable identities of VICINITY users, value-added service and IoT devices;
- end-to-end security and authenticity of exchanged data within neighbourhood;
- access to value-added services and IoT devices controlled by service providers and device owner;
- preserving privacy based on separation of meta-data from actual data;
- private data processing consents to control processing of these data within the neighbourhood;
- supporting privacy features introduced by "EC Regulation 2016/679", the most notably private data rectification, process restriction and erasure.

The high-availability and performance quality measures enable VICINITY to scale-up and scale-out to handle various communication load within neighbourhoods introduced by different applications from building, energy, transport and health domain.

The VICINITY maintainability features, which enable configure, extend, update and adjust VICINITY to constantly changing environment, are built upon standardization on the level of communication interfaces, communication protocols, ontologies and technologies selected, design and architecture patterns applied.

The VICINITY Technical requirements specification together architecture (see D1.6 deliverable) defines the base line for the following implementation of VICINITY components including web-based neighbourhood manager, semantic interoperability gateway, trust, security and privacy services.

2. Introduction

This document provides the technical requirements as developed within "Task 1.4 – Functional & Technical Specification, Architectural design". The architectural design will be covered by D1.6.

This document defines:

- Functional design of the VICINITY solution;
- Set of non-functional requirements and quality considerations.

This document does not define:

- Detailed design of the VICINITY solution;
- Functional requirement of value added services, integrated IoT infrastructures and IoT devices;
- Testing, validation and evaluation topics;
- Implementation management plan.

The relevant tasks (from which technical requirements specification is derived) are as follows:

- Task 1.1 – Elicitation of user requirements and barriers related to IoT interoperability;
- Task 1.2 – Pilot Sites Surveys and extraction of Use Case requirements;
- Task 1.3 – VICINITY Platform User and Business Requirement Definition;
- Task 2.1 – Analysis of available platforms, IoT infrastructures, IoT ontologies and standards.

Thus, it is suggested that readers should be familiar with and have access to the following deliverables:

- D1.1 VICINITY requirement capture framework;
- D1.2 Report on business drivers and barriers of IoT interoperability and value added services;
- D1.3 Report on pilot sites and operational requirements;
- D1.4 Report on VICINITY business requirements;
- D2.1 Analysis of Standardisation Context and Recommendations for Standards Involvement.

This document is prepared as a starting point for technical audience to understand the basic concepts of the VICINITY solution. However, the document will not provide comprehensive and detailed documentation with all the technical information. It should help technical partners within consortium during the development of the VICINITY solution. Consequently, the most important source of information will be the actual source code and its documentation (e.g. user manuals, installation guides and source code comments).

The system will be developed through repeated cycles (iterative) and in smaller portions at a time (incremental), what means that each iteration will contain part of the analysis, implementation and testing. This technical documentation will be also be continuously updated throughout the project.

2.1. Deliverable objectives

The objectives of this deliverable are as follows:

- to define VICINITY functional requirements;
- to define VICINITY non-functional requirements.

2.2. Relation to other Tasks and Deliverables

The following documents have been used to derive requirements:

- D1.1 VICINITY requirement capture framework – defines how the requirements are managed in VICINITY project;
- D1.2 Report on business drivers and barriers of IoT interoperability and value added services – defines constraints for functional specification based on stakeholders' drivers and barriers;
- D1.3 Report on pilot sites and operational requirements – defines constraints deployment and environment constraints for functional specification;
- D1.4 Report on VICINITY business requirements – provides inputs on desired VICINITY functionality.

The following documents will take forward the requirements reported in this document

- D1.6 VICINITY architectural design – uses mostly non-functional requirements to shape the VICINITY architecture;
- D2.2 – Semantic interfaces – technical requirements specification is used as input for the VICINITY Ontology.

The following tasks utilize the requirements reported in this document:

- as use the requirements in detail design and implementation of VICINITY Client components and security services;
- Task 5.1 exploits the requirements to design value added services.

3. Approach

This chapter describes the approach taken to define the technical requirements specification as part of the over-all VICINITY requirement life-cycle defined in the Deliverable D1.1 VICINITY requirement capture framework.

3.1. Methodology

The VICINITY Technical requirements specification collects the following inputs from different deliverables:

- Stakeholders drivers and barriers from D1.2;
- Stakeholders business requirements from D1.4;
- Operational and interface requirements from D1.3;
- IoT platforms, infrastructures, ontologies and relevant standards from D2.1;
- Requirement management methodology from D1.1.

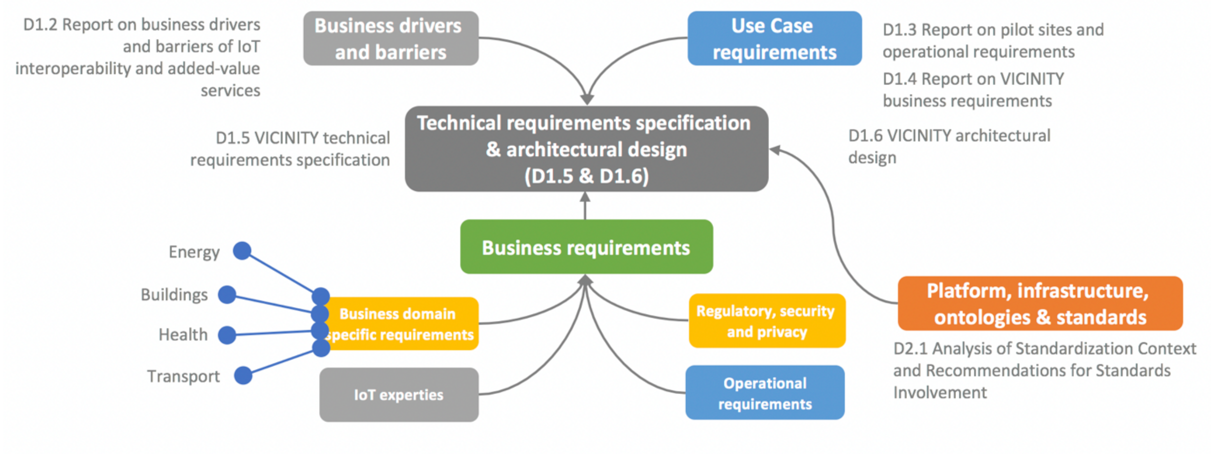

D1.2 Report on business drivers and barriers of IoT interoperability and added-value services
Business drivers and barriers
Use Case requirements
D1.3 Report on pilot sites and operational requirements
D1.4 Report on VICINITY business requirements
D1.5 VICINITY technical requirements specification
Technical requirements specification & architectural design (D1.5 & D1.6)
D1.6 VICINITY architectural design
Business requirements
Energy
Buildings
Health
Transport
Business domain specific requirements
IoT experties
Regulatory, security and privacy
Operational requirements
Platform, infrastructure, ontologies & standards
D2.1 Analysis of Standardization Context and Recommendations for Standards Involvement

Figure 1 Relation of technical requirements specification with other type of requirements

The technical specification is defined in the following steps:

- Chapter 4 defines the VICINITY Functional design as a technical use case (how the system should be used) based on business, use case requirements and stakeholders' drivers and barriers;
- Chapter 5 includes formalized VICINITY functional requirements based on technical use cases and set of VICINITY non-functional requirements.

The summary of relation of technical requirements specification with other type of requirements are shown on Figure 2.

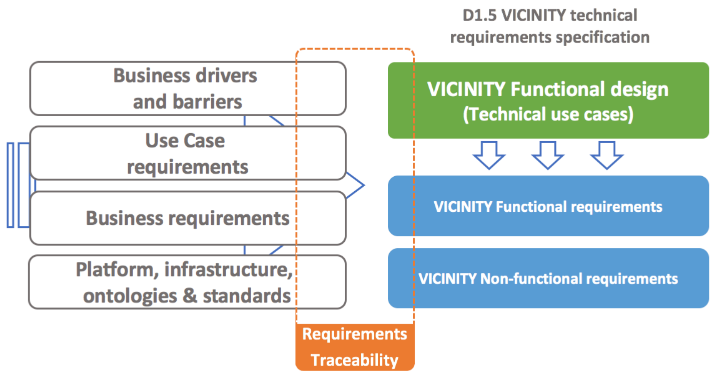

D1.5 VICINITY technical requirements specification
Business drivers and barriers
Use Case requirements
Business requirements
Platform, infrastructure, ontologies & standards
VICINITY Functional design (Technical use cases)
VICINITY Functional requirements
VICINITY Non-functional requirements
Requirements Traceability

Figure 2 Relation upper level requirements and technical requirements

The business drivers and barriers, use case requirements, business requirements and standards will be analysed. From the analysis the list of actors and main technical use cases will be identified (4.1, 4.2, 4.3, 4.4). Each technical use case will be described and decomposed in several detailed use cases. After definition of these use cases, common use cases will be identified (4.5). Short description (the main goal) will be defined for each technical use case. After technical use case completion, the manageable set of functional requirements will be identified and referenced to use cases (5.1).

The non-functional requirements are derived from the analysed documents as follows:

- user requirements (5.2.1) from "D1.2 Report on business drivers and barriers of IoT interoperability and value added services" domains' related drivers and barriers;
- performance, availability and maintainability requirements (5.2.2.1, 5.2.2.2, 5.2.2.3) extracted from "D1.3 Report on pilot sites and operational requirements";
- security requirements (5.2.2.4) from "D1.2 Report on business drivers and barriers of IoT interoperability and value added services" security drivers and barriers;
- privacy requirements (5.2.2.5) from "D1.2 Report on business drivers and barriers of IoT interoperability and value added services" privacy drivers and barriers;
- legal requirements 5.2.3.10) requirement from "D1.2 Report on business drivers and barriers of IoT interoperability and value added services" legal drivers and barriers;
- standardization requirements (5.2.3.2) from "D2.1 Analysis of Standardisation Context and Recommendations for Standards Involvement".

All type of requirements will be managed together. Systems Modelling Language (SysML) was chosen by the VICINITY consortium as a way to keep track of all the requirements and accommodate the changes, track modifications and assess their impact on the overall solution. SysML is a general-purpose modelling language.

3.1.1. Artefacts' identifiers

Every artefact (use case, requirements) of technical requirements specification should have its unique identifier:

- Use case:
    - Main technical use cases UC XX00, where XX is the number of main technical use case;
    - Detailed technical use case UC XXYY, where XX is the number of main technical use case and YY is number of detailed technical use case;
    - Common and supporting use cases: UC AAAXXX, where AAA is notation of the use case type (such as: SEC – Security, COM – Common, NTF – Notification, etc.) and XXX is number of the use case;

- Requirement:
  - A Functional requirement describes the VICINITY functionality/ behaviour: VICINITY-FUNCT-AAAXXX, where AAA is the name of a requirement group and XXX is the number of the requirement.
  - Non-functional requirement describes the property of the VICINITY solution: VICINITY-NFUNC-AAAXXX, where AAA is the name of a requirement group and XXX is the number of the requirement.

Table 1 List of requirements groups

Requirement group	Requirement identifier
Use case requirements	VICINITY-FUNCT-UCR010
User requirements	VICINITY-NFUNC-USR010
Performance	VICINITY-NFUNC-PER010
Availability	VICINITY-NFUNC-AVL010
Maintainability	VICINITY-NFUNC-MNT010
Security	VICINITY-NFUNC-SEC010
Privacy	VICINITY-NFUNC-PRV010
Standardization requirements	VICINITY-FUNCT-STD010, VICINITY-NFUNC-STD010
Licensing	VICINITY-FUNCT-LCS010, VICINITY-NFUNC-LCS010
Value added service	VICINITY-FUNCT-VAS010, VICINITY-NFUNC-VAS010

3.2. UML Design

The VICINITY Technical requirements specification is described by a set of use cases and set of functional and non-functional requirements. The set of use cases is reflecting how the system should be used. Each use case has an actor who performs the action and system on which the action is performed. Consider the following functionality: “VICINITY User is making a login”. This functionality will be described by use case diagram as follows (Figure 3): “VICINITY User” is the Actor, “UC SEC010 – User login” is a use case and “VICINITY” is a system. Moreover, there is another use case “UC SEC013 – Authentication” which performs users authentication. This use case is executed by “UC SEC010 – User login” as depicted by dashed arrow with label “<<include>>”.

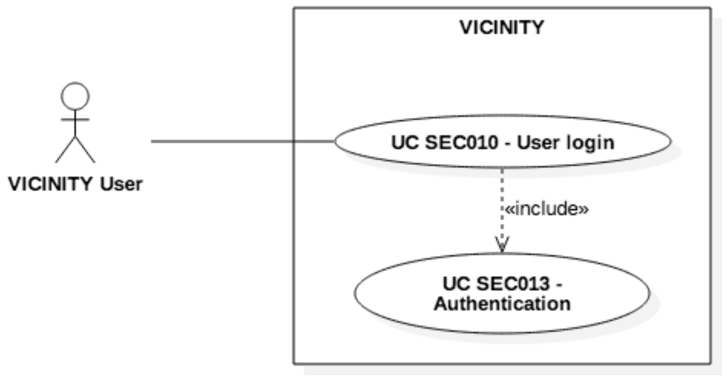


Figure 3 Example of the technical use case diagram

The functional and non-functional requirement will be written directly in VICINITY (Chapter 5) using the following requirement template including:

- requirement identifier,
- requirement title,
- requirement description,
- list of considered requirements, barriers and drivers and use cases:

VICINITY-REQ-0010	<Requirement title>
<Requirement description>	
Considered requirements:	
<List of considered requirements, barriers and drivers numbers>	

4. VICINITY functional design

The VICINITY functional design identifies the following technical use cases and their principal actors:

Table 2 List of actors

Use case	Principal actor	Actor description
Interoperability setup	IoT Operator	IoT Operator primarily controls the social network of devices and services – neighbourhood.
Device register and discovery	Device owner (including household or individuals)	Device owner provides device and its data in VICINITY.
Deploy value added service	Service provider	Service provider provides value added services in VICINITY.
Connecting VICINITY	System integrator (including IoT platform or Device vendor)	System integrator integrates the local IoT infrastructures in the VICINITY.

