Horizon 2020
European Union Funding
for Research & Innovation

CPaaS.io
City Platform as a Service – Integrated and Open

NICT

H2020-EUJ-02-2016
H2020 Grant Agreement Number 723076
NICT Management Number 18302

Deliverable D3.1

Requirement Collection Analysis

Version V1.0

October 14th, 2016

ABSTRACT

This First WP3 deliverable provides a set of functional and non-functional requirements from the Platform perspective. It will be complemented with the requirements received from WP2/Task 2.1 (focussing on the Scenario point of view). Then, both requirement sets will be unified and analysed in the context of Task 3.2 before a first draft of the architecture is elaborated.

Disclaimer

Document Information

Editors	François Carrez (UNIS)
Authors	& Antonio Skarmeta (Odin)
Reviewers	Stephan Haller (BFH), Noboru Koshizuka (UoT)
Delivery Type	R
Dissemination Level	Public
Contractual Delivery Date	30th September 2016
Actual Delivery Date	14th October 2016
Keywords	Requirements, Requirement Engineering

Revision History

Rev.	Date	Description	Contributors
v0.1		Extended ToC	F. Carrez
v0.2	8/09/2016	Security and Requirement contributions	A. Skarmeta
v0.3	12/09/2016	Integration of additional contributions, Introduction/Conclusion	F. Carrez
v0.4	21/09/2016	Update of Security part	A. Skarmeta
v0.5	26/09/2016	Update of Functional Requirements (scenario)	F. Carrez
v0.6	28/09/2016	Consolidation / Cosmetics / Ready for Review	F. Carrez
v0.7	30/09/2016	Review	S. Haller
v1.0	13/10/2016	Final version for submission	F. Carrez & A. Skarmeta

Table of Contents

List of Tables

List of Acronyms

Acronym	Definition
AE	Augmented Entity
BC	Business Case
IoT ARM	IoT Architectural Reference Model
CEP	Complex Event Processing
CPU	Central Process Unit
CSF	Common Service Function
DC	Design Constraints
FC	Functional Component
FG	Functional Group
FREQ	Functional REQuirement
FV	Functional View
HSM	Hardware Security Module
IoT	Internet of Things
IoT-A	Internet of Things - Architecture
IV	Information View
LOD	Linked Open Data
M2M	Machine to Machine
NFREQ	Non Functional REQuirement
PE	Physical Entity
RDF	Resource Description Framework
SCP	(ETSI) Smart Card Platform
SotA	State of the Art
SPARQL	SPARQL Protocol and RDF Query Language
VE	Virtual Entity

1 Introduction

The purpose of this document is to provide the result of the part of the requirement process pertaining to WP3. Indeed it has been agreed between WP2 and WP3 that the former will focus mainly on collecting and unifying the sole requirements gathered at the various stakeholders associated with the CPaaS.io scenarios, while the later will focus on project technical objectives from the platform point of view

The tangible output of this document is therefore a list of functional and non-functional requirements from the platform perspective.

Those results together with requirements from D2.1 will be the essential starting point to task T3.2 which is about requirement analysis, mapping and architecture.

This document is organised as follows: Section 2 introduces the reader with the whole requirements process (requirement collection being the first step) that therefore spans tasks 2.1, 3.1 and 3.2. The description of the requirements is presented in Section 3. Finally Section 4 explains what the next steps are (as part of T3.2) following the two separate requirement collection phases as performed in T2.1 and T3.1.

2 Requirement process

This first main section explains in detail the requirement process as described in IoT-A [4] Deliverable D1.5 [1], Chapter 5. In this section we remind the main concepts and objectives related to requirements; the reader eager to learn more about this process may refer to the IoT-A deliverable for its complete description.

Some of the following steps may be achieved in parallel or at least with a large overlap.

The requirement process consists a requirement collection phase (see Section 2.1) and requirements engineering which itself consists of several steps (see Sections 2.2 to 2.5). Finally a threat analysis (see Section 2.6) phase is performed when a draft of the architecture and its main components has been made available.

2.1 Requirement Collection

The first phase of the requirement process comes with collecting requirements from the different project stakeholders: scenario driven and technology driven ones. The different sources of requirements in CPaaS.io are therefore:

- **Scenario requirements:** they focus on the scenario stories and objectives and identify all kinds of needs and expectations the scenario has, as far as the supporting CPaaS.io platform is concerned, in term of properties, functionalities, interfaces etc. Those scenarios do not focus much on the technology side but rather on the functional aspects (which functionalities are needed in order to implement the story) and also non-functional aspects like performance, privacy, deployment, scalability etc. issues; those requirements should reflect end-users concerns as well;
- **Platform requirements:** platform requirements capture the technical objectives that have been described along the Description of Work documents and which materialise the steps beyond the SotA that the project will undertake to implement;
- **Business requirements:** requirements that stem from business and/or exploitation perspectives;

Requirements are generally classified into three categories as follows:

- **Functional requirements:** directly refer to capabilities of the system in term of what it has to do. Functional requirements are used to build up the Functional View;
- **Non-functional requirements:** directly refers to what the system should be or which properties the system should enjoy. The non-functional requirements are used to build up the Information and Deployment & Operational Views and all system perspectives;
- **Technology / Design constraints:** They impose restrictions on how the system should be designed and which technology should be used and therefore influence both views and perspectives.

In order to facilitate the next steps of the requirement process, it is good practice to capture additional information about each requirement beyond its description and the category it falls into. The table below gives a comprehensive list of fields that are expected to be filled in by requirement issuers. This Table 1 below is taken from the IoT-A deliverable D6.1 and D6.2 [3][8] and the fields described in there are the ones used within the VOLERE template (see supporting tool section hereafter). The VOLERE template for CPaaS.io will be issued after requirements from T2.1 and T3.1 have been analysed and unified as part of task T3.2.

