

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Mobile Healthcare on a M2M Mobile System

Ricardo Jorge Travanca Morgado

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Doctor Ana Cristina Costa Aguiar

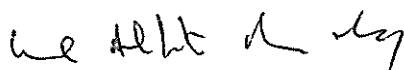
July 31, 2014

A Dissertação intitulada

“Mobile Healthcare on a M2M Mobile System”

foi aprovada em provas realizadas em 22-07-2014

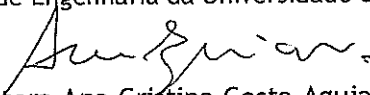
o júri



Presidente Professor Doutor Manuel Alberto Pereira Ricardo
Professor Associado do Departamento de Engenharia Eletrotécnica e de
Computadores da Faculdade de Engenharia da Universidade do Porto

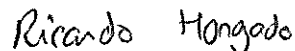


Professor Doutor Adriano Jorge Cardoso Moreira
Professor Associado do Departamento de Sistemas de Informação da Universidade
do Minho Escola de Engenharia da Universidade do Minho



Professora Doutora Ana Cristina Costa Aguiar
Professora Auxiliar do Departamento de Engenharia Eletrotécnica e de
Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.



Autor - Ricardo Jorge Travanca Morgado

Resumo

Os desenvolvimentos tecnológicos na sociedade atual estão a evoluir para um futuro onde objetos mundanos terão a capacidade de comunicar entre si. Estas comunicações nem sempre terão interação humana e serão, provavelmente, autónomas, sendo chamadas de comunicação máquina a máquina (*Machine to Machine*, ou M2M). No entanto, para diferentes serviços, aplicações e dispositivos terem a capacidade de comunicar entre si, existe a necessidade de criar uma base, sobre a qual a comunicação deve ser estabelecida. Isto é conseguido através da standardização de comunicações.

Esta dissertação implementa um dos atuais standards para M2M, criado pelo ETSI (European Telecommunications Standards Institute) e implementa-o num cenário de saúde, com o objetivo de comunicar entre duas aplicações diferentes, criadas de acordo com o standard. Primeiramente, a *Gateway* (GW) móvel M2M é responsável pela conexão e gestão de sensores médicos, enquanto recebe comandos da *Network Application* (NA) móvel M2M que, por sua vez, também mostra os resultados das medições em tempo real. Ambas as aplicações foram implementadas no sistema operativo Android e testadas num *smartphone* com Android 4.4 (KitKat).

No decorrer do projecto surgiu a ideia que a comunicação através do servidor standardizado pelo ETSI, o NSCL, poderia ser ineficiente no que toca aos escassos recursos do *smartphone*. Para colmatar esse problema foi projetada e criada uma ligação local, com o intuito de poupar tráfego nas comunicações e economizar bateria. Os dados transferidos através desta ligação local são guardados para serem, posteriormente, enviados para o NSCL, garantindo que estes não são perdidos. Para testar este conceito num cenário médico, um dispositivo *Bluetooth* de medição de batimentos cardíacos foi integrado com a GW móvel M2M.

O produto desta dissertação é um protótipo funcional de ambas as aplicações, que permite ao utilizador procurar, começar e parar o sensor suportado que esteja nas imediações da GW móvel M2M, usando a interface gráfica básica da NA móvel M2M. A comunicação entre as duas aplicações está funcional, tanto através do NSCL, como através da ligação local projetada. Esta, após medições de tráfego, provou ser eficiente na redução dos dados transmitidos.

Abstract

The technological developments in today's society are evolving into a future where everyday objects will have the ability to communicate with each other. These communications will not always have human interaction, but will most likely be autonomous, hence called Machine-to-Machine communications (M2M). However, for different services, applications and devices to be able to share information, there is the need to create common ground, so they all can communicate with each other. This is accomplished by standardizing these communications.

This dissertation implements one of the current M2M standards, created by ETSI (European Telecommunications Standards Institute), and develops it in a mobile healthcare scenario aiming to be able to communicate between two different applications created according to the standard. Firstly, the mobile M2M Gateway (GW) is responsible for the connection and management of medical sensors, while receiving commands from the mobile M2M Network Application (NA), that also shows the measurement results in real time. Both applications were implemented for Android OS and tested on a smartphone running Android 4.4 (KitKat).

During the course of the dissertation, the idea arose that the communication between the two applications, normally made through an ETSI standardized server entity, the NSCL, could be inefficient to the smartphone's scarce resources, when both applications were running in the same device. Therefore, a local connection was designed and created in order to fulfill this need to save network traffic and battery, while keeping the data stored for delivery to the NSCL in a posterior time, guaranteeing that it is not lost. To test the concept in an healthcare scenario, a bluetooth heart-rate monitor was integrated within the mobile M2M GW.

This dissertation's work presents a working prototype of both applications, that allow the user to search, start and stop the supported sensor near the mobile M2M GW, while using a basic graphical user interface of the mobile M2M NA. The communication between the two applications is functional through the standardized NSCL, as well as through the designed local connection, which after some traffic measurements was proven to be effective in reducing the transmitting data.

Acknowledgments

I would like to dedicate this dissertation to the following people, that in their own way helped me through my academic career and gave me the strength and motivation needed to overcome all the obstacles.

First and foremost, to my parents, to whom I owe everything I am today, and for being my biggest inspiration. Even after all the difficulties they have been through, they keep very humble and it's them I look up to, and wish to be some day.

To my grandparents, which are also very dear to me and who I'm blessed to have in my life. Their selflessness and dedication to family are unimaginable and their personalities also were a great inspiration when growing up.

To my love, Cláudia, I'd like to thank the patience and understanding when the college work came first, and for being there supporting and giving me strength and purpose to keep pushing my own goals.

To Lhorca and Mafalda, for being there for me since the beginning of my degree and helping me every step of the way, being my safe haven in darker times. Your friendship and company has helped me grow into the man I am today.

To Canasta, for being a role-model in dedication and approach to problems, while also showing that it is possible (and essential) to sometimes take a step back and enjoy the company of important people in your life. A great word of appreciation for the effort put into reading this dissertation in order to help me understand where there was room for improvement.

To the *Tuna de Engenharia da Universidade do Porto* and its members, for all the moments and lessons that it provided, that made me grow as a person, and for giving me the opportunity to make the great friends I have today.

To Carlos Pereira, for all the advices, help configuring networks, and for generally being there for me, when I needed help with something related with the project. A special word of appreciation for reviewing this document.

A special word of appreciation for all the help Pedro Rocha and Alberto Correia, from *PT Inovação*, gave me during the development and testing of this project.

To Doctor Ana Aguiar, for mentoring me ever since I started working in *Instituto de Telecomunicações*, and for helping me and teaching me every step of the way, helping to shape me into the professional that I have become.

Ricardo Morgado

“I work on the motto that if something’s not impossible, there must be a way of doing it.”

Sir Nicholas Winton

Contents

1	Introduction	1
1.1	Context of the Project	1
1.2	Motivation	2
1.3	Objectives	3
1.4	Structure	3
2	State of the Art	5
2.1	Publish-Subscribe Paradigm	5
2.2	M2M Communications	6
2.2.1	ETSI M2M Standard	7
2.2.2	ITU M2M Service Layer	10
2.2.3	OMA Lightweight M2M	12
2.3	Communication Protocols	15
2.3.1	Representational State Transfer	16
2.3.2	MQTT	18
2.3.3	AMQP	18
2.4	eHealth	19
2.4.1	Application Revision	20
2.4.2	Communication Technologies	21
2.5	Discussion	23
3	Aspects of ETSI M2M Standard	25
3.1	ETSI M2M Standard	25
3.1.1	Resource Structure	25
3.1.2	M2M Gateway	27
3.1.3	M2M Network Application	28
3.1.4	ETSI M2M Implementations	29
3.2	Marshalling	30
3.2.1	XML	30
3.2.2	JSON	30
3.3	Data Security Mechanisms	31
4	ETSI Compliant Mobile M2M System	33
4.1	Problem	33
4.2	Approach	33
4.2.1	Healthcare Use Case	34
4.2.2	System Design	35
4.2.3	Local Interface	39

4.2.4	Mobile Applications Design	41
4.3	Evaluation	50
5	Implementation	53
5.1	Technologies	53
5.2	ETSI Protocol Implementation	54
5.3	Bootstrap	54
5.4	Web-Server	55
5.5	ETSI Resource Structure Implementation	56
5.6	Sensor Integration	57
5.6.1	Sensor data storage	57
5.6.2	Zephyr data	58
5.7	Communication considerations	58
5.7.1	Subscription Check	58
5.7.2	Persistent Database	59
5.7.3	Commands	60
5.8	Mobile M2M GW	61
5.8.1	Android Implementation	61
5.8.2	Communication	62
5.8.3	Commands handling	64
5.9	Mobile M2M NA	66
5.9.1	Android Implementation	66
5.9.2	Communication	68
5.9.3	Interface Commands	70
5.10	Local Interface	72
6	Results	75
6.1	Proof of Concept	75
6.2	Traffic Measurements	81
6.2.1	Resource Registration	82
6.2.2	Local Interface vs NSCL	83
6.2.3	Discussion	84
7	Conclusions and Future Work	87
7.1	Conclusions	87
7.2	Future Work	89
A	ETSI classes changes	91
B	Mobile M2M GW Logs	93
B.1	NSCL	93
B.2	Local API	96
B.3	Connectivity Lost Test	100
B.4	Disconnection after using local API	106
C	Mobile M2M NA Logs	113
C.1	NSCL	113
C.2	Local API	116
C.3	Connectivity Lost Test	119

C.4 Disconnection after using local API 123

List of Figures

1.1	Smartphone market share by Operating System in 2013. From [1]	2
2.1	Communication comparison	6
2.2	Machine-to-Machine high level architecture. From [2].	8
2.3	M2M Service Capabilities functional architecture framework. From [2].	10
2.4	ITU-T M2M Service Layer in the IoT reference model. From [3].	11
2.5	ITU-T M2M Reference points between device, gateway and network application server. From [3].	13
2.6	Lightweight M2M architecture and protocol stack. From [4].	13
2.7	HTTP packet	16
2.8	CoAP resource-observe. From [5].	17
2.9	CoAP packet	18
2.10	MQTT packet	18
2.11	AMQP functionality overview. From [6]	19
2.12	AMQP packet	19
2.13	IEEE 11073 Framework. From [7].	22
2.14	Continua Alliance profile of standards. From [7].	23
3.1	Illustration of the ETSI M2M resource structure	26
3.2	ETSI M2M smart-home example	26
3.3	Simplified M2M GW resource structure.	27
3.4	Simplified M2M NA resource structure.	29
4.1	UML 2.0 Use Case Diagram for the study scenario	34
4.2	Architectural overview of integrated solution	36
4.3	Mapped mobile M2M GW resource structure.	37
4.4	First Approach.	37
4.5	Second Approach.	38
4.6	Mapped mobile M2M NA resource structure.	39
4.7	NSCL as information forwarder.	40
4.8	Local Interface Communication.	40
4.9	High-level Mobile M2M GW Architecture	42
4.10	Mobile M2M GW Protocol Manager Architecture	43
4.11	Mobile M2M GW Flowchart	44
4.12	Mobile M2M GW Commands Flowchart	45
4.13	High-Level Mobile M2M NA Architecture	47
4.14	Mobile M2M NA Protocol Manager Architecture	48
4.15	Mobile M2M NA Flowchart	49
4.16	Mobile M2M NA Protocol Manager Architecture	50

5.1	Bootstrap Procedures	55
5.2	ECG example, specifying RR interval. From [8]	58
5.3	Procedure to check if subscription created by this mobile M2M GW or NA is present.	59
5.4	Storage Class Diagram.	60
5.5	Class diagram of the structure of the commands.	61
5.6	Sequence diagram of the communication held between the mobile M2M GW and the NSCL.	62
5.7	Mobile M2M GW handling of commands received from the NSCL	65
5.8	Network Application start screen.	66
5.9	Network Application commands screen.	67
5.10	Network Application commands screen with received sensors.	67
5.11	Network Application measurements interface.	68
5.12	Sequence diagram of the communication held between the mobile M2M NA and the NSCL.	69
5.13	Sequence diagram of the mobile M2M NA commands.	71
5.14	Sequence diagram of the communication between the mobile M2M GW and NA using the local interface.	72
6.1	Battery test studies. From [9].	82

List of Tables

4.1	Evaluation metrics	51
6.1	Mobile M2M applications footprint.	81
6.2	Resource Registration Traffic	83
6.3	Network traffic comparison table	83
6.4	Evaluation metrics	85
A.1	Summary of affected classes	92

Abbreviations and Symbols

3GPP	3rd Generation Partnership Project
AMQP	Advanced Message Queue Protocol
API	Application Programming Interface
BPM	Beats per minute
CA	Certificate Authority
CoAP	Constrained Application Protocol
CR	Cardiac Rehabilitation
CRUD	Create, Retrieve, Update and Delete
DSCL	Device Service Capabilities Layer
DTLS	Datagram Transport Layer Security
eHealth	Transfer of health resources and healthcare by electronic means
ETSI	European Telecommunications Standards Institute
FCAPS	Fault, Configuration, Accounting, Performance and Security
FIFO	First-in First-out
GPS	Global Positioning System
GSCL	Gateway Service Capabilities Layer
GW	Gateway
GUI	Graphical User Interface
H2M	Human-to-Machine
HTTP	Hypertext Transfer Protocol
Id(s)	Identifier(s)
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPsec	Internet Protocol Security
ISO	International Organization for Standardization
IT-Porto	<i>Instituto de Telecomunicações, Porto</i>
ITU-T	Telecommunication Standardization Sector of the International Telecommunication Union
JSON	JavaScript Object Notation
LWM2M	Lightweight M2M
M2M	Machine-to-Machine
MQTT	Message Queuing Telemetry Transport
NA	Network Application
NDA	Non-Disclosure Agreement
NSCL	Network Service Capabilities Layer
OEM	Original Equipment Manufacturer
OM2M	Open-source M2M

OMA	Open Mobile Alliance
PKI	Public Key Infrastructures
PKIX	Public Key Infrastructure X.509
PmEB	Patient-Centered Assessment and Counseling Mobile Energy Balance
PTIN	<i>Portugal Telecomunicações Inovação</i>
pub-sub	Publish-Subscribe
R&D	Research and Development
REST	Representational State Transfer
RFC	Request for Comments
RSA	Rivest Shamir Adleman
SASL	Simple Authentication and Security Layer
SCL	Service Capabilities Layer
SCs	Service Capabilities
SNI	Server Name Indication
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

Chapter 1

Introduction

In a society irreversibly marked by the everyday use of technology, new ways to automatically share data and automate systems and decisions are constantly emerging. Machine-to-Machine (M2M) solutions are becoming increasingly popular, for the great scalability they provide, and because they generally make people's lives easier. An easily understandable example of M2M system is GPS (Global Positioning System) [10, 11]. There is no human interaction, and yet by gathering and processing data from satellites, it is able to compute the user's current position. The ease of use of these systems grant them the massive adoption they hold in today's society.

The concept of the Internet of Things (IoT) [12], states that in the near future, everyday objects as simple as food packages, paper documents or even furniture will have the ability to be connected to each other and to the Internet, providing services that can hopefully ease people's lives. M2M systems can act as a possible enabler of this IoT future, by providing the common ground standardization needed for the objects to interact with each other, and enable the creation or services that rely on their information.

The context of the project in which this dissertation is inserted is explained in Section 1.1, followed by the motivation for studying the subject at hand, in Section 1.2. The objectives this dissertation aims to achieve are stated in Section 1.3, before a brief explanation of the document's structure, in Section 1.4.

1.1 Context of the Project

This dissertation is integrated in a joint project between the *Instituto de Telecomunicações, Porto (IT-Porto)*, and *Portugal Telecomunicações Inovação (PTIN)*. M2M communications, which will be further explained in Section 2.2, enable the connection of sensors and services, providing the interfaces and guidelines to allow data to be seamlessly exchanged and stored, to be available to other applications and services authorized to access it. As an example, cities are starting to explore the possibility of using M2M Smart Grids [13, 14], that allow metering data to be automatically uploaded to the service provider, be it water, electricity or gas. There are also other scenarios in smart homes, where M2M communication can be used to autonomously optimize

and manage energy consumption of any appliance that runs on electricity [15] or even control the house temperature (like the Nest¹ equipment line does).

This dissertation's work aims to build two Android OS [16] applications, working in a system to show their potential enabling power for mobile M2M scenarios. The first application is a Gateway (standardized M2M entity) that will gather sensors data, store and process them accordingly for being registered in the M2M Network later. The data is then forwarded to the Network Application (also a standardized M2M entity) that give the user visual feedback, and allows him to perform actions. The healthcare scenario was chosen to allow the better grasp of the standard's specifications, acting as a proof of concept for other possible scenarios.

1.2 Motivation

Back in 1965, Gordon Moore published in a paper what would later be considered Moore's Law [17], where he predicted that the transistor count in microprocessors would roughly double each year. His expectation has been fairly accurate since the paper was published, and the transistor count keeps more or less doubling each year. This means that the processing power is constantly being improved, for every kind of device, and that derived from this, as well the significant improvement of telecommunication technologies (3G, 4G), we've seen an impressive growth in the smartphone industry as shown in Figure 1.1.

Operating System	2013 Units	2013 Market Share (%)	2012 Units	2012 Market Share (%)
Android	758,719.9	78.4	451,621.0	66.4
iOS	150,785.9	15.6	130,133.2	19.1
Microsoft	30,842.9	3.2	16,940.7	2.5
BlackBerry	18,605.9	1.9	34,210.3	5.0
Other OS	8,821.2	0.9	47,203.0	6.9
Total	967,775.8	100.0	680,108.2	100.0

Figure 1.1: Smartphone market share by Operating System in 2013. From [1]

The M2M scenarios mentioned in Section 1.1 have undoubtedly started to appear, but they most of the times are proprietary, difficulting the exchange of data between services and applications the user owns or relies on. Using standardized M2M communications could ease this difficulty in interchanging data, working towards an IoT future, which is limited today, by the fragmentation of M2M solutions [4]. The smartphone can then be an excellent choice as M2M service enabler, since it is permanently around the user. It possesses the necessary processing, battery, and connectivity power needed for interacting as an M2M endpoint, while retaining the mobility necessary to support a new wider range of mobile scenarios. In these, data could be shared between standard compliant M2M devices, services and applications, be it mobile or not.

¹<https://nest.com/>

The scenario in study in this dissertation's project is mobile healthcare, where the user could use its smartphone to effortlessly make health-related measurement using a device he already carries every day to access data from nearby sensors, knowing that the measurements would then be stored online for him or his physician to access. This works as a proof-of-concept for the ability to communicate locally using M2M communications, paving the way for more scenarios to be developed.

1.3 Objectives

This dissertation is going to study how the smartphone can fit into this M2M system, using the ETSI M2M standard. Back when the joint project started in May 2013, it focused on the architectural design and implementation of the mobile M2M Gateway (GW). The functionalities of the M2M GW entity will be further explained in Section 2.2.1. At the beginning of this dissertation (September 2013), a first version of the mobile M2M GW had been released with the MQTT protocol [18] working. Then the project changed its direction to complying with the ETSI M2M Standard, and it became this dissertation's objective to create of a mobile M2M system composed by a mobile M2M GW and a mobile M2M Network Application (NA), working together in a healthcare scenario. Both applications were built on a modular structure that can ease the addition of new protocols, sensors and functionalities.

The mobile M2M Gateway gathers data from compatible healthcare sensors (such as heart-rate), and processes it to be sent to the NSCL (ETSI M2M "cloud" entity). The mobile M2M GW design has most modules running on their own threads, aiming to keep efficiency at a maximum, while featuring a bottleneck in the Internet connection. This was purposefully done to save battery by not having a constant data stream flowing. The mobile M2M NA will then receive the data from the NSCL and show it on the graphical user interface (GUI), as well as commands. With these, the mobile M2M NA will allow the user to send commands to control the mobile M2M Gateway remotely, triggering sensor searches and starting/stopping the readings. The M2M concepts will be further explained in Section 2.2.1 and the use cases will be addressed in Section 4.2.1. Given the constrained nature of smartphones, there is a special interest in saving its scarce resources, and so one of the objectives is also to determine a way to save the battery drain and network usage on 3G/Wi-Fi networks, by using a local bypass in certain situations. The two applications will work together as a system, and regardless of the local bypass being used or not, the feedback given to the user should be consistent, to ease the experience.

1.4 Structure

This document is structured in 7 chapters. After this introduction to the subject of this dissertation, Chapter 2 is focused on the state of the art research related to this project, followed in Chapter 3 by further specifications on the standard and technologies used.

On Chapter 4 the scenario and approach to the problem are explained, followed by the implementation, in Chapter 5, paving the way for results and their discussion in Chapter 6. Last but not least, Chapter 7 will summarize the conclusions to be taken from the work developed for this dissertation as well as possible future work.

Chapter 2

State of the Art

In this chapter there is going to be a closer study of M2M communications, explaining its importance in the evolution of information technology towards an IoT future. This chapter aims to help the reader fully understand the technologies involved in the development of this dissertation's project.

First, an explanation of the publish-subscribe paradigm on which M2M communication are based is explained in Section 2.1. Then, in Section 2.2, there is going to be a study into M2M communications, detailing 3 different standards. Section 2.3 will then detail some communication protocols compatible with the M2M standards or themselves enablers of M2M communications, namely HTTP, CoAP, MQTT and AMQP.

Section 2.4 will review some healthcare applications, followed by some eHealth technology standards that are used to communicate medical data. Finally, Section 2.5 will resume this Chapter's study, with a substantiated discussion on the decisions made for this dissertation's project.

2.1 Publish-Subscribe Paradigm

By 2002, the author of [19] stated that: "Wireless has experienced explosive growth in recent years, and 'push' will be the predominant wireless service delivery paradigm of the future". Push is another term for the publish-subscribe (pub-sub) paradigm that refers to a communication model that have the architecture necessary for users to subscribe to a topic, message, publisher, etc., and receive all the subsequent updates about it. This differs from the web communication model, which is request-response based. In the latter, a request is executed when information is desired (polling), but this is inefficient for some types of communication, such as sensor measurements, as it overloads the servers and networks with unnecessary requests and traffic, slowing down the response times, as well as causing a delay between the information generator (e.g. sensors) and its consumer (e.g. service, application).

