TRESCIMO

science & technology

Deliverable D2.3

TRESCIMO Final Requirements and Scenarios

Editor:	Louis Coetzee, CSIR Meraka
Deliverable nature:	Report
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	31 December 2014
Actual delivery date:	12 January 2015
Suggested readers:	City Planners and Engineers, City Chief Information Officer and Major; Vendors and Technologists, Researchers (Academia), Software and System Engineers, Standard Consortium Parties
Version:	1
Total number of pages:	31
Keywords:	Smart City, Developing Context, Developed Context, M2M, Sensors, Active Devices, Smart City Platform, Delay Tolerant Networks, Smart Energy, Environmental Monitoring, CoAP, oneM2M, Core-Link

Abstract

This document aims to extend previous work related to TRESCIMO requirements and scenarios. It adds new requirements related to experimental facilities (testbeds). The design and creation of the experimental facilities (with components in Africa and Europe) is based on an integrated set of scenarios, extracted requirements, and planned experiments. The functionality of the facilities will be verified through the implementation of selected scenarios and experiments. This document revisits the scenarios and technical requirements published in earlier deliverables. It adds new requirements related to Future Internet Research & Experimentation facilities and introduces the experimental life-cycle which includes the use of these facilities beyond the demonstration of an application (e.g. environmental monitoring). Functional requirements for two planned applications (environmental monitoring and electricity grid stability), and functional requirements for the testbed itself are presented. Non-functional requirements and relevant technologies are also presented.

Disclaimer

The research leading to these results has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n° 611745, and the South African Department of Science and Technology under financial assistance agreement DST/CON 0247/2013.

Impressum

Full project title	Testbeds for Reliable Smart City Machine-to-Machine Communication
Short project title	TRESCIMO
Number and title of work-package	WP2: Final Scenarios and Requirements
Number and title of task	D2.3
Document title	TRESCIMO Final Requirements and Scenarios
Editor: Name, company	Dr. Louis Coetzee, CSIR Meraka
Work-package leader: Name, company	Dr. Louis Coetzee, CSIR Meraka
Estimation of person months (PMs) spent on the Deliverable	3

Executive summary

TRESCIMO – (Testbeds for Reliable Smart City Machine to Machine Communication) is establishing experimental facilities in the context of Smart Cities dealing with mass urbanization in both developed and developing worlds.

These facilities are aimed to investigate the utility of services (e.g. the societal impact) as well as identifying and implementing appropriate architectures for such Smart Cities. TRESCIMO is positioned under the European Union FP7 Future Internet Research & Experimentation (FIRE) initiative.

The objective of this report is to create synthesized specifications and technologies for the TRESCIMO reference architecture.

A previous TRESCIMO deliverable ("TRESCIMO Scenario Specification" [1]) presented detailed scenarios in several domains, with these domains being "energy", "transport", "safety", "health", "education", and "water". From these scenarios detailed technical requirements were extracted ("TRESCIMO User and Technical Requirements" [2]). Similarly, a set of experiments aimed at validating the planned facilities were presented in the document "TRESCIMO Experiment Descriptions" [3].

This document extends on previous work by describing functionality of, and experiments for a testbed, as well as an experimental methodology. Based on the experimental needs, scenarios and use-cases were synthesized after which functional and non-functional requirements (with associated technologies) were extracted and subsequently presented here with the aim of guiding the design of the reference architecture.

The work presented in the collection of deliverables, including this report, guided the creation of architecture for globally interconnected experimental facilities as presented in the report "Architecture Specification (D3.1)" [4].

List of authors

Company	Author	Contribution
CSIR Meraka	Dr. Louis Coetzee	Editor, ToC and overall contribution
Fraunhofer Fokus	Ancuta Corici	ToC, and overall contribution
TUB	Ronald Steinke	ToC and review
TUB	Asma Elmangosh	Contribution to Section 5
UCT	Joyce Mwangama	Contribution to Section 4
UCT	Neco Ventura	Contribution to section 4
I2CAT	Marisa Catalan	Contribution to Section 4,5,6 and 7
I2CAT	Daniel Camps	Contribution to Section 4,5,6 and 7
CSIR MSM	Dawid Oosthuizen	Contribution to Section 5 and general editing
Eskom	Hinesh Madhoo	Contribution to Section 2
Eskom	Shawn Papi	Contribution to Section 2
TUB	Alexander Willner	Contribution to Section 3 and 5
Eurescom	Maria Joao Barros	Review
CSIR Meraka	Andrew Smith	Review

Table of Contents

List of Figures

List of Tables

Definitions, Acronyms and Abbreviations

Name	Description
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks – RFC6282
AM	Aggregate Manager
API	Application Programming Interface
CoAP	Constrained Application Protocol – RFC7252
CoRE	Constrained RESTful Environments – RFC6690
CSIR	Council for Scientific and Industrial Research – www.csir.co.za
CSIR Meraka	CSIR ICT operating unit
DNS	Domain Name Server
DTLS	Datagram Transport Layer Security
DTN	Delay Tolerant Network
ETSI	European Telecommunications Standards Institute
FRCP	Federated Resource Control Protocol
FIRE	Future Internet Research and Experimentation
FITeagle	Future Internet Testbed Experimentation and Management Framework
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronic Engineers – www.ieee.org
IPv6	Internet Protocol, version 6 – RFC2460
KPI	Key Performance Indicator
LLN	Low Power and Lossy Network – described in RFC7102
M2M	Machine-to-Machine communication
MSM	CSIR Materials Science and Manufacturing operating unit
OML	ORBIT Measurement Library
OMSP	OML Measurement Stream Protocol

Name	Description
OpenMTC	Open Machine Type Communication
OpenSDNCore	Research & Testbed for the carrier-grade Network Functions Virtualisation / Software Defined Network environment
OpenStack	Open source cloud computing software
REST	Representational state transfer
RESTful	A REST system adhering to the architectural constraints
RF	Radio Frequency
RPL	Routing Protocol for LLNs – RFC6550
SAF	Store And Forward
SCP	Smart City Platform
SFA	Slice-based Federation Architecture
SSH	Secure Shell
VPN	Virtual Private Network
TRESCIMO	Testbeds for Reliable Smart City Machine-to-Machine Communication
TUB	Technische Universität Berlin
UCT	University of Cape Town

1 Introduction

TRESCIMO (Testbeds for Reliable Smart City Machine to Machine Communication) [5], a project executed as part of the European Union Future Internet Research and Experimentation initiative [6], has as objective to establish intercontinental experimental research facilities in order to demonstrate concepts related to smart and green technological and social innovation.