Table 1: VOLERE template Field description

FIELD	DESCRIPTION
ID	Each requirement is uniquely identified
Requirement Type	1. Functional Requirements (FREQ) 2. Non-Functional Requirements (NFREQ) 3. Design Constraints (DC) It is easier to write an appropriate fit criterion when the type of requirement is established. When one groups all of the known requirements of one type, it becomes readily apparent if some of them are missing or duplicated.
Description	The description is the intent of the requirement. It is a statement about what the system has to fulfil according to the rationale.
Category	Technical or non-technical topic covered by the requirement. List to be decided between partners
Rationale	The rationale is the reason behind the requirement's existence. It explains why the requirement is important and how it contributes to the system's purpose.
Owner / Source / Req. Name	The owner (originator) is the person or organisation who raised the requirement in the first instance, or the person to whom it can be attributed. You should attach the originator's name to the requirements so we have a referral point if questions about the requirement arise or if the requirement is rejected. The person who raises the requirement must have the knowledge and authority appropriate for the type of requirement.
Originating Scenario / BC	ID of the originating scenario / Business Case if relevant
Fit Criterion	A quantification or measurement to assess to which extent the original requirement has been eventually implemented/taken into account within the system. Can be used as the "Definition-of-Done" in agile development practices.
Priority	The priority of a requirement is the decision on the importance of the requirement's implementation. The priority depends highly on the specific domain of the application. Priority takes value from {MUST, SHOULD, MAY} with a decreasing priority level. MUST means compulsory while MAY remains fully optional.
Dependencies	Indicate if the requirement depends on another one. Relations between two or more requirements should be noted and separated by comma(s).
Conflict	Conflicts between requirements imply that there exists contradictions upon system implementation, or one requirement makes the implementation of another requirement less feasible. Values: default "(none)" or requirement number(s), separated by comma(s).
View	One or several views to which the requirement is related.
Functionality Group	One or several functionality groups in the functional decomposition to which the requirement is related.
Functional Component	One or several components in the functional decomposition to which the requirement is related. These functional components are part of the groups listed in the functionality-group field.
Domain Model	One or several domain-model entities to which the requirement is related.
Perspective	One or several perspectives to which a requirement is related.
System Use-Case	ID of the System Use Case that needs the requirement under consideration.
Comment	Any comment providing useful information

NOTE: Green fields are used during the requirement mapping phase (see below).

2.2 Requirement Analysis

As soon as the raw material has been collected (primitive requirements collected from various sources as explained in Section 2.1) there is a need for reviewing those requirements, removing duplicates, unifying/factorizing similar ones and translating/rewriting all requirements using unified terms and concepts as introduced within the IoT ARM (Domain Model in particular). The result of this phase is a collection of unified system requirements collected and summarised within the VOLERE template.

2.3 Requirement View Mapping

Functional unified requirements need to be mapped to the Functional View (FV) where one or more Functional Groups (FG) and Functional Components (FC) can be identified. This will result into a first functional decomposition of the targeted system.

On the other hand, some of the non-functional requirements can be mapped to the Information and Deployment & Operational Views.

2.4 Design Choices

Non-functional qualitative requirements must be mapped to appropriate perspectives. Then suitable tactics/strategies for fulfilling the requirement must be identified (it can be either new or be part of the tactic catalogue as already available in IoT-A D1.5) and subsequent design choices must be made.

2.5 Cross-check

Cross-check is an important continuous step that tracks down requirements that have been taken into account during the architecting process (meaning their respective fit criteria have been met), ensuring none of them is left behind. Cross-check information will be captured through the VOLERE template as well.

2.6 Threat Analysis

This section describes the process which will be used in the context of CPaaS.io for conducting a Threat Analysis [7] (as described as a part of the whole requirements process).

A threat analysis needs to be performed in order to come up with a list of issues (some being identified during requirement collection), which are worth taking into account mitigating between known vulnerability, risks and potential impact. This step will result in a set of security/privacy/trust-focussed requirements.

It traditionally begins with a definition of the elements that have to be protected. Then, an analysis of possible threats is conducted. How identified threats may actually affect elements to be protected, leads to the definition of risks. These risks have to be categorised, considering parameters such as criticality or probability of occurrence.

Several approaches have been proposed for defining a layering model for IoT architectures and the main components for its threat analysis [2] [9].

Here we will focus on the oneM2M approach as follows.

OneM2M was established to develop a single horizontal platform for the exchange and sharing of M2M/IoT data among all applications. OneM2M is creating a distributed software layer which provides a framework for interworking with different technologies [5].

OneM2M defines a security framework from its own *architecture model* [6] which supports end-to-end M2M services. A high level functional view of this architecture model is shown in Figure 1. Starting from this, oneM2M identifies four security domains. Each of these domains provides a set of security measures to address certain threats that may appear on it:

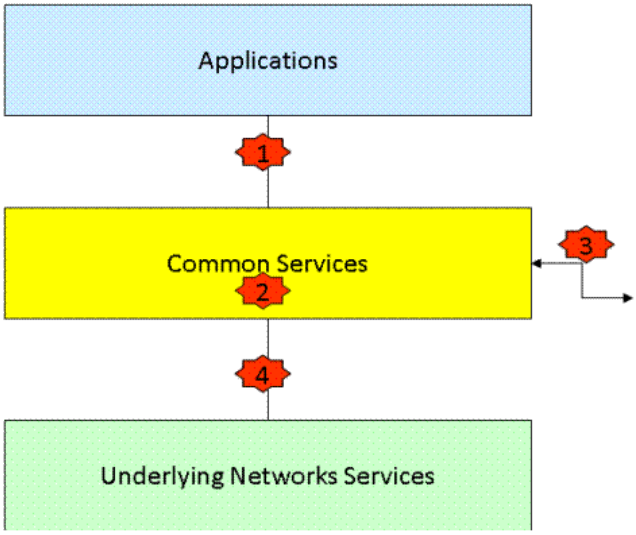

Figure 1: oneM2M context and security domains

- ***Application domain security***: A set of security measures that enable the Application entity and the Common Services entity to securely exchange messages and protect against attacks on (1).
- ***Intra Common Services domain security***: A set of security measures that enable Common Services Functions in the Common Services entity to securely exchange messages and protect against attacks on (2).
- ***Inter Common Services domain security***: A set of security measures that enable messages secure exchange between different Common Services entities and protect against attacks on (3).
- ***Underlying Network security:*** A set of security measures that enable the Common Services entity and the Underlying Network Services entity to securely exchange messages and protect against attacks on (4).

In addition, the oneM2M security architecture consists of the following layers:

- **Security Functions layer**: this layer contains a set of security functions which can be classified into six categories; they are Identification, Authentication, Authorization, Security Association, Sensitive Data Handling and Security Administration;
- **Security Environment Abstraction Layer**: this layer implements various security capabilities such as key derivation, data encryption/decryption, signature generation/verification, security credential read/write from/to the Secure Environments, and so on. The security functions in the Security Functions Layer invoke these functions in order to do the operations related to the Secure Environments. In addition, this layer also provides physical access to the Secure Environments. This layer is not specified in the oneM2M release 1;

- **Secure Environment layer**: this layer contains one or multiple secure environments that provide various security services related to sensitive data storage and sensitive function execution. The sensitive data includes SE capability, security keys, local credentials, security policies, identity information, subscription information, and so on. The sensitive functions include data encryption, data decryption, and so on.

The definition of threats that can appear in each domain, as well as the implementation of security measures to solve them or palliate them, are based on the following aspects: secure storage of sensitive data, sensitive functions executing operations on sensitive data and secure connections allowing the secure transmission of sensitive data. Based on them, oneM2M describes a set of vulnerabilities relevant to the security domains explained above [2]. To do this, a predefined template is used that includes the following information: the issue caused by the threat, a description of the vulnerability, the affected security domains and the list of M2M stakeholders and M2M architecture components which are impacted by the threat. Thus, Table 2 shows an example of a vulnerability defined by oneM2M.

Table 2: Example of vulnerability defined by oneM2M

Threat ID	1
Overview	Long-term service-layer keys are discovered while they are stored in M2M Devices or M2M Gateways and are copied.
Issue	Copied long-term service-layer keys may be used to impersonate M2M Devices and/or M2M Gateways.
Description	Long-term service-layer keys are stored within the M2M Device or M2M Gateway. Those keys are discovered and copied by unauthorized entities and used for illegitimate purposes. Discovery of stored long term service-layer keys may be achieved e.g. by monitoring internal processes (e.g. by Differential Power Analysis) or by reading the contents of memory of the M2M Device or M2M Gateway (by hardware probing or by use of local management commands).
Impacted Use Cases	All
Affected Security domain	Application domain security; Intra Common Services domain security; Inter Common Services domain security; Underlying Network security, if keys are shared with underlying network.
Affected Stakeholders	M2M Application Service Provider; Manufacturer of M2M Devices and/or M2M Gateways; M2M Device/Gateway Management entities; M2M Service Provider; Network Operator, if network operator keys are shared; User/Consumer
Architecture impact	Device / constrained Device : impacts storage of long-term service-layer keys Middle Node / Gateway : impacts storage of long-term service-layer keys Common Services Entity / Function: impacts Security CSF

Once vulnerabilities have been defined, oneM2M provides a set of countermeasures and solutions to prevent these threats. As before, a predefined template is used that includes the following information: related threats that are solved (or are palliated) by the countermeasure, a description of the countermeasure itself, the affected security domains and a list of advantages and disadvantages of the applicability of such countermeasure. Thus, in Table 3 below an example of countermeasure provided by oneM2M is shown.

Table 3: Example of countermeasure provided by oneM2M

Related threats	**Threat 1:** Discovery of Long-Term Service-Layer Keys Stored in M2M Devices or M2M Gateways **Threat 2:** Deletion of Long-Term Service-Layer Keys stored in M2M Devices or M2M Gateways **Threat 3:** Replacement of Long-Term Service-Layer Keys stored in M2M Devices or M2M Gateways
Countermeasure 1	M2M long-term service-layer keys are stored in a HSM (whose tamper-resistance may be certified) residing within the M2M Device / Gateway which renders it infeasible for the attacker to discover the value of keys by logical or physical means.
Applicable Security domain	Application domain security; Intra Common Services domain security; Inter Common Services domain security; Underlying Network security, if keys are shared with underlying network.
Advantages	Resists the attack. A lot of prior art exists in the form of specifications of e.g. ETSI SCP. Other sensitive data / credentials in addition to long-term service-layer keys can be protected
Disadvantages	Additional per-item cost for HSM Need to specify and demonstrate the level of security assurance across the range of manufacturers and their products.

To conclude, the oneM2M security framework can serve as a starting point for performing risk analysis in M2M scenarios. In addition, countermeasures provided by this framework to address security threats detected during such analysis can be used as a basis for implementing solutions that prevent them.