Figure 2.1a shows the client is polling the server for the publisher's content, having the need to poll whenever there is new content. As the client does not know when there is new information, regular polling is needed, to check for updates. On the other hand, in Figure 2.1b, the client must

only subscribe once to receive notifications for the following publications, until the subscription is cancelled.

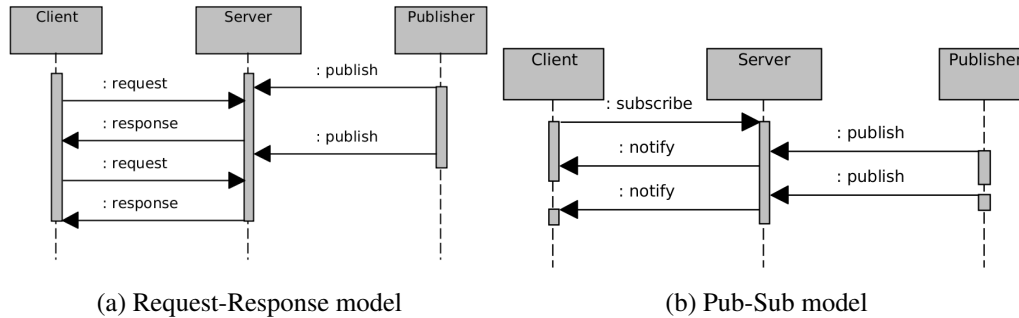


Figure 2.1: Communication comparison

Publishers and subscribers do not need to be actively participating in the interaction at the same time, or even know about each other. They can both produce and consume events in an asynchronous way (i.e. not online at the same time), which allows for very flexible scenarios.

The Request-Response model is ideal for situations where the client only wishes for the information once, like loading a web-page, whereas pub-sub is used when the client wishes to be notified of the publisher's activity updates. This means that the latter fits a sensing M2M scenario best, since it allows the communication to be held by subscribing the publishers that are important to the client, and receive its updates.

2.2 M2M Communications

"The exponential growth of wireless communication devices and the ubiquity of wireless communication networks have recently led to the emergence of wireless machine-to-machine (M2M) communications as the most promising solution for revolutionizing the future "intelligent" pervasive applications", [11]

As technology advances, M2M, which can also be called machine-type communications (MTC), will continue to increasingly replace the traditional human-to-machine (H2M) operations [15]. By definition, *M2M communications* is a term used to refer to data communications without or with limited human intervention amongst various terminal devices such as computers, embedded processors, smart sensors/actuators, mobile devices, software components, services etc. [20]. Since M2M, as opposed to H2M, does not need user interaction, the devices communicating with each other have some degree of decision making, and thus the services provided can be to some extent considered intelligent, as the quote above suggests. M2M communication presents itself as the enabling technology towards an IoT future, by allowing increase of devices that can be uniquely addressed and accessed, enabling them to interact with each other and cooperate to reach common goals [12].

There is interest in the development of M2M communications, since it can impact positively both domestic (e.g. domotics, assisted living, e-health, etc.) and industrial fields (e.g. automation,

manufacturing, logistics, business/process management, etc.) [12]. This can be done by allowing real time information from sensors to produce changes in the subscribing applications, services or devices, connected through the M2M system. This could mean slowing production line if a delivery truck with essential feedstock was delayed, preventing a jam in the production line, or even to trigger the alarm if an uninvited guest showed to the doorstep when the owner is on vacation.

In mobile scenarios, M2M communications can allow for resource optimization, reducing the usage of energy, network resources, computational cost, etc., allowing for more efficient and cheaper solutions for the consumers [5, 11]. These are accomplished by using energy oriented communications, that can have the processing executed in non-constrained devices, while creating those solutions on top of the M2M layer, allowing significant savings in R&D.

The number of M2M-enabled devices (terminals) is increasing exponentially, forecasted to grow from 50 million in 2008 to well over 200 million in 2014, and up to 50 billion by 2020 [15]. As the interest in M2M communications increases, so does the need to standardize it, in order to allow the roughly 50 billion devices to exist in 2020 to communicate with each other, and with services or applications. Naturally, some proprietary M2M solutions have already been proposed and/or deployed, but without an accepted standard used by manufacturers and companies, the IoT is still distant, due to the lack of interoperability. The following sections will give an overview of the current efforts to standardize M2M communications.

2.2.1 ETSI M2M Standard

In 2005, the 3rd Generation Partnership Project (3GPP) started with the standardization of M2M in the Global System for Mobile (GSM) and the Universal Mobile Telecommunications Systems (UMTS). In 2007 the technical report (TR) 22.868 (release 8) [21] completed the study on facilitating M2M communications in 3GPP systems, after which the 3GPP working group for M2M standardization was organized.

In January 2009, the European Telecommunications Standards Institute (ETSI), which is the independent and non-profit standardization organization in the telecommunications industry in Europe, picked up where 3GPP left off and continued the standardizing process of M2M, defining entities and functions to provide interoperable efficient end-to-end information delivery. ETSI published a batch of technical specifications that state the M2M service requirements, its communication, and also several use cases, which will be summarized in this section.

ETSI's M2M architecture provides a Service Capability Layer (SCL) that is common to all M2M applications, devices and services, providing a consistent resource structure that specifies the resources and collections hierarchy, creating the common ground between all M2M nodes (elements that are compliant with the specification). These resources and collections are called Service Capabilities (*SCs*), which is where the information is kept. The documents do not limit the possibilities of these *SCs*, leaving their use open to interpretation.

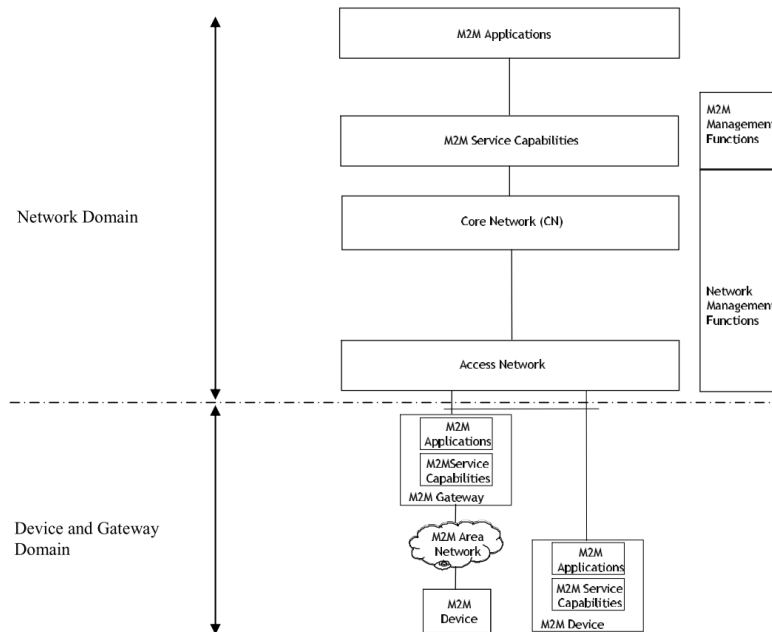


Figure 2.2: Machine-to-Machine high level architecture. From [2].

Figure 2.2 presents the high level view of the ETSI M2M system architecture as specified in [22, 2]. Two domains are depicted: The M2M device and gateway domain, and the network domain. The elements present in the device and gateway domain are:

- **M2M Device:** This entity represents devices that run M2M applications, that use SCs to communicate. These can connect to the Network Domain in two ways:
 - **Direct Connectivity:** Have a direct connection to the Network Domain, performing the management procedures for himself (including registration, authentication, authorization).
 - **Gateway as a Network Proxy:** The M2M device can alternatively connect to an M2M Gateway, through the M2M Area Network. In this situation the M2M Gateway functions as a proxy for the communication with the Network Domain. One device can use multiple M2M Gateways to connect to the Network Domain.
- **M2M Area Network:** Provides connectivity between the M2M Devices and M2M Gateways.
- **M2M Gateway:** An M2M Gateway acts as a proxy for M2M devices, connecting them to the Network Domain. The device can connect directly if it has M2M capabilities, but if it is a legacy (older) device, the M2M Gateway may run an M2M applications that handle the connectivity between that device and the M2M Gateway, allowing him to also communicate with the Network Domain.

The elements present in the Network Domain are:

- **Access Network:** This entity provides access to the Core Network by the Device and Gateway Domain.
- **M2M Core:** The M2M Core is composed by two entities. The Core Network (CN) and the M2M Service Capabilities (SCs):
 - **Core Network:** This provides the access connections, IP connectivity and roaming capabilities with the M2M core.
 - **M2M Service Capabilities:** The SCs allow different applications, that have access to them, to use M2M functions through a set of open interfaces. It simplifies and optimizes application development and deployment by hiding network specificities, working as an application-level protocol.
- **M2M Applications:** These are applications that run services logic, using the SCs accessed through the open interface. This allows for interoperability of sensors and services, being a clear advantage against the verticality of proprietary M2M solutions. These are called NA's (Network Applications) throughout this dissertation.
- **Network Management Functions:** This provides the necessary functions to allow the management of the Access and Core networks. They include Provisioning, Supervision, Fault Management, etc.
- **M2M Management Functions:** This provides the functions required to manage SCs in the Network Domain. It uses a specific M2M SC to manage M2M Gateways and Devices, such as the M2M Service Bootstrap Function (MSBF) or the M2M Authentication Server (MAS).

As seen on Figure 2.3, the interface (called *reference point* in the specifications) between an M2M application in the M2M Device Domain and the M2M Service Capability (SC) in the Network and Application Domain is called *mIa*; *mId* is the interface between an M2M device or M2M gateway and the M2M SC in the Network and Application Domain; finally, the *dIa* is the name of the interface between an M2M device, or M2M gateway on behalf of a device, and the Network Domain. These interfaces define which actions each entity can execute at any given time, and the procedures in case of success and failure. The *mIm*, which is not represented in Figure 2.3, allows two SCL in different domains to be connected, thus extending the reachability of services offered over the *mId* interface.

As mentioned in the M2M Management Functions of the Network Domain, one of its responsibilities is the handling of the Bootstrap procedures. These procedures provide a way to securely connect the different entities using certificates and private keys. Every entity has a pre-provided certificate-key pair that allows them to receive a session key, which is then used to encrypt the communications with TLS (HTTP) or DTLS (CoAP). These are the only two protocols supported by ETSI's standard, and they will be discussed in sections 2.3.1.1 and 2.3.1.2, respectively. These Bootstrap procedures are very important for data to be secure in transit, as they create an encryption channel between the communicating parties, using certificates and private keys (which will be further explained in Section 3.3).

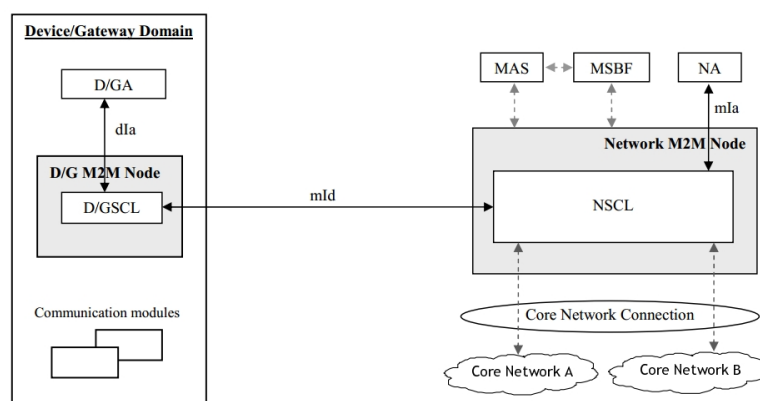


Figure 2.3: M2M Service Capabilities functional architecture framework. From [2].

The NSCL (Network Service Capabilities Layer) connects the Device and Gateway domain to the Network Applications, being a central element in the architecture, by providing the connection between services and devices compliant with the standard. The Device and Gateway also have their own standardized SCL's, respectively Device Service Capabilities Layer (DSCL) and Gateway Service Capabilities Layer (GSCL), which define the resources and actions each entity can have. The M2M Gateway and M2M NA will be further detailed in Section 3.1.

2.2.2 ITU M2M Service Layer

In February 2012 the ITU-T (Telecommunication Standardization Sector of the International Telecommunication Union) created an M2M Focus Group aiming to provide an M2M *Service Layer* focused on e-health use cases [23]. This group released the first batch of documents and studies in April 2014, having first started by executing a study on the current e-health standardization market, to identify which technical areas should be focused for standardization [24], as well as which current e-health ecosystems already rely on M2M (standardized or otherwise) [25].

The main objective of this Service Layer is to provide the solution for some areas that the ETSI standard did not cover, like the lack of specific management, support and security capabilities. Figure 2.4 shows the IoT areas this standard will cover, mentioning the interfaces (called *reference points* in the documentation) that are explained below.

Figure 2.4 shows the architectural overview of the ITU-T Service Layer, which is based on the ITU-T IoT reference model [26]. Below is a brief explanation of the ITU-T IoT reference model [26], which specifies layers and management:

Application Layer: The Application Layer contains the M2M applications, regardless of their type (DA, GA or NA).

Service Support and Application Support Layer: This layer is where this M2M standard builds upon, besides having some management and security capabilities. It is subdivided into two. The generic support capabilities are common capabilities that can be used by different M2M applications, such as data processing or storage. They can also be used to

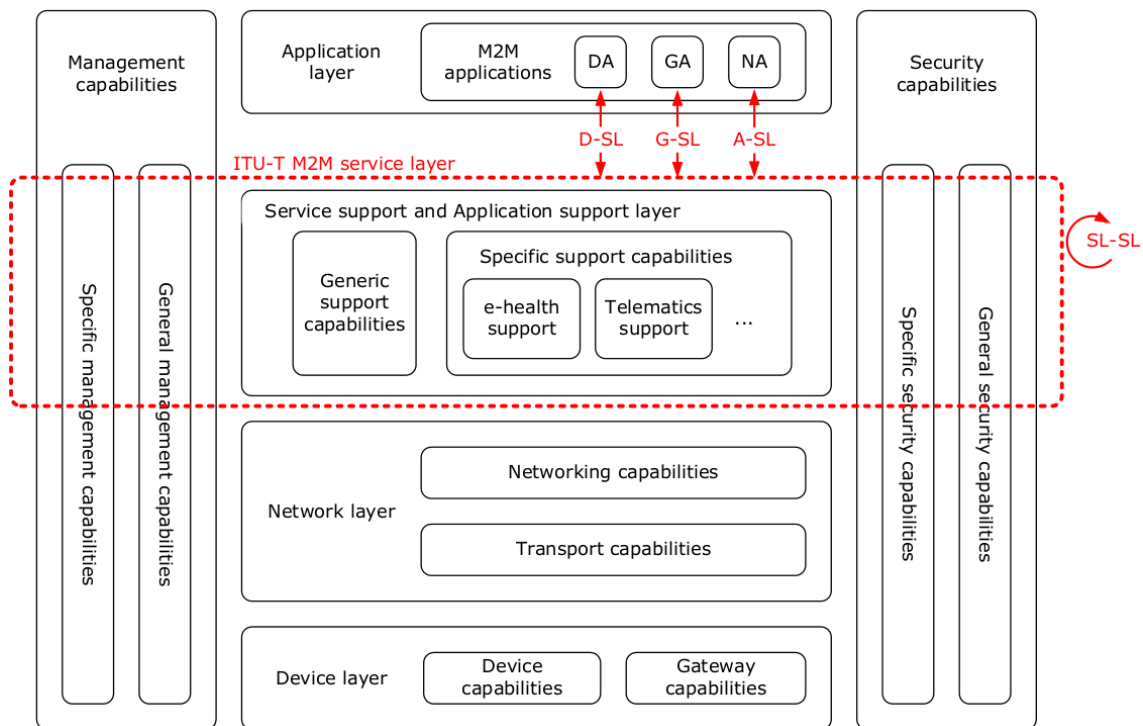


Figure 2.4: ITU-T M2M Service Layer in the IoT reference model. From [3].

build other specific support capabilities. The specific support capabilities are more specific, and serve the requirements of application groups, such as e-health or telematics support, as shown in Figure 2.4.

Network Layer: This layer is subdivided into two: Network capabilities provide control functions over network connectivity, such as access and transport resource control functions, mobility management or authentication, authorization and accounting; Transport capabilities provide connectivity for the M2M service transport and application specific data, and also M2M related management information.

Device Layer: This layer is subdivided into two kinds of capabilities: Device capabilities provide direct or indirect (through a gateway) access to the communication network, as well as ad-hoc networking and sleeping functionalities (to save energy); Gateway capabilities support multiple interfaces for the device (for example Zigbee, Bluetooth and Wi-Fi) and for the network layer (e.g. 2G,3G and LTE), as well as protocol conversion. The protocols need to be converted if the communications at device level use different layer protocols (e.g. one uses Bluetooth and the other Zigbee), or if the protocol at device layer and network layer is different (e.g. device layer uses Bluetooth and network layer uses 3G technology).

Management Capabilities: Besides the traditional management capabilities like FCAPS (fault, configuration, accounting, performance and security) this layer also provides the M2M with

generic capabilities to manage devices (remote activation and de-activation and diagnostics firmware/software upgrading), manage the local network topology, and traffic and congestion management (detecting network overflows and implementing time-critical and/or life-critical data flows).

Security Capabilities: Security capabilities are divided into generic and specific groups. The specific security capabilities depend on application specific requirements, such as mobile payments or medical data. Generic capabilities are independent from the applications, but depend on the layer:

- **Application Layer:** Authorization, authentication, application data confidentiality and integrity protection, privacy protection, security audit and anti-virus;
- **Network Layer:** Authorization, authentication, use data and signalling data confidentiality, and signalling integrity protection;
- **Device Layer:** Authentication, authorization, device integrity validation, access control, data confidentiality and integrity protection.

Similar to ETSI's mId, mIa and dIa reference points, the ITU-T Service Layer, provides its own reference points for the communication between the architectural entities.

D-SL This reference point allows a device application in a *Device* to access the ITU-T M2M *Service Layer* in the same device or in a *Gateway*. It can also connect *Legacy Device* that does not have the ITU-T M2M service layer capabilities to a *Gateway* [3].

G-SL This reference point allows a gateway application in a *Gateway* to access the ITU-T M2M *Service Layer* in the same *Gateway* [3].

A-SL This reference point allows a network application server to access the ITU-T M2M *Service Layer* in the same network application server [3].

SL-SL This reference point allows the ITU-T M2M *Service Layer* in a *Device*, *Gateway* or *Network Application Server* to access the ITU-T M2M *Service Layer* in a different *Device*, *Gateway* or *Network Application Server* [3].

These reference points connections to each other are illustrated in Figure 2.5.

The ITU-T *Service Layer* defines a set of requirements for its API, to allow the use of HTTP and CoAP to communicate between entities [27]. However, since these documents are still very recent, there is still no more information on the specificities of the API or the Architecture of the *Service Layer* itself.

2.2.3 OMA Lightweight M2M

In December 2013, the Open Mobile Alliance (OMA), released the first candidate version documentation on the Lightweight M2M (LWM2M©) standard proposal. The LWM2M is an application layer communication protocol that attempts to fulfill the client-server needs that fit the

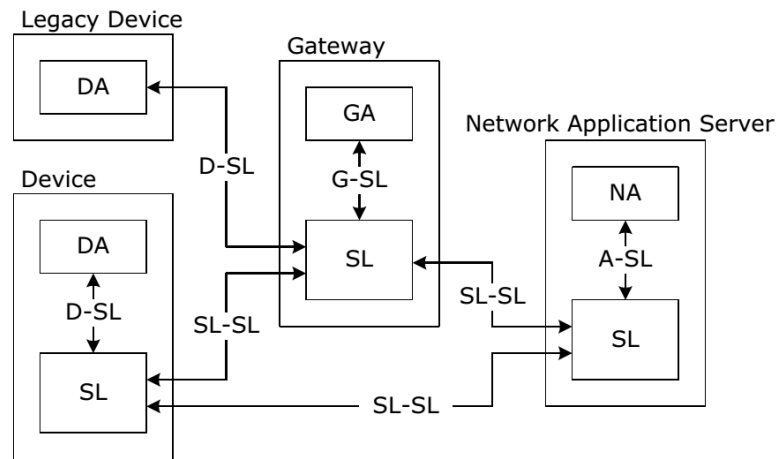


Figure 2.5: ITU-T M2M Reference points between device, gateway and network application server. From [3].

overall M2M architecture from oneM2M alliance [28]. The latter is composed by the standardization entities from around the world (including the aforementioned ETSI), as well as industry companies, and aims to gather everyone’s efforts into creating an universally accepted Service Layer that can then be incorporated in future devices and software. The LWM2M solution is referred by OMA as an *Enabler*, aiming to have its LWM2M Server incorporated into private or public data centers while the LWM2M Client built into modules or devices.

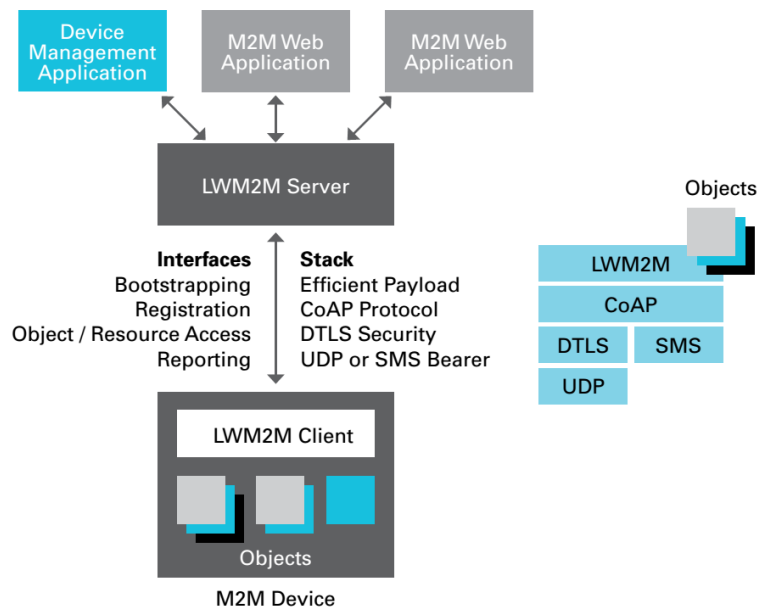


Figure 2.6: Lightweight M2M architecture and protocol stack. From [4].