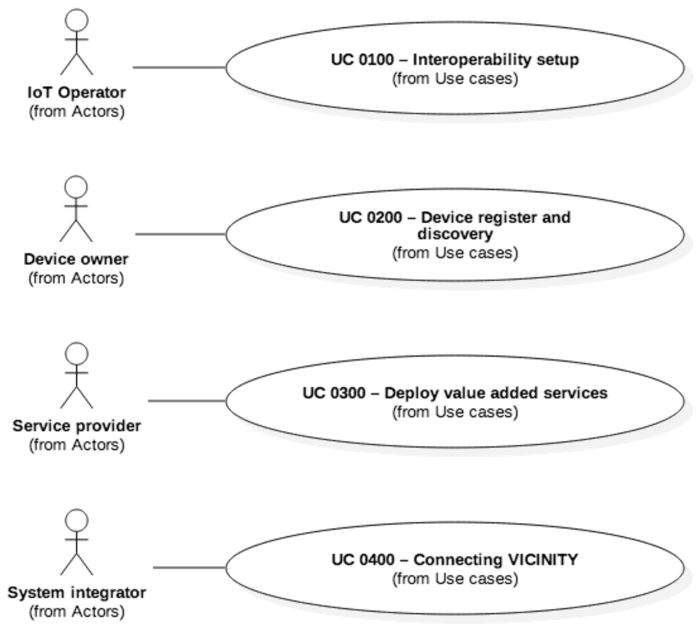


Figure 4 Main VICINITY Actors and technical use cases

4.1. UC 0100 – Interoperability setup

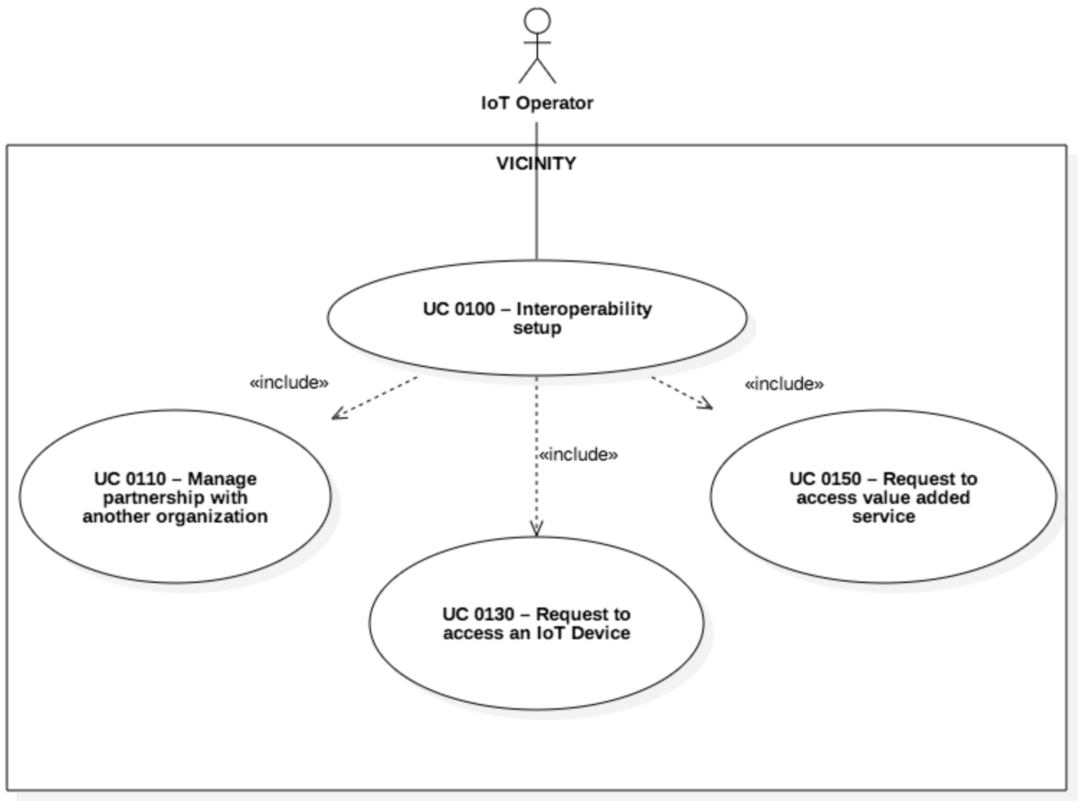

IoT Operator
VICINITY
UC 0100 – Interoperability setup
«include»
«include»
«include»
UC 0110 – Manage partnership with another organization
UC 0150 – Request to access value added service
UC 0130 – Request to access an IoT Device

Figure 5 UC 0100 – Interoperability setup

Principal actor: IoT Operator

Description: The goal of the interoperability setup use case is to create and maintain a social network of IoT objects called a “VICINITY neighbourhood” where IoT Operators should share IoT devices and value added services on behalf of their organization. The use case encompasses the following features: manage partnerships, manage access to IoT devices and value added service.

Partnership management includes searching for new partners (UC SRC010 – Search in VICINITY), sending partnership request, its accepting or declining. If a partnership is not needed any more, it can be removed in VICINITY. Partnership management includes setting up visibility of IoT devices and value added services within the neighbourhood.

Management of access to IoT devices and value added service enables the control of sharing of IoT devices and value added services within neighbourhood up to IoT object level. IoT Operator can request access to IoT objects managed by different oraganization. Request can be accepted or rejected. Moreover, accepted request can be rejected any time.

where there is exchanging or processing sensitive/ private data, consent needs to be provided by IoT Operator (data subject).

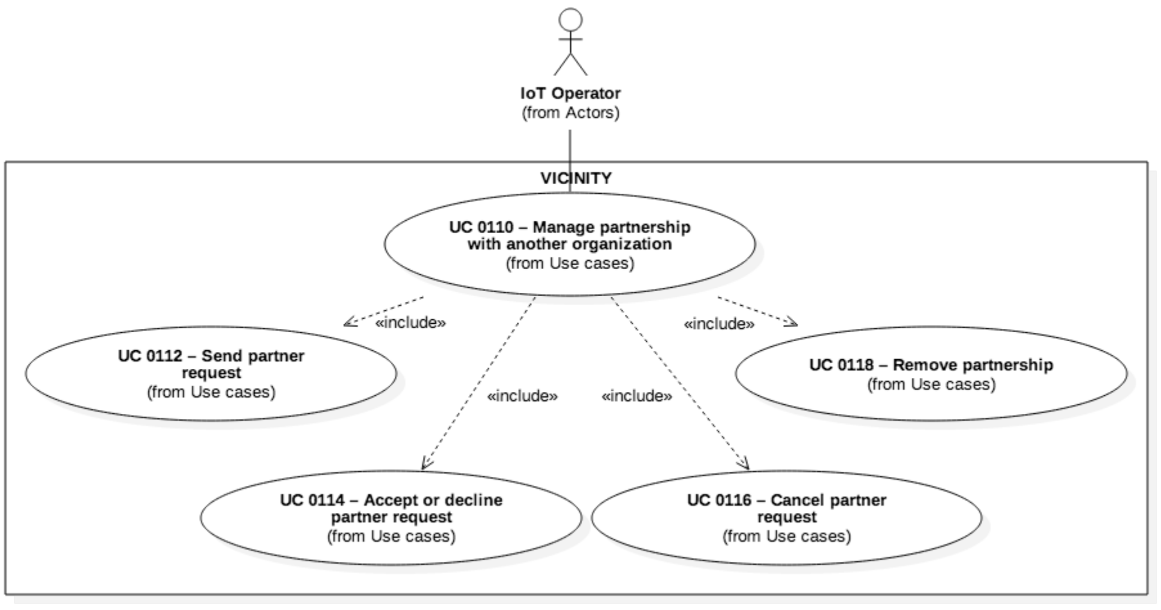
List of specific and common use cases:

- UC 0110 – Manage partnership with another organization;
  - UC 0112 – Send partner request,
  - UC 0114 – Accept or decline partner request,

- UC 0116 – Cancel partner request,
- UC 0118 – Remove partnership,

- UC 0130 – Request to access an IoT Device,
    - UC 0134 – Send request to access IoT devices,
    - UC 0136 – Accept or decline request to access IoT device,
    - UC 0138 – Cancel request to access an IoT device,
    - UC 0139 – Remove access to IoT device,
- UC 0150 – Request to access value added service,
    - UC 0154 – Send request to access value added service,
    - UC 0156 – Accept or decline request to access value added service,
    - UC 0158 – Cancel value added service access request,
    - UC 0159 – Remove access to value added service.

4.1.1. UC 0110 – Manage partnership with other organization



IoT Operator
(from Actors)
VICINITY
UC 0110 – Manage partnership with another organization
(from Use cases)
«include»
«include»
«include»
«include»
UC 0112 – Send partner request
(from Use cases)
UC 0118 – Remove partnership
(from Use cases)
UC 0114 – Accept or decline partner request
(from Use cases)
UC 0116 – Cancel partner request
(from Use cases)

Figure 6 UC 0110 – Create partnership with other organization

The goal is to create partnership between two organizations in VICINITY to open access to other IoT devices and services. The use case includes sending partner request between organizations, accepting and declining of such partner request and removing exiting partnership.

4.1.1.1. *UC 0112 – Send partner request*

IoT Operator should send a partner request to include an organization in its neighbourhood. The partner request might include the acknowledgement of terms and conditions or privacy management concerns as well. The managing IoT Operator is notified about any pending partner requests.

4.1.1.2. *UC 0114 – Accept or decline partner request*

The IoT Operator should accept or decline any partner request for her organization. The requesting IoT Operator is notified about any result of the request. The requesting organization becomes part of the neighbourhood after accepting the partner request. Partnered organizations might have visibility/

access to all IoT device and value added services with visibility/ access set to neighbourhood level. Declination of the request does not block any future partner requests.

4.1.1.3. *UC 0116 – Cancel partner request*

IoT Operator can withdraw any partner request at any time before it is accepted or declined. Accepted or declined requests cannot be cancelled. Any confirmed terms and conditions or privacy management concerns regarding the request will be automatically rejected.

4.1.1.4. *UC 0118 – Remove partnership*

A partner in a VICINITY network may withdraw their membership at any time. The IoT Operator should remove an organization from neighbourhood if requested by its organisation. Any access or visibility to IoT objects (IoT devices and value added services) should be removed automatically except those with public access or visibility. After removing access to IoT objects, there shoul be no further processing of new information from device acces via that partnership.

4.1.2. UC 0130 – Request to access IoT Device

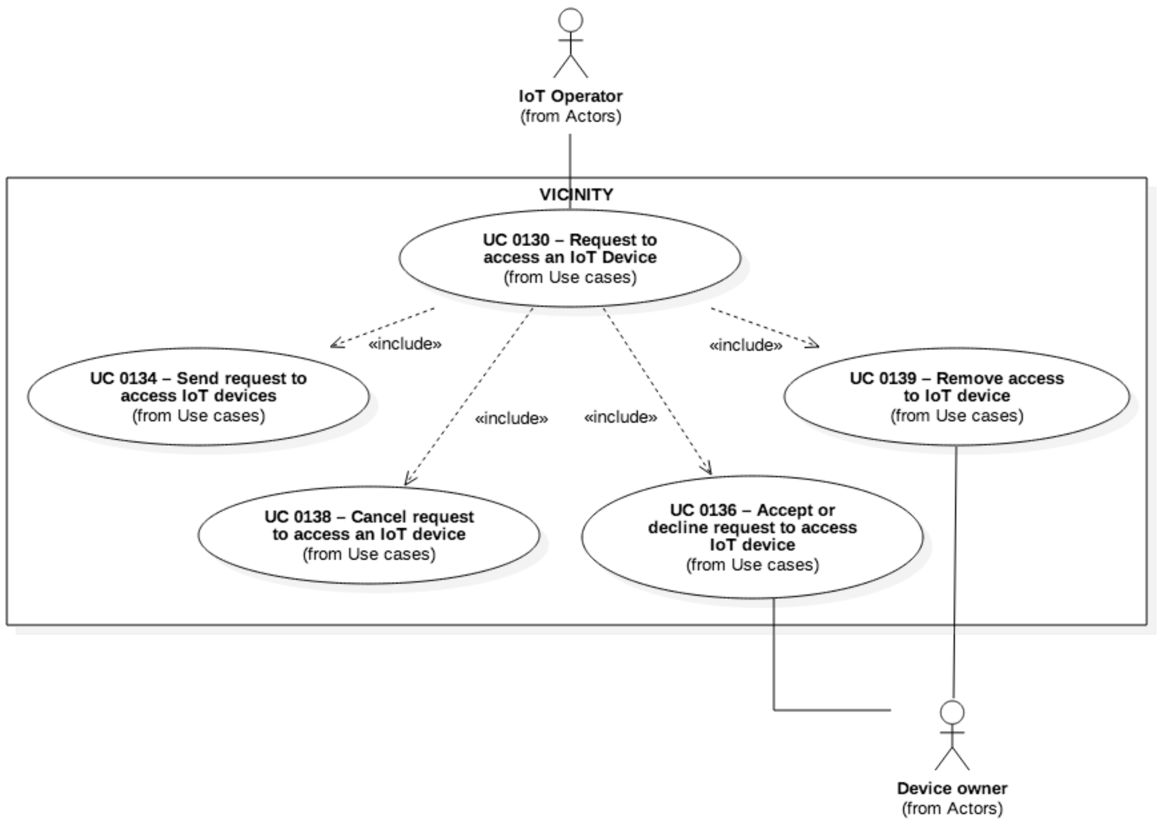

IoT Operator
(from Actors)
VICINITY
UC 0130 – Request to access an IoT Device
(from Use cases)
«include»
«include»
«include»
«include»
UC 0134 – Send request to access IoT devices
(from Use cases)
UC 0139 – Remove access to IoT device
(from Use cases)
UC 0138 – Cancel request to access an IoT device
(from Use cases)
UC 0136 – Accept or decline request to access IoT device
(from Use cases)
Device owner
(from Actors)

Figure 7 UC 0130 – Request to access IoT Device

4.1.2.1. *UC 0134 – Send request to access IoT devices*

The IoT Operator should send an IoT device access request to the Device owner. The acknowledgement of terms and conditions or privacy management concerns can be included as part of the request. The

Device owner (or manager) is notified about any pending access requests. Note that an IoT Operator can send IoT device access request only on visible devices. The device visibility can be setup by Device owner (UC SEC050 - Update accessing rules of IoT device / group of devices).

4.1.2.2. *UC 0136 – Accept or decline request to access IoT device*

The Device owner should accept or decline any request to access IoT devices he/she is owning. The requesting IoT Operator is notified about any result of the request. The processing of information from an IoT device can proceed after accepting of the request.

4.1.2.3. *UC 0138 – Cancel request to access IoT device*

An IoT Operator can withdraw any IoT device access request any time before it is accepted or declined. Accepted or declined request cannot be cancelled. Any confirmed terms and conditions or privacy management concerns regarding the IoT device request should be automatically rejected.

4.1.2.4. *UC 0139 – Remove access to IoT device*

The Device owner may remove access to any IoT device he/she owns. Access to IoT devices should be removed automatically if the Device owners' organizations are not in neighbourhood anymore. The access to the IoT device should not be renewed automatically when previously removed partnership is requested again. After removing access to the IoT device, the processing of new information should be constrained. The IoT Operator administering organization neighbourhood can remove access to any IoT device as well.[1]

[1] This functionality should be in-line with common behavior of devices in IoT infrastructures, such as change device status or appearing and disappearing of device in proximity network, etc.