This Threat Analysis task will be performed as soon as a draft of CPaaS.io architecture is available (i.e. at the beginning of Task 3.2. when the requirement mapping activity takes place). It will then challenge the draft architecture w.r.t. the security objectives extracted from the security requirements collected in this deliverable and complement it in term of precise security architecture.

2.7 Supporting Tools

This additional section introduces a supporting tool used for summarizing the outcomes of the various steps described above, in the form of an Excel sheet. This is an outcome of Task 3.2, after various requirements have been unified. It is then updated during the mapping phase.

The VOLERE template used in CPaaS.io is the version used by the IoT-A project for collecting requirements (see Figure 2 below) which is itself an extension of the original VOLERE template [15], augmented in order to comply with the Rozanski & Woods methodology using Views, Viewpoints and Perspectives [11].

In particular this extension allows us to capture the essence of the View / Perspective mappings as explained in Section 2.3 and 2.4

	Volere Template							Traceability regarding the architectural reference model							
UNI ID	Requirement Typ	Categor	Description	Rationale	Fit Criterion	Dependencie	Conflicts	View	Perspective	Functionality Group	Functional Compone	Domain Mod	Originating Business Scenario	Demonstration Business Scenari	System use case
UNI.001	Non-functional Requirements	Privacy	A system built using the ARM shall provide a means to allow people to use Internet of Things services anonymously	"Citizens want to protect their private data"	It is possible for a user to use the system without having any Personaly Identifiable Information being tracked	UNI.322, UNI.504, UNI.423, UNI.623, UNI.424, UNI.624	UNI.605	Functional	Security and Privacy	Security	Identity Management	User, Service, Resource, Device	Smart city	(none specific)	(none specific)
UNI.002	Non-functional Requirements	Privacy, Usage	Users have control how their data is exposed to other users	"Citizens want to protect their private data"	The system lets users select which personal data is accessible to other users	UNI.413, UNI.507		Functional	Security and Privacy	Security	Authorisation	Human User, Service, Resource	Smart city	(none specific)	(none specific)
UNI.003	Non-functional Requirements	Self-description, Semantics	A system built using the ARM shall enable the provision and exchange of semantics between services in order to support the design of new applications	"I would like a way to create and exchange semantics between objects in order to design new applications"	Semantic descriptions of services in the system are available	UNI.070		(none)	Evolution and Interoperability	(none specific)	(none specific)	Service, Resource	Smart city	(none specific)	(none specific)
UNI.004	Non-functional Requirements	Self-description, Semantics	A system built using the ARM shall enable the semantic description of physical entities	"I would like a way to create and exchange semantics between objects in order to design new applications"	Semantic descriptions of physical entities registered in the system are available			Information	(none)	(none specific)	(none specific)	(none)	Smart city	(none specific)	(none specific)
UNI.005	Functional Requirements	Autonomicity, Data handling & Communication	A system built using the ARM shall support event-based, periodic, and/or autonomous communication	"The remote monitoring device gathers patient measurements, data and or events. Data may be communicated each time the device gathers the data, accumulated measurements may be communicated periodically (e.g., hourly, daily), or data may be delivered upon request or upon certain events"	The considered system must be able to use or combine event-based, periodic and autonomic communication	UNI.301		Functional	(none)	IoT Service	IoT Service	(none)	e-Health	(none specific)	(none specific)
UNI.008	Non-functional Requirements	Interoperability	A system built using the ARM shall be able to run applications and services in an interoperable manner	"The problem is to provide a framework, a set of scenarios where these applications could be developed in harmony, in an interoperable way and in a way that responses to the real needs of organization and people"	The system should consistently use standardized or at least known interfaces			(none)	Performance and Scalabiltiy	(none specific)	(none specific)	Service	(none specific)	(none specific)	(none specific)

Figure 2: Fragment of the IoT-A Unified Requirements [14]

3 CPaaS.io Requirements Collection

In this section we give a list of requirements, which translate a pure **platform point of view,** taking into account in particular the project technical objectives as described in the Description of Work document. In this section we have split the requirements into two categories: Non Functional and Functional, and for each of those two categories into Run-time and non Run-time sub-categories. At an early stage of Task 3.2, those requirements will be unified with the requirements (following the scenario point of view) found in D2.1.

In the two main following Sections 3.1 and 3.2, we address and explain global classes of requirements and then break them into more concrete ones (as listed in the "refines into requirements" headers). Those concrete requirements are also listed respectively in Table 4 and Table 5 and the end of their respective sections.

3.1 Non-Functional Requirements

The non-functional requirements as introduced in this sub-section are partly adapted from [10].

3.1.1 Run-time Quality Requirements

3.1.1.1 Scalability Requirements

Description: The ability of the CPaaS.io system to execute its task within its expected performance profile and to handle on-demand increased processing volumes of data and service requests.

Rationality/Drivers: The CPaaS.io platform has ambitious performance requirements. Such platforms must cope with user's demand for data and services, capture real-time data that will be catalysed by a myriad of sensors. It is very difficult to have clear performance characteristics due to the ubiquity, heterogeneity high connectivity of devices and end-users.

Refines into requirements: NFREQ.1, NFREQ.2, NFREQ.3, NFREQ.4

Dependencies with requirements: n/a

3.1.1.2 Availability and Reliability Requirements

Description: The ability of the CPaaS.io platform to be fully or partly operational as and when required and to effectively handle failures that could affect system availability.

Rationality/Drivers: Build a reliable foundation for "on demand" exploitation of data that has complex or extended availability requirements, complex recovery processes, or a high profile.