The LWM2M Enabler is focused on constrained devices (low power micro-controllers and small amounts of RAM) that aims to provide a simple and low-cost remote management and service-enablement mechanism [4]. It provides a light and compact secure communication inter-

face that uses CoAP over DTLS (which will be explained in Section 2.3.1.2) or Short Message Service (SMS [29]) for communicating between the LWM2M Server and Client, like shown in Figure 2.6's protocol stack. It was also thought to be usable with non-IP based local networks such as ZigBee [30] or 6LoWPAN [31], which are built over the IEEE 802.15.4 [32] for low cost and/or low-speed communication between devices within the same Wireless Private Networks (WPAN). The implementation of the LWM2M Enabler with UDP is mandatory, whereas with SMS is optional. Also shown in Figure 2.6, the LWM2M Enabler specifies four logical interfaces between server and client, namely [33]:

Bootstrap: This interface is used to provide the LWM2M Client with the necessary information for him to *Register* with one or more LWM2M Servers. There are four bootstrap modes in the LWM2M Enabler: Factory Bootstrap, Bootstrap from Smartcard, Client initiated Bootstrap and Server initiated Bootstrap. The LWM2M Clients must support one of these, while the LWM2M Server must support at least the Client initiated and Server initiated modes.

Device Discovery and Registration: This interface specifies how the interactions to *Register*, *Update* or *De-Register* must take place between the LWM2M Client and Server. The LWM2M Server must support all interactions, while the LWM2M Client must implement the *Register* and *Update* but should also provide the *De-Register* interaction.

Device Management and Service Enablement: This interface, which must entirely be supported by both the LWM2M Client and Server, specifies how the server can access the client's objects and resources, through the use of *Create*, *Read*, *Write*, *Delete*, *Execute*, *Write Attributes*, or *Discover* operations.

Information Reporting: This interface, which must entirely be supported by both the LWM2M Client and Server, is used by the LWM2M Server to *Observe* changes in the LWM2M Client's resources, receiving notifications when new values are available. This *Observe* operation is initiated by the LWM2M Server, for a specific LWM2M Client resource or object, and ends with the *Cancel Observation* operation.

Using CoAP, the LWM2M Enabler has the ability to use RESTful communication (RESTful systems use the REST paradigm, which will be explained in Section 2.3.1) on constrained devices, providing asynchronous communication using UDP. UDP does not guarantee message delivery, but reduces the packet sizes. Some CoAP responses are supported by LWM2M, and together with its resource discovery, are encoded in a simple binary format, providing their functionalities starting with a 4 byte overhead [4].

The LWM2M Enabler defines a simple resource model where the information contained in the LWM2M Client or Server is a *Resource* that it organized into *Objects*, where every *Resource* must belong to an *Object*. The first release of the OMA LWM2M standard specifies the following set of device management objects [4]:

LWM2M Security: Handles security aspects between management servers and the LWM2M client on the device.

LWM2M Server: Defines data and functions related to the management servers.

Access Control: Defines for each of several permitted management servers the kinds of access rights they have for each data object on the client.

Device: Details resources on the M2M device related to device specific information.

Firmware: Details resources on the M2M device useful for firmware upgrades.

Location: Groups those resources that provide information about the current location of an M2M device.

Connectivity Monitoring: Groups together resources on the M2M device that assist in monitoring the status of a network connection.

Connection Statistics: Groups together resources on the M2M device that hold statistical information about an existing network connection.

With the LWM2M Enabler, OMA aims to generate market growth and cost efficiency to the industry Original Equipment Manufacturers (OEM), providing an easy way to provide services and manage devices for users and corporations, simplifying the communications while reducing the payloads. OMA feels that LWM2M may be the missing standardization for the massification of M2M devices, aiming towards an IoT future [4].

2.3 Communication Protocols

Communication protocols define how information is exchanged between nodes (any device connected to the network), where these nodes do not need special software or hardware requirements, allowing different devices, or software written in different programming languages, to be able to exchange data with each other. Protocols are thus present from the moment of information collection, by the sensors, up to its delivery to the service or application. During that transition, several protocols may be used to communicate between nodes, and mastering all of them in one system leads to difficulties in the development and deployment of end-to-end M2M/IoT solutions. The proposed M2M standards reviewed in Section 2.2 act as application layer protocols, that aim to allow easier integration between devices, services and applications, while relying on other communication protocols at their core, some of which have been developed specifically for M2M communication. This section aims to review the protocols on which the standards are built upon, and other M2M-focused protocols existing today. HTTP (explained in Section 2.3.1.1) is used in the ETSI and ITU-T standards, while CoAP (explained in Section 2.3.1.2) is used in all the reviewed M2M standards. MQTT and AMQP, explained respectively in Sections 2.3.2 and 2.3.3, were created purposefully for M2M systems even though the reviewed standards do not adopt them. Since HTTP and CoAP share the REST paradigm, Section 2.3.1 will describe its features.

2.3.1 Representational State Transfer

Representational State Transfer, or just REST [34], is the dominant approach in client-server communications, that allows for stateless communication (meaning it does not need to be synchronized nor have an open session), cacheable resources and resource operations.

RESTful protocols rely on Uniform Resource Identifiers (URI) [35] to uniquely identify names of web resources and collections (groups of resources), to be able to execute methods on them. The *CRUD* methods are then used to Create, Retrieve, Update and Delete resources.

Clients perform operations on resources stored on a server by means of request and responses exchanges. RESTful protocols implement four types of requests:

GET Retrieves the content of an existing resource or list a collection of resources;

POST Creates a new resource;

PUT Changes or updates the content of an existing resource;

DELETE Deletes/removes an existing resource or an entire collection;

This way, a client can have an abstraction layer on how the server handles the resources in its database, while having a guideline on how to perform the actions needed. Being stateless, the server does not save the state of the client after each request, since the messages carry all the important data, leaving the client to manage the information and its flow as it wishes, which removes stress from the server.

2.3.1.1 HTTP

Hypertext Transfer Protocol, or HTTP [36, 37], is one of the most used protocols in today's Internet. Websites are usually transmitted over HTTP, or its secure version HTTPS, which adds TLS (Transport Layer Security [38]) to encapsulate the HTTP packets.

HTTP started to be used in the early 1990's World-Wide Web global information initiative only being proposed as standard in 1997 [39], receiving that status in 1999 with version 1.1 [36]. It can use TCP [40] or UDP [41] as Transport Layer protocols, but TCP is much more common because of the reliability it provides. HTTP can be used in a variety of communications, including M2M, despite having a big header overhead, on top of an already large TCP packet size. A typical HTTP packet, is illustrated in Figure 2.7. This figure does not account for the TCP connection and setup traffic, as well as re-transmissions, and only illustrates the size of a HTTP message, on the application layer.

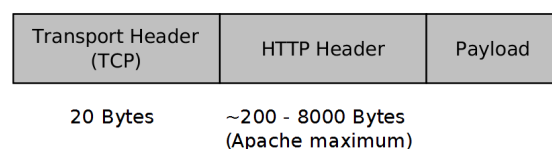


Figure 2.7: HTTP packet

2.3.1.2 CoAP

The Constrained Application Protocol (CoAP) [42] is a lightweight protocol designed for constrained devices (with low memory, processing power, etc) and constrained networks (e.g low power, lossy), and specially fulfilling M2M requirements. Recently, the protocol's proposal for standardization was approved, leading to RFC 7252 [42].

Its simple interface and applicability was demonstrated in the 2013 ETSI plug-tests where it was shown that in an event with eight companies with several different CoAP implementations of clients and servers, the interoperability rate was 94.1% [43].

CoAP derives from Representational State Transfer (REST), explained in Section 2.3.1, adapted for the use in constrained networks and nodes in M2M applications [5]. CoAP is also easily translated to HTTP for integration with the web while accomplishing specialized requirements, such as multicast support, built-in resource discovery, block-wise transfer, observation, and simplicity for constrained environments [44]. This allows for compatibility with existing systems.

Unlike HTTP, where to discover a resource structure, the client has to poll the server, CoAP has an asynchronous approach to support pushing information from servers to clients: observation (see Figure 2.8). In constrained environments, polling wastes too many resources, and CoAP addresses this with a special GET request, where a client can indicate its interest in further updates from a resource by specifying the *OBSERVE* option. If the server accepts this option, the client becomes an observer of this resource and receives an asynchronous notification message each time it changes. Every notification is identical in structure to the response to the initial GET request [44]. This model is an attempt to achieve stateless pub-sub communication, ideal for devices with fewer resources.

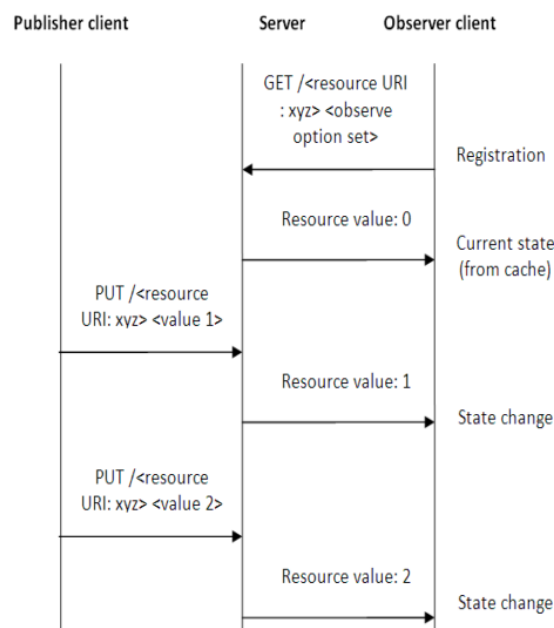


Figure 2.8: CoAP resource-observe. From [5].

Furthermore, in order to implement CoAP in constrained small devices (memory available, computational and power consumption restrictions), the transport protocol is User Datagram Protocol (UDP) [45] and the protocol overhead in the header fields can be reduced to 4 bytes. Reliability can be implemented by an optional stop-and-wait protocol, and security by the use of Internet Protocol Security (IPsec) [46] or Datagram Transport Layer Security (DTLS) [47]. CoAP also supports TCP [48] connections, which add reliability of delivery, but increases the overhead of each message, which is not ideal for constrained situations.

The CoAP packet is illustrated in Figure 2.9.

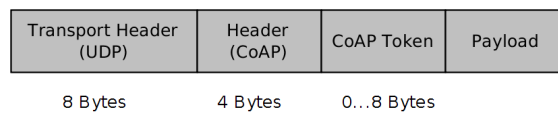


Figure 2.9: CoAP packet

2.3.2 MQTT

MQ Telemetry Transport (MQTT) [18] is a lightweight broker-based publish-subscribe messaging protocol designed to be open, simple, lightweight and easy to implement. The message broker manages and routes the messages to subscribers depending on the message topic. These characteristics make it ideal for use in constrained environments just like a smartphone. It is a protocol developed by IBM to address the issue of reliable M2M communications [18].

MQTT is connection oriented, used over the TCP/IP protocol and features 3 quality of service (QoS) levels for assuring delivery (no retransmission, re-transmit once and re-transmit until received). Being based in pub-sub, it can save constrained devices to have to answer to requests, saving messages, and thus processing power and battery. In terms of security, MQTT provides the ability of using a username and password identification, built into the protocol. For point-to-point encryption, SSL/TLS can be used, but it is independent of the protocol.

The underlying TCP connection causes MQTT to have a bigger overhead than other protocols such as CoAP, which is evidenced in the protocol comparison tests done in [5]. The CoAP packet is illustrated in Figure 2.10.

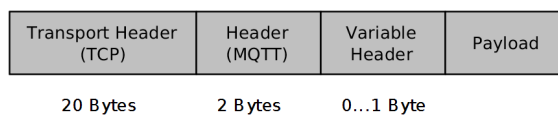


Figure 2.10: MQTT packet

2.3.3 AMQP

The Advances Message Queue Protocol (AMQP) [49] is an open standard application layer asynchronous protocol for message oriented middleware. It uses the pub-sub model defined earlier in Section 2.1

AMQP uses brokers (servers) to receive messages from publishers, and route them to consumers. Figure 2.11 gives a high level perspective of the protocol. [6] has a simple explanation of the functionality of the protocol: "Messages are published to exchanges, which are often compared to post offices or mailboxes. Exchanges then distribute message copies to queues using rules called bindings. Then AMQP brokers either deliver messages to consumers subscribed to queues, or consumers fetch/pull messages from queues on demand".

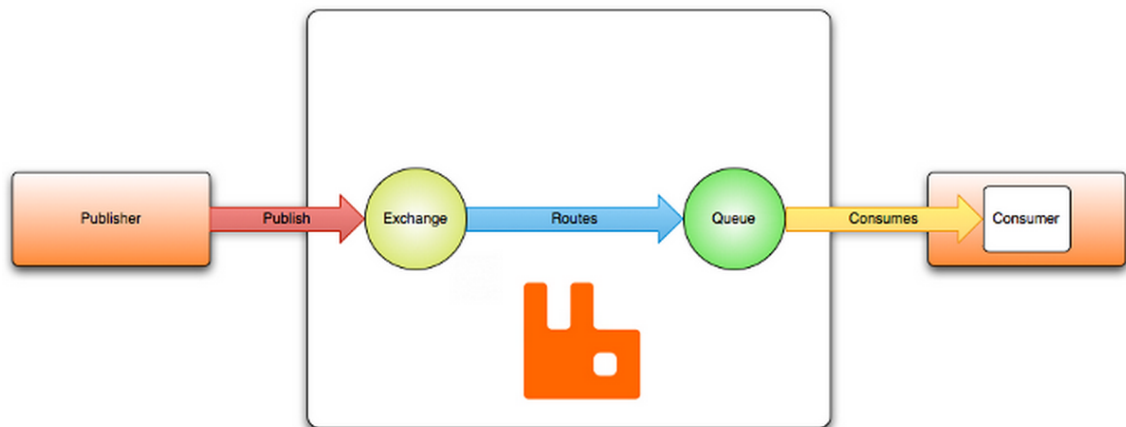


Figure 2.11: AMQP functionality overview. From [6]

In terms of security, AMQP implements Simple Authentication and Security Layer (SASL) [50] and TLS [51]. Figure 2.10 illustrates the AMQP packet size.

Transport Header (TCP)	Header (AMQP)	Extended Header	Payload
20 Bytes	2 Bytes	0...8 Bytes	

Figure 2.12: AMQP packet

2.4 eHealth

This dissertation uses a mobile eHealth (or healthcare) scenario to drive the development of M2M solutions. But what exactly is eHealth? The World Health Organization (OMS) defines it as "the transfer of health resources and healthcare by electronic means" [52], and is grouped into three areas: using telecommunications to exchange health information between health professionals and health consumers; improve public health services by online commerce (production, distribution, marketing, sale or delivery of goods and services by electronic means, i.e. conducting business over the Internet) and education and training health professionals; using online commerce and businesses practices in health systems management [52].

This dissertation focuses on the first area, of exchanging of healthcare information using telecommunications, on an M2M scenario. A review of mobile enabled healthcare applications

is presented in Section 2.4.1, followed by some communication standards for transmitting eHealth data, in Section 2.4.2.

2.4.1 Application Revision

This section will review some healthcare studies that use mobile eHealth applications data to enable real-time medical advices to improve/keep the users health and wellness, while also enabling medical research over the gathered data. These reviews will provide a baseline for the communication, for comparison with M2M approaches.

Section 2.4.1.1 will review a wellness diary, aimed to educate the users on weight management, while Section 2.4.1.2 will review an application that provides remote cardiac monitoring of patients. Then, Section 2.4.1.3 will review an application that monitors caloric balance in real time.

2.4.1.1 Mobile Wellness Diary

Developed countries have numerous health issues, due to their sedentary lifestyle and demanding jobs, the most important of which are obesity, stress and sleep disorders[53]. All these are very serious threats, since obesity is estimated to account for 2%-8% of healthcare in Europe, while stress and sleep disorders have been associated with cardiovascular diseases and type II diabetes. Since current healthcare systems do not have the means or resources to prevent or manage these health risks, it was necessary to individually address each case. The mobile wellness diary was created for Symbian, Nokia's OS, and it served as the base for a study of the habits of 29 subjects [53]. The application created received input of users on their input of calories ingested and regular weight measurements. It was also associated with a study that gathered information on the heart-rate (through a connected blood pressure meter) and a pedometer (through a connected device). To transfer the sensors information to the researchers, it first had to be transferred onto a computer, while the data generated from user input was sent via SMS or e-mail.

The data collected by this study were shown to be positive in educating the subjects' on their less healthy habits, instilling them to take better care of their health and wellness. Results during the study also shown a decline of weight during the tests, showing that the subjects were receptive to the advices and suggestions of the application [53].

2.4.1.2 Cardiac Telemonitoring of Patients

Cardiovascular disease is the leading cause of global mortality in both developed and developing countries, mostly due to heart attack and hypertension, while cardiac arrhythmia is also thought to be responsible for most of the sudden cardiac deaths that occur [54]. Cardiac rehabilitation (CR) can help lower the risk of future heart problems, and is aimed at patients recovering from heart attacks or heart surgery. It is based on the education and counseling of patients on how to increase their physical fitness, in order to reduce cardiac symptoms and future complications. For this study, the CR involved outdoor walking sessions with the duration of approximately 6

minutes. A system composed by a portable electrocardiogram (ECG) and a GPS receiver, connected to a smartphone via Bluetooth recorded the walking sessions, and streamed the data to the secure server through 2G/3G connectivity. This system thus enabled real-time visualization of the patient's data in order to monitor his progress [55]. The aim was to diminish costs of CR that are usually only executed in hospitals, with all the transport and accommodation costs that accompany it.

The study, published in 2011, showed that a real-time system with these characteristics is technically feasible and that it can complement the hospital-based programs. It also showed that during the 6 minute walks, despite not having readily available doctors as would happen in the hospital, the GPS data allied with voice contact through the smartphone could offer some level of security for the patient's medical care [55].

2.4.1.3 Monitoring Real Time Caloric Balance

Given the obesity problem in developed countries (e.g. 65% of US adults are either overweight or obese [56]), self monitoring is an important skill for success in weight management. To provide a real-time application that allowed users to monitor their caloric balance the Patient-Centered Assessment and Counseling Mobile Energy Balance (PmEB) was created. This application relies solely on the user input of the calories consumed and time spent doing physical activity. The PmEB consists of a smartphone client, a server application and a web interface that allows the user to personalize the mobile client. This client personalization is needed since calculating the overall calories consumed depends on the person's body (height, metabolic rate, etc.), and for identifying probable work-out scenarios. Then, once a day, the server application signals the mobile client to upload the registered data, in order to store it. Locally, besides allowing the addition of physical activity and food intake, the client allows the consultation of the caloric balance, and set a goal to fulfill. A more comprehensive web interface on the server is available, allowing the user to consult the caloric balance on specific time intervals (e.g. for the past n days) [56].

This study proceeded to develop a detailed discussion on the research and development leading to the creation of the PmEB, and concluded that using mobile applications for real-time monitoring of caloric balance had a good adoption from the users, and comparing with paper questionnaires that yielded the same balance. Users felt that usability of the PmEB was better, specially when used over long periods of time.

2.4.2 Communication Technologies

As there are multiple manufacturer's for medical devices, there needed to be some standardization on how they should communicate their data, for security reasons and for that data to be available for the connected applications and services. Section 2.4.2.1 presents the ISO/IEEE 11073 family of standards, while the

2.4.2.1 ISO/IEEE 11073

The ISO/IEEE 11073 is the family of standards for medical devices that was originally called IEEE 1073, or just X73 [57]. It arose in 1982 with the need for easy to use (plug-and-play) medical grade equipment for operating rooms or bedside monitoring in intensive care units (ICU), aiming for real-time and efficient exchange of data [58]. In the past decade, the IEEE 11073 has focused on developing Personal Health Devices (PHD) standards, standardizing functions for each of the OSI layers [59].

Figure 2.13 shows the IEEE 11073 Framework as of 2012, where several device specializations can be represented. The main goal of all these standardizations is to facilitate the development of equipment to monitor people’s well-being inside or outside of hospitals and provide interoperability on medical grade equipments. These standards are not available for public scrutiny, and consequently details on them are unavailable. On Android OS version 4.0 (API 14), a Bluetooth Health API was introduced, facilitating the integration with devices following the IEEE 11073 specifications [60].

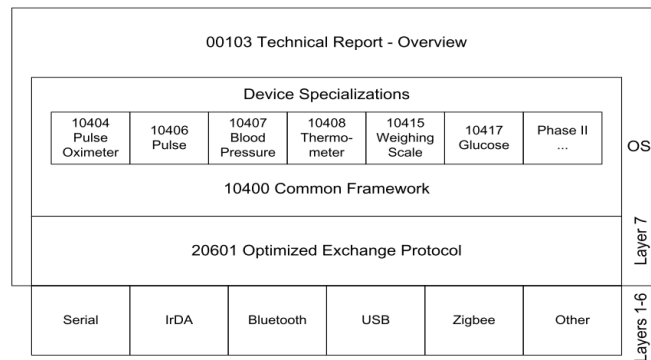


Figure 2.13: IEEE 11073 Framework. From [7].

2.4.2.2 Continua Alliance

The Continua Health Alliance [61] is a consortium that created a profile of standards built on top of the IEEE 11073 family. Its goal is to provide the application layers with semantic interoperability, to further allow communication between devices and services. Figure 2.14 shows how the protocol builds around IEEE 11073 to allow for greater interoperability with a wider range of devices [7].

Currently the Continua Alliance has a set of guidelines for manufacturers to follow, to provide them with the proper certification, ensuring the compatibility of all Continua certified devices. The details of these guidelines and of the resulting communication protocol are proprietary, and thus not available for deeper discussion in this dissertation.

Telehealth		Telecare		Home controls
IEEE 11073				
USB	BT	BT-LE	ZigBee healthcare profile	ZigBee home automation profile
	2.4GHz			868MHz

Figure 2.14: Continua Alliance profile of standards. From [7].

2.5 Discussion

Although the all the M2M standards discussed in this chapter could fit the needs of this project, ETSI's was chosen given the partnership with *PTIN*, since they had already implemented the NSCL to communicate with. This also drove the protocol choice of HTTP and CoAP, as ETSI does not support any other protocols.