4.1.3. UC 0150 – Request to access value added service

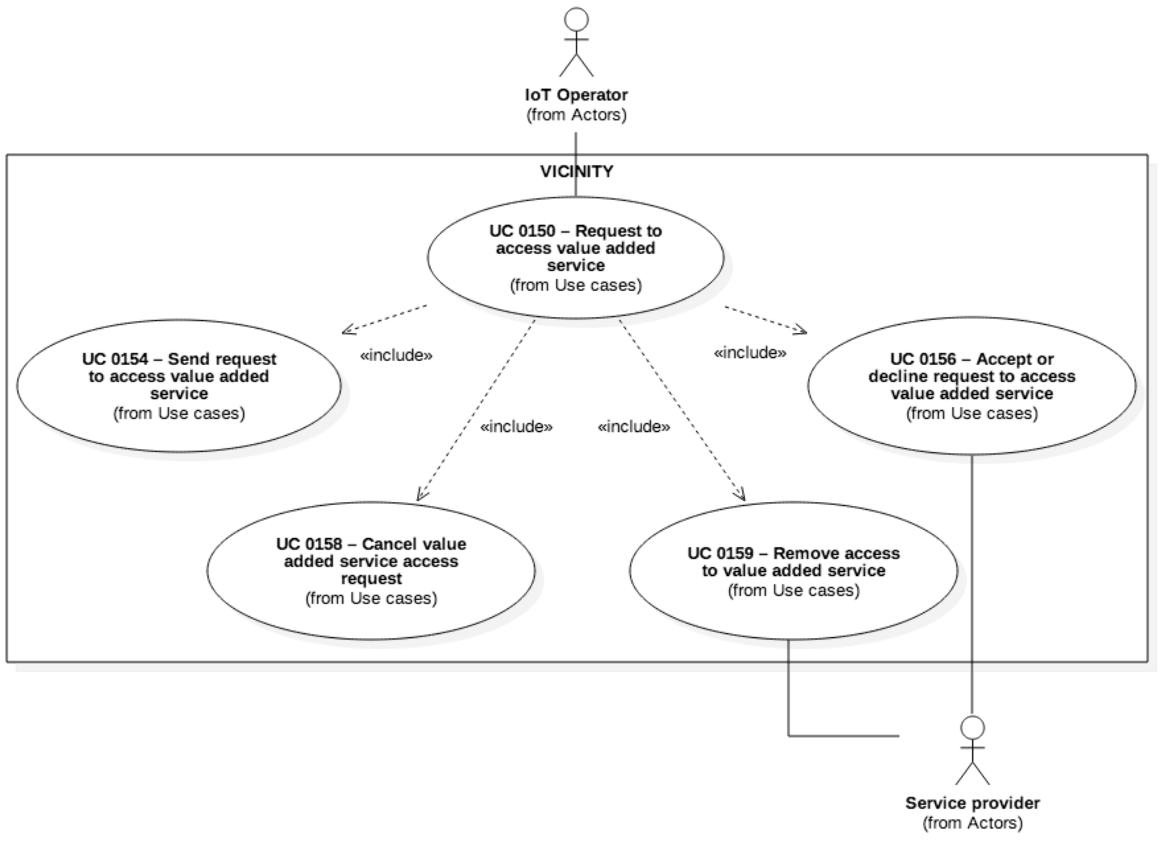

IoT Operator
(from Actors)
VICINITY
UC 0150 – Request to access value added service
(from Use cases)
«include»
«include»
«include»
«include»
UC 0154 – Send request to access value added service
(from Use cases)
UC 0156 – Accept or decline request to access value added service
(from Use cases)
UC 0158 – Cancel value added service access request
(from Use cases)
UC 0159 – Remove access to value added service
(from Use cases)
Service provider
(from Actors)

Figure 8 UC 0150 - Request to access value added service

4.1.3.1. *UC 0154 – Send request to access value added service*

An IoT Operator may send a value added access request to the Service provider administering the service. The access request for service should include an acknowledgement of terms and conditions or privacy management concerns. The Service provider is notified about any pending access requests. Note that IoT Operator can only send request to access visible value added services. The service visibility can be setup by Service provider (UC SEC040 - Update accessing rules of the service / group of services).

4.1.3.2. *UC 0156 – Accept or decline request to access value added service*

The Service provider should accept or decline any request to access value added service he is administering. The requesting IoT Operator is notified about any result of the request. The processing of information from a value added service can proceed after the request has been accepted.

4.1.3.3. *UC 0158 – Cancel value added service access request*

An IoT Operator can withdraw any value added service request any time before it is accepted or declined. Accepted or declined requests cannot be cancelled. Any confirmed terms and conditions or privacy management concerns regarding the request will be automatically rejected if a request is cancelled.

4.1.3.4. *UC 0159 – Remove access to value added service*

The Service provider may remove access to value added service he/she is administering. Access to value added service should be removed automatically if organizations are not in the neighbourhood anymore. The access to value added services should not be renewed automatically when previously removed partnership is requested again. After removing access to value added service, the processing of new information should be constrained. An IoT Operator should be able to remove access to value added service its organization is providing or using at any time.[2]

4.2. UC 0200 – Device register and discovery

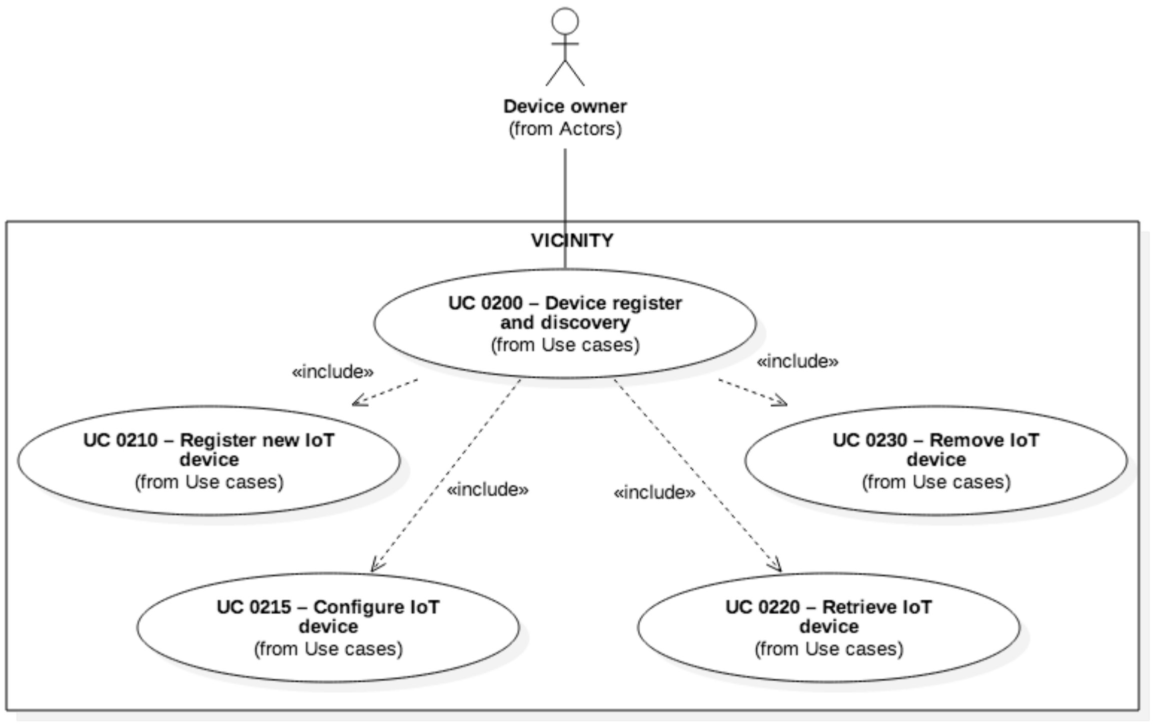

Device owner
(from Actors)
VICINITY
UC 0200 – Device register and discovery
(from Use cases)
«include»
«include»
UC 0210 – Register new IoT device
(from Use cases)
UC 0230 – Remove IoT device
(from Use cases)
«include»
«include»
UC 0215 – Configure IoT device
(from Use cases)
UC 0220 – Retrieve IoT device
(from Use cases)

Figure 9 UC 0200 – Device register and discovery

Principal actor: Device owner

Description: The goal of the use case is to manage any IoT device lifecycle within VICINITY, from automatic discovery or manual registration of any new IoT device in the VICINITY Client's infrastructure to its potential removal. In case it is necessary, the VICINITY Agent should be reconfigured automatically to support the data exchanging protocol utilized by the IoT device, even after the IoT device communication protocol update. The device owner should be able to configure the profile of the devices (such as communication protocol, description, security, privacy attributes and initial visibility within neighbourhood). Access to the IoT device can be constrained by terms and conditions or privacy management procedures. Note that a domain (network) owner (or their agent) may need

[2] This functionality should be in-line with common behavior of devices in IoT infrastructures, such as change device status or appearing and disappearing of device in proximity network, etc.

to carry out security and authorisation checks on an IoT device before it is added to their system. Details are explained in the section on Security use cases (UC SEC000 – Security).

List of specific and common use cases:

- UC 0210 – Register new IoT device;
- UC 0215 – Configure IoT device;
- UC 0220 – Retrieve IoT device;
- UC 0230 – Remove IoT device.

4.2.1. UC 0210 – Register new IoT device

The new IoT device should be recognized by the VICINITY Client in an integrated infrastructure. A discovery should be performed on each newly recognized IoT devices. The IoT device profile should be stored in the repository, based on identification and semantic matching. Security checks and private attributes should be initialized in the profile based on predefined rules. The IoT device profile should be verified by the Device owner. VICINITY Client should be configured to support the IoT device communication protocols and semantic data mediation mapping should be updated if necessary.

4.2.2. UC 0215 – Configure IoT device

The IoT device profile should include the static, security and privacy attributes. The Static attributes should include at least name, avatar, type of the device, device group membership and list of data sources provided by the device. Security and privacy attributes should include at least the device authorization rules and privacy attributes. The Device owner should be able to change the IoT device configuration manually. VICINITY Client should provide the device's configuration automatic updates if possible. VICINITY should perform semantic matching and VICINITY Client semantic data mediation mapping configuration in case of changes of IoT device.

4.2.3. UC 0220 – Retrieve IoT device configuration

The Device owner should be able to retrieve static, dynamic attributes, security and privacy attributes and audit logs stored in or associated with the IoT device profile. The IoT device configuration should be visible to all VICINITY users based on the IoT device or profile attributes visibility/accessibility rules.

4.2.4. UC 0230 – Remove IoT device

VICINITY should be able to remove any IoT device automatically based on its removal from the integrated infrastructure (e.g. VICINITY should receive IoT device removal notification from integrated infrastructure) or manually by its device owner. The existence of the removed device should be traceable through audits and logs. The removal of the device is notified to other IoT Operators within the neighbourhood which have access to the device.

4.3. UC 0300 – Deploy value added services

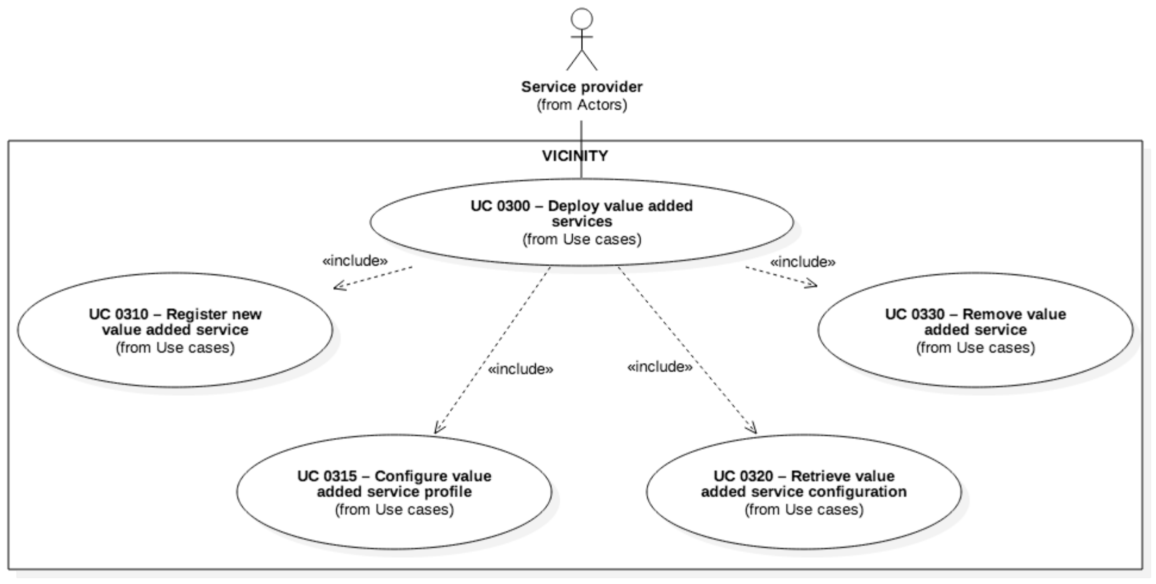

Service provider
(from Actors)
VICINITY
UC 0300 – Deploy value added services
(from Use cases)
«include»
«include»
«include»
«include»
UC 0310 – Register new value added service
(from Use cases)
UC 0330 – Remove value added service
(from Use cases)
UC 0315 – Configure value added service profile
(from Use cases)
UC 0320 – Retrieve value added service configuration
(from Use cases)

Figure 10 UC 0300 – Deploy value added services

Principal actor: Service provider

Description: The service provider should be able to manage the whole life cycle of a value added service connected to VICINITY from its discovery or registration a service to its removal from VICINITY. Registered value added service profile attributes (such as communication protocol, description, security, privacy attributes and initial visibility within neighbourhood) should be configured by the service provider. The service provider can manage terms and condition to use the value added service.

List of specific and common use cases:

- UC 0310 – Register new value added service,
- UC 0315 – Configure value added service profile,
- UC 0320 – Retrieve value added service configuration,
- UC 0330 – Remove value added service.

4.3.1. UC 0310 – Register new value added service

The availability of a new value added service should be recognized by the VICINITY Client in an integrated infrastructure. For each value added service a discovery should be performed. Based on identification and semantic matching during the discovery, the value added service profile should be stored in the repository. Security and privacy attributes should be initialized in the profile based on predefined rules where applicable. The value added service profile should be verified by the service provider. VICINITY Client should be configured to support the value added service communication protocols and semantic data mediation mapping should be updated if necessary.

4.3.2. UC 0315 - Configure value added service profile

The value added service profile should include the static, security and privacy attributes. The Static attributes should include at least name, avatar, type of the service, service group membership and list of data sources provided by the service. Security and privacy attributes should include at least service authorization rules and privacy attributes. The service provider should be able to change the value added service configuration manually. VICINITY Client should obtain automatic updates of configuration from the value added service if possible. VICINITY should perform semantic matching and VICINITY Client semantic data mediation mapping configuration if necessary.

4.3.1. UC 0320 – Retrieve value added service configuration

The Service provider should be able to retrieve static, dynamic attributes, security and privacy attributes and audit logs stored in or associated with the value added service profile. The value added service configuration should be visible to all VICINITY users based on the service attributes visibility/accessibility rules.

4.3.2. UC 0330 –Remove value added service

VICINITY should be able to remove any value added service automatically based on its removal from the integrated infrastructure (e.g. VICINITY should receive value added service removal notification from an integrated infrastructure) or its manually removal by the service provider. The existence of the removed service should be traceable through audits and logs. The removal of the value added service is notified to other IoT Operators within the neighbourhood.