Refines into requirements: NFREQ.5, NFREQ.6, NFREQ.7, NFREQ.8

Dependencies with requirements: n/a

3.1.1.3 Trust Requirements

Description: A quality related to the user's belief in the reliability, integrity and ability of the functional behaviour of the platform. Gain understanding of what influences user's experience while interacting with services provided.

Rationality/Drivers: Relevant to the systems that share and collect information that may raise public concern. In some cases, trust has to do with the reliability of data and their providers, whereas in other cases trust can be associated with the security and privacy of the technology that was deployed. Trust affects the reputation of the platform besides its dissemination and maturity on the market.

Refines into requirements: NFREQ.16, NFREQ.17

Dependencies with requirements: FREQ.25, FREQ.30-32

Dependencies with requirements: n/a

3.1.1.4 Security Requirements

Description: Ability of the system to enforce the intended confidentiality, trust, integrity and service and data access policies, and to detect and recover from failure in these security mechanisms.

Rationality/Drivers: The CPaaS.io platform may become a valuable target for attackers which can potentially leave huge swathes of information exposed. This could potentially undermine trust in the governance model and damage its reputation. Manage the data and services in a way that ensures its integrity, and compliance with data protection regulations.

Refines into requirements: NFREQ.9 to NFREQ.15

Dependencies with requirements: FREQ.23, FREQ.26

3.1.1.5 Privacy Requirements

Description: CPaaS.io platform should protect the vulnerability aspects of volunteered citizen's data. Ability of the system to ensure that the collection and transmission of personal data is minimized and handled in accordance with user's expectation and regulations.

Rationality/Drivers: Ensuring users privacy is protected positively influences user's experience, acceptance and continuous use of the platform. Besides other factors, the reputation of the platform depends on how well user's information is secure and preserved.

Refines into requirements: NFREQ.18, NFREQ.19, NFREQ.20, NFREQ.21, NFREQ.22

Dependencies with requirements: FREQ.20-21, FREQ.23-24

3.1.2 Non Run-time Quality Requirements

3.1.2.1 Evolvability Requirements

Description: The ability of the platform to withstand and to easily adapt when new requirements and changes are introduced.

Rationality/Drivers: CPaaS.io is expected to be highly evolvable in order to accommodate future emerging technologies and avoid interoperability issues. Ensure the platform is able to accommodate additional functionality and emerging technologies at later stage at a fair and transparent cost.

Refines into requirements: NFREQ.23

Dependencies with requirements: NFREQ.24-NFREQ.26

3.1.2.2 Extensibility Requirements

Description: The flexibility of the system to allow services and functionality to be extended and augmented by service providers in order to increase value of services to both platform providers and end-users. Identify integrated approaches to design and service delivery which ensures that services fit together and those synergies can be exploited by the CPaaS.io strategy.

Rationality/Drivers: Important for longer- lived and more widely used systems. CPaaS.io is expected to be highly evolvable in order to accommodate future emerging technologies and avoid interoperability issues. Stakeholders can extend the services provided by the CPaaS.io platform, so that partnerships can be built to deliver holistic and interoperable solutions..

Refines into requirements: NFREQ.24, NFREQ.25, NFREQ.26

Dependencies with requirements: n/a

3.1.2.3 Inter-operability Requirements

Description: in order to come up with a knowledge layer where all data is linked using LOD principles, a common set of disjoints ontologies must to be agreed upon and consequently unique schema must be used when annotating the data.

Rationale / Driver: CPaaS.io is a federated platform meaning that data from various sources and domains (incl. platform) maybe be aggregated and linked into a whole data set, meaning that data inter-operability must be ensured.

Refines into requirements: NFREQ.27

Dependencies with requirements: FREQ.13 & FREQ.14

3.1.3 List of Non-Functional Requirements

This Table 4 below summarises the Non-Functional Requirements as introduced, motivated cross-referenced in Section 3.1.1 to Section 3.1.1.5 for the Run-Time Requirements and in Section 3.1.2.1 and 3.1.2.2 for the non Run-Time Requirements.

Table 4: List of Non-Functional Requirements

ID #	Description	Perspective	Priority	Category
NFREQ.1	Support different service level agreements (SLA)	Scalability	SHOULD	Platform, Infrastructure
NFREQ.2	Process services and events on a set of	Scalability	SHOULD	Platform,

	distributed nodes			Infrastructure
NFREQ.3	Continuously monitor quality of service at runtime	Scalability	SHOULD	Platform, Infrastructure
NFREQ.4	Balance its load at runtime	Scalability	SHOULD	Platform, Infrastructure
NFREQ.5	Provide high availability	Availability	SHOULD	Platform, Infrastructure
NFREQ.6	Guarantee infrastructure availability	Availability	SHOULD	Platform, Infrastructure
NFREQ.7	Ensure network availability	Availability	SHOULD	Platform, Infrastructure
NFREQ.8	Be able to perform self-healing	Availability	SHOULD	Platform, Infrastructure
NFREQ.9	Expose data and services to authorized users	Security	MUST	City Data, Platform
NFREQ.10	Ensure services are always accessible to entitled users	Security	MUST	City Data, Platform
NFREQ.11	Ensure data freshness	Security	MUST	Platform
NFREQ.12	Support access control mechanisms	Security	MUST	Platform
NFREQ.13	Have security mechanisms to protect data transmission	Security	SHOULD	Platform, Infrastructure
NFREQ.14	Make it difficult to spy on communicated messages	Security	SHOULD	Platform, Infrastructure
NFREQ.15	Be able to perform to detect threats at runtime	Security	SHOULD	Platform, Infrastructure
NFREQ.16	Provide trusted and secure communication and information management	Trust	SHOULD	Platform, Infrastructure
NFREQ.17	The platform infrastructure and services shall be trustable	Trust	SHOULD	Infrastructure
NFREQ.18	Allow users to use free services anonymously	Privacy	SHOULD	Societal Needs, Platform
NFREQ.19	Allow people to use free services anonymously	Privacy	SHOULD	Societal Needs, Platform
NFREQ.20	Allow users to control which data they are willing to provide and how their data should be used	Privacy	SHOULD	Societal Needs, Platform
NFREQ.21	Keep users access-control rights/ policies	Privacy	SHOULD	Platform