To this end, deliverables D2.1 (TRESCIMO Scenario Specification) [1], and D2.2 (TRESCIMO User and Technical Requirements) [2] analysed the context with related concepts wherein TRESCIMO operates. In an iterative fashion, the context influences the reference architecture and implementation for TRESCIMO [4].

With the aim of creating intercontinental future internet research facilities, the scenarios and use-cases are means to test and exploit the facilities in addition to influencing the reference implementation. Similarly, the envisioned experiments as presented in D4.1 [3] impact on the architecture and specifications. The exploitation and verification of the research facilities are realised through experiments.

This report ties together the above deliverables and serves as a final user and technical requirements specification. It also presents requirements as linked to the creation of experimental facilities ("testbeds") which also impacts on the reference implementation.

Section 2 provides an overview of scenarios as well as user and technical requirements as extracted from the scenarios. Section 3 presents details pertaining to experimental research facilities. Section 4 contains functional requirements for experimental facilities as well as from an end-user perspective. Section 5 analyses the non-functional requirements while Section 6 investigates the technological requirements. Section 7 is a synthesized response to previous comments regarding specifications. Section 8 concludes.

2 Scenario and User and Technical Requirements overview

This section summarises and highlights the requirements from deliverable D2.1 (TRESCIMO Scenario Specification) [1] and deliverable D2.2 (TRESCIMO User and Technical Requirements) [2]. These deliverables focus on functionality as from a "user" point of view which will be executed as validation of the testbed. Furthermore, the testbed in itself has a set of requirements which are explored and described in Section 3. Functional requirements are captured in Section 4.1.

2.1 Deliverable D2.1 – TRESCIMO Scenario Specifications

D2.1 explores the need for Smart Cities, provides a definition of the Smart City concept, investigates challenges in deploying Smart Cities, and lists a number of scenarios applicable to a Smart City. The objective of the document was to provide a "foundation" for the development of technical building blocks which could result in a Smart City architecture that can be deployed successfully in both developed and developing countries.

Some developed countries have already begun deploying Smart City technologies/applications (e.g. SmartSantander [7]) but a number of challenges remain to be overcome before full benefit of such systems can be realised. These challenges include the lack of interoperability between devices/sensors from various technology vendors, limited robustness and reliability of devices, and the reliance on communication across constrained and unreliable networks. Support, maintenance and cost also limit wide deployment of Smart City applications.

Developing countries have also identified the need for, and the benefits of, Smart City platforms. However, the maturity of standardised platforms and infrastructures, aspects related to maintenance

and support, as well as localisation of technologies impact on the direct implementation of available Smart City concepts. With this in mind, there is a need for a system that takes into account requirements for developed and emerging countries. This is the departure point for deliverable D2.1 and subsequent TRESCIMO deliverables.

The numerous Smart City definitions provided in deliverable D2.1 indicate that this is still an emerging concept and that there is no global consensus. However, for purposes of the TRESCIMO project deliverable D2.1 concludes that "the smart city is about how people are empowered, through using technology, for contributing to urban change and realising their ambitions".

As methodology, deliverable D2.1 used scenarios to set the scene for specific user interactions in a Smart City. A variety of scenarios over multiple domains from both developed and developing countries were investigated. The domains were "energy", "health", "transport", "safety", "education: and "water". In the scenario a user challenge was described, a possible intervention and the outcome of that intervention, all specifically related to the user. An example from deliverable D2.1 is the Johnie scenario (energy):

> ***Johnie works at the local power Utility. His main role is to ensure that the "lights stay on" and to "keep the wheels turning", commonly used terms in his environment indicating that energy is quite often scarce, and that every effort needs to be made to ensure that no blackouts occur and that the economy is not impacted if problems arise.***
>
> ***Johnie is in the warm and dry control room enjoying a cool drink. It is cold and wet outside, not very typical weather for this time of the year. He is worried about the drop in energy supply and the increase in demand from municipalities, industry and home-owners.***
>
> ***There is unplanned maintenance at one of the power stations that has resulted in an unstable grid due to a decrease in generation capacity. Johnie realises that he has to act in order to ensure the stability of the grid. Johnie calls up the aggregated view of both industrial as well as residential areas. Through the aggregated view he identifies areas and entities of high utilisation. Johnie notifies individuals through a demand management signal to reduce consumption. He notices a drop in demand, but not enough to ensure the stability of the grid. As a next step Johnie sends a command to devices that previously have been nominated by their owners as "controllable". This emergency intervention has the desired outcome as the grid returns to a stable state. Johnie is happy that he has managed to "keep the lights on".***

A second example is environmental monitoring for health through the Linde scenario:

> ***Linde has asthma and is always worried about pollution in the air. It hasn't rained in many months, it is dry and without wind. She has to go into town today, but isn't sure it is safe. She has no-one to call to get pollution status and decides to risk it. As she enters the city centre using the taxi, she is alerted of the very high pollution in this area. She immediately stops the taxi, disembarks and by using a different taxi returns to a place where it isn't as polluted. She is happy that she has avoided the possibility of an asthma attack.***

A subset of these scenarios was further broken down into user and technical requirements as described in Section 2.2.

2.2 Deliverable D2.2 - TRESCIMO User and Technical Requirements

In deliverable D2.2, the scenarios were broken down into use-case diagrams through which the various actors and systems involved were identified. Figure 1 provides a graphical representation of the method of analysis used in deliverable D2.2.