4.4. UC 0400 – Connecting VICINITY

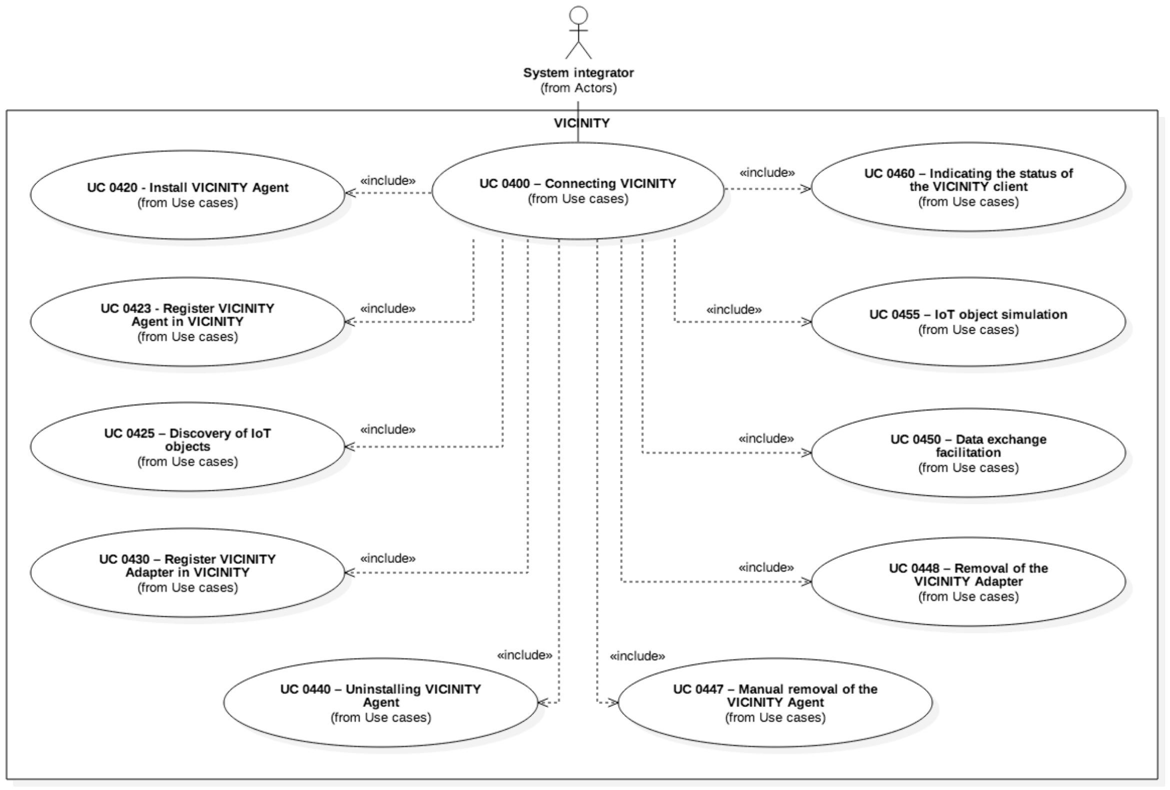

System integrator
(from Actors)
VICINITY
UC 0400 – Connecting VICINITY
(from Use cases)
UC 0420 - Install VICINITY Agent
(from Use cases)
UC 0460 – Indicating the status of the VICINITY client
(from Use cases)
UC 0423 - Register VICINITY Agent in VICINITY
(from Use cases)
UC 0455 – IoT object simulation
(from Use cases)
UC 0425 – Discovery of IoT objects
(from Use cases)
UC 0450 – Data exchange facilitation
(from Use cases)
UC 0430 – Register VICINITY Adapter in VICINITY
(from Use cases)
UC 0448 – Removal of the VICINITY Adapter
(from Use cases)
UC 0440 – Uninstalling VICINITY Agent
(from Use cases)
UC 0447 – Manual removal of the VICINITY Agent
(from Use cases)
«include»

Figure 11 UC 0400 - Connecting VICINITY

Principal actor: System integrator

Description: The objective of this use case is to setup the connection between VICINITY and VICINITY Client infrastructures.

There are two options on how VICINITY can be integrated in VICINITY client infrastructures.

- The VICINITY Agent is a software component implemented for a specific platform add-on. This is usually part of an open source integrated infrastructure platform. The VICINITY Agent is still under full control of the integrated infrastructure owner; however, it can provide several automatic features such as automatic installation and configuration. The VICINITY Agent should be installed by the system integrator or provided by the VICINITY Client infrastructure. The installed VICINITY Agent should be registered within VICINITY. The VICINITY Agent should be able to perform initial and continuous discovery of IoT objects (IoT devices and value added services).
- VICINITY Adapter is software component implemented and fully managed under governance of integrated infrastructure owner handling exchange of data within VICINITY. The VICINITY Adapter should be registered in VICINITY by a system integrator. The VICINITY Adapter should perform the initial and continuous discovery of IoT objects (IoT devices and value added services).

VICINITY Clients (VICINITY Agent and/or VICINITY Adapter) should support initial set-up of communication channels with other VICINITY Clients. The communication channels set-up should be continuously updated according to interoperability set-up of neighbourhood. Thus, VICINITY Clients should exchange information only based on the actual set-up of communication channels.

List of specific and common use cases:

- UC 0420 - Install VICINITY Agent;
- UC 0423 - Register VICINITY Agent in VICINITY;
- UC 0425 – Discovery of IoT objects;
- UC 0430 – Register VICINITY Adapter in VICINITY;
- UC 0440 – Uninstalling VICINITY Agent;
- UC 0447 – Manual removal of the VICINITY Agent;
- UC 0448 – Removal of the VICINITY Adapter;
- UC 0450 – Data exchange facilitation;
- UC 0455 – IoT object simulation;
- UC 0460 – Indicating the status of the VICINITY client.

4.4.1. UC 0420 – Install VICINITY Agent

The system integrator should be able to download and install the VICINITY Agent from the platform specific market place or internet if market place is not available.

4.4.2. UC 0423 – Register VICINITY Agent in VICINITY

The VICINITY Agent should be registered within the VICINITY automatically or manually. Automatic registration should use IoT Operator credentials if possible. Manual registration should be performed by the IoT Operator through the VICINITY Agent user interface if possible or directly in VICINITY Server user interface. During the registration process the VICINITY Agent profile should be created. After registration, the initial discovery of IoT objects should be performed.

4.4.3. UC 0425 – Discovery of IoT objects

During the initial discovery, the search of all IoT objects accessible through VICINITY Client (Agent or Adapter) should be performed. The initial discovery should not need to be constrained, while the goal is to get a super set of accessible IoT objects. For each discovered object, the object identification is performed including the semantic matching of the IoT object descriptor. An initial version of the IoT object profile is created and stored in the respective IoT object profile repository. Based on the IoT object profile, the VICINITY Client semantic data mediation mapping and communication protocol support should be updated if necessary.

4.4.4. UC 0430 – Register VICINITY Adapter in VICINITY

The VICINITY Adapter should be registered within the VICINITY automatically or manually. Automatic registration should use IoT Operator credentials if possible. Manual registration should be provided and performed by the IoT Operator through the VICINITY Adapter user interface if possible, or directly in VICINITY. During the registration process the VICINITY Adapter profile should be created. After registration, the initial discovery of IoT objects should be performed.

4.4.5. UC 0440 – Uninstalling VICINITY Agent

The System integrator should be able to uninstall the VICINITY Agent. The Agent uninstallation should remove all associated IoT objects (IoT devices, value added service) from VICINITY and update the interoperability set-up including communication channels set-up. VICINITY should keep audit trails and logs regarding the VICINITY Agent even after its removal.

4.4.6. UC 0447 – Manual removal of the VICINITY Agent

The IoT Operator should be able to remove the registered VICINITY Agent. The Agent removal should remove all associated IoT objects (IoT devices, value added service) from VICINITY and update the interoperability set-up including communication channels set-up. VICINITY should keep audit trails and logs regarding the VICINITY Agent even after its removal.

4.4.1. UC 0448 – Removal of the VICINITY Adapter

An IoT Operator should be able to remove the registered VICINITY Adapter. All associated IoT objects (IoT devices, value added services) should be removed from VICINITY and the interoperability set-up including communication channels set-up should be updated. VICINITY should keep audit trails and logs regarding the VICINITY Adapter even after its removal.

4.4.2. UC 0450 – Data exchange facilitation

VICINITY Client should be able to facilitate the data exchange with other VICINITY Clients using common standardized protocols. The data exchange channels are set-up based on interoperability set-up (UC 0100 – Interoperability setup).

4.4.1. UC 0455 – IoT object simulation

VICINITY Client should be able to simulate supported shared IoT object within existing IoT infrastructure.

4.4.2. UC 0460 - Indicating the status of the VICINITY Client

The VICINITY Client should notify VICINITY about initial and any changes of status of associated IoT Objects if possible.

4.5. Common and supporting use cases

This section consists of the following groups use cases which are available across the whole VICINITY solution:

- Legal,
- Security and privacy,
- IoT devices and value added services grouping,
- Organization and user management,
- Search and User notifications.

Use cases are supportive to previously described business driven use cases (4.1, 4.2, 4.3, 4.4), thus the following generic actors needs to be introduced:

Table 3 List of supportive actors

Actor	Description
VICINITY User	Any user interacting with VICINITY.
VICINITY Organization Administrator	VICINITY User which has a right to set-up organization profile, assign VICINITY roles to other users within her organization.

VICINITY User is a generic actor of all other actors, i.e. the VICINITY function provided to a VICINITY User is also provided to any other actor (Figure 12).

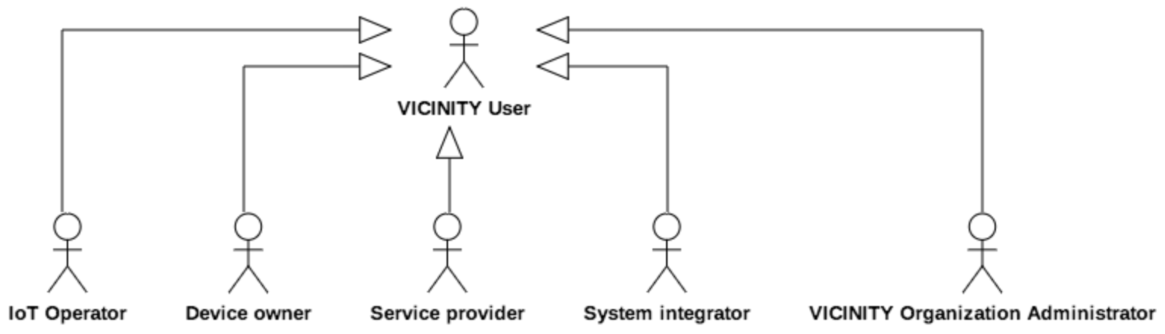


Figure 12 Relation between actors

4.5.1. UC LEG000 – Legal

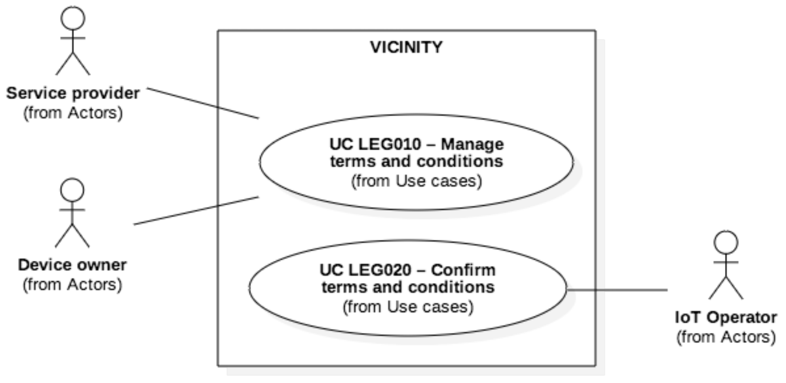


Figure 13 UC LEG000 – Legal

4.5.1.1. *UC LEG010 – Manage terms and conditions*

The Device owner and the service provider should be able to manage different terms and conditions. The terms and conditions might be associated with specific value added services and IoT devices. They should be part of the IoT object profile. Their visibility should be the same as the visibility of the associated IoT object's profile.

4.5.1.2. *UC LEG020 – Confirm terms and conditions*

The IoT Operator should be able to confirm the terms and conditions of the value added service and/or IoT device during the interoperability setup. The confirmation of terms and conditions is stored in VICINITY according to the security requirements. The IoT Operator should not be able to revoke confirmed terms and conditions, except as part of a termination of an agreement.

4.5.2. UC SEC000 – Security

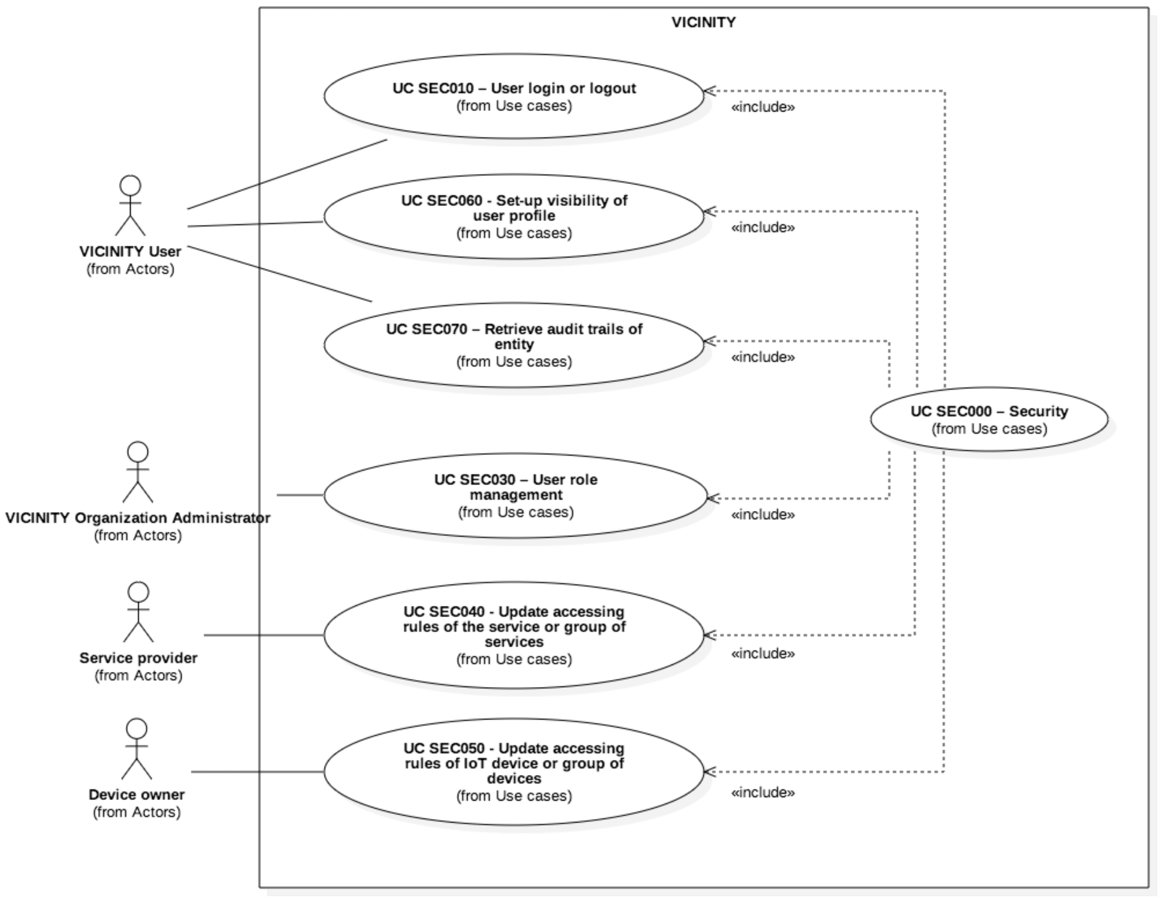

VICINITY
UC SEC010 – User login or logout (from Use cases)
UC SEC060 - Set-up visibility of user profile (from Use cases)
UC SEC070 – Retrieve audit trails of entity (from Use cases)
UC SEC030 – User role management (from Use cases)
UC SEC040 - Update accessing rules of the service or group of services (from Use cases)
UC SEC050 - Update accessing rules of IoT device or group of devices (from Use cases)
UC SEC000 – Security (from Use cases)
«include»
VICINITY User (from Actors)
VICINITY Organization Administrator (from Actors)
Service provider (from Actors)
Device owner (from Actors)

Figure 14 UC SEC000 – Security

4.5.2.1. *UC SEC010 – User login/ logout*

The VICINITY User should be able to login and logout in VICINITY using at least user name and password credentials.