The three mobile eHealth applications reviewed in this chapter give only a small example of the possibilities that they can yield with their users. However, as shown, they all presented their own specific communication servers and specificities, which severely limits the re-usability of the data gathered by them. M2M communications allied with those applications could ease the usability but mostly their interoperability, by allowing the data collected to be shared between devices, services and applications. This could increase the amount of applications the user has access to, that could greatly enhance the health insights he has access to. If the mobile eHealth applications use sensors that implement the IEEE11073 or the Continua Health Alliance guidelines, using M2M communications would allow for the easier integration of further medical grade sensors (compliant with the same standards) on the healthcare system.

Chapter 3

Aspects of ETSI M2M Standard

This Chapter will take a deeper look into the ETSI M2M standard's specifications, in Section 3.1. Section 3.2 will detail Marshalling, which is an important mechanism in M2M communications, followed by an overview of the security mechanisms required by the standard, in Section 3.3.

3.1 ETSI M2M Standard

The ETSI standard resource structure will be further detailed in Section 3.1.1, followed by the explanation of the differences between the Gateway and a Network Application in Sections 3.1.2 and 3.1.3. Finally, in Section 3.1.4 some ETSI M2M implementations found will be presented.

3.1.1 Resource Structure

The ETSI M2M standard defines a variety of resources and how they connect to each other, in order to allow several M2M scenarios to take place. Figure 3.1 does not contain all the resources or attributes standardized, but it illustrates the connections between them. The *Scls* resource has a collection of *Scl* resources, where each contain an *Applications* resource, with the same structure as its homonym connected to *SclBase* in the figure. This is recursive in the resource structure, and shown in *Applications*, as there is a collection of *Application* resources, that in turn contains *Containers* resources. So, this versatile resource structure means that with every new device, gateway, or network application to be registered (locally or remotely, through the NSCL), there may be a large number of resources created, remembering that Figure 3.1 shows only a small fraction of the totality of the resources standardized.

In a smart-home example, such as the illustrated by Figure 3.2, the boiler and the washing machine would register their DSCL's on the M2M Gateway's GSCL's *Scls* resource, using the dIa interface. The M2M Gateway could also connect to some legacy devices that did not support the dIa interface, in which case a Gateway Application (GA) would create their resources on the GSCL's *Scls* resource also using the dIa interface. These sensors would in turn be registered by the M2M Gateway on the NSCL, making them reachable by M2M Network Applications. Figure 3.2

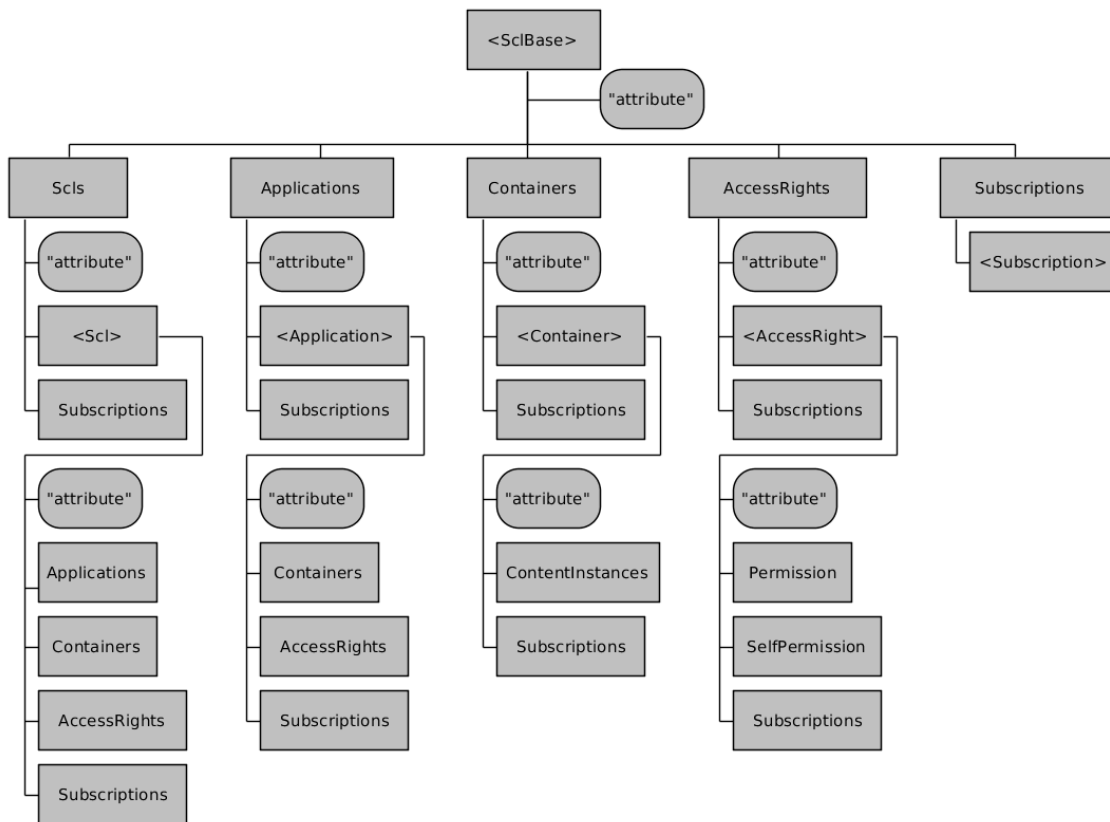


Figure 3.1: Illustration of the ETSI M2M resource structure

suggests an M2M NA that would control the heating and washing cycles, and another from the devices manufacturer’s (non-legacy), that could trigger software updates on their devices. Being in the same SCL than the NSCL, the M2M NA’s would register themselves in the *Applications* resource of the NSCL.

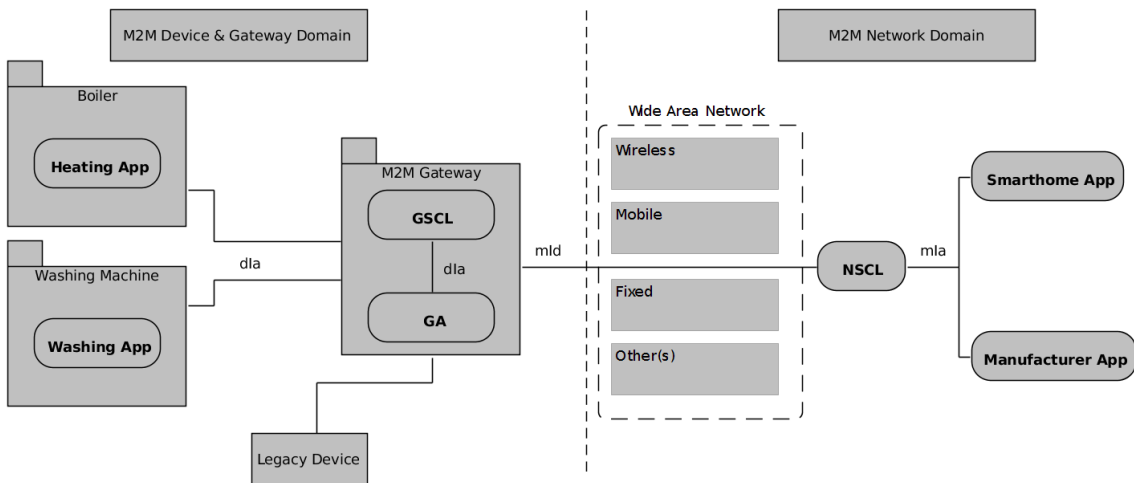


Figure 3.2: ETSI M2M smart-home example

3.1.2 M2M Gateway

The M2M Gateway entity communicates through the *dIa* reference point with network capable sensors, and communicates with the NSCL using the *mId* reference point. The latter defines the procedures to connect the M2M Device and Gateway Domain to the M2M Application Domain, as shown earlier in Figure 2.3 [2, 62]. The *mId* interface offers the following functions for the M2M GW to interact with the NSCL [2]:

- Registration of a SCL (DSCL or GSCL) to the NSCL.
- Request to Read/Write information in the NSCL, GSCL or DSCL. This requires proper authorization.
- Request device management actions (e.g. software upgrade, configuration management).
- Subscribe and get notified for those subscriptions and for specific events.

The M2M Gateway entity has its own standardized M2M SCL, the GSCL, which details how their SC's should be created and managed. However, given the limited time of this dissertation, a complete implementation of the GSCL resource structure and the *mId* and *dIa* interfaces was not feasible. So, the resources relevant for this dissertation's work were identified and are represented in Figure 3.3. Note that the resources connected to the *SclBase* (*Scls*, *Applications*, *Containers*) were abstracted as collections, since for the purpose of this dissertation they act as arrays of their child resources (respectively *Scl*, *Application* and *Container* resources). How this simplified resource structure is mapped into actual devices and concepts will be discussed in Section 4.2.2.3.

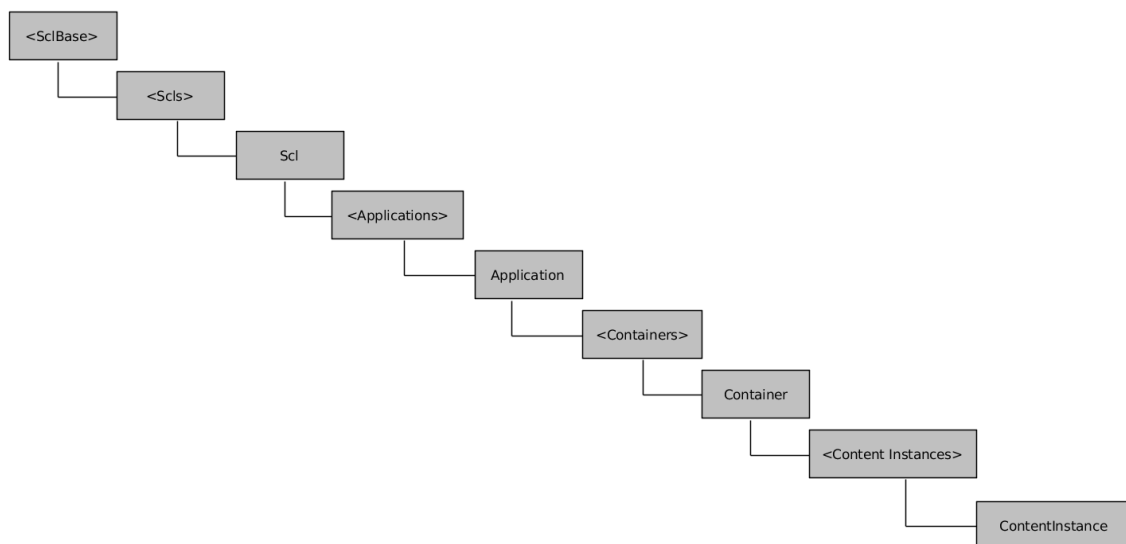


Figure 3.3: Simplified M2M GW resource structure.

For the Gateway to fully interact with the NSCL using this resource structure, it has to perform a series of actions defined in the *mId* reference point:

1. Register the *scl* resource (with a specific '*sclId*') under the */sclBase/scls* target on the NSCL host.

2. Receive the OK response for that registration.
3. Receive a subscription on the *scl* resource just created.
4. Register the *application* resource (one or more) under */sclBase/scls/sclId/applications*.
5. Receive the OK response for each *application* resource registered.
6. Receive a subscription on the (one or more) *application* resource registered.
7. Register a certain *container* resource belonging to the *application* '*appId*' under */sclBase/scls/sclId/applications/appId/containers*.
8. Receive the OK response for that registration.
9. Receive a subscription on the *container* resource just registered.
10. Register a *contentInstance* resource belonging to the *container* with '*containerId*' under */sclBase/scls/sclId/applications/appId/containers/containerId/contentInstances*.
11. Receive the OK response for that registration.

Being based in the publish-subscribe paradigm, subscriptions are needed to trigger the following registration. These actions illustrate the path taken to create the registration for one specific resource. The last resource (*contentInstance*) carries the data regarding the registered resource (for example temperature or heart-rate measurements) encoded in Base64 [63].

3.1.3 M2M Network Application

The M2M NA entity communicates through the m1a reference point with the NSCL (also shown in Figure 2.3), and through it can also communicate with D/GSCL, if needed. A M2M NA can represent a service or application, that can produce, consume or treat data within the M2M domain, adding functionalities to it [2, 62]. It is also possible for a NA to have an external interface to make the M2M data available outside of the scope, but this external interface is not mandatory or standardized.

The m1a reference point offers the following functions for the M2M NA to interact with the NSCL [2]:

- Registration of NA on the NSCL.
- Request to Read/Write information in the NSCL, GSCL or DSCL. This requires proper authorization.
- Request device management actions (e.g. software upgrade, configuration management).
- Subscribe and get notified for those subscriptions and for specific events.

M2M NA's belong to the Network Domain, as shown in Figure 3.2, and how its SC's are organized are detailed in the NSCL. As with the M2M GW, Figure 3.4 illustrates the simplified version of the M2M NA resource structure, whose map into relevant concepts for this dissertation's project

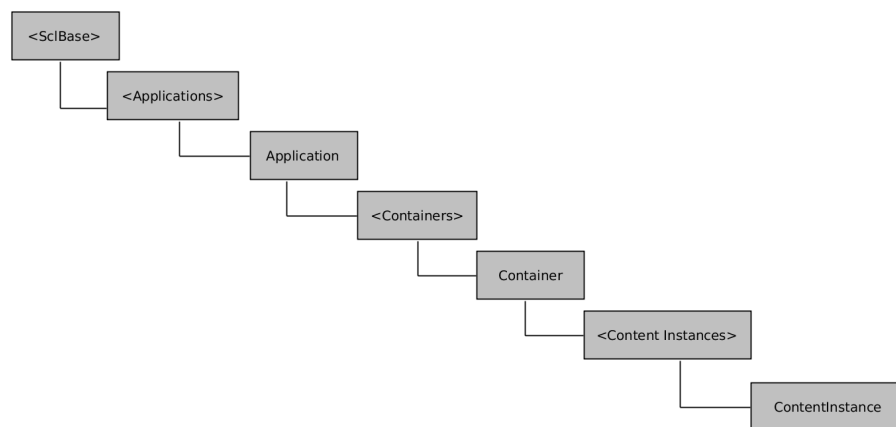


Figure 3.4: Simplified M2M NA resource structure.

will be discussed in Section 4.2.2.3. The collections in this simplification are also abstractions, to facilitate the understanding of this resource structure in the coming Chapters.

For the M2M NA to fully interact with the NSCL using this resource structure, it has to perform a series of actions:

1. Register the *Application* resource (with a specific '*appId*') under the */sclBase/applications* target on the NSCL host.
2. Receive the OK response for that registration.
3. Register a certain *Container* resource under */sclBase/applications/[appId]/containers*.
4. Receive the OK response for that registration.
5. Register a *ContentInstance* resource belonging to the *Container* with '*containerId*' under */sclBase/applications/[appId]/containers/'containerId'/contentInstances*.
6. Receive the OK response for that registration.

3.1.4 ETSI M2M Implementations

Being an open standard, there are a few ETSI M2M open-source implementations available online, each with their natural limitations. This section will review the found implementations.

OM2M

The Open-source M2M (OM2M™) [64] is an attempt from the Eclipse foundation to reach a common ground on how the ETSI standard is interpreted, and is still in proposal stage, covering limited functionalities. The idea behind the project is to implement the ETSI standard and allow its functionalities to be accessed via an API, allowing for easier implementation and deployment of solutions that integrate this standard. There is currently not much information on this project, but it implements the standard's Service Capabilities Layer, as well as a Gateway and a Device according to the ETSI architecture. This project aims to become the basis for future implementation of the OneM2M standard.

Cocoon

The Cocoon™ [65] solution, provides an implementation of the ETSI M2M standard. It includes the tools to install the ETSI M2M protocol on existing Gateways, or the ability to purchase equipment with the protocol already installed. The goal is to allow the manufacturer's to easily include this protocol on their hardware, and to provide a graphical user interface that developers can use to interact with the Gateways and devices, learning the necessary tools to build M2M NA's. Actility, the company behind Cocoon, has a marketplace where these M2M NA applications are available for purchase, aiming to get revenue on the platform they provide.

3.2 Marshalling

Marshalling, or serializing, is the process of encapsulating data for storage or transport, so that it can be processed by different systems. Marshalling techniques allow for interoperability of services by using standards readable by both machines and humans [66]. The most common types of marshalling languages are XML and JSON, which will be explained in the sections 3.2.1 and 3.2.2 respectively, and the ETSI M2M standard supports both of them.

Some high level programming languages like Java, have their own serialization procedures allowing the storage or transport of memory into binary objects. These techniques are essential in M2M scenarios in order to be able to transparently exchange information between entities, assuring that the destination is able to decode the original message, improving the interoperability of the system.

3.2.1 XML

The eXtensible Markup Language, commonly known as XML, was first released in a World Wide Web Consortium (W3C) draft in 1996 [67] as a derivation from the Standard Generalized Markup Language (SGML), which had been standardized in the International Organization for Standardization (ISO) in ISO 8879 [68]. XML 1.0 was originally submitted for standardization on an RFC (Request for Comments) in 1998 [69] and has since become one of the most widely used markup languages. It is, for instance being used in Android OS for detailing the layout specifications and the permissions needed to run the application.

3.2.2 JSON

JavaScript Object Notation (JSON), was originally specified in [70] as a less verbose alternative to XML, featuring pairs of attribute-value, and it is currently in use in the many server-application communication systems. It is used primarily to transmit data between a server and web application, as an alternative to XML.

3.3 Data Security Mechanisms

Data security is a very important issue, for most foreseen M2M applications and specially for sensitive data like healthcare. One of the ways to provide secure communications is data encryption, and it has two major concepts:

Symmetric Encryption This is the name given to communication where the sender and the receiver use the same key and similar processes to encrypt and decrypt data. If the keys were always unique and known only to the two intervenients, this would be the way to always communicate sensitive information. However, given the need to share keys, asymmetric encryption was developed. The most commonly used symmetric encryption ciphers are the 3DES (more cryptographically secure than the original Data Encryption Standard) and AES (Advanced Encryption Standard) [71]. Data encrypted with AES or 3DES is nearly impossible to decode without special information.

Asymmetric Encryption This kind of encryption is based on number theory and the computational difficulty of today's processing units to invert discrete logarithms [71]. This provides a way to have two keys that complement each other, where one public and available for everyone, and one private know only to the specific machine, and its strengths rely on the fact that is easy to generate the public key from the private key, but infeasible to revert it without special information (not impossible, but extremely difficult). The most common asymmetric encryption algorithm is RSA (Ron Rivest, Adi Shamir and Leonard Adleman [72]). Asymmetric encryption's major downside is processing time, since it is computationally much more demanding than symmetric encryption in order to yield similar results. Because of this, it is mostly not used to communicate, but to share symmetric encryption keys.

Besides allowing for the exchange of symmetric keys, there are two functionalities of asymmetric encryption. If the destination's public key is used to encrypt the information, since only its private key can decode the message, the data is sent securely. But it can also act as an authentication agent, if the source private key is used, since everyone with access to its public key can decode the message to verify its authenticity.

Symmetric and asymmetric encryption often work together to create a secure system, where the keys for the symmetric encryption are shared with asymmetric encryption. TLS and DTLS use this method to agree on a symmetric session key to encrypt the subsequent communications. After the public keys are created, the need to distribute them led to the creation of certificates, and Certificate Authorities (CA). Certificates consist of a storage for public keys, amongst other information about the key's owner, which is then signed by a CA, by hashing the owner's information with its (CA's) private key. CA's are used to store and provide authenticity to the stored certificates. This process allows anyone with a certificate to know the owner's information (amongst which is its public key), while also being able to verify its authenticity by checking that after using the CA's public key to decrypt the information, the hash value coincides with the one present in the certificate [71].

X.509 Certificates specifies the certificate format, and how the public keys and other information should be stored, as well as protocols on how to sign and check the signature of the certificate. Public Key Infrastructures (PKI) are systems that use CA's to create, manage, store and revoke certificates in a network, making use of asymmetric encryption [71]. The Public Key Infrastructure X.509 (or PKIX [73]) defines a generic model that is suitable for deploying a certificate-based architecture in web services. ETSI's M2M standard makes use of X.509 Certificates.

Chapter 4

ETSI Compliant Mobile M2M System

This chapter presents the problems this dissertation's work proposes to solve, in Section 4.1, followed by the detailed approach taken, in Section 4.2. Finally, the evaluation metrics of this work in Section 4.3.

4.1 Problem

The ETSI M2M standard has been in development since 2011, and some implementations have already naturally emerged, aiming to ease the development of ETSI standardized solutions, as shown in Section 3.1.4. These, even though some examples are given, still leave the standard's resources unmapped, difficulting the development approach.

Mobile scenarios, relying on a mobile M2M Gateway allow for a new range of applications, but they needed to be thoroughly thought to use the ETSI M2M standard's architecture. A way to manage communication with sensors and use them following the ETSI M2M architecture is the challenge this dissertation aims to solve.

4.2 Approach

This section explains the system concept composed by a mobile M2M GW and a mobile M2M NA in an ETSI compliant approach. From now on, when referring to mobile M2M GW or NA, this dissertation is not referring to the ETSI entities, but rather the mobile Android OS application that implement those entities.

Specifying how we wanted to map the ETSI standard into a functional mobile M2M GW and NA system prototype, needed a specific focus. Section 4.2.1 will detail the healthcare use case thought to be the implementation target of the standard. Then, the system design will be detailed in Section 4.2.2, with information on the requirements, architecture and mapping needed to use the standard in a mobile scenario. Section 4.2.3 will detail the designed local interface, aimed to save resources, followed by the Android applications architecture in Section 4.2.4.

4.2.1 Healthcare Use Case

Imagine a user, let us call him John, that is concerned about his health. John goes into a pharmacy, into the corner where the check-up equipment is and wants to weigh himself, or check his blood pressure. He picks up his smartphone and connects to the sensors wirelessly, executing the desired measurement, getting the result right in his smartphone. This measurement is then stored online, and enables a service that provides him access to his medical records, in order to keep track of his health evolution. John may also want to go to the park for a run, where he does not have Wi-Fi connection or interest in spending his 3G/4G *plafond*, while still keeping track of his heart-rate measurements. John should be able to keep track of his running locally, storing the results to add to his medical history later.