	secured.			
NFREQ.22	Provide privacy protection for users interacting with the platform	Privacy	SHOULD	Societal Needs, Platform
NFREQ.23	Provide communication confidentiality	Privacy	SHOULD	Platform, Infrastructure
NFREQ.24	Be extensible for future technologies.	Evolvability	MUST	Societal Needs, City Data, Platform, Infrastructure
NFREQ.25	Provide standard interfaces for service providers	Extensibility	SHOULD	Platform, Business Needs
NFREQ.26	Be able to provide services in an interoperable manner	Extensibility	SHOULD	Platform
NFREQ.27	Data must be interoperable across the different parties (data / service producers)	Inter-operability	MUST	Platform

3.2 Functional Requirements

3.2.1 Run-time Requirements

3.2.1.1 Load/network resource balancing

Description: In CPaaS.io, the concept City platform as a service solution suggests that it is possible for the various city actors to access data and then run and perform analytics on top of it. In order to make optimal use of the platform and network resources and to comply with some privacy issues, mechanisms must be provided in order to run and eventually migrate tasks (e.g. analytics or even CEP tasks) at various places of the whole system, including in the cloud, in gateways and even closer to the edge (where e.g. the data is captured and fed to the platform). Decision making must take into account various parameters including the global context, network capabilities and topology, data distribution (how many nodes are involved and where) and access / privacy restrictions about the data or its issuer. In addition the decision making must consider at the same time the level of trust of the (eventually hosting) network nodes and the level of trust of the task to be eventually deployed remotely.

Rationale / Drivers: Elasticity and flexibility are two main targeted characteristics that ensure that optimal use of the resources is made. Related decision must comply with access and privacy policies and node inner physical assets (CPU, available bandwidth…).

Refines into requirements: FREQ70, FREQ71, FREQ72

Dependencies with requirements: n/a

3.2.1.2 Discovery and Look-up

Description: Discovery and look-up of Virtual Entities (VE) based on characteristics of the objects they are representing (maybe location of the Physical Entity or more complex/abstract properties) and of the services they are associated with. Desired information must be granted according to context of request and prior allocated permissions. This group of requirement has therefore strong link with security and privacy ones. In order to provide rich query requests and efficient information retrieval, semantic descriptions must be associated with both VE and IoT Services.

Rationale/Drivers: In order to ensure high level of security and privacy the "need to know" strategy is used, meaning only parties fully granted access to some data/service can get informed about the sole existence of that data/service (instead of filtering them in or out at the actual access level). As for access, access-rights may be bypassed according to relevant contexts like state of emergency.

Refines into requirements: FREQ.1, FREQ.2, FREQ.3, FREQ.4 and FREQ.5

Dependencies with requirements: FREQ.23, FREQ.31, FREQ.32

3.2.1.3 Data and Service/VE/Device Description Queries

Description: A SPARQL end-point will be used in order to provide data consumers (incl. accessing service, VEs and device semantic descriptions) with powerful queries.

Rationale/Drivers: SPARQL [13] is an ideal and widely used tool for getting the full-fledge of semantic annotations as far as querying semantic data from a database is concerned. Using this type of end-points does not preclude the compulsory use of a triple-store as some highly efficient and scalable relational databases do propose a SPARQL end-point.

Refines into requirements: FREQ.12

Dependencies with requirements: FREQ.13

3.2.1.4 Controlled Access to Object (Virtual Entities), Data and Services

Description: Access to object (e.g., accessing VE properties), retrieving data and invoking services must be granted or denied according to the access policies dictated by the owner of the object, data or services, upon proper authentication of all involved parties, including the issuer of the request.

Rationale/Drivers: After look-up and discovery, a second step of security enforcing needs to decide whether access to the data, service or object maybe granted or not, in order to implement the various access and privacy policies.

Refines into requirements: FREQ.30 to FREQ.39

Dependencies with requirements: FREQ.23, FREQ.26

3.2.1.5 Accounting

Description: CPaaS.io should provide mechanisms that allow keeping track of user's action like service invocation or access to information for eventual billing purpose.

Rationale/Drivers: Enable the emergence of business models based on exchange and production of information, high-end data analysis, etc.

Refines into requirements: FREQ.20

Dependencies with requirements: n/a

3.2.1.6 Accountability

Description: The platform must provide some mechanisms supporting non-repudiation.

Rationale/Drivers: For liability issues (especially regarding access to private information or resource actuation) it is important to keep track of who (person or service) is doing what.

Refines into requirements: FREQ.90

Dependencies with requirements: n/a

3.2.2 Non Run-Time Requirement

3.2.2.1 Discovery and Look-up

Description: Each Virtual Entity and service must be (semantically) described with a degree of detail that allows powerful search criteria and complex retrieval.

Rationale/Drivers: rich semantic description of objects, services and data allows for fine grain access and invocation, making the whole system more usable.

Refines into requirements: FREQ.10

Dependencies with requirements: n/a

3.2.2.2 Control Right and Privacy

Description: Data owners must be able to specify and manage their access policies, however it must be possible that the policy is bypassed in case of emergency (Run-time aspect of access control). As part of privacy and access policy, the user may want the data to be eventually locally encrypted and then accessed but either in an anonymised way (use of pseudonym e.g.) or passed-on in an encrypted form. In addition access policies must be protected so that they are never publicly available.