4.5.2.2. *UC SEC030 – User role management*

The VICINITY Organization administrator should be able to associate the VICINITY Users with the organization and update the user's roles within the organization according to its needs. The VICINITY Organization administrator should be able to assign at least with the following roles: VICINITY Organization administrator, VICINITY IoT Operator, VICINITY Device owner and VICINITY Service provider. A VICINITY User may have more than one role.

4.5.2.3. *UC SEC040 - Update accessing rules of the service / group of services*

The Service provider should set-up accessing rules for each value added service or value added service group. At least the following accessing rules should be considered:

- Public visibility and public access;
- Public visibility and access for partners;
- Public visibility and access upon request;
- Public visibility and private access;
- Visibility to partners and access for partners;
- Visibility to partners and access upon request;
- Visibility to partners and private access;
- Private visibility and private access.

4.5.2.4. *UC SEC050 - Update accessing rules of IoT device / group of devices*

The Device owner should set-up accessing rules for each IoT device or IoT device group. At least the following accessing rules should be considered:

- Public visibility and public access;
- Public visibility and access for partners;
- Public visibility and access upon request;
- Public visibility and private access;
- Visibility to partners and access for partners;
- Visibility to partners and access upon request;
- Visibility to partners and private access;
- Private visibility and private access.

4.5.2.5. *UC SEC060 - Set-up visibility of user profile*

The VICINITY User should be able to setup visibility of its profile. The visibility organization profile should not influence visibility of organization’s users’ profiles.

4.5.2.6. *UC SEC070 – Retrieve audit trails of entity*

The VICINITY User should be able to view entity (she has access to) audit trails including chronological order of important events of the entity (value added service, IoT device, user access rules, etc.) life cycle such as: registration, change of configuration, removal, etc.

4.5.3. UC PRV000 – Privacy

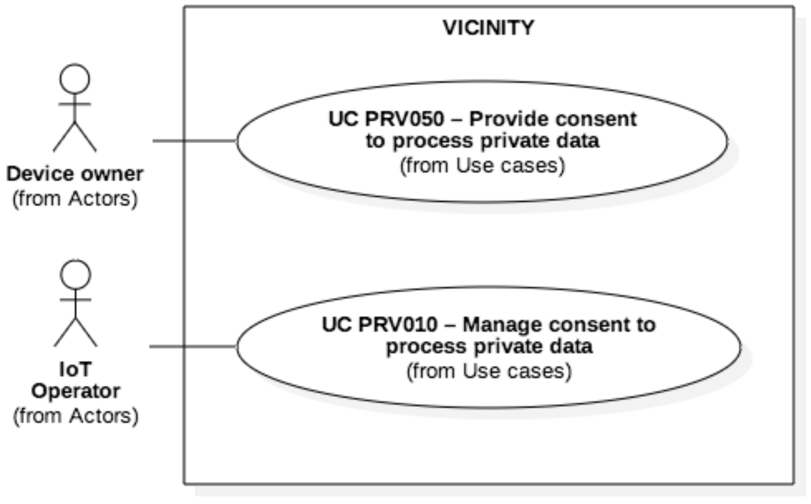

VICINITY
UC PRV050 – Provide consent to process private data (from Use cases)
UC PRV010 – Manage consent to process private data (from Use cases)
Device owner (from Actors)
IoT Operator (from Actors)

Figure 15 UC PRV000 – Privacy

4.5.3.1. *UC PRV010 – Manage consent to process private data*

The IoT Operator should manage privacy throughout the life-cycle using consent to process private data. These consents should be associated with value added services. Confirmed consents should be part of the value added service's profile. They should be visible only to the IoT Operator and the Service provider.

4.5.3.2. *UC PRV050 – Provide consent to process private data*

The Device owner should be able to confirm consent to process private data by a value added service or organization which has access to the IoT device. The Device owner should be able to manage all confirmed consents and revoke them individually.

4.5.4. UC GRP000 – IoT devices and value added services grouping

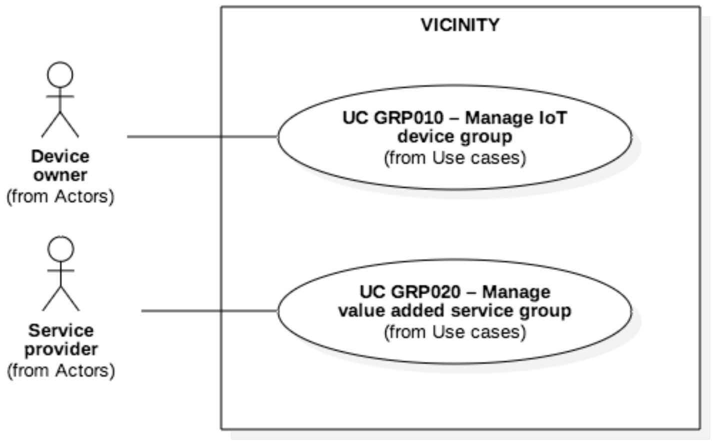

VICINITY
UC GRP010 – Manage IoT device group (from Use cases)
UC GRP020 – Manage value added service group (from Use cases)
Device owner (from Actors)
Service provider (from Actors)

Figure 16 UC GRP000 – IoT devices and value added services grouping

4.5.4.1. *UC GRP010 – Manage IoT device group*

The Device owner should be able to create (or remove) an IoT device group and add or remove IoT devices from this group. The device owner should be able to perform group actions on IoT devices such as change IoT device access rules. Each IoT device group should have a profile including at least name, avatar, description, device group visibility, accessibility rules, optionally terms and conditions and/or consent to process private data template.

4.5.4.2. *UC GRP020 – Manage value added service group*

The Service provider should be able to create (or remove) a value added service group and add (or remove) value added services from this group. The service provider should be able to perform group actions on value added services such as changing value added service access rules. Each value added service group should have a profile including at least name, avatar, description, value added service group visibility, accessibility rules, optionally terms and conditions and/or consent to process the private data template.

4.5.5. UC COM000 – Organization and user management

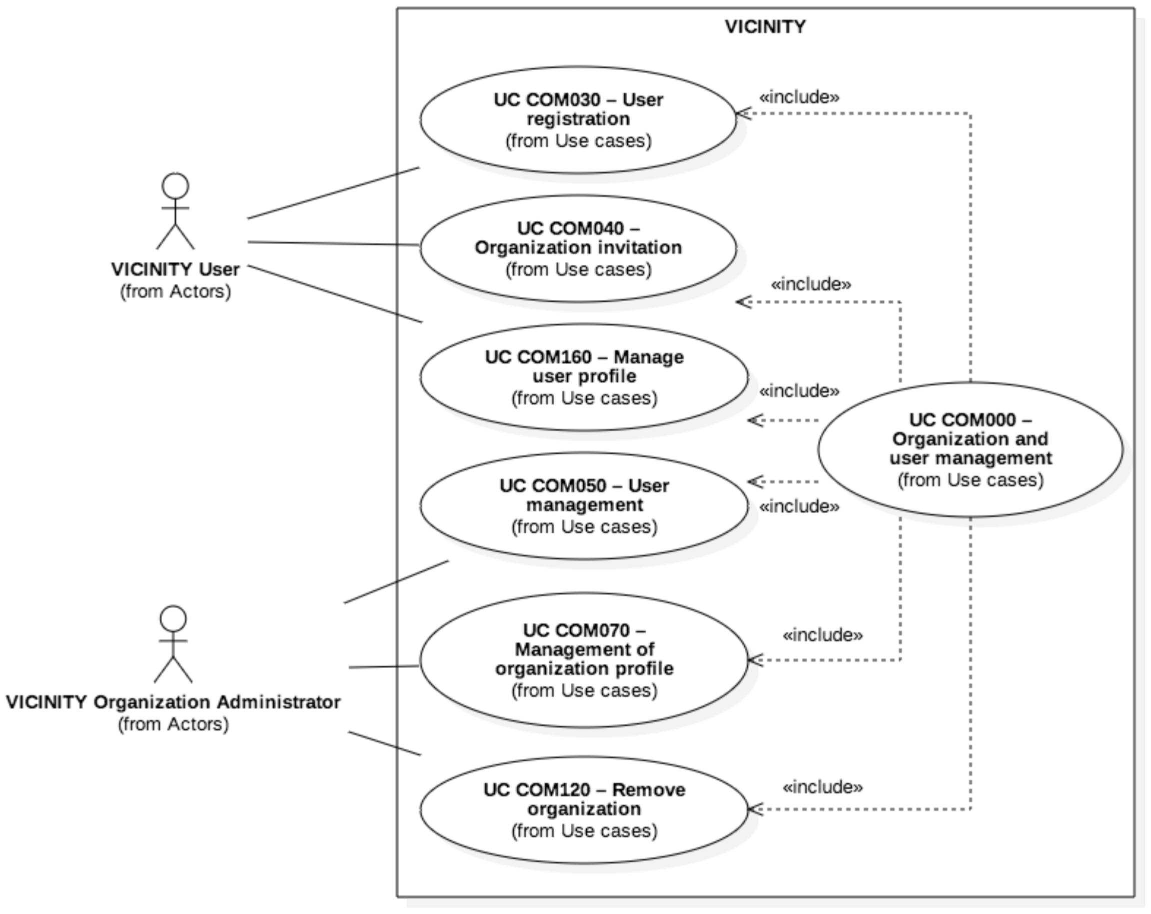

VICINITY
UC COM030 – User registration (from Use cases)
UC COM040 – Organization invitation (from Use cases)
UC COM160 – Manage user profile (from Use cases)
UC COM050 – User management (from Use cases)
UC COM070 – Management of organization profile (from Use cases)
UC COM120 – Remove organization (from Use cases)
UC COM000 – Organization and user management (from Use cases)
«include»
«include»
«include»
«include»
«include»
«include»
VICINITY User (from Actors)
VICINITY Organization Administrator (from Actors)

Figure 17 UC COM000 – Organization and user management

4.5.5.1. *UC COM030 – User registration*

The VICINITY User should be registered using the registration form. The user should be able to register only with a valid email address.

4.5.5.2. *UC COM040 – Organization invitation*

The VICINITY User should be able to invite other organizations to join the VICINITY network or create the organization by himself/herself.

4.5.5.3. *UC COM050 – User management*

The VICINITY Organization administrator should be able to invite other users (not registered in VICINITY) to join the VICINITY under his/her organization. The VICINITY Organization administrator should be able to add (already registered users) or remove users from the organization.

4.5.5.4. *UC COM070 – Management of organization profile*

The VICINITY Organization administrator should be able to view the organization profile based on the visibility of rules. The organization profile should include at least the organization name, description, avatar and location. The organization profile should include associated entities such as users, IoT devices, value added services and partnered organizations. The organization profile can be modified by the VICINITY Organization administrator only.

4.5.5.5. *UC COM120 – Remove organization*

The VICINITY Organization administrator should be able to remove the organization from their VICINITY network. The organization removal should invalidate all the registered VICINITY Clients, IoT devices and value added services. Moreover, it should remove the organization's neighbourhood including interoperability and communication channels set-ups. New registration of the organization will not renew any removed profiles, configurations and set-ups. All association of VICINITY users with an organization should be removed. Removal of the organization should not remove the relevant audit trails or logs.

4.5.5.6. *UC COM160 – Manage user profile*

The VICINITY user should be able to update only his/her own user profile.

4.5.6. UC SRC010 – Search in VICINITY

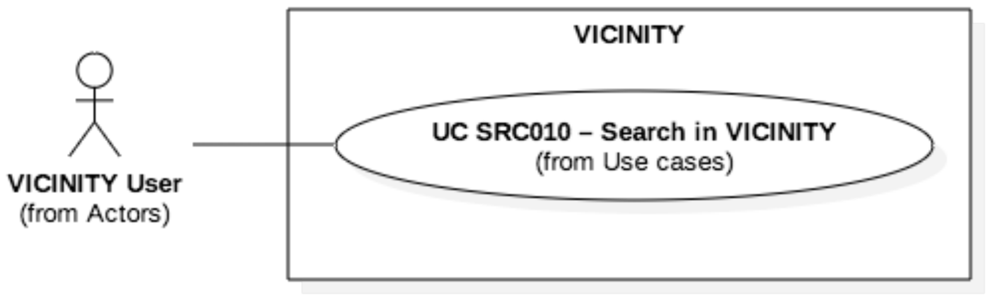


Figure 18 UC SRC010 – Search in VICINITY

The VICINITY User should be able to search in VICINITY at least the following entities: users, organizations, IoT devices, IoT devices groups, value added services, value added services groups. VICINITY network should filter search results based on the access rules of each entity and the VICINITY User performing the search. The VICINITY search might be constrained to their location or neighbourhood.

4.5.7. UC NTF010 - User notifications

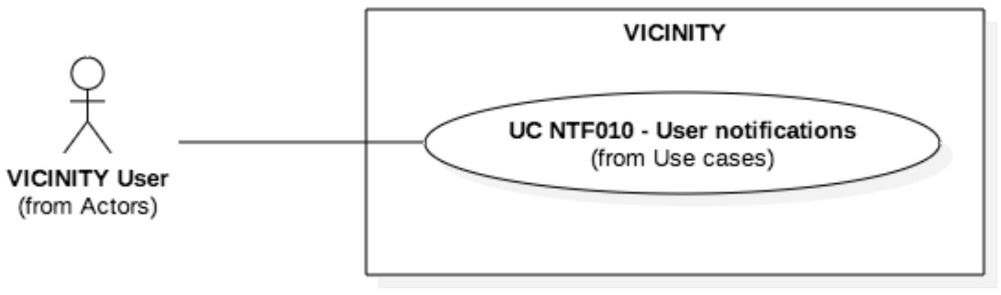


Figure 19 UC NTF010 - User notifications

The VICINITY User should be able to receive notifications regarding important events in VICINITY such as partnership or access requests, new IoT device or value added service notifications. The VICINITY User should be able read only those notifications associated with his role and relevant notification object. The user notification should include at least sender, recipient and role of notification, notification subject and object.

4.6. List of actors and use cases

This chapter summarizes the list of actors with assigned list of use cases.