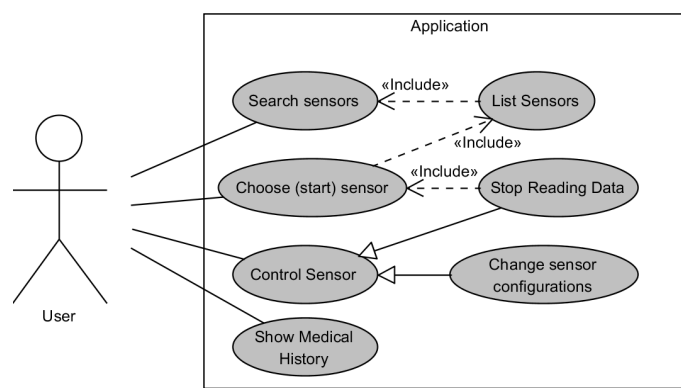


Figure 4.1: UML 2.0 Use Case Diagram for the study scenario

Given this scenario, the user actions presented in Figure 4.1 were defined. The user will be able to use the phone's Bluetooth capabilities to search supported sensors that have been paired with the smartphone. Found sensors will be listed and choosing a sensor will start its measurements.

The sensors measurements will be shown in the application and the user will be able to stop them at any time. If the measurements are running, the user will also be able to change the three sensor configurations:

1. Maximum time the sensor must execute a new measurement (for sensors that support it);
2. Maximum time between the sensor measurements are delivered to the web-server;
3. Maximum size (in bytes) to be stored in the smartphone's memory.

When it is time to send the data, either by reaching the maximum time, or because the maximum size was reached, it is then forwarded to a web-server, that can provide a service with his medical records, which can be consulted by himself or by his physician.

In order to save bandwidth in the jogging situation, the application should also allow the user to continue to use the sensors offline, saving the measurements to add to the medical records when he regains connectivity. Working offline, the application would also end up saving battery, since 3G and Wi-Fi are some of the components that drain more battery [9].

4.2.2 System Design

This section will detail the system that was designed with the healthcare use case in mind. Section 4.2.2.1 will present the functional and non-functional requirements of this system, followed by its high-level architecture in Section 4.2.2.2. Then, Section 4.2.2.3 will detail the map that was made to adapt the use case to comply with the ETSI M2M standard, paving the way for the sensor procurement, in Section 4.2.2.4.

4.2.2.1 Requirements

To implement the use case the following **functional** requirements are needed, in descending order of priority:

1. Allow the following user actions:
 - Search sensors;
 - Select sensor;
 - Start sensor;
 - Stop sensor;
 - Change sensor configurations;
 - Show medical history.
2. Support healthcare external sensors:
 - Heart-rate monitor;
 - Sphygmomanometer (Blood pressure monitor);
 - Scale.

The **non-functional** requirements that the system has to fulfill are, also in descending order of priority:

1. Be compliant with the ETSI M2M standard;
2. Have a modular design, aiming for easy integration of new features, sensors or protocols;
3. Support offline storages, so that data is not lost when connection is;
4. Support offline connection to sensors, not needing Internet connection to make measurements.
5. Support multiple protocols:
 - HTTP;
 - CoAP;
 - MQTT.

4.2.2.2 High-Level Architecture

This section explains the design of this dissertation’s system mapping the use cases into the ETSI M2M architecture. The high-level system architecture is shown in Figure 4.2. Here, the mobile M2M GW will take advantage of the smartphone’s connectivity abilities, connecting to sensors via Bluetooth while communicating with the NSCL through the active network connectivity (Wi-Fi or 3G) using the mId interface. The mobile M2M NA will also communicate with the NSCL through the active network connectivity, but using the mIa interface. It will also possess a graphical user interface (GUI) to allow the user to easily control the connection to supported healthcare devices and see information relevant to him.

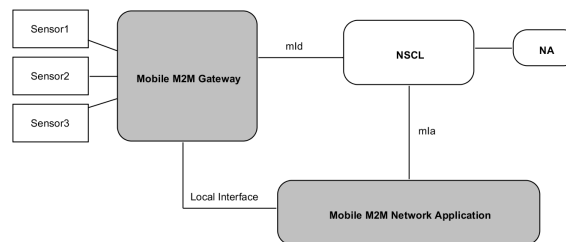


Figure 4.2: Architectural overview of integrated solution

The mobile M2M GW and mobile M2M NA work as an M2M system to fulfill the requirements defined in the previous section, where the mobile M2M GW handles the connectivity to the sensors, and how their data is processed, and the mobile M2M NA receives the user’s input on the action to be taken. The connection to the healthcare service that stores the user’s medical records, can be done with aid of an M2M NA, which does not need to be mobile, that can treat the information and re-introduce it to the NSCL under different resources, or even forward them to an external service. Being a separate project and mapping altogether, and given the limited time to execute this dissertation, this implementation fell out of the scope of this dissertation.

4.2.2.3 ETSI Resource Structure Map

This section will present how the ETSI resource tree was mapped into actual devices and concepts, aiming to fulfill the requirements of this project:

M2M GW Resource Map:

Given the resource tree structure specified in Section 3.1.2, it made sense that the lower on the tree, the closest the mapping would be of a physical device. Given the nature of the *ContentInstance* resource, which has an attribute that stores data encoded in Base 64, its parent resource *Container* was mapped as a specific sensor (such as a Bluetooth heart-rate monitor).

The *Application* resource had to somehow generalize the sensors, and as such it made sense that it grouped the different kinds of sensors the mobile M2M GW can connect to. Relating to the above example of a *Container* as a heart rate sensor, the application it would be associated with would be referring to health care monitoring. Other sensors of similar purposes would also be added here.

Because of all the different kinds of sensors possible, all handled by the same mobile device, the *ScI* resource represents the smartphone, being the "father" of all the other resources. The mobile M2M GW mapping is illustrated in Figure 4.3.

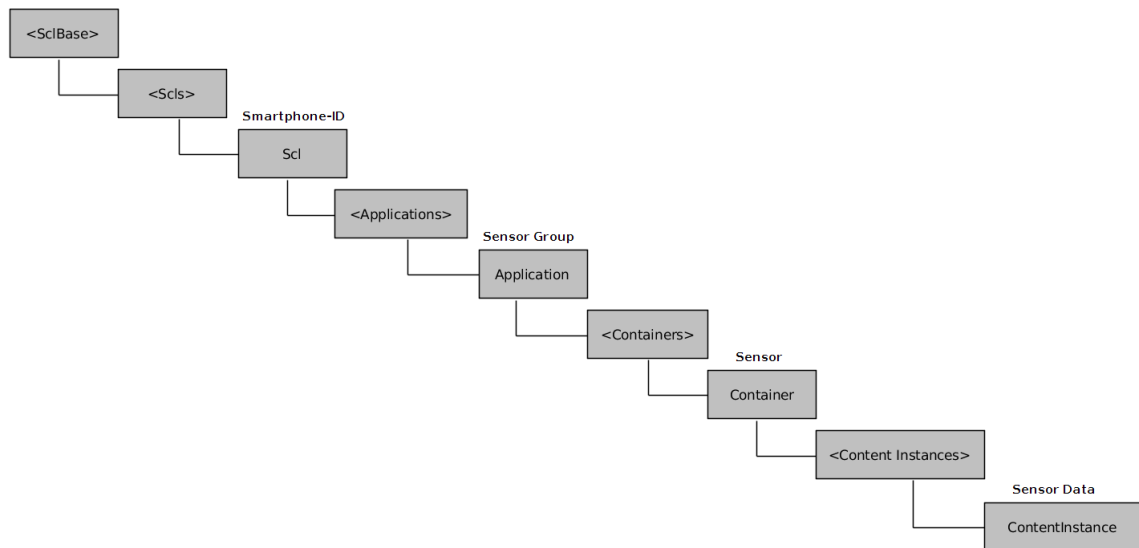


Figure 4.3: Mapped mobile M2M GW resource structure.

M2M NA Resource Map:

The mobile M2M NA also needed to have the resources specified in Section 3.1.3 to have some mapping of its own. To allow the mobile M2M NA to execute commands on the mobile M2M GW, the interaction had to somehow be mapped in the ETSI's data structure. Two approaches were discussed with the *PTIN* partners:

First Approach:

In this approach, the data flow would start with the subscription of the resource *Applications*, by the mobile M2M GW. This subscription would be limited in the NSCL by Access Rights, providing only the relevant mobile M2M NA information back to the mobile M2M GW. This means that when the mobile M2M NA changes its own resources, the mobile M2M GW will be notified with the relevant actuation data. This approach is illustrated in Figure 4.4.

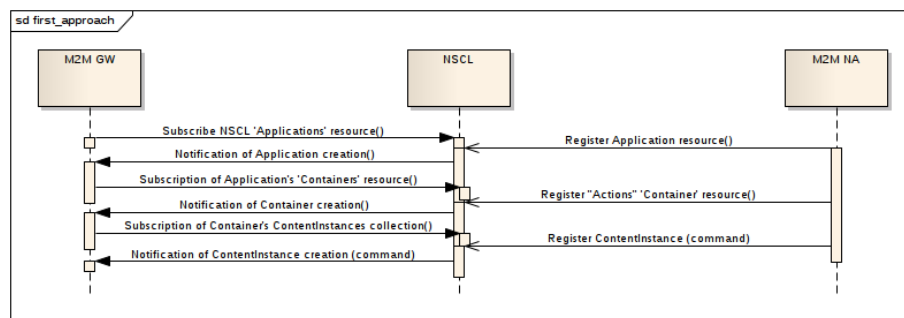


Figure 4.4: First Approach.

Second Approach:

In this approach, when the mobile M2M GW is in the process of registering the *Container* resources, two different types of resources would be created, one for the mobile M2M to write in, that would be subscribed by the mobile M2M NA, and another for the latter to write in, that would be subscribed by the mobile M2M GW. The sensor data would be sent in the *Container* in which the mobile M2M GW would write in, and the acting commands would be sent by the mobile M2M NA's *Container*, triggering the notification for the intended destination, thus communicating. This approach is illustrated in Figure 4.5.

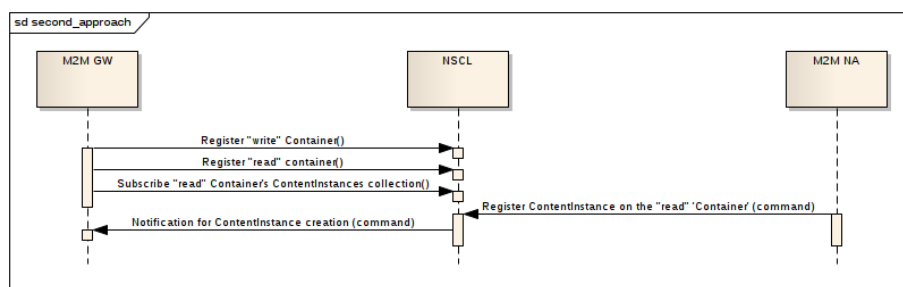


Figure 4.5: Second Approach.

Some considerations were taken into account to the decision:

- On the first approach, without the *Access Rights* limitation by the NSCL, the mobile M2M GW would have an enormous amount of information to deal with, to subscribe all active *Application* resources, increasing the network traffic with possibly useless information.
- The first approach is considerably more scalable, since as long as the mechanism for subscribing the M2M NA's (all the ones present in the NSCL database) is working, it will work on every case.
- The second approach would have the mobile M2M NA write data in a *Container* that did not belong to itself, which could possibly cause *Access Rights* issues in the future.
- The second approach would have the mobile M2M GW subscribe its own resource, because it would be changed by the mobile M2M NA.

The first approach was chosen, with the actions to be performed stored in a *Container* resource for that purpose. The scalability of the second approach was the decisive factor, as two *Container* resources would have to be managed.

The mapping evolved around the actions *Container*, meaning that the *ContentInstance* resources will be the actions themselves. The *Application* resource has a mapping similar to the mobile M2M GW's *Scl*, with attributes unique to the smartphone (model and serial number), to assure its uniqueness on the NSCL database. The resulting map is shown in Figure 4.6.

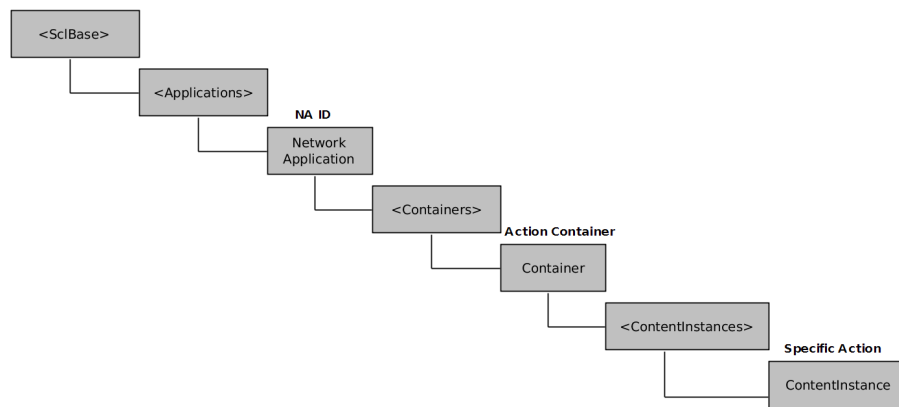


Figure 4.6: Mapped mobile M2M NA resource structure.

4.2.2.4 Sensor Procurement

In order to implement the healthcare use cases that will act as proof-of-concept for the ETSI resources map, a procurement of medical sensors to be integrated was necessary. Given the requirements defined in Section 4.2.2.1, three types of sensors could be used:

- **Heart-Rate Monitor:** This sensor would allow the user to keep continuous track of his heart-rate.
- **Sphygmomanometer (Blood pressure monitor):** This sensor would allow the user to make measurements that yield the maximum and minimum blood pressure.
- **Scale:** This sensor would allow the user to make punctual measurements of his weight.

A first search attempted to find sensors certified by the Continua Health Alliance (detailed in Section 2.4.2.2), but given the closed nature of the protocol, the *PTIN* partners have discarded that option for this project. In order to be faster integrated in the project, a heart-rate monitor already available in *IT* and with no Non-Disclosure Agreement (NDA) or proprietary protocol was chosen. The *Zephyr™ HxM Heart-Rate monitor* [74] has an open API with Java libraries, which was ideal for integration with this project. Then the scale chosen was the *ForaCARE W310*¹, which was owned by *PTIN*, with known protocol specifications. This scale measures the body mass index, as well as body fat, besides the normal user weight. However, bureaucracies due to an NDA *PTIN* had signed in order to be granted access to the protocol delayed the availability of the sensor, leaving no time for it to be implemented.

4.2.3 Local Interface

As argued in Section 4.2.1 there is a great interest in having the possibility of a local connection between the two applications when running in the same smartphone, since the NSCL acts as a intermediary between the mobile M2M GW and the mobile M2M NA, as shown in Figure 4.7.

¹More information in <http://www.foracare.com/weightscale-W310.html>.

The figure illustrates how communicating through the NSCL wastes bandwidth, when a local connection could be used instead.

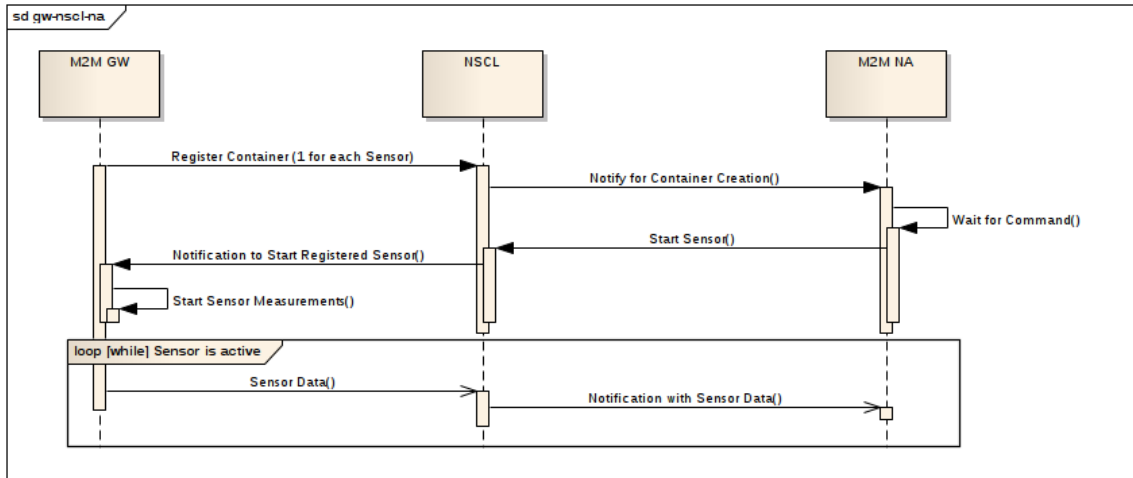


Figure 4.7: NSCL as information forwarder.

The proposed local interface builds upon the ETSI standard to allow the user to use the mobile M2M NA transparently, sending the commands to the mobile M2M GW either locally or through the NSCL, depending on which mode is chosen. Figure 4.8 illustrates this concept's approach.

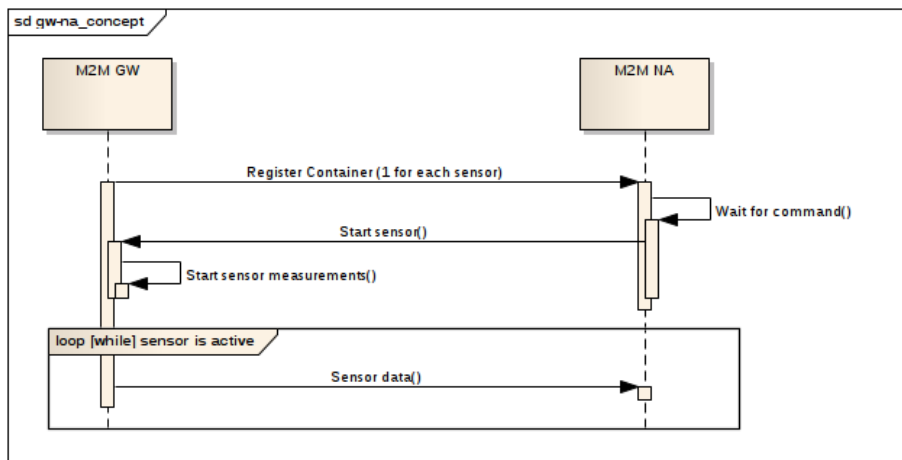


Figure 4.8: Local Interface Communication.

This *bypass* solution does not mean, however, that the sensor information will be lost, as the mobile M2M GW will store all the information sent locally in a database file. This data can then be delivered in two different situations:

- If the user stops the mobile M2M GW while the sensor is running, or there is no connectivity when the stop command is received, the mobile M2M GW will store this data in the smart-phone's memory. This file will be opened the next time the mobile M2M GW is started, and has connectivity, so that the saved resources are registered in the NSCL.

- If the user sends a command to stop the sensor, the data will be sent to the NSCL immediately, if the mobile M2M GW has connectivity at that time.

If the local interface is used, the only communication to pass through the NSCL is the session establishment, where the mobile M2M GW and NA establish the TLS session to register their resources. After this, the *Container* creation (each regarding one sensor), and the delivery of commands and sensor data (by registering *ContentInstance* resources) is entirely done locally. This saves bandwidth for four reasons:

1. Creation of resources, either *Container* or *ContentInstance*. The *Container* is only created once, so the major savings will come from the *ContentInstance* resources.
2. The NSCL answer to the registration of these resources, will also not be received, saving an amount very similar to the previous reason (1.).
3. The notification triggered by the creation of resources will not happen, saving a bandwidth slightly larger than steps 1. and 2., since the *Containers* and *ContentInstances* collections are slightly larger than its "child" resources.
4. The answer to the notifications, which is approximately the same as the previous reason (3.).

The amount of savings obtained will then depend on the length of the sensor information, and how frequently that data is sent, since one *ContentInstance* delivery triggers the sequence of four message exchanges mentioned above.

$$ContentInstance_size = Transport\&Protocol_headers + ContentInstance_info + sensor_or_command_data$$

$$Notification_size = Transport\&Protocol_headers + ContentInstances_info + ContentInstance_size$$

The savings for each sensor message or command sent will be roughly: $2 * ContentInstance_size + 2 * Notification_size$, as shown above.

4.2.4 Mobile Applications Design

A big part of this approach is the design and implementation of the mobile Android OS applications that communicate with the sensors and with each other. Section 4.2.4.1 will present the concept details of the mobile M2M GW, while Section 4.2.4.2 will present the mobile M2M NA specificities.

4.2.4.1 Mobile M2M Gateway

The mobile M2M GW acts as a proxy for the supported Bluetooth sensors, by announcing them, gathering their data, and processing it for registration in the NSCL, using ETSI's mId interface, as explained in Section 3.1.2. ETSI standardizes an entity called Gateway Application (GA) that has the responsibility to manage this connection between legacy sensors (older, non ETSI

compliant devices) and the M2M GW, through the dIa interface (as seen in Figure 3.2). For the purposes of this dissertation, we decided to converge the M2M GA and M2M GW in a single Android OS application, focusing on the mId connection to the NSCL and on the mapping needed to fulfill the requirements defined in Section 4.2.2.1. The mobile M2M GW application is explained below, where its detailed architecture is presented.

The mobile M2M GW design is based on the concept of services and threads to run entirely in the background, with no GUI. The aim was to build a mobile M2M GW as modular as possible, easing the addition of supported features, sensors and protocols so that, as the mobile M2M GW becomes more complex, changes and extensions could be easily accommodated.