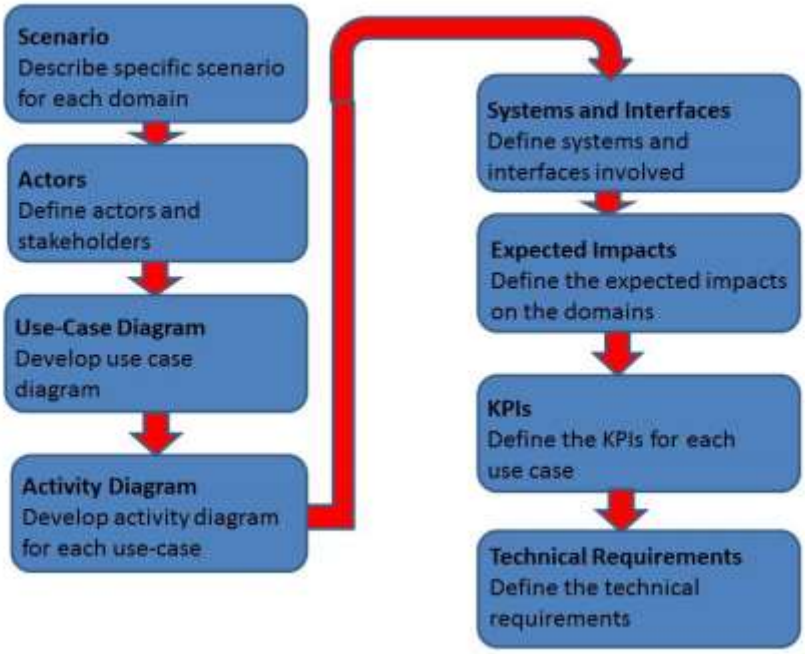

Figure 1 Method of analysis used in **deliverable** D2.2

The system/device identification process was essential in that it informed the technology decision makers within TRESCIMO as to which hardware and software technology platforms are necessary to implement the envisioned Smart City framework. In addition, the information provided a solid base for the development of technical specifications and definition of functional requirements applicable to the selected technologies. Following this, activity diagrams were developed to demonstrate the sequence of actions necessary to solve the problems presented in each scenario.

Further analysis and comparison of the use-cases revealed a number of common elements within the various Smart City domains. These elements enable the development of generic functionalities that can be applied to a number of domains. However, elements specific to a particular domain were also discovered and will typically be catered for by means of domain-specific applications. The rest of this document presented functional requirements of each system/device identified in deliverable D2.2.

3 Functional Future Internet Research Experimentation

The overarching goal of TRESCIMO is to offer a Smart City software stack as a Service for experimental Future Internet research in a holistic way. A very specific focus hereby lies on the one hand in the re-usage of existing software components and on the other hand in the scenarios and requirements for Smart Cities in the developing and developed world.

The latter have been analysed in deliverable D2.1 [1] and were refined in deliverable D2.2 [2]. Further, in deliverable D2.2, the technical requirements for standards, technologies, and testbeds have been described in order to identify the software components that could be re-used.

Together with the experiment descriptions from deliverable D4.1 [3], an additional set of requirements is identified. These are realized through the following tests:

- Validation of the infrastructure as to its ability to support the broader set of experiments.
- Configuration: to validate the performance of the implemented service under various environments, by conducting different measurements with configurable conditions.
- Functionality: to validate the specified functionality of utilized components and technologies of the system.
- End-user testing: to gain insight into the effect of status communication back to the end-user (e.g. the user's current energy consumption).
- Repeatability: or test–retest reliability is the variation in measurements taken by a single person or instrument on the same item and under the same conditions.
- Scalability: to test the infrastructure based on its ability to cope with increasing number of users while satisfying individual user's metrics.

Therefore, this section describes the requirements that arise in the context of the generic experiment life-cycle (cf. Figure 2).

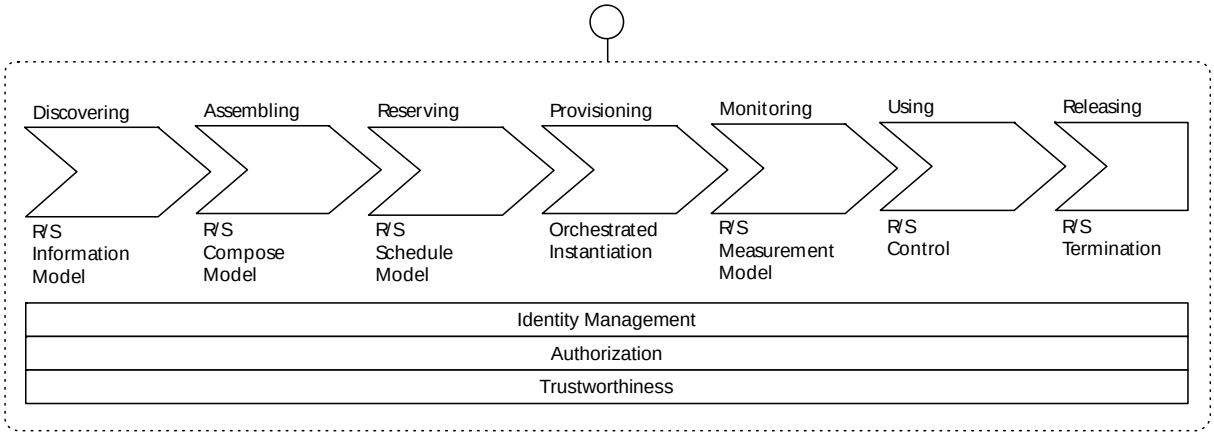

Figure 2: Experiment Life-Cycle

3.1 Resource Discovery and Provisioning

It has to be possible to describe resources and services that are available within the TRESCIMO federation and to provide this information to potential users. Further, an experimenter needs means to discover, select and provision a subset of them for a specific duration in order to make use of the resources (acting) or the related information (monitoring). Here an exclusive and non-exclusive approach is feasible and requirements for reservations in the future are not foreseen. Initially, these functionalities are required within the TRESCIMO consortium and should later be opened for further users.

A potential candidate to fulfil these requirements is implementations of the SFA AM APIs [8].

3.2 Resource Control / Usage

After the successful discovery and provisioning of the selected resources, utilisation becomes possible. The particular requirements depend on the type of resource. Three non-exclusive use-cases can be identified:

a) API access to dynamically instantiated or static software components (such as a Smart City API or a M2M platform),
b) Access to live/historical monitoring data from sensors, and
c) Automated, repeatable control of resource parameters.