Table 4 List of actors and associated use cases

Actor	Assigned use cases
IoT Operator	UC 0110 – Manage partnership with another organization; UC 0112 – Send partner request; UC 0114 – Accept or decline partner request; UC 0116 – Cancel partner request; UC 0118 – Remove partnership; UC 0130 – Request to access an IoT Device; UC 0134 – Send request to access IoT devices; UC 0136 – Accept or decline request to access IoT device; UC 0138 – Cancel request to access an IoT device; UC 0139 – Remove access to IoT device; UC 0150 – Request to access value added service; UC 0154 – Send request to access value added service; UC 0156 – Accept or decline request to access value added service; UC 0158 – Cancel value added service access request; UC 0159 – Remove access to value added service; UC LEG020 – Confirm terms and conditions; UC PRV010 – Manage consent to process private data;

Device owner	UC 0200 – Device register and discovery;
	UC 0210 – Register new IoT device;
	UC 0215 – Configure IoT device;
	UC 0220 – Retrieve IoT device;
	UC 0230 – Remove IoT device;
	UC 0136 – Accept or decline request to access IoT device;
	UC 0139 – Remove access to IoT device;
	UC LEG010 – Manage terms and conditions;

Actor	Assigned use cases
	UC SEC050 – Update accessing rules of IoT device / group of devices; UC PRV050 – Provide consent to process private data; UC GRP010 – Manage IoT device group;
Service provider	UC 0300 – Deploy value added services; UC 0310 – Register new value added service; UC 0315 – Configure value added service profile; UC 0320 – Retrieve value added service configuration; UC 0330 – Remove value added service; UC 0156 – Accept or decline request to access value added service; UC 0159 – Remove access to value added service; UC 0159 – Remove access to value added service; UC LEG010 – Manage terms and conditions; UC SEC040 – Update accessing rules of the service / group of services; UC GRP020 – Manage value added service group;

System integrator	UC 0400 – Connecting VICINITY;
	UC 0420 – Install VICINITY Agent;
	UC 0423 – Register VICINITY Agent in VICINITY;
	UC 0425 – Discovery of IoT objects;
	UC 0430 – Register VICINITY Adapter in VICINITY;
	UC 0440 – Uninstalling VICINITY Agent;
	UC 0447 – Manual removal of the VICINITY Agent;
	UC 0448 – Removal of the VICINITY Adapter;
	UC 0450 – Data exchange facilitation;
	UC 0455 – IoT object simulation;
	UC 0460 – Indicating the status of the VICINITY client;

Actor	Assigned use cases

VICINITY User	UC SEC010 – User login/ logout;
	UC SEC060 - Set-up visibility of user profile;
	UC SEC070 – Retrieve audit trails of entity;
	UC COM030 – User registration;
	UC COM040 – Organization invitation;
	UC COM160 – Manage user profile;
	UC SRC010 Search in VICINITY;
	UC NTF010 - User notifications;

VICINITY Organization Administrator	UC SEC030 – User role management;
	UC COM050 – User management;
	UC COM070 – Management of organization profile;
	UC COM120 – Remove organization;

5. VICINITY Interoperability platform requirements

5.1. Functional requirements

The manageable set of functional requirements is extracted from the use cases defined in chapter 4.

5.1.1. Interoperability set-up requirements

The chapter summarizes the functional requirements of the main use case UC 0100 – Interoperability setup.

VICINITY-FUNC-UCR010	**Neighbourhood building based on partnership network.**

VICINITY should support the creation of a social network based on partnership between organizations. The social network should be built based on:

- partnership request;
- acknowledgement or rejecting of partnership requests;
- cancelling unnecessary partnerships.

Considered requirements:

UC 0112, UC 0114, UC 0116, UC 0118, UC NTF010, VICINITY-B&D-BLD06, VICINITY-B&D- ENR06

VICINITY-FUNC-UCR020	**Device visibility and accessibility set-up within neighbourhood.**

VICINITY should make visible devices and their data sources within the neighbourhood, and provide access to these devices and information. The visibility and accessibility of any device should be distinguished on the following levels:

- Public visibility and public access;
- Public visibility and access for partners;
- Public visibility and access upon request;
- Public visibility and private access;
- Visibility to partners and access for partners;
- Visibility to partners and access upon request;
- Visibility to partners and private access;
- Private visibility and private access.

Considered requirements:

UC SEC050, VICINITY-BR-040

VICINITY-FUNC-UCR030 Request based device accessibility.

VICINITY should provide means to manage access to any devices based on specific approval of access requests sent to the IoT Operator. VICINITY should provide means to send, approve, reject access requests and revoking approved access request.

Considered requirements:

UC 0134, UC 0136, UC 0138, UC0140, UC NTF010, VICINITY-B&D- ENR06, VICINITY-B&D- HLT04, VICINITY-BR-040

VICINITY-FUNC-UCR040 Value added service visibility and accessibility set-up within neighbourhood.

VICINITY should allow configure visibility and accessibilty to any value added service within the neighbourhood. The visibility and accessibility of any value added service should be distinguished on the following levels:

- Public visibility and public access;
- Public visibility and access for partners;
- Public visibility and access upon request;
- Public visibility and private access;
- Visibility to partners and access for partners;
- Visibility to partners and access upon request;
- Visibility to partners and private access;
- Private visibility and private access.

Considered requirements:

UC SEC040, VICINITY-B&D- ENR06, VICINITY-BR-040

VICINITY-FUNC-UCR050 Value added service accessibility request.

VICINITY should provide means to manage access to any value added service based on specific approval of access request sent to the IoT Operator. VICINITY should provide means to send, approve, reject access requests and revoking approved access request.

Considered requirements:

UC 0154, UC 0156, UC 0158, UC0160, UC NTF010, VICINITY-B&D- ENR06, VICINITY-B&D- HLT04, VICINITY-BR-040

5.1.2. Value added service lifecycle requirements

This set of requirements defines the registration functionality of a new value added service, its profile management in VICINITY and its removal when it is not necessary anymore.

VICINITY-FUNC-UCR060	**VICINITY should support the registration of a new value added service.**

VICINITY should provide means to register a new value added service. Registration might be manual or automatic.

The registration process should include at least:

- the semantic matching of the value added service,
- creating value added service profile and
- initialize the value added service visibility and accessibility.

Considered requirements:

UC 0310, UC 0320, UC 0425

VICINITY-FUNC-UCR070	**VICINITY should support value added service profile management.**

VICINITY should manage (create, provide, update, remove and list) the profile for each value added service visible and accessible in VICINITY. The value added service profile should include at least:

- service name;
- service avatar;
- type of service;
- service group membership;
- list of data sources;
- identification of data owners;
- authorization rules and privacy attributes.

Considered requirements:

UC 0315, UC 0320, VICINITY-BR-ENR050

VICINITY-FUNC-UCR080	VICINITY should support value added service semantic data mediation mapping.
VICINITY should provide means to semantically map the value added service and provide semantic data mediation mapping of services throughout the whole lifecycle of the device including device registration or device change.	
Considered requirements:	
UC 0315, UC 0310, ORE_001, ORE_002, ORE_004, ORE_004, ORE_005, ORE_006, ORE_007, ORE_009, ORE_010, ORE_011, ORE_012, ORE_014, ORE_015, ORE_016, ORE_017, ORE_018, ORE_020, ORE_021, ORE_022, ORE_024, ORE_026, ORE_027, ORE_028, ORE_030, ORE_031, ORE_033, ORE_035, ORE_036, ORE_038, ORIO_001, ORIO_005	

VICINITY-FUNC-UCR090	VICINITY should support removal of the value added services.
VICINITY should support removal of any value added service. VICINITY should notify the service consumers. VICINITY should store trace audit trail for each service removal.	
Considered requirements:	
UC 0330	

5.1.3. Device lifecycle requirements

This set of requirements defines functionality for registration of a new IoT device, its profile management in VICINITY and its removal when it is not necessary anymore.

VICINITY-FUNC-UCR100	VICINITY should support registration of the new IoT device.
The VICINITY should provide means to register a new IoT device. Registration might be manual or automatic. The registration process should include at least: • the semantic matching of the IoT device, • creating the IoT device profile and • initialize the IoT device visibility and accessibility.	
Considered requirements:	
UC 0210	

VICINITY-FUNC-UCR110	VICINITY should support the IoT device profile management.

VICINITY should manage (create, provide, update, remove and list) a profile for each IoT device visible and accessible in VICINITY. The IoT device profile should include at least:

- device name,
- device avatar,
- type of device,
- device group membership,
- list of data sources,
- identification of data owners,
- authorization rules and privacy attributes.

Considered requirements:

UC 0215, UC 0220, VICINITY-BR-ENR050

VICINITY-FUNC-UCR120	VICINITY should support IoT device's semantic data mediation mapping.

VICINITY should provide means to semantically map the IoT device and provide semantic data mediation mapping for devices throughout the whole lifecycle of the device including device registration or device change.

Considered requirements:

UC 0215, UC 0210, UC 0425, ORE_001, ORE_002, ORE_003, ORE_004, ORE_005, ORE_006, ORE_007, ORE_009, ORE_010, ORE_011, ORE_012, ORE_013, ORE_014, ORE_015, ORE_017, ORE_019, ORE_020, ORE_021, ORE_025, ORIO_002, ORIO_004

VICINITY-FUNC-UCR130	VICINITY should support removal IoT devices.

VICINITY should support removal of IoT devices. VICINITY should notify IoT device data consumers. The VICINITY should store the audit trail for each IoT device removal.

Considered requirements:

UC 0230

5.1.4. VICINITY Clients requirements

VICINITY-FUNC-UCR140	**VICINITY should provide means to manage lifecycle of the VICINITY Client.**

The VICINITY Client lifecycle should include at least:

- Installing of VICINITY Agent;
- Register VICINITY Agent/Adapter;
- Manual removal or uninstalling of the VICINITY Agent from VICINITY;
- Removal of the VICINITY Adapter;

Events of the VICINITY Client lifecycle changes should be stored in audit trails.

VICINITY should provided status of each VICINITY Client such as online, offline, unknown.

Considered requirements:

UC 0420, UC 0423, UC 0425, UC 0430, UC 0440, UC 0447, UC 0448, UC 0460

VICINITY-FUNC-UCR145	**VICINITY Client should provide means to facilitate exchange of data between integrated infrastructures using semantic interoperability.**

The VICINITY Client should be able to facilitate data exchange (such as data measurements, data measurements history, device comments) between other VICINITY Clients.

The communication protocols should be standardized per domain. The communication channels should be in line with interoperability setup rules.

The VICINITY Client should be able to simulate supported type of IoT objects from neighbourhood in integrated infrastructure.

Considered requirements:

UC 0450, UC 0455, ORE_003, ORE_008, ORE_013, ORE_014, ORE_015, ORE_028, ORE_029, ORIO_004, ORIO_005, VICINITY-BR-BLD010, VICINITY-BR-BLD020, VICINITY-BR-BLD030, VICINITY-BR-BLD060, VICINITY-BR-BLD080, VICINITY-BR-ENR010, VICINITY-BR-ENR020, VICINITY-BR-ENR030, VICINITY-BR-ENR040, VICINITY-BR-TRA070, VICINITY-BR-TRA110, VICINITY-BR-OR001, VICINITY-BR-OR002, VICINITY-BR-OR003, VICINITY-BR-OR004, VICINITY-BR-OR005, VICINITY-BR-OR006, VICINITY-BR-OR007, VICINITY-BR-OR008, VICINITY-BR-OR009, VICINITY-BR-OR010, VICINITY-BR-OR011, VICINITY-BR-OR012, VICINITY-BR-OR014, VICINITY-BR-OR015, VICINITY-BR-OR016, VICINITY-BR-OR017, VICINITY-BR-OR019, VICINITY-BR-OR024, VICINITY-BR-OR026, VICINITY-BR-OR028, VICINITY-BR-OR029, VICINITY-BR-OR030, VICINITY-BR-OR031, VICINITY-BR-OR032, VICINITY-BR-OR033, VICINITY-BR-OR034, VICINITY-BR-OR036, VICINITY-BR-OR037, VICINITY-BR-OR039, VICINITY-BR-010, VICINITY-BR-040

5.1.5. Supporting requirements

VICINITY-FUNC-UCR150	**VICINITY should support grouping of IoT objects.**
The grouping of IoT objects enables to perform one action on the whole group. The grouping should support: • create a group; • add and remove objects from the group; • remove a group.	
Considered requirements:	
UC GRP010, UC GRP020	

VICINITY-FUNC-UCR160	**VICINITY should support searching functionality for IoT objects.**
VICINITY should support users to search IoT objects at least based on: • name; • type; • location.	
Considered requirements:	
UC SRC010	

VICINITY-FUNC-UCR165	**VICINITY should support digital signature of terms and conditions to connecting to VICINITY.**
VICINITY should support at least: • Management of terms and conditions to connecting to VICINITY; • Confirmation of terms and conditions by VICINITY Organization Administrator.	
Considered requirements:	
VICINITY-BR-HLT040	

VICINITY-FUNC-UCR170	**VICINITY should support management of access to IoT devices and value added services based on terms and conditions.**

VICINITY should support at least:

- Association of terms and conditions with a device or a service;
- Confirmation of terms and conditions.

Considered requirements:

UC LEG010, UC LEG020

VICINITY-FUNC-UCR180	**VICINITY should support the management of consent to process private data.**

VICINITY should support to process private data by:

- Provide consent for processing of private data;
- Revoke consent for processing of private data.

Considered requirements:

UC PRV010, UC PRV050, VICINITY-BR-TRA020, VICINITY-BR-TRA040

VICINITY-FUNC-UCR190	**VICINITY should support management of users and organizations and their profiles and roles.**

VICINITY should support users’ life cycle at least with the following functionalities:

- User registration in VICINITY;
- User profile editing and visibility set-up;
- User role management;
- User removal.

VICINITY should support organization life cycle at least with following functionalities:

- Organization registration;
- Organization profile editing and visibility set-up;
- Organization’s user association;
- Organization removal.

Considered requirements:

UC SEC030, UC SEC060, UC COM030, UC COM040, UC COM050, UC COM070, UC COM120, UC COM160

5.2. Quality considerations and non-functional requirements

5.2.1. User experience

An excellent user experience with the VICINITY solution should increase the user efficiency and satisfaction, reduce users training and support costs. Thus, functional design should bear in mind the look and feel of the application and usability.

The user experience is mostly covered by learnability, efficiency and memorability requirements. Learnability minimises the learning curve of the basic features and memorability focus on minimising the re-learning curve. Efficiency requirements define the time needed to perform tasks once the user has mastered them. Challenges posed by user experience can be addressed by designing the solutions around information users need to access. Moreover, functionality should be designed based on best practices in particular functional domains (such as: configuration files for administrators personal and a simple user interface for managing neighbourhood).