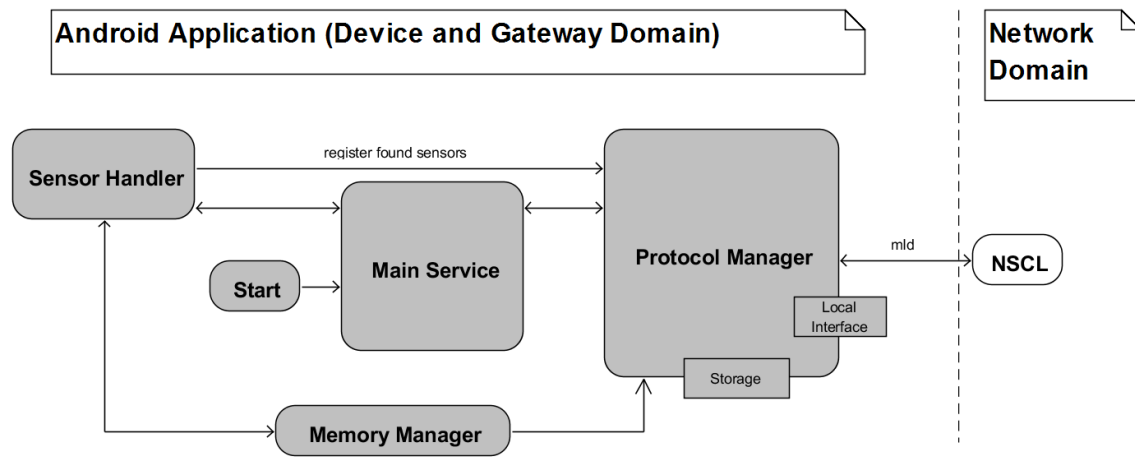


Figure 4.9: High-level Mobile M2M GW Architecture

Figure 4.9 represents the high-level architecture of the mobile M2M GW and the modules' roles are described below:

Start: This module has the responsibility to start the Main Service on start-up or upon receiving a specific Android OS command;

Main Service: This module assures that the application stays active, despite not having a GUI, while also managing the remote commands, previously interpreted by the Protocol Handler. It is also responsible for receiving and processing Android inter-process commands from the mobile M2M NA, that start or stop the mobile M2M GW.

Sensor Handler: This module has the responsibility to manage the connections to sensors, including internal and Bluetooth searches, as well as controlling which ones are active at each time, and manage their threads. It also handles the GPS measurements, that are present in the sensor data;

Memory Manager: This module handles the memory buffers assigned for each sensor, and manages when the data should be sent;

Protocol Manager: This module handles all the communication aspects of the mobile M2M GW. It is responsible for checking the network connectivity, marshalling the sensors data, and communicating with the NSCL to register itself, the sensors and the their data. It is also

responsible for managing the commands received and for storing the sensors data if connectivity is lost. It also handles when the local interface is used.

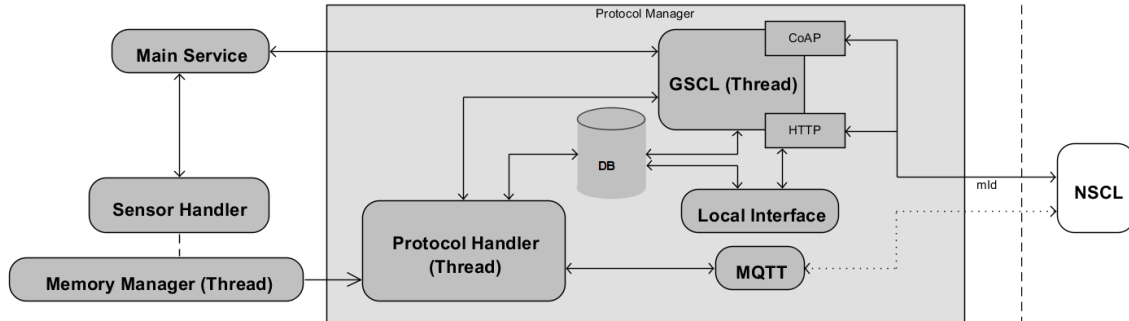


Figure 4.10: Mobile M2M GW Protocol Manager Architecture

Figure 4.10 shows the detailed architecture of the Protocol Manager, which is the most intricate module of the project. Its modules details are as follows:

Protocol Handler: This module has three main roles in the mobile M2M GW:

1. When the Memory Manager flushes sensor data to be delivered to the NSCL, this module handles the JSON marshalling of that data, depending on the sensor, so it can be properly read afterwards.
2. Know whether an Internet connection exists. If it does not, and the application is starting, it forces the application to wait for an available connection. If there is no connection when sensors data is flowing to be registered in the NSCL, the module saves the marshalled payload on the smartphone's memory. When connectivity returns, the marshalled data is then properly registered onto the NSCL.
3. This is also the module that depending on the protocol chosen to communicate, relays the marshalled information to the proper module.

GSCL: This module partially implements the ETSI GSCL, handling the connectivity with the NSCL through the mId interface using the supported protocols (HTTP or CoAP). It has an internal memory with the created resources and it stores them in a local database, in order to save traffic when the resource is already registered in the NSCL. By having its own thread, this module assures that the communication protocols are accessed in a synchronized manner, only allowing one outgoing request at a time to be executed. This module is responsible for the creation of the *Container* resource with the name of the sensors, and the *ContentInstance* resources with the marshalled sensor data.

MQTT: This module implements the MQTT protocol, and it was already completed when this dissertation started. It currently does not support the communication of the sensors, since it is not supported by ETSI, nor by the NSCL.

Local Interface: This module represents the local interface connection explained in Section 4.2.3, which is executed over HTTP, through the localhost interface of the smartphone. This module also has access to the database, since it stores the sensors data so it can later be delivered to the NSCL.

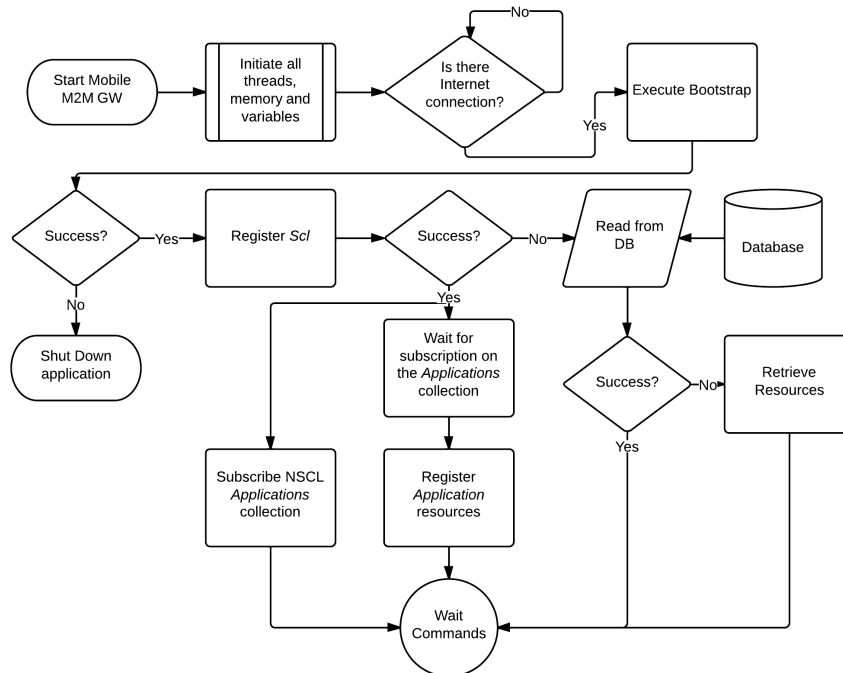


Figure 4.11: Mobile M2M GW Flowchart

Figure 4.11 illustrates the information flow inside the mobile M2M GW. After starting the application, the threads corresponding to the modules present in Figure 4.9 (except Start), and all their variables and memory buffers are started. The modules have their own threads in order to effectively reduce race conditions and delay of processing, since the instructions for the threads to run are stored and handled in a first-in first-out (FIFO) pile. When there is an Internet connection, the mobile M2M GW proceeds to execute the Bootstrap procedures, using one of the supported ETSI protocols. If the Bootstrap is unsuccessful, the mobile M2M GW shuts down (since it has no way to communicate to the NSCL), but if it is successful, the mobile M2M GW proceeds to register its *Scl* resource. If the registration fails (meaning that the resource already exists on the NSCL), the mobile M2M GW attempts to retrieve those resources from its local database. If the resources stored somehow got corrupted, or do not match the requirements, the mobile M2M GW retrieves the correct resources from the NSCL, and proceeds to await for commands. However, if the *Scl* resource registration is successful (meaning that it is the first time this particular device is connecting or that the NSCL database was wiped), the mobile M2M GW subscribes the NSCL *Applications* collection (where the NA's are stored), and then awaits for a subscription on its own *Applications* collection. Once it receives this subscription, it registers its *Application* resources

corresponding to the groups of sensors that are supported (in this dissertation's case only health sensors are supported, so only one *Application* resource is registered). After this registration, the mobile M2M GW awaits for commands, that will trigger its functionalities.

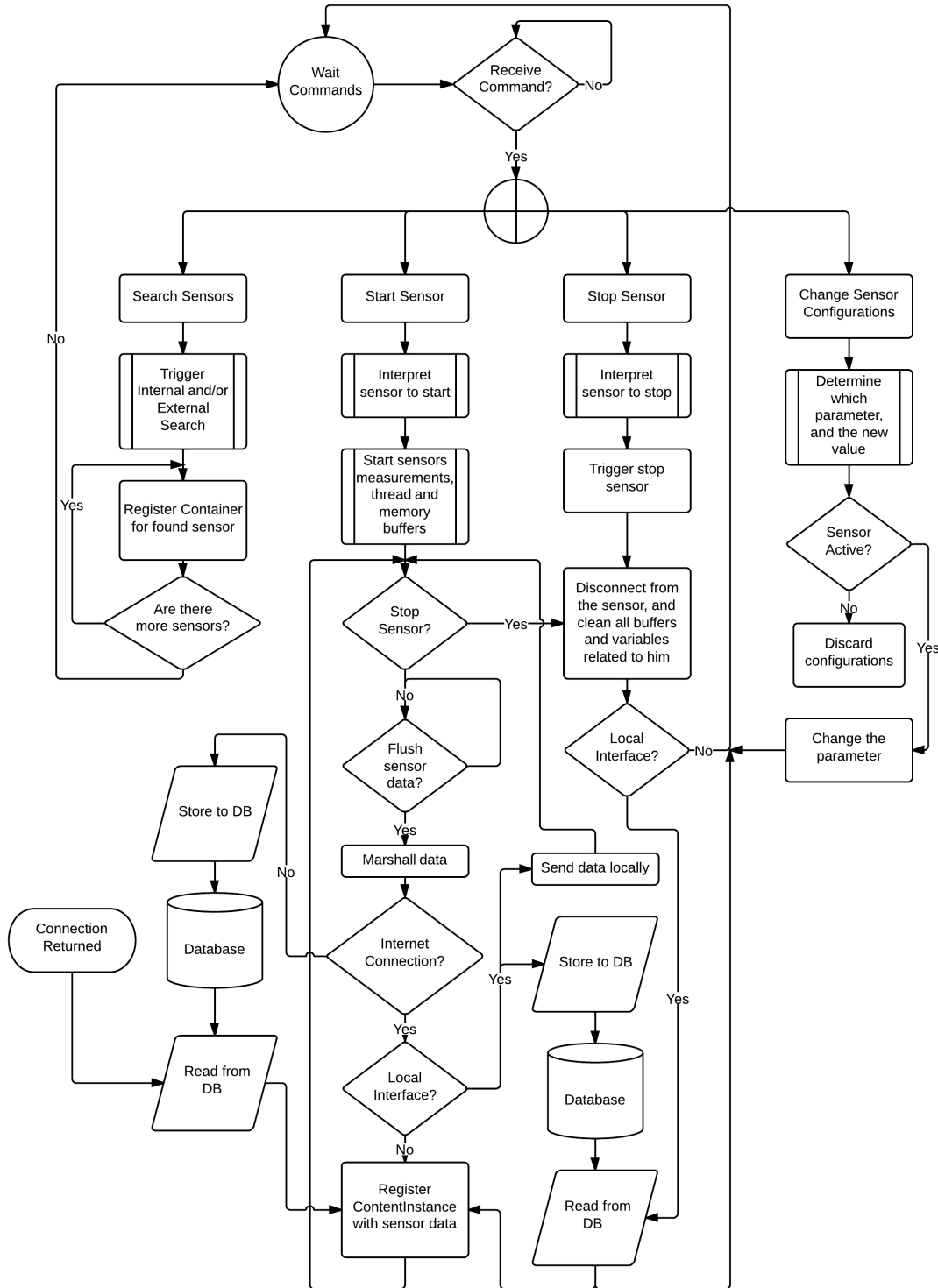


Figure 4.12: Mobile M2M GW Commands Flowchart

Figure 4.12 represents the data flow produced by the incoming commands, each carrying some specific information: The *search sensors* command contains information on whether the search to be executed should be only internal, only external (through Bluetooth), or both. The *start sensor* command contains the name of the *Container* to start, that because of the map explained in Section 4.2.2.3 represents the sensor name. The *stop sensor* command also contains the name of the *Container* to stop; The *change sensor configurations* command contains the name of the *Container* to act, and the sensor configurable parameters that will be described below.

If a *search sensor* command is received, the Sensor Handler will proceed to execute the required local and/or Bluetooth search, and process the found sensors. It then delivers the found sensors to the Protocol Manager's GSCL module, in order to register one *Container* resource for each sensor found. This procedure only announces that the sensors are present in the vicinity of the mobile M2M GW, but not started immediately.

When a *start sensor* command is received, the Sensor Handler starts the procedures to connect to it, start its own thread, and trigger the Memory Manager to start the buffers and variables regarding that sensor. Each sensor has a memory buffer with the following configurable parameters:

Reading Granularity : Maximum interval between two sensor readings.

Maximum Transmission Granularity : Maximum interval between two sensor *flushes*.

Maximum Buffer Size : Maximum amount of physical memory that the buffer storing the sensor information is allowed to be.

The data on the buffers stays stored until the transmission granularity is reached or the buffer gets full, and the trigger for delivery is whichever comes first. Once this data is *flushed*, it is sent to the Protocol Manager for marshalling and dispatch. If there is no connectivity when the information reaches the Protocol Manager, it is stored in the local storage, until the mobile M2M GW has the ability to properly register the data on the NSCL. The marshalling process starts with the packaging of the sensors data into a JSON object. The basic output is shown below with an example from a Zephyr HxM Bluetooth sensor. The values array is specific to each sensor, where the relevant data can be sent, whereas the remaining information about the sensor and GPS is common between different types of sensors:

```
{
  "values": [
    {
      "timeStamp": "1397491419169",
      "RRinterval": [
        "728"
      ],
      "heartRate": "87"
    }
  ],
  "sensorModel": "HxM BT Heart Rate Monitor",
  "sensorSerial": "12:34:56:78:90:AB",
  "GPS_Latitude": 40.6299045,
  "sensorType": "ZEPHYR",
  "GPS_Accuracy": 20,
  "GPS_Longitude": -8.6460066
}
```

This marshalled sensor data is used on the creation of a *Content* resource, which is an attribute of the *ContentInstance* resource. The *Content* automatically encodes the payload in Base64 [63], and the GSCL module then registers the *ContentInstance* on the NSCL, as explained in Section 3.1.2, using one of the ETSI supported protocols. In case there is no connectivity, the marshalled information is stored in the database file initiated at start up. When the connection returns, this storage file is chached into memory, and the messages are dispatched. Using the local interface, the sensor data is automatically stored on the database, while being sent locally to the mobile M2M NA.

Upon receiving a *stop sensor* command, the sensor connection is terminated, along with the memory buffers associated with it. If this command is sent while using the local interface, the data stored in the database by the local interface is then registered on the NSCL, provided there is connectivity. If there is no connectivity, the data remains in the database until the next application start up.

The *change sensor configurations* command contains the information on the target sensor, the configuration to be changed, as well as the new value. If the target sensor is not active at the time, the configuration request is discarded, otherwise the Sensor Handler manages the change of the configurations.

4.2.4.2 Mobile M2M Network Application

The mobile M2M NA implements the mIa ETSI interface, as mentioned in Section 3.1.3. Having a mobile M2M NA was essential to complete the system, to allow the user to have visual feedback (through the GUI), besides the ability to command the connection to sensors. The mobile M2M NA receives the relevant information about the healthcare sensors by notification from the NSCL, after the proper subscription of the Gateway's resources. It also sends commands to the Gateway to trigger the actions needed to fulfill the use case scenario. The mobile M2M NA Android application architecture design is presented below.

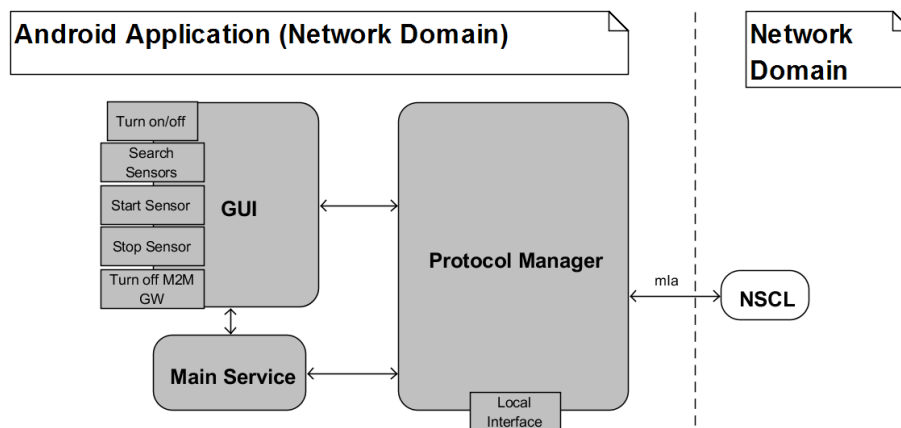


Figure 4.13: High-Level Mobile M2M NA Architecture

Figure 4.13 represents the high-level architecture of the mobile M2M NA, and its modules roles are:

GUI: This module handles the user input, that depending on the interface the application is on, can represent turning it on or off (with options to turn off only the mobile M2M NA or also the mobile M2M GW), or send the commands to search, start and stop sensors.

Main Service: This module, inherited from the mobile M2M GW's design, manages the active threads and the information flow between different user interfaces.

Protocol Manager: This module handles the Internet connection, and how the mobile M2M NA communicates with the NSCL. It is also responsible for the local interface communication, when active.

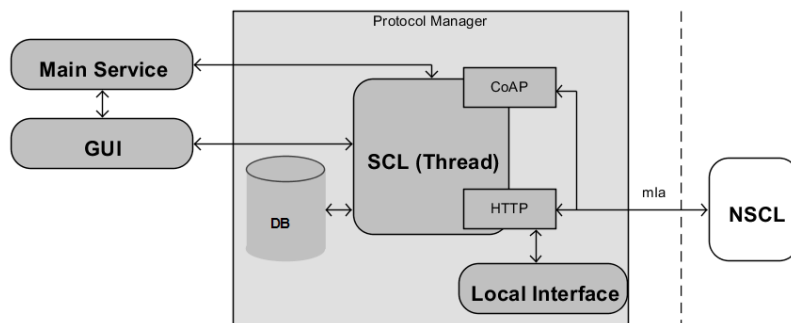


Figure 4.14: Mobile M2M NA Protocol Manager Architecture

Figure 4.14 illustrates in greater detail the mobile M2M NA's Protocol Manager and its sub-modules. Their roles in the application are:

SCL: This module handles the connectivity with the NSCL through the mIa interface using the supported protocols (HTTP or CoAP). It has an internal memory with the created resources and it stores them in a local database, in order to save traffic when the resource is already registered in the NSCL. By having its own thread, this module assures that the communication protocols are accessed in a synchronized manner, only allowing one outgoing request at a time to be executed. This module is also responsible for creating the *ContentInstance* resources corresponding to the commands triggered by the user's input on the GUI.

Local Interface: This module is responsible for the local connection to the mobile M2M GW, when it is active. The local interface can deliver commands and receive sensor data, allowing the mobile M2M NA to function just as if it was communicating through the NSCL.

Figure 4.15 represents the information flow of the mobile M2M NA. After the user starts the application, the modules present in Figure 4.13 are started, with their corresponding threads. If there is Internet connection at start up, the Bootstrap procedure is initiated. Being a critical operation, if it fails, the mobile M2M NA shuts down, as it has no way to communicate with the NSCL. If the Bootstrap is successful, the mobile M2M NA proceeds to the registration of its unique *Application* resource onto the NSCL. If this registration is not successful, meaning that it was already

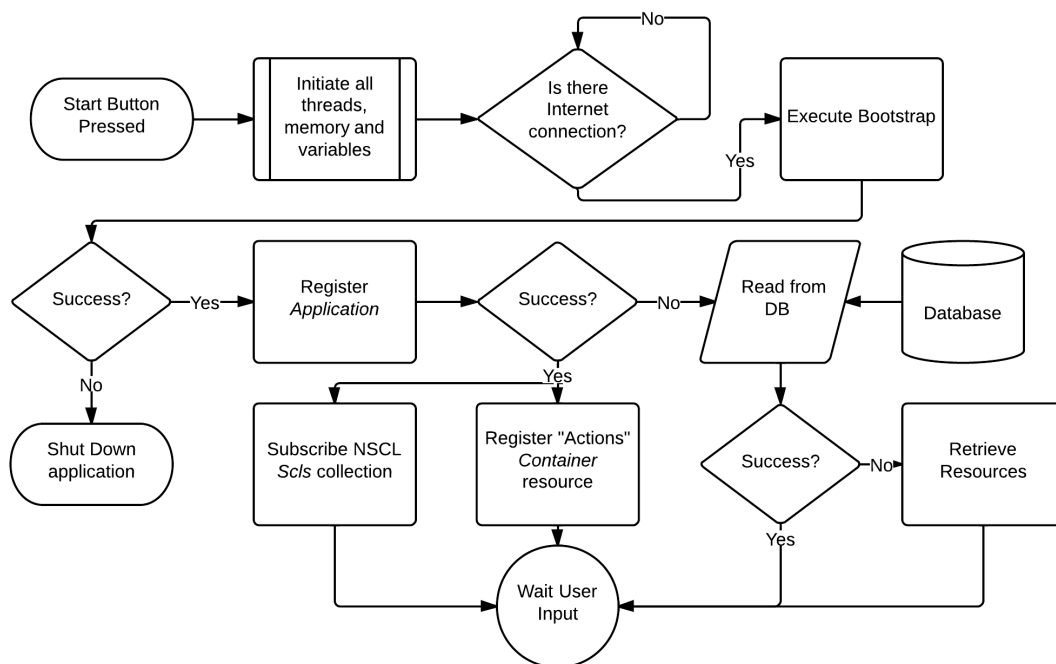


Figure 4.15: Mobile M2M NA Flowchart

completed before, the mobile M2M NA retrieves the data stored in its database. If somehow that data was corrupted or deleted, the mobile M2M NA then retrieves the necessary *Application* and *Container* resources from the NSCL. If, however, the *Application* resource registration is successful, the mobile M2M NA then registers the "Actions" *Container*, and also subscribes to the *Scls* collection of the NSCL, where the mobile M2M GW registers itself.