Given the specific regulations (such as restrictive software licenses) or types of resources (such as AirBase Air Quality devices) within the TRESCIMO consortium, in particular direct access to resources via SSH is generally not required / desired.

A potential candidate to fulfil the last requirement in particular is an implementation of the FRCP API [9].

3.3 Resource Reserving and Releasing

It must be possible to reserve resources for an experiment run so that results from different parallel experiments do not influence the final results. The particular requirements depend on the type of resource. We will consider reserving only the physical or emulated end devices (sensors, actuators). Thus two requirements can be identified:

a) API to reserve physical or emulated devices in terms of collected data, configuration and commands sent to the device.
b) Handling of simultaneous reserve request using queueing of requests or other algorithms.

3.4 Resource Monitoring

In order to gain scientific knowledge within a (repeatable) experiment, specific metrics have to be defined and monitored. Depending on the actual resource usage (see above), two different types of monitoring scenarios are required:

a) Access to measurable information via API (use-case 3.2a and use-case 3.2b), and
b) Push/storage of experiment specific data.

In conjunction with 3.2b a potential candidate to fulfil these needs is an implementation of OMSP streams[1].

3.5 Authentication and Authorization

Direct access to the provided implementations should be avoided (cf. 3.2), and misuse of the infrastructure should be prevented and documented. Therefore, un-authorized access to the resources is not envisioned – although it is planned to open most of the available resources and services to a broader community.

Therefore, a potential candidate to fulfil these requirements is implementations of the SFA AM APIs.

3.6 Trustworthiness

In order to increase the confidence of users in the TRESCIMO facility, the availability of the different testbeds, resources and services should be monitored and published.

A potential candidate to publish the availability and reliability information to, is the trescimo.eu web site and further the Fed4FIRE first level support page fls.fed4fire.eu.

[1] http://oml.mytestbed.net/doc/oml/latest/doxygen/omsp.html

4 Synthesized Functional Requirements

This section presents a synthesis of functional requirements related to the testbed experimental requirements, and those as extracted from scenarios and use-cases. Section 4.1 presents experimental requirements while Section 4.2 presents requirements as associated with scenarios. For the purpose of brevity requirements only related to “energy” and “health” (as initially presented in deliverable D2.1 [1] and repeated in Section 2.1) are presented. As the TRESCIMO architecture has to support a variety of scenarios and use-cases requirements from “energy” and “green environment” are presented in an integrated manner.

4.1 Functional Experimental Requirements

Analysis of the user requirements and architecture, as presented in deliverables D2.1 [1] , D2.2 [2] and D3.1 [4] respectively, resulted in the development of a set of experiments which are presented in deliverable D4.1 [3] “Experiments Descriptions”. After formulating these experiments, functional requirements relating to the experiments were fed back into the mentioned deliverables through an iterative methodology. The following subsections highlight the key functional requirements related to the planned experiments.

4.1.1 Requirements based testing types

Requirements can be described based on the types of testing that will be performed. The types of testing to be performed are stated and further why these tests are needed is described below:

Table 1: Experimental Function Requirements

Requirement	Description
Configuration and functionality	Configuration testing provides the ability to validate the performance of the implemented service under various environments. Functionality testing provides the ability to validate the specified functionality of utilized infrastructure, components and technologies of the system. Both these types of tests are important to validate that the system is performing as was designed.
End-user testing	To gain insight into the effect of status communication back to the end-user (e.g. the users’ current energy consumption). These tests are important to be able to see if the system gives users a good or bad experience (as these types of issues are not always apparent to system developers). These tests also provide a way to pick up on application/system navigation ease (or difficulty) for the intended audience of users.
Repeatability	To retest reliability is the variation in measurements taken by a single person or instrument on the same item and under the same conditions.
Scalability	To test the infrastructure based on its ability to cope with an increasing number of users while satisfying individual users’ metrics. Scalability testing is important to gauge the capability of the system to handle heavy workloads, or to identify the point at which the system performance begins to degrade (or crashes). Scalability tests also show whether the system is able to scale up or scale out when it is needed.

4.1.2 Requirements based on testbed interconnection

For the testbed interconnections, certain requirements needed to be taken care of. These can be seen as two phases. For example:

1. In the stage 1 VPN only environment -
    - that the networks are reachable,
    - For UCT having the perimeter firewalls and proxy issues resolved,
    - CSIR complying with the IPv6 requirement, and
    - For TUB, that the VPN is running in the OpenStack.
2. In the stage 2 federated environment -
    - That each location has to have an OpenStack installation,
    - That each location has images for the OpenMTC and SCP,
    - That TUB have an OpenSDNCore to discover and control the OpenStack installations, and
    - That this OpenSDNCore have an interface with FITeagle for FIRE.

4.2 Functional Scenario and Use-case Requirements

4.2.1 Actors

The “energy” and “environment” aspects presented the following actors:

Table 2: Actors

Actor	Example
Service Provider	• Municipality. • Operator.
Customer	• Citizen. • User. • Application.
Appliance	Device to be controlled e.g.: • Geyser. • Kettle.
Sensor	• Environmental sensor. • Actuator on appliance.
Network	• Delay tolerant network embodied through a mobile entity (e.g. bus). • Fixed network.

4.2.2 Activities

These actors interact through the following different activities:

Table 3: Activities

Actor	Activity
Service Provider	• Request measurements. • Initiate actions. • Update system configuration.
Customer	• Request measurements. • Process data and present results leading to actions.
Appliance	• Control appliances.
Sensor	• Measure variables. • Report measurements.
Network	• Communicate readings & commands.

4.2.3 Systems

In order to analyse functional requirements, we note the following technical components, or systems, that are involved:

Table 4: Systems

System	Role
Application	• Able to process incoming data, present views to stakeholders (e.g. a mobile device) and initiate an action in the environment.
Back-end service	• A platform able to host applications, collect data and interact with Machine to Machine. • The Smart City Platform is an example of such a back-end).
Machine-to-Machine service	• System able to create a link between edge node devices (e.g. the “aggregator service”) and a back-end platform. • Manage connectivity state. • Such a data-provider could be a Machine-to-Machine middleware platform.
Gateway service	• A system realised through gateways (both static and mobile). • The gateways collect data from sensors before communicating it through the “machine-to-machine middleware” to the ”back-end”. • Data transport is through a permanent connection or delayed data transport through a delay tolerant mechanism.
Sensing service	• A system consisting of numerous sensors (devices able to measure and/or control) which link to the gateways.