VICINITY-NFUNC-USR010	**VICINITY should support easy to learn approach for any VICINITY User.**
VICINITY design should be support: • look and feel helping users (regardless of their role) to easily identify possible actions; • reuse widely implemented use interface design patterns.	
Considered requirements:	
VICINITY-B&D-BLD07, VICINITY-B&D-TEC01, ORE_039, ORI_001, ORI_003, ORI_006, VICINITY-BR-TRA060, VICINITY-BR-HLT020, VICINITY-BR-OR025	

VICINITY-NFUNC-USR020	**VICINITY should support efficient usage by any VICINITY User.**
VICINITY design in terms of functionality, look and feel should support users (regardless of their role) to perform learned basic tasks quick with minimum error (such as: personalization of displayed information, localization of the user interface, user interface responsive to type of device and its display, etc.).	
Considered requirements:	
VICINITY-B&D-BLD07, VICINITY-B&D-TEC01, ORE_039, ORI_001, ORI_003, ORI_006, VICINITY-BR-TRA060, VICINITY-BR-HLT020, VICINITY-BR-OR025	

VICINITY-NFUNC-USR030	**VICINITY should support memorable design by any VICINITY Users.**
VICINITY design in terms of functionality, look and feel should support users (regardless of their role) to perform basic tasks quickly with minimum error, such that users can recall the process to follow, even after a period no using the interface.	
Considered requirements:	
VICINITY-B&D-BLD07, VICINITY-B&D-TEC01, ORE_039, ORI_001, ORI_003, ORI_006, VICINITY-BR-TRA060	

5.2.2. Technical requirements

5.2.2.1. *Performance*

Performance requirements define time (speed, latency) and space (scalability, capacity, stability) characteristics of the VICINITY platform. By identifying such constraints, the tasks being handled by the VICINITY framework will be able to operate under optimal conditions towards accomplishing the designated functions in a fast, scalable, stable and reliable manner. To that end, a set of requirements have been identified either from previous experiences of related tasks or from the relevant stakeholder engagement as described in D1.2.

The identified performance constraints that can be taken into consideration while designing, implementing and evaluating the VICINITY framework are described below:

VICINITY-NFUNC-PER010	**Response time of the VICINITY framework should be sufficient to support robust real-time operation with minimum latency.**

The VICINITY components should ensure that the communication among them or with their respective databases or third party software will provide real-time[3] data exchange supporting the following:

- Real-time search/querying in stored semantic data towards allowing smooth user experience;
- The semantic search should cause the minimum delay to the system's operation (should be put in another asynchronous thread to not delay the process);
- Fast enough response time should allow the creation of value added services that depend on processing of real-time data streams (e.g. triggering of events on abnormal data sequence detection);
- Data prioritization for safety or emergency situations.

Considered requirements:

VICINITY-B&D- TEC02, ORI_005

[3] Meaning real-time depends on the application context of a data exchange. For room temperature control application, real-time data exchange response time 30 seconds might be enough. However, VICINITY User would consider response time of IoT device search as sluggish.

VICINITY-NFUNC-PER020	**The VICINITY Platform should be scalable to unobtrusively cope with various operation levels and under strain conditions.**
The overall VICINITY framework system functionality and/or capability (e.g. responsiveness, stability, robustness, etc.) should not be compromised under extended usage (i.e. huge number of concurrent users). Furthermore, the system should be able to face unanticipated peak loads (i.e. number of concurrent users accessing simultaneously the system due to unpredictable spikes), without deteriorating in terms of efficiency, functionality, speed, etc.	
Considered requirements:	
ORI_005	

VICINITY-NFUNC-PER030	**The integrated VICINITY framework as well as the VICINITY components should be able to perform with great stability and cohesion.**
The system should able to run for long periods[4] of time, without data corruption, slowdown, or servers needing to be rebooted. Each VICINITY component should be able to handle internally erroneous issues ensuring stability for the integrated overall framework.	
Considered requirements:	
ORI_005	

[4] The length of the period depends on VICINITY components. Components which are not under control of VICINITY maintenance personnel (such as VICINITY Agent) should be able to run very long periods (e.g. months or year). Components under control of VICINITY maintenance personnel will be proactively monitored.

VICINITY-NFUNC-PER040	**The overall VICINITY system should support recover from out of service periods.**

The VICINITY framework should be reliable and robust and therefore should support to overcome the following situations:

- If the VICINITY framework or one of its components comes back online after scheduled or unscheduled downtime, the users should be able to see/do everything they expect (system able to resume at the correct point).
- if some parts of the network are unreachable/disconnected, the system should be able to manage the whole operation with historically stored/temporary data.
- If one of the IoT Infrastructures transmits unreliable data due to issues with the connection or to a malfunction of a component out of the VICINITY framework, the system should be able to interpret (to a degree) the data using historically stored information to continue its operation.

Considered requirements:
ORI_005

VICINITY-NFUNC-PER050	**The VICINITY system should be able to store and retrieve large amounts of meta-data**

Due to the nature of the VICINITY platform, large amounts of data and meta-data (actual IoT data should be stored separately) are expected to be exchanged in real-time operation. Thus, it is essential that the information exchange within the VICINITY framework should not be affected by the amount of data.

Considered requirements:
ORI_005

Note, that these performance requirements define constraints for VICINITY components. There are also requirements abstracted from environment conditions and constraints which cannot be influenced (e.g. low bandwidth between VICINITY Clients, etc.).

5.2.2.2. *Availability*

Availability of the system defines the system status when a user can obtain the promised services/ goods/ products of the system. If the system cannot deliver the intended service is considered to be "unavailable". Availability of the system is influenced by various factors such as: user's device used to access VICINITY, communication between device and VICINITY, VICINITY software components and hardware platform where VICINITY components are running, etc. Some of these factors such availability of VICINITY software components and hardware platform can be fully or partially influenced by the design and implementation of the VICINITY service. However, factors such as user's device or connected interoperable infrastructures, quality of connection cannot be influenced. Thus, the VICINITY availability should be calculated from the availability of VICINITY software components, platform where VICINITY software components are deployed and communication connections between components within the platform and connection to closest public internet connection.

Based on findings during stakeholders' workshop summarized in D1.2 the availability of the VICINITY services should be 24x7 including acceptable downtimes according to application context.

High-availability of VICINITY services can be achieved only if the high-available principles are applied during design, implementation and deployment of VICINITY components.

VICINITY-NFUNC-AVL010	**VICINITY should support 24x7 service availability with acceptable downtime 2%.**
The VICINITY service availability should be achieved through high-available VICINITY components and underlying infrastructure such as: • VICINITY components horizontal and vertical scalability or • infrastructure service load balancing and virtualization.	
Considered requirements:	
ORI_009	

VICINITY-NFUNC-AVL020	**Principles for elimination of single point of failure should be applied during design VICINITY components.**
Elimination of the single point of failure and reliable crossovers means that failed component does not mean failure of the entire system.	
Considered requirements:	
ORI_009	

VICINITY-NFUNC-AVL030	**Means to recognize VICINITY components failure or other issues should be provided by VICINITY.**
VICINITY should provide means (such as: activities logins, monitoring interfaces, etc.) that can provide: • information about important events, • warning about system functionality degradation, • failure of one functionality, component or whole VICINITY solution. VICINITY should be able to monitor the external interfaces as well as internal failures, changes on these interfaces can cause degradation or failure of VICINITY functionalities.	
Considered requirements:	
ORI_009	

5.2.2.3. *Maintainability*

Maintainability requirements defines ability of the system to react on changes in environment where system is deployed, changes and issues in system itself. Maintenance requirements influence the structure of the system, operational functionalities of the system and requirements.

VICINITY-NFUNC-MNT005 **VICINITY Adapter should be implemented below average software engineer**

VICINITY Adapter will be implemented in an integrated infrastructure by personnel who do not have extensive knowledge in VICINITY Architecture, IoT interoperability and semantics, etc.

Implementation of the VICINITY Adapter should be supported by:

- guidelines and
- examples[5].

Considered requirements:

ORI_002

VICINITY-NFUNC-MNT007 **VICINITY should adopt modular architecture**

VICINITY architecture should be broken down into self-contained modules, which can be separately:

- designed,
- implemented,
- tested,
- validated,
- updated and
- in certain cases replaced.

Considered requirements:

ORI_003

[5] Automatic configuration of the VICINITY Adapter and VICINITY Gateway API should be considered during VICINITY Architecture design.

VICINITY-NFUNC-MNT010	**VICINITY Components should provide means to identify and resolve resource budget issues**

VICINITY should provide means (such as: activities logins, monitoring interfaces, etc.) that can provide resource budgets (current, average) on level of:

- Hardware (such as CPU, Memory, Storage space, Bandwidth and energy consumption);
- Platform on which VICINITY components are running or using (such as platform resource usage, data fragmentation, event measures, etc.);
- VICINITY components' specific resources and internal / external VICINITY interfaces;

Considered requirements:

ORI_003

VICINITY-NFUNC-MNT020	**VICINITY Components should be designed to anticipate most probable resource budget issues.**

Design of VICINITY Components should consider best practices to reduce impact of the resource budget on various level:

- HW or ICT level (e.g. DNS load balancing);
- Virtualization or platform as a service (e.g. dynamic resource provision);
- Component architecture (e.g. component vertical or horizontal clustering).

Considered requirements:

ORI_003

VICINITY-NFUNC-MNT030	**VICINITY should adopt and use open & interoperable standards on internal and external interfaces or isolate dependencies by design to support reusability and portability.**

VICINITY should adopt existing open & interoperable standards or "de-facto" standards on following layers:

- HW, ICT and Platform;
- Application / Service level.

On application / service level VICINITY should apply dependencies isolation techniques such as adapter or bridge patterns.

Considered requirements:

ORI_004, ORI_007

VICINITY-NFUNC-MNT040 **VICINITY should provide means to migrate its components and data to different version.**

Each component and interface of VICINITY tends to change in time, thus there should be means such as:

- guidelines,
- tools or
- automatic migration mechanisms

to allow updating of components and managed data.

VICINITY Component should support backward compatibility on external interfaces by design.

Considered requirements:

ORI_004

VICINITY-NFUNC-MNT050 **VICINITY should provide tools to simplify the installation of VICINITY components**

VICINITY should provide means to build, package, deploy/ install VICINITY components. Building, packaging and deploying / installing should preferably be implemented by automated tools (such as: mobile device application stores, platform deployment tools offered by Heroku cloud platform, maven/npm packagers, installation guides, etc.).

Considered requirements:

ORI_003

VICINITY-NFUNC-MNT060	**VICINITY Components should allow changing their behaviour off-line or in runtime.**

A VICINITY solution and its components should be designed to support components functionality changes through design patterns introducing high flexibility and configurability such as: integration patterns, plug-in architecture, dependency injection, etc.

From an implementation point of view, the VICINITY solution should be able to configure its interfaces (internal / external) and components itself during run-time or offline. Configured attributes might be:

- HW, ICT or PaaS level: number of servers, processors, memory being used, load balancers;
- Data storage level: connections strings, authentication mechanism configuration (certificates, credentials);
- Components level: logging configuration (file location / level), vertical/ horizontal cluster configuration, external and internal interfaces configuration (such as location), other application specific settings.

Considered requirements:

ORI_003

5.2.2.4. *Security considerations*

The VICINITY is potentially exposed to security threats including:

- Masquerade;
- Eavesdropping;
- Unauthorized access;
- Loss or corruption;
- Repudiation;
- Forgery;
- Denial of service.

According to ITU-T Rec. E.408 (05/2004)[6] the following security measures minimize the impact of the threats:

- Verification of identities of VICINITY users, VICINITY Clients infrastructures and IoT objects;
- Controlled access and authorization of VICINITY users and VICINITY Clients;
- Protection of confidentiality of information exchanged between VICINITY Clients infrastructures and with VICINITY in general;
- Protection of data integrity exchanged between VICINITY Clients and meta-data integrity stored within VICINITY;
- Accountability services that VICINITY users and VICINITY Clients are responsible only for activities they performed;

[6] http://www.itu.int/rec/T-REC-E.408-200405-I

- Activity logging and audit of operation performed by VICINITY users and VICINITY Clients on VICINITY components;
- Alarm reporting of potential selected security breaches.

5.2.2.4.1. Verification of identities

VICINITY-NFUNC-SEC010	**VICINITY should provide capabilities to establish and verify the claimed identity and integrity of any actor interacting with any external or internal interface of VICINITY Components.**

Identities of actors that access external or internal interfaces of VICINITY Components should be verified. Three types of identity verification might be considered:

- User authentication
- Peer entity authentication and
- Data origin authentication.

Authentication should provide proof on the actor's identity for the certain instance of a time and identity should be repeatedly verified.

The credential management capabilities should be provided to generate, change, recover, revoke and delete credentials.

For each authentication mechanism, a specific policy needs to be defined.

Threats addressed: masquerade, unauthorized access

Considered requirements:

VICINITY-BR-SEC010, VICINITY-BR-SEC020, VICINITY-BR-SEC030, VICINITY-BR-SEC040, VICINITY-B&D-SEC04, ORI_008, VICINITY-BR-TRA050, VICINITY-BR-020

Note, that verification of identities such as IoT objects should be performed as well, e.g. IoT objects verification during discovery.

5.2.2.4.2. Controlled access and authorization

Support for both identity-based and role-based access management should be provided by VICINITY. Access control should enable fine grained definition of which IoT objects (their data fields if necessary) or meta-data elements can be visible for different actors

Management of access control privileges associated with the entities that are authorized for access, privilege management should be utilized.

VICINITY-NFUNC-SEC020	**VICINITY should provide capabilities to ensure that actors are prevented from gaining access to information or resources that they are not authorized to access.**

The privilege management enabling mechanisms should have following properties:

- Dedicated owner
- Reasonable defaults
- Explicit grant
- Privilege granting only by the owner
- Privilege recovery

Threats addressed: Masquerade, Denial of service

Considered requirements:

VICINITY-BR-SEC010, VICINITY-BR-SEC020, VICINITY-BR-SEC030, VICINITY-BR-SEC040, VICINITY-BR-SEC050, VICINITY-B&D-SEC01, ORI_008, VICINITY-BR-HLT060, VICINITY-BR-020, VICINITY-BR-040

Note, that a blocking mechanism for IoT objects, and any request sending should be considered as well in detailed design to mitigate of the spamming or flooding risk.

Note, that in many countries, there is a legal requirement that historic meta data which relates to communications relevant to an identifiable individual must be made available to security services as necessary (see UK Regulation of Investigatory Powers Act 2000).

5.2.2.4.3. Protection of confidentiality

VICINITY-NFUNC-SEC030	**VICINITY should provide capabilities to ensure the confidentiality of stored and communicated data.**

VICINITY should ensure that data is not disclosed to any actors unless they have been authorized to access those data.

The level of confidentiality should be considered on the following levels:

- System confidentiality (e.g. reducing the knowledge of security architecture, technology (including version) used);
- Connection level (e.g. end-to-end security);
- On attributes level (e.g. confidentiality of selected attributes of data).

Considered requirements:

VICINITY-BR-SEC010, VICINITY-BR-SEC020, VICINITY-BR-SEC030, VICINITY-B&D- ENR07, ORI_008, VICINITY-BR-ENR060, VICINITY-BR-BLD040, VICINITY-BR-TRA040, VICINITY-BR-020

5.2.2.4.4. Protection of data integrity

Data integrity function is to ensure that the data has not been changed, destroyed, or lost in an unauthorized or accidental manner.

VICINITY-NFUNC-SEC040	**VICINITY should be able to guarantee the integrity of systems and stored and communicated data.**

The VICINITY should consider following data integrity properties:

- Arrange for modifications to be detectable;
- Standard cryptographic one-way hash functions and digital signature algorithms;
- Openness, e.g. all algorithms shall be published;
- No proprietary verification requirements;
- Different algorithm for digital signatures (data integrity) and encryption (data confidentiality).