Rationale/Drivers: to preserve the data owners' privacy end-to-end, i.e. from storage to delivery over the network.

Refines into requirements: FREQ.20 & FREQ.23 to FREQ.26

Dependencies with requirements: n/a

3.2.2.3 Data Format and Annotation / Data Federation

Description: The data at the platform level must be annotated using the same schema (ontologies) and in the form of Linked Open Data, where data coming from different sources and eventually different platforms is eventually aggregated into a consistent whole dataset.

Rationale/Drivers: In order to be interoperable and linkable the same ontologies must be used for all data produced. The translation to the right schema can be achieved after the raw data has been produced though, assuming the translation rules would be clearly defined but the data owner.

Refines into requirements: FREQ.13 & FREQ.14

Dependencies with requirements: n/a

3.2.2.4 Data Queries

Description: Data search and retrieval can be made upon specification of meta-data (as part of data annotations) which must include some location, quality (of information) information and some information about the reputation and trustworthiness of the resource delivering the data.

Rationale/Drivers: Not all information comes with the same quality (accuracy, level of availability or sampling rate for instance), trustworthiness (which may depend on the issuer's own reputation). It is important for the sake of efficiency to select/filter out data according to a comprehensive enough amount meta-data (data annotation).

Refines into requirements: FREQ.11

Dependencies with requirements: FREQ.13, FREQ.10

3.2.2.5 Data Subscription

Description: The CPaaS.io platform must provide subscription mechanisms that allow an application / an end-user to access live data as it is produced by any data provider. Subscription criteria must include at least geographical area (indicating which area the data relates to), data type and originator of that data.

Rationale: Scenario applications (as they are envisaged by the different project scenarios) must be able to reflect a certain situation based on near-real time data consumption. Criteria must be able to filter out non-relevant data from the plethora of data produced by the different data producers participating to CPaaS.io as data sources.

Refines into requirements: FREQ.90

Dependencies with requirements: n/a

3.2.2.6 Complex Event Processing

Description: It must be possible for the CPaaS.io system to process/trigger complex events out of an incoming stream of live data so that when certain criteria/conditions are met, events are produced for further processing within the platform.

Rationale: Scenario applications like emergency and natural disaster scenarios must be able to received synthetized information reflecting the raise of particular events based on specific conditions like e.g. water level exceeding a certain value or seismic waves exceeding certain thresholds with certain periodicity.

Refines into requirements: FREQ.91

Dependencies with requirements: n/a

3.2.2.7 Added-value Services

Description: The CPaaS.io platform must provide mechanisms that allow to detect and to predict events out of a large set of incoming data. It must also provide mechanisms that can be used to analyse historical data and preform analytics.

Rationale / Drivers: One of the main objective of CPaaS.io platform is to correlate aggregate, link information from various sources in order to perform event detection and prediction

Refines into requirements: FREQ.80

Dependencies with requirements: n/a

3.2.3 List of Functional Requirements

The following Table 5 below provides a list of functional requirements from the platform point of view (meaning they will need to be consolidated/unified) with stakeholders and scenario requirements (see Conclusion & Next Steps Section). The list is partly taken /adapted from the SMARTIE Project [12]

Table 5: List of Functional Requirements

ID #	Description	Priority	Category
FREQ.1 [REQ25]	Discovery and lookup service of IoT systems should allow locating the physical entities based on geographical parameters.	SHOULD	Discovery Look-up
FREQ.2 [REQ31]	The look-up service of CPaaS.io shall withhold or grant information depending on context such as application involved, requesting entity, and security permissions.	MUST	Discovery Look-up
FREQ.3 [REQ36]	The IoT system shall enable the dynamic discovery of relevant virtual entities and their related services based on respective specifications.	MUST	Discovery Look-up
FREQ.4 [REQ37]	The IoT system shall enable the dynamic discovery of relevant VEs and their related services based on a geographical location scope.	MUST	Discovery Look-up
FREQ.5 [REQ38]	The IoT system shall enable the lookup of service descriptions of specified services for Virtual Entities with the VE identifier as key for the lookup.	MUST	Discovery Look-up
FREQ.10	Object (Virtual Entities), services and resources shall be	MUST	Interoperability

	semantically described		
FREQ.11	Data search and retrieval shall be achieved upon a collection of criteria that includes location (of the Physical entity concerned), characteristics of the underlying IoT Resource reputation and the quality of information.	MUST	Interoperability
FREQ.12	The platform shall provide a SPARQL end-point for accessing semantic information (either annotated data or semantic descriptions)	MUST	Inter-operability
FREQ.13	Any data at the platform level shall be following a common schema (ontology)	MUST	Inter-operability
FREQ.14	The platform shall be able to aggregate data from various sources using LOD principles	MUST	inter-operability
FREQ.20 [REQ19]	User anonymity shall be provided in order to enforce privacy (e.g. at communication level)	MUST	Security/Privacy & Trust
FREQ.21 [REQ81]	Personal data in servers should be ciphered by a private key.	SHOULD	Security/Privacy & Trust
FREQ.22 [REQ84]	Secure storage of data should be ensured.	SHOULD	Security/Privacy & Trust
FREQ.23 [REQ34]	Data owners shall be able to set access-control rights/ policies (set up by data owners) to their data stored on resources.	MUST	Security/Privacy & Trust
FREQ.24 [REQ35]	Access-control rights/ policies (set up by data owners) should not be published publicly.	SHOULD	Security/Privacy & Trust
FREQ.25 [REQ63]	Communicated data shall remain confidential.	MUST	Security/Privacy & Trust
FREQ.26	Service providers shall be able to set access-control rights/ policies (set up by service owners) to their services	MUST	Security/Privacy & Trust
FREQ.30 [REQ116]	CPaaS.IO shall be able to evaluate access request depending on access control policies.	MUST	Security/Privacy & Trust
FREQ.31 [REQ117]	CPaaS.IO shall provide an authorization policy to data, object and service for the different users or devices.	MUST	Security/Privacy & Trust
FREQ.32 [REQ 59]	User or device as part of the platform deployment shall be registered before using any services provided by the platform.	MUST	Management Security/Privacy & Trust
FREQ.33 [REQ60]	User or device shall be identified.	MUST	Management
FREQ.34 [REQ62]	User or device shall be authorized to use a service.	MUST	Security/Privacy & Trust