4.2.4 Functional Requirements

For each of these components, the following functional requirements are identified:

Table 5: Functional Requirements

System	Role
Application	• Will process and present data to customers.
Back-end service	• Will allow customers to retrieve historical and current data about individual sensor and or aggregated data from sensors, where reporting periods should be configurable. • Will be able to store large amounts of data as collected from many devices. • Will allow users to discover and control individual devices. • Will link applications and the Machine-to-Machine platform.
Machine-to-Machine service	• Provide appropriate and optimised communication between devices and other platforms using Store and Forward mechanism for M2M data but also for device management, transport using block wise transfer over CoAP. • Flexible transport: handling policies for changing the transport protocol from CoAP in the front end to HTTP. • Adaptable transport: by enabling device management to provision the location of the services in the cloud (e.g. the gateway is informed to which server to send dynamically). • Provide device management mechanism for retrieving device information (manufacturer, battery level), firmware update, location retrieval, enabling of features and devices or even lock and wipe. • Provide secure transport: CoAP with DTLS or HTTPS for protecting the user data privacy. • Provide subscribe-notify as well as pull mechanism to retrieve data, notification aggregation.
Gateway service	• Has connectivity to the Machine-to-Machine platform through broadband or intermittent mobile connections. • Has the ability to discover devices in close proximity. • Will be able to report information from sensors and mobile gateway to the back-end. • Will be able to cache the last measurements reported from a given sensor. • Will be able to store and forward measurements on demand. • Will be able to wake up sensors and collect stored measurements. • Will be able to relay commands from the back-end to the sensor devices. • Will be able to determine its position via GPS. • Will be able to receive policies influencing data collection.
Sensing service	• Will support and have a collection of sensors (e.g. energy, pollution, heat, humidity). • Will be able to store collected measurements to communicate to the

	"aggregator service" (e.g. the mobile gateway). • Will be able to communicate readings as per demand. • Will support "sleep" and "wakeup" functions.

4.3 Domain Specific Requirements

In addition to the generic functional requirements presented above (experimental and end-user), domain specific requirements are also of influence. They include:

Table 6: Domain Specific Requirements

Domain	Requirements
Smart Energy	• Ability to measure intervals of time. • Ability to control remote appliances. • Ability to measure energy consumption of households. • Ability to measure energy consumption of individual appliances. • Mobile application for individual user. • Data analytics to interpret and visualise historical data collected from users. • Ability to detect when resources need to be scaled according to system loads.
Smart Environmental Monitoring	• Ability to measure energy consumption of devices. • Ability to measure connectivity duration lengths of devices. • Ability to keep track of communication range to device. • Ability to measure air pollution. • Ability to measure other environmental data such as noise, temperature, humidity, atmospheric pressure and light. • Ability to measure data transfer in bytes.
Health	• Ability to monitor patient data and detect anomalies. • Ability to track time taken for system to alert a doctor, and he/she responds. • Ability to detect when resources need to be scaled according to system loads.
Infrastructure	• Ability to identify system resources. • Ability to configure experiments. • Ability to control various system entities and processes.
Education	• Ability to setup lectures and experiments for students. • Ability to deliver multimedia content.

Cross Domain	• Ability to interconnect different domains (both domains and applications).

5 Non-functional requirements

This section considers the non-functional requirements as presented in deliverable D2.2 [4] as well as the newly added requirements after the first WP3 integration activities concluded.

5.1 Generic non-functional requirements

Table 7: Generic non-functional requirements

Non-functional requirement	Description
Reliability	As presented in deliverable D2.2 [4] the solutions should be able to compensate for unreliable data transmission through aspects of delay tolerant networks (DTN) and Store and Forward (SAF).
Scalability	Scalability relates to the ability of systems to seamlessly cater for higher demand in computing resources of data, devices, people and applications.
Extensibility	Extensibility of all the system components has to be taken into account, from providing hardware platforms to integrate multiple sensors to middleware software to testbed infrastructures. The M2M middleware should be capable of receiving the data from multiple types of sensors: physical or emulated. The Smart City Platform should be able to support not only multiple applications for the different use-cases, but also cross-domain applications. Specific applications computing statistics and anonymization, should also be able to interact with the Smart City Platform if necessary.
Addressability	To ensure that the system software components can easily integrate and communicate, the cloud platform will be able to assign IP addresses accessible from the outside. The software components running on the cloud will have to recognize the requests destined to the IP address visible from outside.
Maintainability	The system should expose functionality to the infrastructure maintenance to update the functionality. For example the cloud platform should be able to deploy new instances with new software versions.
Mobility	The system has to be able to establish connection also when one end is moving, for sensors that are stationary and mobile data collectors. Mobility can also be considered regarding the cloud deployed components that have to adapt to newly deployed components.
Testability and Ability to Experiment	In order to offer the ability to experiment from small scale to large scale, the system has to support heterogeneity of applications and devices and even support cross-domain applications on the high level.

5.2 Risk and mitigation

Table 8: Non-functional requirements risk and mitigation

Risk	Impact	Mitigation
Connectivity loss for mobile devices, buses, sensors	Data is not transmitted.	Store the data until connectivity is back again.
Heterogeneity of IP versions in the system	Connectivity between multiple sites or testbeds is hindered.	Develop interworking proxies.
Heterogeneity of hardware	System is too complex to manage installation and deployment.	Use a cloud platform to abstract the storage, computing and networking resources.
Security of user data not ensured (as described in deliverable D2.2 [2])	Users will not use/endorse the system.	Transmission using encryption, either CoAP with DTLS or HTTPS.
Privacy (as described in deliverable D2.2 [2])	User data is open to third party and user does not use/endorse the system.	Using access rights over data exposed by the Smart City Platform to the applications.
Interoperability (as described in deliverable D2.2 [2])	Ineffective data transmission.	Using standard protocols like CoAP, HTTP, or system platforms like ETSI M2M, oneM2M. Clear direction of the information mapping is required.
Reliability	Untrusted environment.	Provide manual override in environments. Provide a network health overview (observable resources). Utilize Store and Forward (SAF) capability at gateways.
Scalability	Inability to perform operations at the required speed resulting in delayed executions.	Use a cloud platform.
Extensibility	Unable to deal effectively with new components.	Provide published APIs to allow for interworking proxies.
Addressability	Unable to address the various components.	Utilise standard Internet components (DNS) as well as those provided by cloud services.