The functionality of the mobile M2M NA then depends on the user's input, and is illustrated in Figure 4.16. After the successful registration of resources and the mobile M2M NA is in possession of the resources needed to send commands, specifically the "Actions" *Container*, the *search sensors* button is activated. The buttons to shut down the mobile M2M NA or both applications are always active, and if pressed they will trigger the proper termination of the selected application(s).

If the user presses the *search sensors* button, a command *ContentInstance* with that action will be registered on the "Actions" *Container* on the NSCL. The mobile M2M NA then disables the *search sensor* button and enables the *start sensor* button, and awaits for the NSCL notifications on new *Container* resources (if there were any), containing the information on the found sensors. These sensors are presented to the user in a list, and pressing the *start sensor* enables him to choose which sensor to start. The choice of a sensor registers a *ContentInstance* resource with that action on the "Actions" *Container*, followed by the subscription of the sensor's *ContentInstances* collection, for receiving the posterior notifications with the sensor *ContentInstance* resources. Still derived from the *start sensor* trigger, the GUI changes to give the user feedback on the received data, and activates the *stop sensor* button. Pressing the *stop sensor* button registers a *ContentIn-*

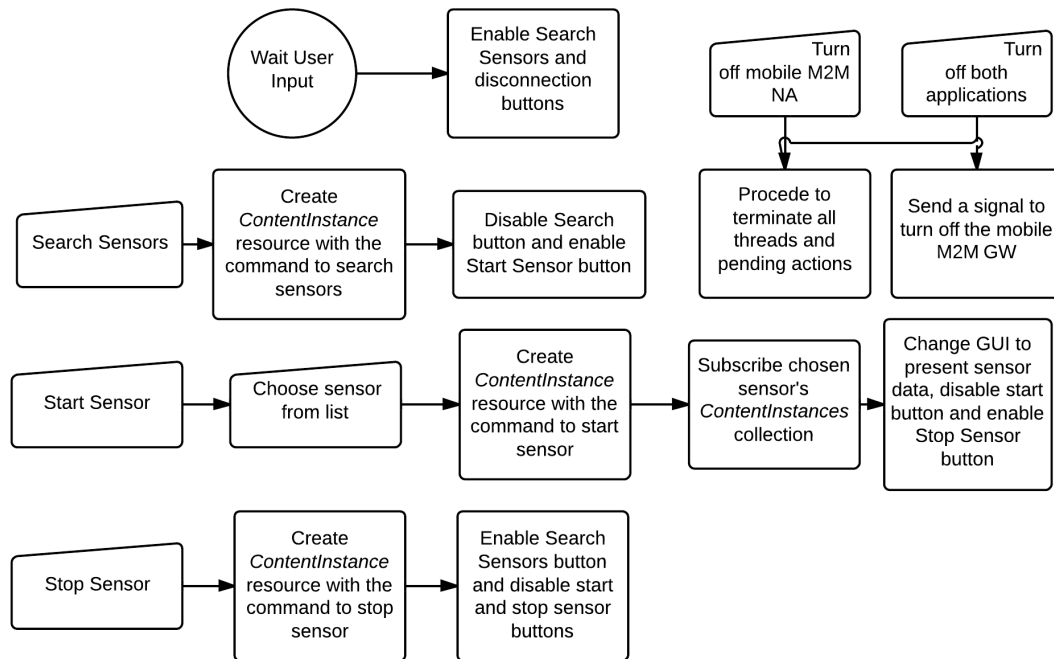


Figure 4.16: Mobile M2M NA Protocol Manager Architecture

stance with that action on the "Actions" *Container*, followed by re-enabling the *search sensors* button and disabling the *stop sensor* and *start sensor* buttons, allowing the user, to re-execute a search, if necessary. Using the local interface poses absolutely no changes to the information flow just described, and the user does not need to make any changes to the way he interacts with the application.

4.3 Evaluation

The proposed solution is going to be evaluated according to the goals mentioned in Section 1.3. To determine to which degree the mobile M2M GW and mobile M2M NA are functional, Table 4.1 defines the requirements, organized by priorities. The traffic measurements will evaluate if the it effectively saves bandwidth, as it was designed to do.

Target Application	Priority	What to Evaluate
<u>Basic Use Case</u>		
Both	High	Communicate securely with the NSCL using at least one supported protocol
Both	High	Register itself and its resources in the NSCL
Both	High	Support a web-server that is able to receive subscriptions and messages
Mobile M2M GW	High	Support at least one medical sensor
Mobile M2M GW	High	Have the ability to receive and process commands according to the Use Case scenario
Mobile M2M NA	High	Have the ability to send commands to the Gateway and process its answers
Mobile M2M GW	Low	Support more medical sensors, widening the test base
<u>Traffic Reduction Use Case</u>		
Both	High	Implement the local interface
Both	High	Compare communication traffic through the NSCL and through the local interface
Mobile M2M GW	Medium	Use a local storage to safeguard the sensor data when the connection drops

Table 4.1: Evaluation metrics

Chapter 5

Implementation

The development of the solution shown in Chapter 4 had its share of obstacles and difficulties, which are always present in this kind of project. This Chapter aims to explain the development of each feature and functionality, what were the main difficulties, and how they were overcome.

To begin with, the technologies needed for the implementation process are presented in Section 5.1. Then the implementation of the communication protocol on which the ETSI M2M standard relies for its communications is explained in Section 5.2, followed by the Bootstrap procedure, in Section 5.3. Section 5.4 details the web-server created within the mobile M2M GW and mobile M2M NA to receive the necessary subscriptions, before the implementation details of the ETSI's resource structure in Section 5.5.

Section 5.6 focuses on the integration of the healthcare sensors, paving the way for the explanation of the mobile M2M GW communication with the NSCL as well as the functionalities of the commands, in Section 5.8. The mobile M2M NA's communication is then detailed in Section 5.9 detailing how different actions are shown in the user interface, ending with the local interface details in Section 5.10.

5.1 Technologies

Both the mobile M2M GW and the mobile M2M NA are being built as Android Operating System applications. This choice was made given its open source nature, that provides easy integration with the external sensors, and because the programming language is Java, which was already familiar. The wide range of devices currently running Android OS was also an important factor since the result of this dissertation may evolve into a commercially available product.

The main software used in the development was:

- Android Development Kit (SDK) [75];
- IntelliJ Integrated Development Environment (IDE) [76];
- Apache Maven, for the dependency management, and easier integration of the several frameworks/libraries [77];

- Subversion version control [78].

The frameworks used to implement the use cases were:

- Native Apache HTTP implementation;
- Californium CoAP framework [79];
- Paho-mqtt library (Release 0.4.0, MQTT version 3) [80];
- Jackson is the library used to execute the marshalling and de-marshalling of ETSI resources.

Also, as communication technologies, Bluetooth, cellular (2G/3G/4G) and Wi-Fi capabilities to communicate with the NSCL.

5.2 ETSI Protocol Implementation

The Protocol Manager of both applications was designed in a modular way, as explained in Chapter 4, to allow the use of either HTTP or CoAP protocols to communicate with the NSCL. Initial development efforts focused on CoAP, but there were some issues with the Californium CoAP library (suggested by *PTIN* partners because they were also using it) and the use of DTLS. The problem was that the framework did not provide a useful API to determine where the configuration and certificate files were located, which forced the import of the framework's source code into the project. However, since the NSCL didn't fully support CoAP at the time, this integration was put on hold, and efforts focused on the integration of HTTP.

With HTTP, the first stage was to find a library that could connect over TLS using the certificates retrieved from the NSCL. *PTIN* was using an Oracle library, that yielded dependency issues in Android, which ultimately led to the use of the native Apache HTTP libraries, shipped with the Android Operating System. The choice was fruitful, since the connections were successfully implemented using TLS (v1.2) and also implementing a TLS extension (and ETSI requirement) called Server Name Indication (or SNI). This extension allows a web-server, in this case the NSCL, to provide different services over the same IP address, provided that the client specifies its destination in the connection. In this dissertation's case, the NSCL URI was *https://phonegw.nscl.m2m.ptinovacao.pt*, with SNI *phonegw.nscl.m2m.ptinovacao.pt*.

On Android there were some issues with this feature implementation, where the well documented Android fragmentation [81] had impact, since the Apache features that allow the SNI to be properly implemented in Android were only introduced in API 17 (Android version 4.2), meaning that the introduction of this requirement limits the number of devices that can use the final application.

5.3 Bootstrap

As mentioned in Section 2.2.1 the Bootstrap is the first procedure to be executed, either by the mobile M2M GW or by the mobile M2M NA (the procedure is the same for both), as it

yields the application's session key (which is a X.509 Certificate with the NSCL public key, and the requesting application's RSA private key) to be used in the secure connection with the NSCL. These security concepts were explained in Section 3.3. Since this procedure is protocol dependent, it was only implemented for HTTP, as were the rest of the communications, as mentioned in Section 5.2.

In order to correctly execute the Bootstrap, the mobile M2M GW or mobile M2M NA have a pre-provisioned Certificate-Key pair, that is used to fetch the session keys. This pre-provisioned pair is stored as a project resource, and is compiled with the applications to allow them to execute this procedure. After receiving the NSCL session keys, they are stored in the smartphone's memory and then used to create the TLS enabled HTTP client that will be used to execute the communication with the NSCL. This procedure is illustrated in Figure 5.1.

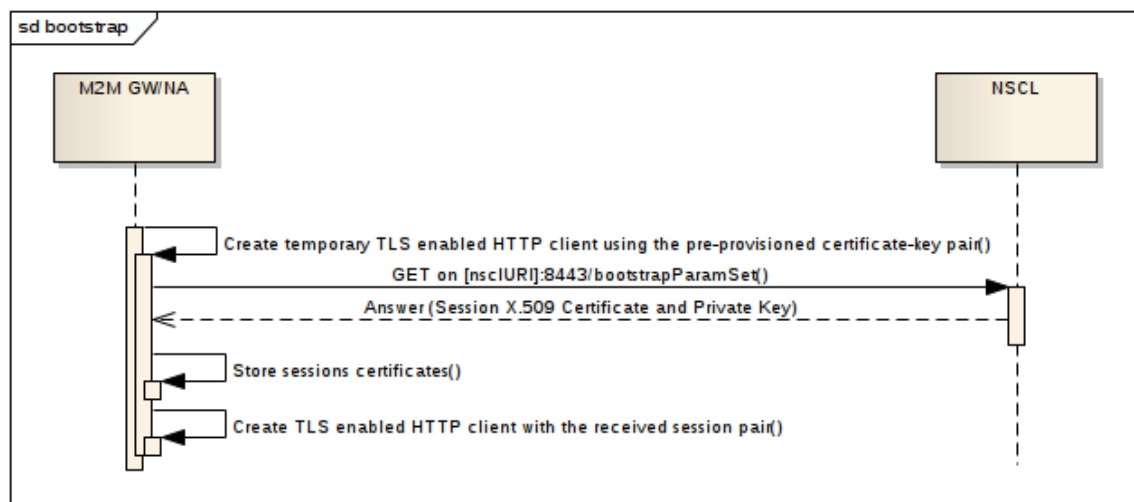


Figure 5.1: Bootstrap Procedures

Both the pre-provisioned and the session key are created by the NSCL to the specific client (GW or NA) and they are the client's own private key and the NSCL public key to use in the TLS handshake. During this connection, the mobile M2M GW or mobile M2M NA use the NSCL public key to initiate a secure handshake, to then share the symmetric key to be used for the session encryption where the actual data is exchanged. The same certificate-key pair is used to generate the secure HTTPS web-server, allowing the NSCL to also communicate securely over TLS.

5.4 Web-Server

The web-server exists both in the mobile M2M GW and mobile M2M NA to allow the NSCL (or each other using the local interface) to subscribe their resources. These are naturally protocol specific, and the CoAP web-server was never implemented, because the communication development focused on HTTP for the reasons mentioned in Section 5.2.

There was some research on HTTP lightweight implementations, being NanoHTTPD [82] the most interesting. This library was implemented, but the lack of SSL/TLS support led to its eventual

dismissal. Since the client was already functional with the Apache libraries, the web-server was also implemented using these, with and without support for TLS. However, because the NSCL still uses an insecure connection to register its subscriptions or send notifications, the TLS feature on the mobile applications is disabled.

For the communication to properly work, the web-servers needed to be accessible from the outside of the FEUP network, to allow the subscriptions and subsequent notifications to reach the applications. This required a configuration of the firewall, to open the ports used by the applications, as well as the establishment of routes to direct that traffic to the the right place. This led to some bureaucracy steps¹ within FEUP, as well as configuration challenges², to allow the web-server to be accessed from the outside of the internal network (from *PTIN*, in Aveiro). This introduced an unexpected delay, that shed light on one of the big problems mobile M2M solutions may face, since normally every device is behind a firewall ou router that does not allow him to become *visible* to the exterior. The end-user normally does not mess with router configurations, limiting the scalability of solutions such as this one. To overcome this issue, the ETSI M2M standard provides a *long-polling* mechanism that allows for the subscriptions and subsequent notifications to be delivered using only a HTTP or CoAP client, by creating a persistent connection started by the applications, which can always reach the NSCL. This could be used in the future, but since the NSCL still did not implement it, the web-servers were the approach chosen.

The mobile M2M GW web-server listens to port 8080 while the mobile M2M NA listens to port 9090. This was an arbitrary choice taken and can be easily changed, with the natural safeguards towards the configuration needed on the routing stand point for the testing to continue being possible from within the FEUP network.

5.5 ETSI Resource Structure Implementation

Implementing the ETSI resource structure was the first big milestone of this dissertation's, since it was a requirement for communicating with the NSCL.

The ETSI documents, specifically the TS 102 921 [62], besides the useful information about communication, also provided XML schemas for every resource of the structure. With JAXB (XML marshalling library) it was possible to convert these schemas into Java classes with annotations, for automatic marshalling and de-marshalling. The problem faced was that Android could not understand the annotations because of missing XML libraries. On Android, there were problems, since although it implements XML files to build its user interfaces, it does not have the core XML libraries (javax.xml.*) and does not allow them to be compiled along with the application. Given this issue, there were two possible approaches:

¹Acknowledgment of the significant help of Superior Technician Isidro Ribeiro in requesting the port openings to CICA.

²Acknowledgment of the significant help of Paulo Vaz (IT technician), and Carlos Pereira (Doctorate student in FEUP) for the help given with this configuration procedure.

- Import the XML libraries under a different package (such as `xml.libraries.*`) so that they could get compiled along with the application.
- Use a different library and figure out the best way to make use of the provided classes.

The solution chosen was to use the Jackson library, because of its fast performance on Android OS [83] (even when compared with the native Android JSON library), and because both approaches demands editing some or all of the provided 120 classes: The first approach requires changing the imports of every class, while the second requires commenting out the XML annotations and include a few Jackson when needed (only known by trial and error).

The Jackson JSON approach seems like the best option on the long run, even though it is clear that if there are new changes to those classes, they will have to be implemented by hand, prevent having the same annotation issue as before. A comprehensive list of the changes executed in the ETSI generated classes can be found in Appendix A.

After all the changes were completed and the applications were successfully marshalling and de-marshalling resources using Jackson, the focus shifted on the communication of the mobile M2M GW with the NSCL, but not before integrating a medical sensor to test the data with.

5.6 Sensor Integration

The mobile M2M GW's sensor handler was built in a modular way, and each sensor connected to it has its own thread to prevent race condition issues. When the sensors are searched, for each sensor found, a new *Container* is created. The *Container* name includes the sensor name (e.g. Zephyr), and a timestamp of the moment it was created. The sensors are not connected until a specific *start* command is received. Only then does the mobile M2M GW connect to the sensor and start collecting data to build and send the *ContentInstance* resources with. This was done for scalability purposes, because no sensor is going to consume battery unless specifically started. Section 5.6.1 will present the implementation details for the database created to store the sensors information when there is no connectivity.

As mentioned in Section 4.2.2.4, the only integrated sensor was the Zephyr™ HxM Heart-Rate monitor, and given its open API, that integration was easily executed with the modular mobile M2M GW. Section 5.6.2 will detail the information gathered and stored by the Zephyr sensor.

5.6.1 Sensor data storage

As mentioned in Section 4.2.4.1, the ability to store the sensor data for when no connectivity is available is an important feature for this project's use case scenario, to assure that no data is lost. To execute this, the data to be sent is stored in the smartphone's memory when the connection is lost, and re-sent when the it returns. If the mobile M2M GW is stopped before the connection returns, the data is kept in the smartphone's memory and sent the next time the mobile M2M GW is started (with connectivity), so that no data gets lost. This storage keeps the sensor data already

properly marshalled, along with the *Container* name they belong to, and the target URI, which is the necessary information for that data to be properly delivered, when connectivity is regained.

5.6.2 Zephyr data

The information being collected by the Zephyr sensor, when it is connected, is a single average heart rate value (in beats per minute, or bpm), as well as an array of RR intervals, which holds the exact time difference (in milliseconds) between *R* heart beats, as shown in Figure 5.2.

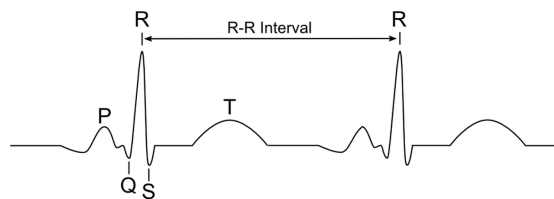


Figure 5.2: ECG example, specifying RR interval. From [8]

The average value allows for a quick glance on what the heart-rate is at any given time, and is calculated within the sensor itself whereas the RR interval measurement allows for more precise data collection and analysis, since they are the base measurement for the heart-rate calculation:

$$\text{HeartRate} = \frac{60000\text{ms}}{\text{RRinterval}} (\text{bpm})$$

5.7 Communication considerations

During the implementation of the communication, some features common to the mobile M2M GW and the mobile M2M NA had to be developed to surpass some problems. These features will be described in this Section.

5.7.1 Subscription Check

Since the communication depends almost entirely in subscriptions and notifications, there had to be a way to check if every desired resource is subscribed, to prevent the situation, where the information is not received because the resource was created and no subscription was in place to trigger the notification. To tackle this issue, a series of functions are called whenever the applications (mobile M2M GW or mobile M2M NA) need to know if there already is a subscription with their URI to the desired resource or collection. This procedure is shown in Figure 5.3.

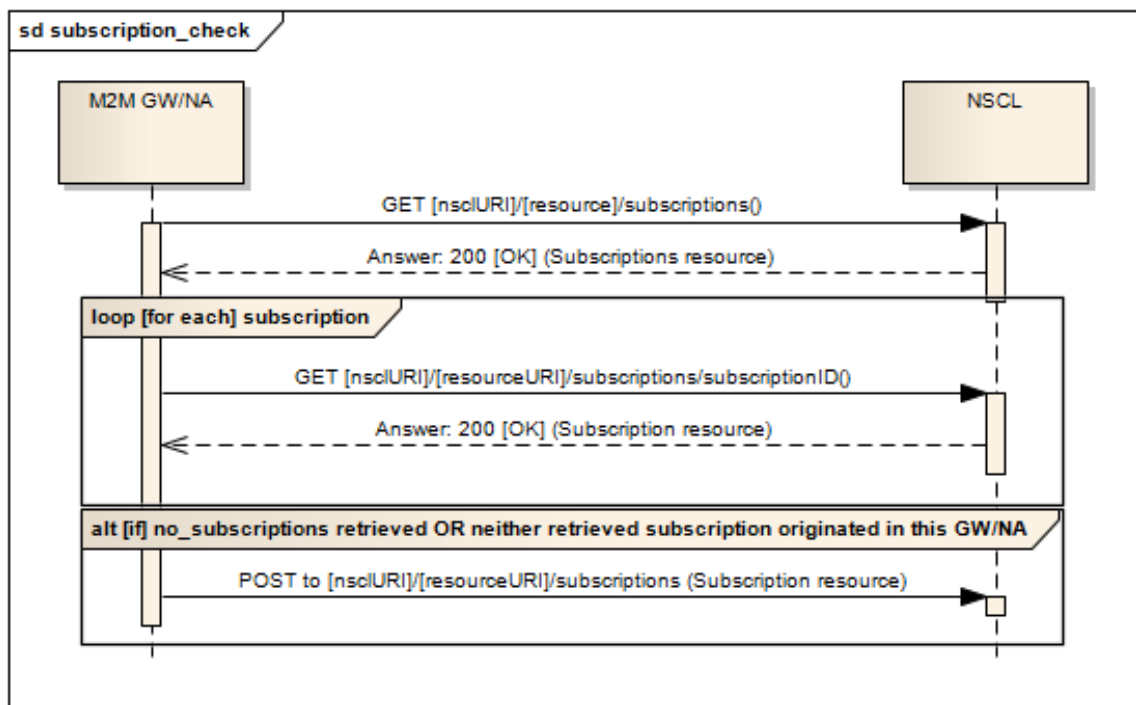


Figure 5.3: Procedure to check if subscription created by this mobile M2M GW or NA is present.

Firstly, a GET request is executed to retrieve the *Subscriptions* collection, which contains the list of all *Subscription* resources. Then, the application (mobile M2M GW or mobile M2M NA) retrieves every single *Subscription* resource, to compare the URI to its own. After checking all the resources, only if there is no *Subscription* from its URI, does the application (mobile M2M GW or mobile M2M NA) create its own. This assures that a *Subscription* is present while preventing creating multiple subscriptions with the same URI, which would result in multiple notifications for every action occurred.

5.7.2 Persistent Database

Given the mobile nature of the mobile M2M GW and mobile M2M NA of this dissertation, the mobility is a huge aspect of the design, which may cause the Internet connection to drop. The mapping discussed in Chapter 4 was designed to have unique Id's for each smartphone's mobile M2M GW and mobile M2M NA, but since the ETSI standard forces its resources, such as the *Scf* and *Application*, to have an unique identifiers (Ids) (returning the HTTP code 405 (Method not allowed) if the Identifier (*Id*) is in use), there was the need to develop a way to be able to keep this resources after they were created. To tackle this situation, a persistent database was created on both the mobile M2M GW and the mobile M2M NA to store the created unique resources on the smartphone's memory. This database stores the needed information in serialized Java objects. The classes created to fulfill this need are pictured in Figure 5.4.