Threats addressed: Loss or corruption of information

Considered requirements:
VICINITY-BR-SEC020, VICINITY-BR-SEC030, VICINITY-B&D-SEC02, ORI_008, VICINITY-BR-020

5.2.2.4.5. Accountability

VICINITY-NFUNC-SEC050	**VICINITY should provide means so that an entity cannot deny the responsibility for any of its performed actions as well as their effects.**

Its function is to ensure that the actions of a system entity may be traced uniquely to the originating entity, which can then be held responsible for its actions.

It is important that all actions shall be attributable to an authenticated entity.

Threats addressed: Repudiation and forgery

Considered requirements:
VICINITY-BR-SEC040, VICINITY-B&D-LEG05, ORI_008, VICINITY-BR-TRA040, VICINITY-BR-020

5.2.2.4.6. Activity logging and audit

Activity logging and audit function is to provide chronological record of system activities that is sufficient to enable the reconstruction and examination of the sequence of environments and activities surrounding or leading to an operation, procedure, or event in a security-relevant transaction from inception to results. Such audit information can be used to analyse security issues or disputes[7].

[7] In some countries there is an obligation to provide meta-data to security services upon reasonable request.

VICINITY-NFUNC-SEC060	**VICINITY should provide a capability to track activities on the system with a record of individuals or entity that instigated the activity.**

The activity logging and audit should have following properties:

- "Append only" mode;
- Detectable modifications;
- Precise ordering and timing;
- Open and standardized format;
- Multi-tenant support;

Threats addressed: Masquerade, Unauthorized access, Repudiation, Forgery and Denial of Service

Considered requirements:

VICINITY-BR-SEC040, VICINITY-BR-SEC050, VICINITY-B&D- HLT03, VICINITY-B&D-SEC03, ORI_008, VICINITY-BR-HLT030, VICINITY-BR-020

5.2.2.4.7. Alarm reporting

Alarm reporting function is to send notification of selected security event (possible breach of security) to selected recipients.

VICINITY-NFUNC-SEC070	**VICINITY should provide means to send notification in case selected security event occurs.**

Notification should be attributed to the subject and object of a security event including the context in which the event occurred.

Any alarm should be subject of activity logging and audit.

Threats addressed: Masquerade, Unauthorized access, Lost or corruption of information and Denial of Service.

Considered requirements:

VICINITY-BR-SEC010, VICINITY-BR-SEC020, VICINITY-BR-SEC030, VICINITY-BR-SEC040, VICINITY-BR-SEC050, ORI_008

5.2.2.5. *Privacy considerations*

Privacy by design[8] must be the heart of the VICINITY architecture and implementation. Privacy requirements define:

- constraints for processing of sensitive data within the VICINITY;
- means to foster privacy between integrated VICINITY Clients infrastructures.

Data exchanged between integrated VICINITY Clients infrastructures can be twofold:

- information provided by shared IoT objects (such as IoT devices or value added services) and,
- meta-data defining means to exchange information (such as information type, location, events accepting, security measures applied, etc.).

Note, that information provided by shared IoT objects may include sensitive information such as private data. On the other hand, it is assumed that some meta-data cannot be linked to an identifiable person, or components known to be owned by an identifiable person. Such meta-data is considered not to include any sensitive data. It contains only meta-data necessary to exchange information.

Interoperability between VICINITY Clients' infrastructures should be set-up based on impersonal meta-data information provided by IoT objects. Based on the interoperability set-up the VICINITY set-up communication channels and facilitates exchange of information between VICINITY Clients' infrastructure.

Separating the information and meta-data enables to facilitates exchange of information between VICINITY Clients' infrastructures based on meta-data without need to know the content of exchanged information is an aspect of "privacy by design";

VICINITY-NFUNC-PRV010	**VICINITY should only process and manage meta-data necessary to facilitate exchange of information between VICINITY Clients' infrastructures from privacy point of view.**

Meta-data describing exchanged information should include at-least:

- identification of IoT Object providing information;
- location of IoT Object (such as URI);
- meta-data about information structure (such as encoding);
- properties of information (such as name, units, etc.);
- security means description to access information.

Considered requirements:
ORI_008, VICINITY-BR-020

Exchanging sensitive information should be done on peer-to-peer basis via interoperability set-up. The interoperability is set-up based on willingness of information exchange, however it can be guided based on sensitivity of the information and trust between peers. Handling of higher sensitivity

[8] https://www.iab.org/wp-content/IAB-uploads/2011/03/fred_carter.pdf

information is directly proportional to need for higher trust between peers. While the level of sensitivity information and trust between peers is strongly subjective, VICINITY should provide the following means to support interoperability set-up decisions.

The Data controllers should provide information on how they process sensitive information for IoT object. Based on this information the data subject can decide to share its sensitive information by providing a consent to process private or sensitive information.

Moreover, data controllers should provide means to manage sensitive information collected and erase that information for other data subjects.

5.2.2.5.1. Private data processing information

VICINITY-NFUNC-PRV020	**VICINITY should provide means to facilitate availability of private data processing information.**
The service provider or IoT Operator should provide to device owners information on how the private data are being processed in VICINITY Client infrastructures. The availability of data processing information is not mandatory. The provided information should be in-line with EC Regulation 2016/679[9] (GDPR).	
Considered requirements:	
ORI_008, VICINITY-BR-020	

5.2.2.5.2. Consent of data subject

VICINITY-NFUNC-PRV030	**VICINITY should provide means to create, submit and withdrawn of consent to process private data.**
The consent should be submitted by a device owner to allow the processing of the potential private data by service provider or IoT Operator. The consent may be withdrawn by the device owner. The consent withdrawal might require erasure of all historic private data. The content of the consent should be in-line with EC Regulation 2016/679. (GDPR)	
Considered requirements:	
VICINITY-B&D-LEG02, ORI_008, VICINITY-BR-020	

[9] http://ec.europa.eu/justice/data-protection/reform/files/regulation_oj_en.pdf

5.2.2.5.3. *Access to private data, their rectification and process restriction*

VICINITY-NFUNC-PRV040	**VICINITY should provide means to facilitate exchange of private data between a device owner and the service provider or IoT Operator.**

VICINITY should provide means for the device owner to facilitate:

- Access to private data;
- Private data rectification, where the data subject or owner detects errors;
- Private data process restriction.

The private data provision or rectification can be manual or supported by automatic means of VICINITY Client infrastructure.

Considered requirements:

VICINITY-B&D-LEG01, ORI_008, VICINITY-BR-020

5.2.2.5.4. *Private data erasure*

VICINITY-NFUNC-PRV050	**VICINITY should provide means to facilitate device owner to erase private data process by service provider and/or IoT Operator.**

The private data erasure can be manual or supported by automatic means of VICINITY Client infrastructure.

The provided information should in-line with EC Regulation 2016/679. (GDPR)

Considered requirements:

VICINITY-B&D-LEG04, ORI_008, VICINITY-BR-020

5.2.3. Legal and Standardization requirements

This section is focusing on the VICINITY solution legal (VICINITY Licensing rules) and standardization (reuse and openness) principles, which should be followed throughout its detailed design and implementation.

5.2.3.1. *VICINITY Licensing*

VICINITY-NFUNC-LCS010	**VICINITY components for the Pilot trials should be licensed on terms in-line with consortium agreement.**
Consortium agreement covers the exploitation and access right principles and background within consortium. Moreover, VICINITY components should be in-line with dissemination level. Example of open source licenses can be found in the ANNEX I.	
Considered requirements:	
ORI_002	

5.2.3.2. *Standardization requirements*

Standards relevant for the VICINITY project have been elaborated in deliverable D2.1 Analysis of Standardisation Context and Recommendations for Standards Involvement. Standards relevant to VICINITY components should be considered in the architecture design, detailed design and implementation. For selection of the standards the following principles should be followed:

VICINITY-NFUNC-STD010	**VICINITY should consider reuse of the existing, and most used ontologies in building, energy, transport, health and IoT domain.**
The goal of VICINITY is not to build a new VICINITY Ontology from the scratch, however to reuse the existing most used or de facto standard ones.	
Considered requirements:	
VICINITY-B&D- HLT05, VICINITY-BR-HLT050, VICINITY-BR-TRA100, VICINITY-BR-040	

VICINITY-NFUNC-STD020	VICINITY should consider reuse of open industrial standard security and privacy solutions, methods and approaches.

The aim of the VICINITY is to reuse existing open standardized solutions (such as OpenSSH, OpenId), methods (such as ITU-T Recommendation E.408 [10]) and approaches (such as HTTP over TLS, SingleSignOn) in security and privacy.

Considered requirements:

ORI_002

Note that best practice in Privacy by Design is still being determined, for example work started in December 2016 in the IEEE p7002 question about Privacy by Design.

VICINITY-NFUNC-STD030	VICINITY should consider reuse of open industrial standards for internal and external communication interfaces.

The open communications standards should be used whenever possible to ensure solution openness, flexibility and maintainability.

Considered requirements:

VICINITY-B&D-TEC03

Architecture and software engineering standards elaborated in D2.1 have been selected to use common domain language (such as Internet of Things Reference Architecture) and representation notations (such as UML).

5.3. Value added services requirements

Value added service are from VICINITY solution point of view external components which will be connected to VICINITY through VICINITY Agent or Adapter. These services should be able to exchange information with VICINITY Clients' infrastructures in a secure and privacy preserving way. Exchange of information should be facilitated by VICINITY. Thus, it is necessary to introduce the minimum set of high level functional, security and privacy requirements that should be supported by value added service.

[10] http://www.itu.int/rec/T-REC-E.408-200405-I

VICINITY-FUNC-VAS010	Value added service should support communication to exchange information through a VICINITY Agent or Adapter.

Value added service should be able to exchange information with other peers (VICINITY Clients' infrastructures) in VICINITY. The communication patterns, technology, encoding should be standardized.

Considered requirements:

ORE_001, ORE_002, ORE_004, ORE_004, ORE_005, ORE_006, ORE_007, ORE_009, ORE_010, ORE_011, ORE_012, ORE_014, ORE_015, ORE_016, ORE_017, ORE_018, ORE_020, ORE_021, ORE_022, ORE_024, ORE_026, ORE_027, ORE_028, ORE_030, ORE_031, ORE_033, ORE_035, ORE_036, ORE_038, ORIO_001, ORIO_005

VICINITY-NFUNC-VAS020	Value added service should provide meta-data about information being exchanged through a VICINITY Agent or Adapter.

Value added service meta-data should be described and used to facilitate interoperability within VICINITY. At least the following meta-data should be considered:

- Type of the service based on the VICINITY ontology;
- Name of the service;
- URIs of the service;
- Encoding of provided data from the service;
- Security properties of the service;
- Information sources properties (such as: type, name, units, etc.).

Considered requirements:

ORIO_004, ORIO_005

VICINITY-NFUNC-VAS030	Value added service should support authentication and authorization mechanisms to access meta-data and information.

Constrained access to data should protect value added service against unauthorized access to meta-data and/or information.

Considered requirements:

VICINITY-BR-SEC010, VICINITY-BR-SEC020, VICINITY-BR-SEC030, VICINITY-BR-SEC040, ORI_008

VICINITY-NFUNC-VAS040	Value added service should support end-to-end encryption of exchanged data with a VICINITY Agent and/or Adapter.
The exchanged data must be protected against disclosure to unauthorized identities. The protection should be considered on the level of connection and/or data.	
Considered requirements:	
VICINITY-BR-SEC010, VICINITY-BR-SEC020, VICINITY-BR-SEC030, ORI_008	

VICINITY-NFUNC-VAS050	Value added service should support privacy preserving of processed data.
The following privacy preserving features should be considered: • authorized access to private data; • rectification capability for private data; • restriction of processing of private data; • erasure of private data;	
Considered requirements:	
ORI_008	

6. Conclusion

The goal of the VICINITY Technical requirements specification is to define the functional design of the VICINITY solution including formalized functional requirements and non-functional requirements (quality considerations).

The VICINITY functional design was divided into the following main functional blocks (technical use cases): Connecting to VICINITY, Discovery of new devices and value added service and VICINITY Interoperability setup. Each functional block has identified actors such as:

- System integrator (Connecting IoT infrastructure into VICINITY),
- Service provider (Integrate new value added service in VICINITY),
- Device owner (Provide device and its related data within the neighbourhood)
- IoT Operator (Build device and service social network in neighbourhood).

These main technical use cases have been described in detail, focusing on their principal functions. Functional design is concluded with common (supportive) use cases in terms of legislation, security, privacy, user and group management.

Complementary to functional design, quality considerations are elaborated in following fields:

- security transparently protects VICINITY itself and exchanged data, including role based access to data, data integrity and end-to-end security;
- privacy by design to protect privacy when exchanged data between peers utilizing the VICINITY concept;
- user experience focuses on learnability, efficiency and memorability of the VICINITY solution;
- performance considers time and space characteristics of the VICINITY solution and its deployment environment;
- availability supports accessibility of the VICINITY solution services 24x7 with accessible downtimes according to application context;
- maintainability promotes a flexible and extensible VICINITY solution able to react to future changes;
- legal & standardization principles should be followed throughout detailed design and implementation of the VICINITY solution such as licensing, openness, reuse of industrial standards.

The technical specification is concluded with high-level requirements for the value added service to support their detail design and implementation in WP 5 “Value added Services Implementation”.

VICINITY Technical requirements specification is an input to the VICINITY Architecture design (D1.6) in for identification of:

- system components based on the functional design,
- high-level information model (including abstraction of physical devices and services),
- the internal and external interfaces resulted from detailed analysis of VICINITY behaviour and
- selection of architecture patterns based on non-functional requirements in security, privacy, user experience, performance, availability and maintainability fields.

ANNEX I Example of Open source licenses

This annex summarizes and elaborates[11,12] the most used open source licenses, however the extended lists of licenses are publicly available[13]. The following aspects have been elaborated on selected license models:

- Possibility to link with code published under a different license;
- Ability to release changes under a different license;
- Patent grant - protection of licensees from patent claims made by code contributors regarding their contribution, and protection of contributors from patent claims made by licensees;
- TM (Trademark grant) - use of trademarks associated with the licensed code or its contributors by a licensee;
- Private use - whether modification to the code must be shared with the community or may be used privately (e.g. internal use by a corporation);
- Possible modification of the code by a licensee;
- Releasing changes under different license.

Table 5 List of Open source licenses

	Apache 2.0	Eclipse	BSD	MIT	GNU GPL v3
Linking with code with different license	Yes	Limited, see [14]	Yes	Yes	No
Release changes under different license	Yes	Limited, see 1	Yes	Yes	No
Patent grant	Yes	Yes	Manually, see [15]	Manually, see [16]	Yes
Private use	Yes	Yes	Yes	Yes	Yes
Modification of the code by a licensee	Yes	Limited, see 1	Yes	Yes	Copylefted
GPL v3 compatibility	Yes (only GPL v3!!!)	No	Yes (modified BSD) No (original BSD)	Yes	Yes
TM	No	Manually	Manually	Manually	Yes
Release changes under different license	Yes	Limited, see 1	Yes	Yes	No

[11] https://opened.pressbooks.com/chapter/ open-software-licenses-high-level-comparison-of-open-licenses/

[12] http://forum.xprize.org/t/apache-public-license-vs-eclipse-public-license/1016/4

[13] https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses

[14] http://www.eclipse.org/legal/epl-v10.html

[15] https://opensource.org/licenses/BSD-3-Clause

[16] http://opensource.org/licenses/MIT