Maintainability	Unable to effectively support and ensure stability of environment.	Create and implement management APIs.
Mobility	Inability to communicate between cloud and edge devices.	Utilise delay tolerant mechanisms.
Testability	Inability to effectively test impact of various configurations.	Utilise internationally accepted testbed frameworks.

6 Technology Requirements

Functional requirements, as described in Section 4, lead to technological building block selections. Technology related to future internet experimentation, Smart Cities, networks, protocols, testbeds, and other components, are evolving at a rapid rate. There is a multitude of possibilities and options.

In deliverable D2.2, preliminary "Technical Requirements" were presented. Using influences from planned experiments (deliverable D4.1) and the analysis presented in this report, the following technologies and associated standards were identified.

6.1 Cloud Environment

In order to provide testbed infrastructures and tools (for running experiments with an increased level of flexibility) for presenting the TRESCIMO software components as a service to the experimenter, several cloud platforms have being analysed. Some of the platforms are open source, for example OpenNebula [10] and OpenStack [11], and some are non-open source, e.g. Citrix [12] and Rackspace [13]. For enabling access to a larger scientific community, only open source projects were selected. When comparing OpenNebula to OpenStack, it appears that OpenNebula is more stable and easier to install, yet OpenStack has gained significant momentum as it is now driven by vendors as well. In the recent releases, OpenStack has evolved significantly on the networking abstraction component called Neutron to provide OpenFlow capabilities, VPNaaS, FirewallaaS with plugins/modules provided by vendors. Plugin functionality can be substituted with that of a vendor or open source version and the solution as a whole acts in a transparent way. In order to attract the attention of possible industry partners, OpenStack was chosen.

6.2 Secure Testbed Connection

To enable a secure connection between the various testbed deployments, and to overcome NAT addressing challenges between the endpoints, a tunnelling solution is necessary. We have distinguished two suitable major technologies with these being VPN and IPSec. IPSec is based on IETF standards [14], whereas VPN is considered easier to configure and is also supported on the Linux and Windows operating system as well as Android for mobile phones. With this in mind, a VPN solution is planned for the integrated experimental facilities [15].

6.3 Dynamic Configuration

For providing dynamic configuration of the nodes (related to the time they should transmit or the devices they should interact and moreover enable software and firmware management as well as log retrieval) device management protocols have been analysed in the TRESCIMO deliverable D3.1 [4].

The OMA LW M2M protocol linked to CoAP, with support for queued transmission for end devices that are not currently connected or in idle mode, was chosen in preference to OMA DM (that is based on HTTP and could put in danger a back end that is managing a large scale deployment).

6.4 Low Power Communication: IEEE 802.15.4

IEEE 802.15.4 specifies the physical layer and media access control for low-rate wireless personal area networks. It is the foundation for a number of higher level specifications including 6LoWPAN (used in this application), ZigBee and WirelessHART.

The physical layer implementation defines three possible unlicensed frequency bands:

- 868.0 - 868.8 MHz: primarily used in Europe, has a maximum rate of 100 Kbit/s, and allows one communication channel.
- 902 - 928 MHz: primarily used in North America, has a maximum rate of 250 Kbit/s, and allows up to 30 communication channels.
- 2400 - 2483.5 MHz: permitted worldwide, has a maximum rate of 250 Kbit/s, allows up to 16 communication channels, and used in this application.

The specification also defines the media access control that includes a management interface, physical channel access and network beaconing.

The network model defines the following node types:

- A full-function device (FFD) as a common node that can communicate with any other node and relay messages, implemented in devices such as ActivePlug and ActiveDIN.
- A full-function device, acting as a private area network (PAN) coordinator. This is referred to as a border-router in our application, implemented in the ActiveGate device.
- A reduced-function device (RFD), designated for nodes that have limited energy and resources. These nodes cannot act as PAN coordinators, and may also be implemented such that they are in a sleeping/powered down mode most of the time.

802.15.4 networks may be implemented as star or peer-to-peer topologies. This implementation is as a peer-to-peer topology.

In the DTN-based Spanish Trial, the communication between the wake-up sensor devices and the mobile collectors will be performed using 802.15.4 raw data frames at 2.4GHz. This solution is chosen in front of 6LoWPAN or ZigBee in order to optimize data exchange in terms of duration and message size, what is critical in the DTN scenario, where there is a short communication window between the mobile gateway (installed in the bus) and the sleeping wake-up sensor.

6.5 IPv6

Due to the exhaustion of the IPv4 address space, it makes sense to design the future internet devices to make use of IPv6 from the start. In this implementation, the ActiveGate device acts as an IPv6 router between an upward IPv6 network (connecting multiple gateways) and a localised IPv6 device network for the sensors and actuators. The border-router provides a 64 bit IPv6 network space to the devices which use their unique 64 bit identifiers as the host part of their IPv6 addresses. This allows all devices to be directly addressable from a public IPv6 network.

6.6 6LoWPAN

In order to make use of IPv6 on an 802.15.4 network, an adaptation layer is required. This is due to the 127 octet maximum physical payload size limitation of 802.15.4 and the 1280 octet minimum size payload requirement of IPv6. The Internet Engineering Task Force (IETF) published specifications (RFC6282) for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) to ensure interoperable implementations [16]. These included compression and management mechanisms.

6.7 RPL

A routing protocol is required to create and maintain routes in the low power RF network as these networks are typically characterised by high loss rates, low data rates and instability. The IPv6 Routing Protocol for Low power and Lossy Networks (RPL) was designed and specified (RFC6550) for devices with constrained resources and constrained connectivity [17]. The protocol enables the automated configuration of IPv6 addresses (by specifying the network prefix to each node) as well as management of routes. Routes are defined as Destination Oriented Direct Acyclic Graphs (DODAGs) that allow upward and downward data flow. This implementation has the border-router set up as the Direct Acyclic Graph (DAG) root.