FREQ.35 [REQ69]	Objects should authenticate a person or object that try to access its data.	SHOULD	Security/Privacy & Trust
FREQ.36 [REQ77]	Authentication of user or service shall be carried out by devices before delivering data.	MUST	Security/Privacy & Trust
FREQ.37 [REQ 82]	Authentication of user or service shall be carried out to access a service.	MUST	Security/Privacy & Trust
FREQ.38 [REQ123]	User's data shall be processed only with the user's consent.	MUST	Security/Privacy & Trust
FREQ.39 [REQ133]	The IoT system should support context-aware access policies.	MUST	Security/Privacy & Trust
FREQ.50 [REQ90]	Services shall maintain a log of the operations done by users. This log shall not be modified by attackers.	MUST	Security/Privacy & Trust
FREQ.60 [REQ20]	Time stamps should be supported.	SHOULD	Platform
FREQ.70 [REQ131]	The IoT system shall ensure that processing of information takes place on trusted nodes.	MUST	Security/Privacy & Trust Load Balancing & Bandwidth Optimization
FREQ.71 [REQ132]	The IoT system should optimize the processing of information with respect to a cost function, e.g. communication, computation, energy, etc.	SHOULD	Load Balancing
FREQ.72	The CPaaS.io platform shall be able to provide mechanisms for migrating tasks to nodes participating the CPaaS.io systems according to its characteristics	MUST	Load Balancing & Bandwidth Optimization
FREQ.80	CPaaS.io platform shall be able to perform analytics, and event detection/prediction using “live” and historical data	MUST	Platform
FREQ.90	The CPaaS.io platform shall provide publish / subscribe functionality and criteria for filtering in/out data based (at least) on geographical area, data type and data source	MUST	Platform
FREQ.91	The CPaaS.io platform shall provide Complex Event Processing techniques that are able to infer high level events out of an incoming flow of data, based on various criteria (condition for that event to be triggered)	MUST	Platform

4 Conclusions & Next Steps

This deliverable provides a first comprehensive list of requirements that shall, or in cases should be met when elaborating (during next WP3 phase) the system architecture for the CPaaS.io project. As already mentioned, this list will be complemented with the result of a similar task (Task 2.1 "Requirement Collection") carried out in WP2 with focus on Scenario-specific requirements. Many additional functional requirements are expected to be added from that work, which is based on the assumed scenarios (which represent the final users of the platform, in terms of data consumption and object/service use).

Despite the title "Requirement Collection **and Analysis**", the "Analysis" part is the actual next step carried out in the context of Task 3.2 (starting M04) where all requirements (D2.1 & D3.1) are unified/factorised, reformulated for some of them and then mapped to Views, Functional Groups/Functional Components and Perspectives, leading to a first Functional Decomposition of the CPaaS.io platform. The VOLERE template summarising the project Unified Requirement is one of the outcomes of this next deliverable D3.2 (available end of M06).

5 References

[1] F. Carrez *et al.*, "IoT-A Deliverable D1.5 – Final Architectural Reference Model for the IoT v3.0", www.iot-a.eu/public/public-documents/

[2] ETSI TR 103 167Machine-to-Machine Communications (M2M); Threat analysis and counter-measures to M2M service layer

[3] Ho, E. & al., "IoT-A Deliverable D6.1 – Requirements list", www.iot-a.eu/public/public-documents/

[4] IoT-A web site: http://www.iot-a.eu

[5] oneM2M white paper, January 2015, http://www.onem2m.org/images/files/oneM2M-whitepaper-January-2015.pdf

[6] oneM2M Functional Architecture, http://www.onem2m.org/images/files/deliverables/TS-0001-Functional_Architecture-V1_6_1.pdf

[7] OneM2M security solutions", oneM2M-TR-0008-Security-V1.0.0, 2014

[8] Pastor,A. "Updated Requirements List", www.iot-a.eu/public/public-documents/

[9] OneM2M security solutions", oneM2M-TR-0008-Security-V1.0.0, 2014

[10] "Requirements Specification for Urban Plarforms v2.2" prepared by Demand-Side Engagement Stream I the context of the European Innovation Partnership for Smart Cities & Communities (EIP_SCC) Integrated Infrasturcture Action Cluster – Urban Platform

[11] N. Rozanski and E. Woods, "Applying Viewpoints and Views to Software Architecture" , Addison Wesley, 2011

[12] SMARTIE Project Deliverable D2.2 "Requirements"

[13] SPARQL Protocol And RDF Query Language http://www.w3.org/TR/rdf-sparql-query/

[14] UNIs: List of IoT-A UNIfied requierementS. Available at http://www.iot-a.eu/public/requirements

[15] VOLERE Requirements Resources, http://www.volere.co.uk/