6.8 CoAP

CoAP (RFC7252) [18] is used at various levels of the system due to its suitability for Machine-to-Machine (M2M) communications: multicast, low overhead cost and simplicity. To facilitate the implementation on constrained devices, a number of design choices have been considered including: i) the use of UDP as the transport layer protocol to avoid the overhead of connection oriented protocols; ii) efficient packing of protocol information in the header, which can be as small as 4 bytes. It maps easily to HTTP, enabling simple proxy implementations. The sensing devices publish their resources in the CoRE link format with CoAP as the service protocol. CoAP is used by an interworking proxy to collect these and push them into the M2M Gateway, and to facilitate communication between the M2M Platform and the Smart City Platform.

In the DTN-based Spanish trial, CoAP is used in the mobile collector installed in the bus to implement the binding between the internal gateway logic and the OpenMTC gateway.

6.9 CoRE link format

RFC6690 defines the Constrained RESTful Environments (CoRE) Link Format [19]. In particular, the Well-Known ‘core’ URI is implemented in the sensing and actuating devices to provide a list of resources that are available.

6.10 Wake-Up Radio (WuR)

The Wake-Up Radio technology used in the Spanish DTN pilot has been developed by I2CAT, based on the initial work described in [20]. This technology is based on the AS3932 IC from Austria Microsystems, which is a low frequency wake up receiver operating at 125 KHz. This IC has a correlator that can be programmed with a 16 bit long address. Thus, upon address detection the AS3932 triggers an interruption that wakes up the main host micro-processors, which then transmits the cumulated data through the IEEE 802.15.4 radio.

In order to allow for longer ranges, the gateway transmits the 125 KHz wake up address modulated as the envelope of an 868 MHz carrier, which can be detected at the receiver through a properly tuned adaptation circuit, achieving thus ranges of up to 40 meters. The range of the wake up radio shall be enough to wake the sensor and recover the data at typical bus speeds.

6.11 IEEE 802.11

The mains powered environmental sensors in the Spanish DTN pilot will use an IEEE 802.11 radio [21] for data transmission. Both IEEE 802.11n or 802.11g operating at 2.4GHz, and providing data rates of at least 20 Mbps, are considered appropriate for the intended purposes. IEEE 802.11 is chosen because in this case power is not a concern, and the available data rates favour the transmission of the large data payloads potentially generated by environmental sensors, in the short communication window available between the bus and the sensor. The 802.11 client interface shall be able to

recognize the Beacons from the 802.11 Access Point running in the bus and autonomously start the association process. Transmission will be performed over UDP.

6.12 Cellular networks (2.5G/3G/4G)

The mobile collector used in the Spanish DTN pilot will transfer data to/from the remote platform through a cellular link. In the project, a commercial mobile provider will be used. The reason for providing the cellular link is to allow permanent connectivity in order to minimize the time between the collection of data and its transmission to upper platforms. This will also allow sending requests to the collector and sensor devices. In a first instance, a GPRS/UMTS connection will be used; though this could be upgradable in future if the dimensions or data requirements of the deployment increase.

6.13 FITeagle

FITeagle is a framework for managing and federating testbeds. . It's implemented as an extensible and modularized architecture on top of the Java Enterprise Edition (J2EE) platform. It currently focuses on, but is not limited to, offering FIRE-related APIs for testbeds to support the experiment life-cycle of the heterogeneous facility's resources and services.

In the context of the TRESCIMO project the FITeagle framework serves multiple purposes. First, it acts as a Slice-based Federation Architecture (SFA) compliant Aggregate Manager (AM) for the whole TRESCIMO federation. This allows to offer the services of the VPN-interconnected testbeds to the FIRE and GENI community (the integration into the Fed4FIRE federation has been started). Second, it acts as a middleware between external users and the actual service provisioning that is being conducted by the Fraunhofer OpenSDNCore Orchestrator. Third, it will provide distributed testbed monitoring information for first level support (FLS) purposes (i.e. pushing monitoring data to other federations like Fed4FIRE and collecting and visualizing internal monitoring information).

6.14 ETSI M2M

ETSI M2M standardization was based on service requirement for M2M in different use cases, such as Smart Metering and E-Health [22]. Recently, ETSI M2M work is transferred to oneM2M consortium aiming to consolidate the standardization work in M2M communications and develop technical specifications addressing the needs for a common M2M service layer.

6.15 OneM2M

OneM2M aims to be the world wide accepted standard for M2M/IoT communication. The oneM2M consortium was established in 2012, with the aim of consolidating the standardization work in M2M communication [23]. More than 260 participating partners and members has joint oneM2M to participate in the standardization work of M2M communication system. OneM2M specifies a high-level architecture at both the field and infrastructure domain, to support end-to-end M2M services. The oneM2M functional architecture comprises of: Application Entities (AE) that are responsible of providing end-to-end M2M logic solution, Common Service Entities (CSE) comprises a set of Common Service Functions (CSF) that are common to the M2M environment and exposed to other entities through four Reference Points named Mca, Mcn, Mcc and Mcc', and the Underlying Network Services Entity (NSE): to provide services to the CSEs, such as device management, location services and device triggering

The Scope of oneM2M work includes providing open standard interfaces and APIs to enable access independent view of end-to-end services.

In the TRESCIMO context it is envisioned that both ETSI M2M and oneM2M functionalities will be supported.

6.16 Smart City Platform

Once reliable communication has been established (linking the physical environment with the broader Internet), a platform is required on which a collection of applications can be run. A large collection of such platforms exists; see for example [24], [25]. Standards are yet to be created. In the context of re-using existing building blocks, an in-house CSIR Smart City Platform is used. This platform provides the ability to link to a variety of application and communication mechanisms. In addition, it provides for big-data storage and retrieval as well as a model of the real-world.

7 Synthesized Comments Regarding Specifications

Analysing the above functional and non-functional requirements (as linked to the aim of creating a testbed able to support both experiments in which the environment is configured and controlled, as well as typical applications within a real-world context, for example the environmental scenario), the following comments can be made:

- The architecture considers the reliability of the system by including a Delay Tolerant Network Gateway in the components that will run in the mobile entities (e.g. Bus) (also referred to as the “Aggregator service” above).
- During transmission, the Security and Privacy of user data will be ensured by means of the CoAP protocol together with the DTLS or HTTPS protocols when transmitting data from sensors to M2M Middleware, between the M2M Middleware (“Machine to Machine service”) and the Smart City Platform (“back-end service), and from the mobile application (“application”) to the Smart City Platform (“back-end service”).
- In order to provide extensibility of the testbeds, the cloud platform will abstracted out of the hardware and the cloud platform will contain a component that will be able to allocate/instantiate topologies on multiple datacenters/cloud deployments following a uniform approach. To this end, the OpenSDNCore orchestrator (being a tool built by FOKUS and TUB) will interface with the FITeagle in order to provide one installation of the orchestrator which is capable of managing resources that span over multiple datacenters.
- For increased reliability (apart from DTN and SAF technologies) we have to consider that the transmission might have unexpected retransmissions caused by the time window being exceeded. Retransmissions might be triggered just when the acknowledgement would come, causing duplicated data to be transmitted.
- Interoperability was introduced in deliverable D3.1 [5] using Interworking Proxies in order to adapt between two software components and make the translation to the M2M middleware. This leads to a) an Interworking proxy serving the Smart City Platform and the M2M Middleware, b) several Interworking proxies from the DTN gateway from the environmental monitoring or the gateway as extracted from the smart building/energy use-case.
- In the architecture from deliverable D3.1 [5] a device management client running on the M2M gateway and a management server, interacting with the M2M server. This combination provides configuration management for devices on the bus route. Functionality as firmware and software management is built on the above components and the end devices can implement the appropriate actions when receiving the trigger. Support for queued device management messages to the device will be introduced in extensions to the reference implementation.
- For increased mobility of the software components, the OpenSDNCore orchestrator will interact with a Device Management Server to announce the presence of the M2M middleware components so that the end devices, running a device management client, can send the data to the right location in terms of IP and port.

8 Conclusion

Compiling a comprehensive list of both functional and non-functional requirements as these relate to an experimental framework for Smart City (as implemented in developing and developed worlds) is a daunting task. For TRESCIMO, a methodology of creating scenarios, use-cases, and experiments was used to extract a snapshot of applicable requirements. This report contains a synthesis and expansions on previous deliverables. The combination of all the requirements created through an iterative process led to an architecture specification. This architecture evolved as the requirements have evolved. Using a combination of all the inputs received and the resultant architecture [4], a reference implementation is being developed. Once again an iterative process will be followed to refine the reference architecture to ensure that the appropriate functional and non-functional requirements are addressed.

References

[1] L. Coetzee, “TRESCIMO Scenario Specification (D2.1),” 2014.

[2] H. Madhoo, “TRESCIMO User and Technical Requirements (D2.2),” 2014.

[3] J. Mwangama and N. Ventura, “Experiment Description (D4.1),” 2014.

[4] A. Corici, “Architecture Specification (D3.1),” 2014.

[5] “TRESCIMO,” [Online]. Available: www.trescimo.eu.

[6] “Future Internet Research & Experimentation,” [Online]. Available: http://www.ict-fire.eu.

[7] “SmartSantander,” [Online]. Available: http://www.smartsantander.eu/.

[8] L. Peterson, S. Sevinc, J. Lepreau and R. Ricci, “Slice-based Federation architecture,” GENI, 2009.

[9] T. Rakotoarivelo, G. Jourjon and M. Ott, “Designing and Orchestrating Repro- ducible Experiments on Federated Networking Testbeds,” *Computer Networks, Special Issue on Future Internet Testbeds,* 2014.

[10] “OpenNebula,” [Online]. Available: http://opennebula.org.

[11] “OpenStack,” [Online]. Available: http://www.openstack.org.

[12] “Citrix,” [Online]. Available: http://www.citrix.com/products/cloudplatform/overview.html.

[13] “Rackspace,” [Online]. Available: http://www.rackspace.com/cloud.

[14] “Security Architecture for the Internet Protocol,” [Online]. Available: https://tools.ietf.org/html/rfc4301.

[15] “OpenVPN,” [Online]. Available: https://openvpn.net/.

[16] “IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs),” [Online]. Available: https://datatracker.ietf.org/doc/rfc4919/.

[17] “RPL: IPv6 Routing Protocl for Low-Power and Lossy Networks,” [Online]. Available: https://tools.ietf.org/html/rfc6550.

[18] “The Constrained Application Protocol (CoAP),” [Online]. Available: https://tools.ietf.org/html/rfc7252.

[19] “Constrained RESTful Environments (CoRE) Link Format,” [Online]. Available: https://tools.ietf.org/html/rfc6690.

[20] J. Oller, I. Demirkol, J. Casademont and J. Paradells, “Design, development, and performance evaluation of a low-cost, low-power wake-up radio system for wireless sensor networks,” *ACM Trans. Sen. Netw. ,* vol. 10, 2013.

[21] “802.11-2012 - IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Spec”.

[22] “ETSI TS 102 689 V1.1.2: Machine-to-Machine communications (M2M): Service Requirements,” ETSI, 2011.

[23] “oneM2M,” [Online]. Available: http://onem2m.org/.

[24] “Postscapes: Tracking the Internet of Things,” [Online]. Available: http://postscapes.com/internet-of-things-platforms.

[25] C. Doukas, “Building Internet of Things with the Arduino,” [Online]. Available: http://www.buildinginternetofthings.com/internet-of-things-list-of-platforms/.

[26] O. Mehani, G. Jourjon, J. White, T. Rakotoarivelo, R. Boreli and T. Ernst, “Characterisation of the Effect of a Measurement Library on the Performance of Instrumented Tools,” NICTA (4879), 2011.

[27] M. Singh, M. Ott, I. Seskar and P. Kamat, “ORBIT Measurements framework and library (OML): motivations, implementation and features,” in *IEEE Trident- com: First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2005.

[28] T. Rakotoarivelo, G. Jourjon, M. Ott and I. Seskar, “OMF: a control and management framework for networking testbeds,” *Operating systems review,* 2009.