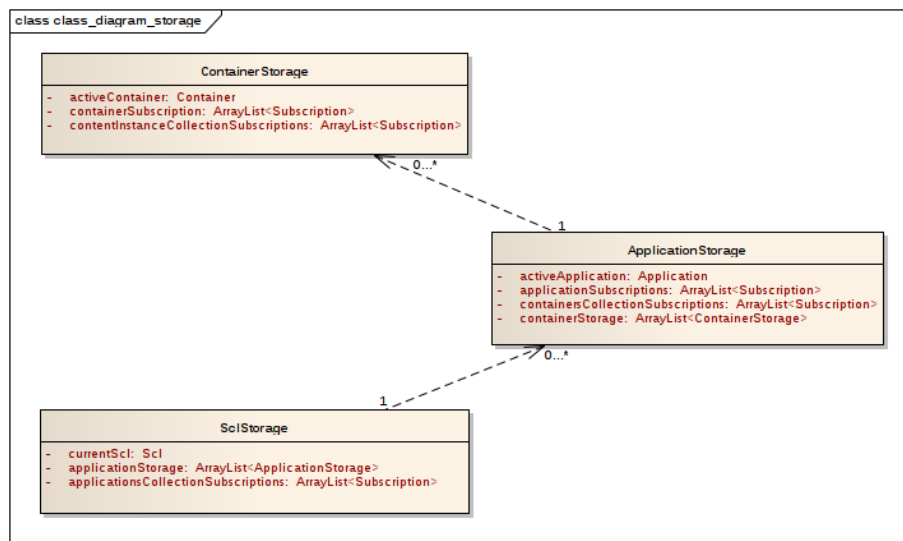


Figure 5.4: Storage Class Diagram.

The mobile M2M GW stores a *ScIStorage* instance, that holds the active *ScI* resource, has an array that contains all its active *ApplicationStorage* objects as well as an array for the subscriptions made on the *Applications* collection. The *ApplicationStorage* holds the information on the active *Application* resource, as well as *Subscription* arrays that stores the subscriptions to the active *Application* and to the *Containers* collection. An array of *ContainerStorage* instances is also stored within the *ApplicationStorage*, holding the information on the active *Container* resource, and arrays for that *Container's* subscriptions and its respective *ContentInstances* collection. All this information is relevant to store, allowing the mobile M2M GW and mobile M2M NA to hold the information needed to successfully re-connect to the NSCL with the same unique *Id*. Since the mobile M2M NA does not own an *ScI* resource, it only stores an *ApplicationStorage* instance, which was one of the reasons for this array centered design.

To accomplish the implementation of these databases, a few more changes had to be executed on the ETSI classes, adding the ability for Java Serialization, for easier and faster storage and retrieval from the database. The list of classes that needed these changes is presented in Appendix A.

5.7.3 Commands

To implement the important feature of the commands delivery, by adding *ContentInstances* to the mobile M2M NA's actions *Container* as explained in Section 4.2.2.3, common ground had to be created between the mobile M2M GW and the mobile M2M NA. To do this, the *ActionCommands* class was created, which was then extended by the specific action to be taken, as depicted in Figure 5.5. This super class has the *action* attribute that allows the mobile M2M GW to recognize which subclass it is receiving, to then retrieve the action's attributes, which allow it to act accordingly. Also, to assure that the mobile M2M GW is not acting on commands that were previously sent, the super class also has the creation time attribute, which holds the current time in milliseconds since January 1, 1970, as well as a life span attribute (also in milliseconds) that stores

for how long that command should be active. To the purpose of this dissertation's tests, the life span was 2 minutes, to prevent commands from previous tests to interfere with.

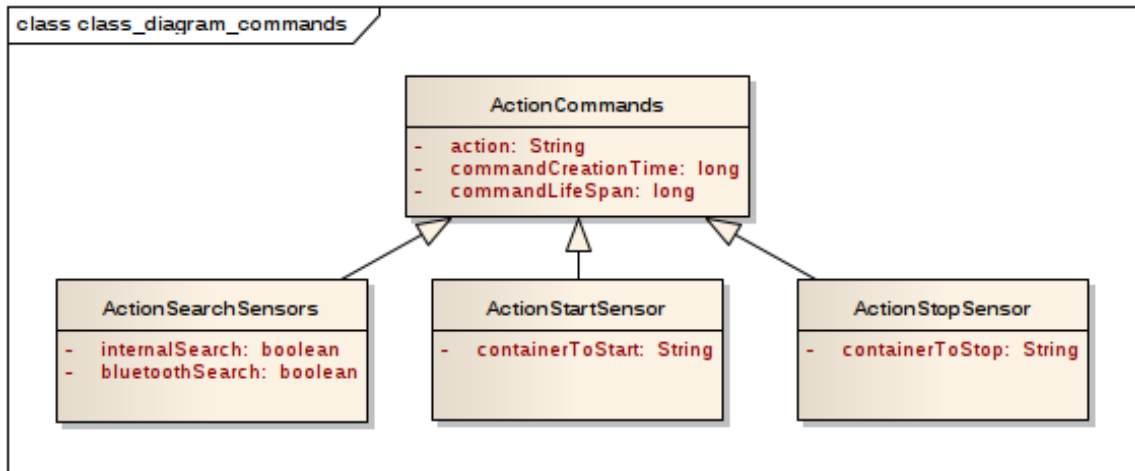


Figure 5.5: Class diagram of the structure of the commands.

To prove the concept created for the commands was functional, and regarding the list of commands presented in Section 4.2, the change of sensor configurations was not implemented since it was not a priority to test the functionalities of the sensors, that fulfill the use cases.

The action specific attributes are very simple. The *ActionSearchSensors* has two booleans to configure whether the search to be executed is internal (to use the smartphone's accelerometer, gyroscope, or magnetometer) or external (Bluetooth supported sensors) or both. The *ActionStartSensor* and *ActionStopSensor* commands are very similar, with the attribute signaling the *Container* resource on which they are acting, either to be started, or stopped. This architecture is also modular, aiming to ease the future addition of commands and features to the system.

5.8 Mobile M2M GW

This section explains how the mobile M2M GW was implemented in Android OS, in Section 5.8.1, followed by the communication specifications in Section 5.8.2. Then, how the mobile M2M GW handles the commands received is explained in Section 5.8.3.

5.8.1 Android Implementation

The mobile M2M GW concept explained in Section 4.2.4.1 was successfully implemented on Android, with some specificities. Since it has no GUI, in order to allow it to be started in the event that the mobile M2M NA is not installed in the same smartphone, an application launcher was created. This is just an icon that is present in the applications drawer of the smartphone and looks like any other Android application for the user. However, when the user selects it, no GUI is opened, and it only sends the intent (Android inter-process mechanism) to start the service that drives the mobile M2M GW.

Android services require the creation of a persistent notification, to assure that the process is not stopped. This was put to use, so that if the user presses the notification of the mobile M2M GW the application is shut down, solving the problem of having no GUI. The mobile M2M GW can also be stopped by the mobile M2M NA, by sending a specific intent that stops the application.

5.8.2 Communication

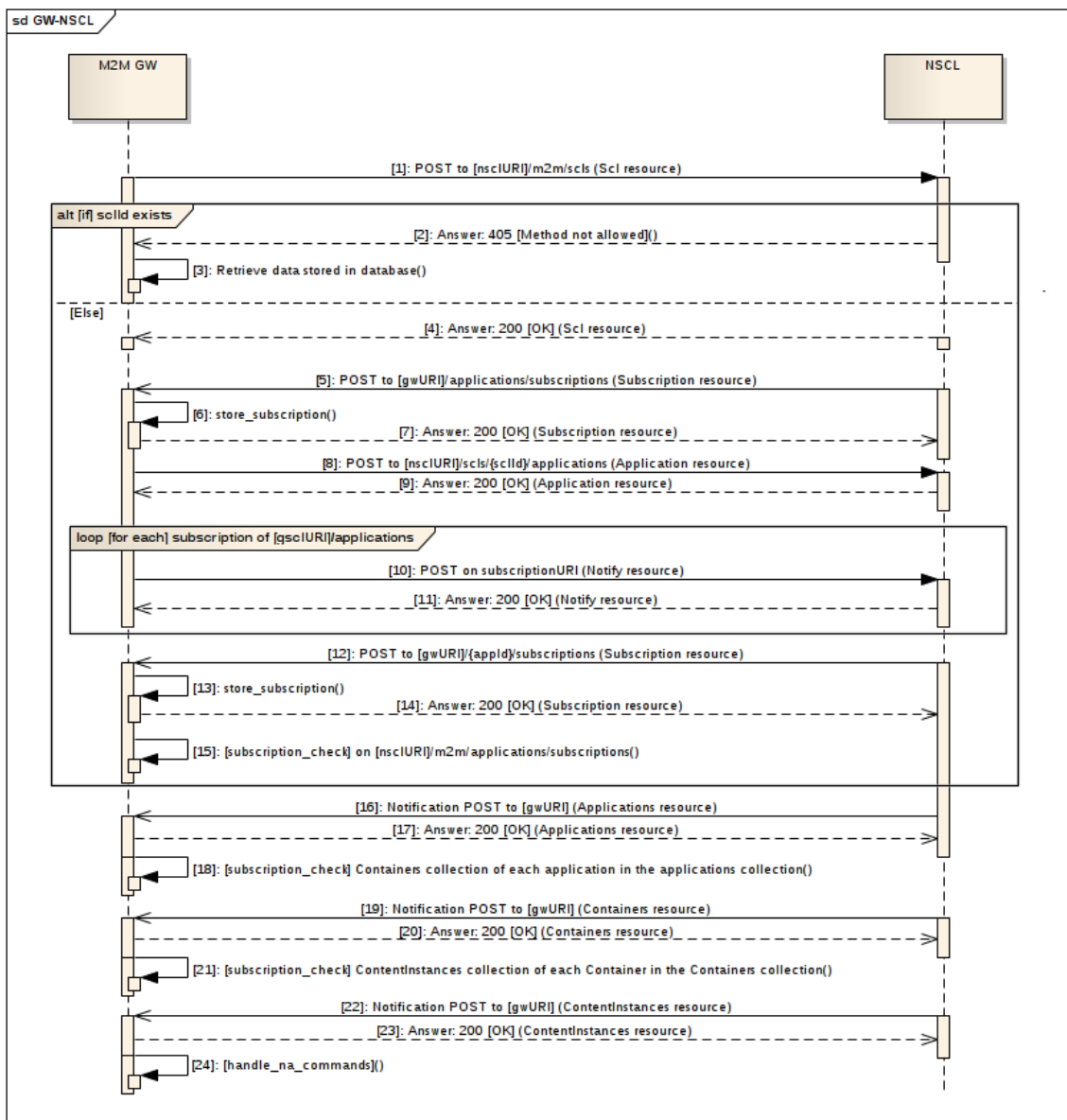


Figure 5.6: Sequence diagram of the communication held between the mobile M2M GW and the NSCL.

The sequence diagram of the communication between the mobile M2M GW and the NSCL is shown in Figure 5.6. An explanation of the messages exchanged between the mobile M2M GW and the NSCL is below, with aid of the figure's numeration.

- [1]: The first message sent by the mobile M2M GW is the POST to create an *Scl* resource (on [nslURI]/m2m/scls). The *sclId* of the mobile M2M GW references to the device model and serial number, to assure it is unique and there are no two equal IDs.
- [2 & 3]: If this smartphone was previously connected to the NSCL and the resource already exists, the NSCL returns the HTTP code 405 (Method not Allowed), specifying that it was a Bad Request. The mobile M2M GW then retrieves the data stored on the smartphone's memory (procedure previously explained in Section 5.7.2) allowing the communication to continue using the original resources, without issues.
- [4]: If the mobile M2M GW had never connected to the NSCL before or the new uniquely generated *sclId* was successfully recognized, the NSCL returns the HTTP code 200 (OK), with the complete *Scl* resource in the message body. The following messages are exchanged only if the resources have not yet been created in the NSCL, as show in Figure 5.6.
- [5]: After the successful creation of the *Scl* resource, the NSCL uses the "link" attribute to subscribe to the *Applications* collection of that resource, by sending a POST message with the *Subscription* resource to [gwURI]/applications/subscriptions. This *Subscription* is stored in the memory of the mobile M2M GW, in order to later notify the subscribers, and the next step of the its registration is triggered.
- [6, 7 & 8]: The mobile M2M GW stores the *Subscription* resource on the *Applications* collection and returns the HTTP code 200 (OK) back to the NSCL, with the resource received in the message body, as required by the ETSI standard. It then registers its *Application* resources (one or more) on the NSCL *Applications* collection, by sending a POST message to [nslURI]/scls/[sclId]/applications, with the attributes needed to successfully create the application in the NSCL.
- [9]: The NSCL creates the *Application* resource locally, and generates the its specific attributes, returning the HTTP code 200 (OK), with the created resource marshalled in the message body.
- [10 & 11]: The mobile M2M GW stores the *Application* resource registered in [9] locally, and notifies every subscriber of the *Applications* collection sending a POST request with the Notify resource to the specific *subscriptionURI*, which is an attribute of the *Subscription* resource. The NSCL then returns the HTTP code 200 (OK) for the Notification of each subscriber, with the resource created in the message body.
- [12]: The NSCL subscribes each *Application* resource registered with a POST request to [gwURI]/appId/subscriptions with a *Subscription* resource in the message body.
- [13 & 14]: Stores the received subscriptions of the *Application* resources and returns the HTTP code 200 (OK), with the *Subscription* resource in the body.

- [15]: Then, the mobile M2M GW executes a *Subscription Check* procedure (described in Section 5.7.1) to check if the current Scl (with the same URI) has already subscribed to the [nscURI]/m2m/applications/subscriptions collection, which holds the subscriptions the *Applications* collection.
- [16 & 17]: The NSCL notifies the mobile M2M GW about changes on the mobile M2M NA *Applications* collection, sending a POST message with the marshalled resource in the body. From this point on, the mobile M2M GW behavior is the same for either situation, if it was just registered in the NSCL, or if it retrieved the information from storage. The mobile M2M GW then returns the HTTP code 200 (OK), with the received resource in the body, as required by the ETSI standard.
- [18]: The mobile M2M GW executes a *Subscription Check* procedure on the *Containers* collection of each *Application* resource referenced in the received *Applications* collection.
- [19 & 20]: The NSCL notifies the mobile M2M GW about changes on the mobile M2M NA *Containers* collection, sending a POST message with the marshalled collection in the body. The mobile M2M GW then returns the HTTP code 200 (OK), with the received resource in the body, as required by the ETSI standard.
- [21]: The mobile M2M GW executes a *Subscription Check* procedure on the *ContentInstance* collection of each *Container* resource referenced in the received *Applications* collection.
- [22, 23 & 24]: The NSCL notifies the mobile M2M GW about changes on the mobile M2M NA *ContentInstances* collection, sending a POST message with the marshalled collection in the body. The mobile M2M GW then returns the HTTP code 200 (OK), with the received resource in the body, as required by the ETSI standard. The *ContentInstance* resources present within the collection are the specific commands that trigger actions in the mobile M2M GW. The handling of these commands is explained below, in Section 5.8.3.

5.8.3 Commands handling

Once the mobile M2M NA's *ContentInstances* collection is properly subscribed, the mobile M2M GW receives the notification for its changes. Given the mapping of the ETSI resources, those changes (triggered by the creation of new *ContentInstance* resources) contain the commands that control the mobile M2M GW's actions. This interaction is illustrated in Figure 5.7.

After parsing the received command as explained in Section 5.7.3, the 3 types of commands have a specific trigger:

Search Sensors If the command is to search sensors, the mobile M2M GW first decodes if the mobile M2M NA wants the internal, Bluetooth or both. Then, it triggers the search accordingly. On this dissertation's healthcare scenario, only the Bluetooth sensors are being searched. For each supported sensor found, the mobile M2M GW creates a new *Container* resource on the NSCL, with a unique *Id* with a timestamp as well as the name of the sensor.

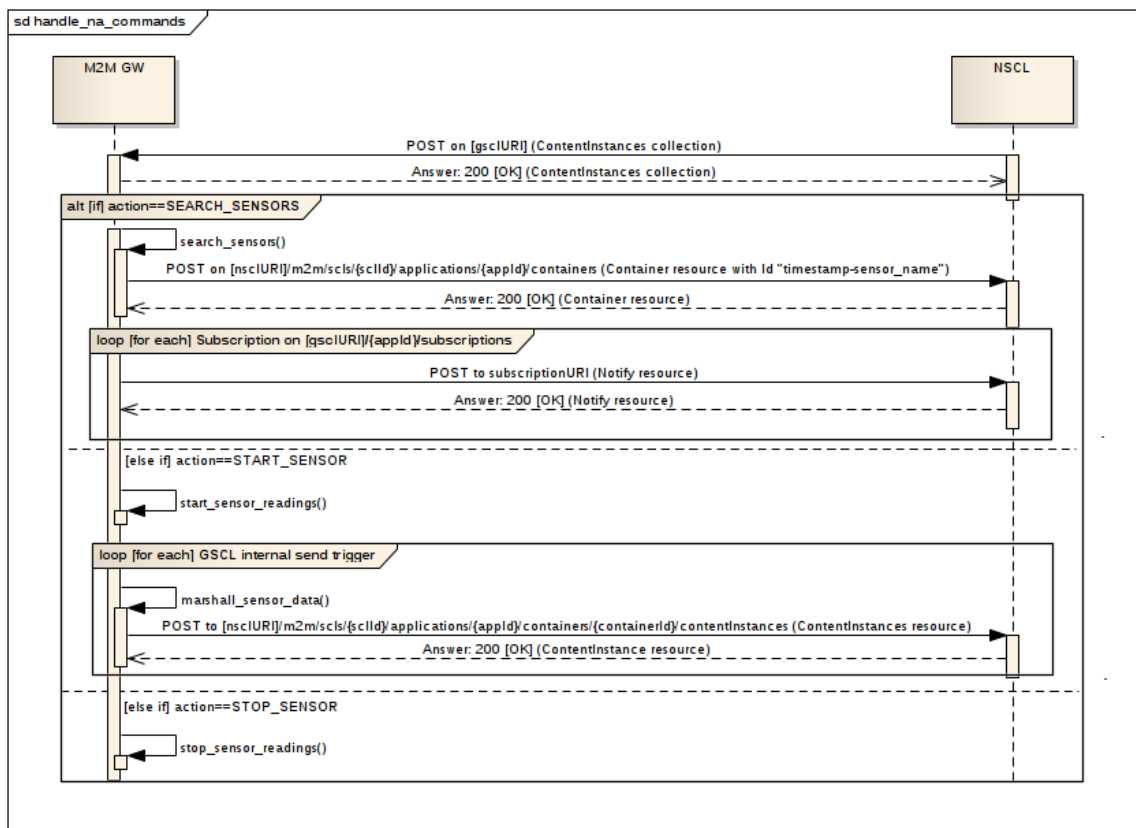


Figure 5.7: Mobile M2M GW handling of commands received from the NSCL

This was done so that the mobile M2M NA has the ability to triage which sensors were found recently and are still able to be acted upon. For this dissertation's purpose, the sensor command is considered valid for up to 2 minutes, which is maximum time for the mobile M2M NA to receive the *ContentInstance* notification. The two minutes were arbitrarily chosen, just like on the *Container* life span. For each Subscription to [gwURI]/appId/subscriptions stored in the mobile M2M GW's memory, a *Notify* resource is delivered.

Start Sensor If the command is to start a specific sensor, the mobile M2M GW first decodes which *Container* it refers to. It starts the measurements procedures, starting the appropriate memory buffers and threads related to that sensor. For as long as the sensor stays turned on, its data keeps being flushed to the NSCL, either by reaching the maximum send granularity, or by surpassing the maximum buffer size allowed for that sensor, as explained in Section 4.2.4.1.

Stop Sensor If the command is to stop a specific sensor, after decoding which one, besides stopping the measurements, the memory buffers related to it are cleaned, and its thread is also stopped.

5.9 Mobile M2M NA

This section explains how the mobile M2M NA was implemented in Android OS, in Section 5.9.1, followed by the communication specifications in Section 5.9.2. Then, how the interface impacts the information flow is detailed in Section 5.9.3.

5.9.1 Android Implementation

Following the design of the mobile M2M NA discussed in Section 4.2.4.2, the application required a user interface that was simple to use, but gave the user the necessary information at any time. Figure 5.8 shows the start-up interface of the application. It was designed to be clear when it is possible to use the local interface connections, graying out the check-box when it is not installed in the same smartphone as it's clear from the comparison between Figures 5.8a and 5.8b.

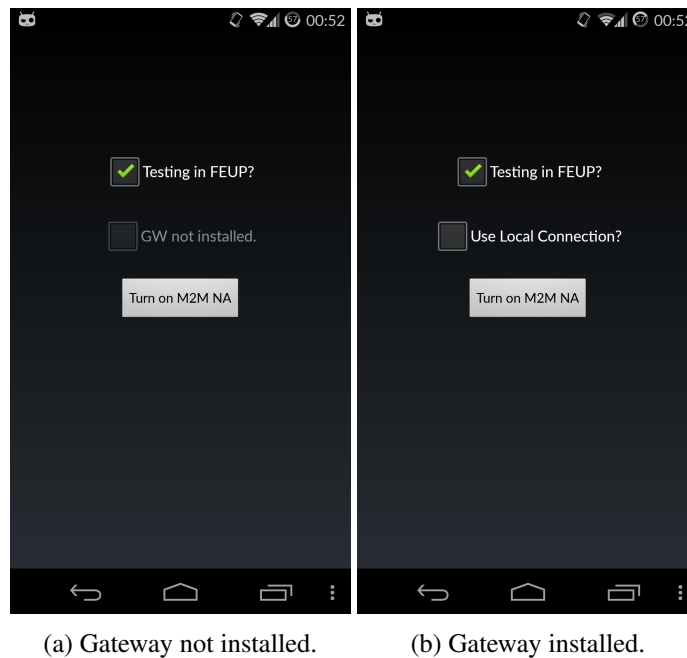
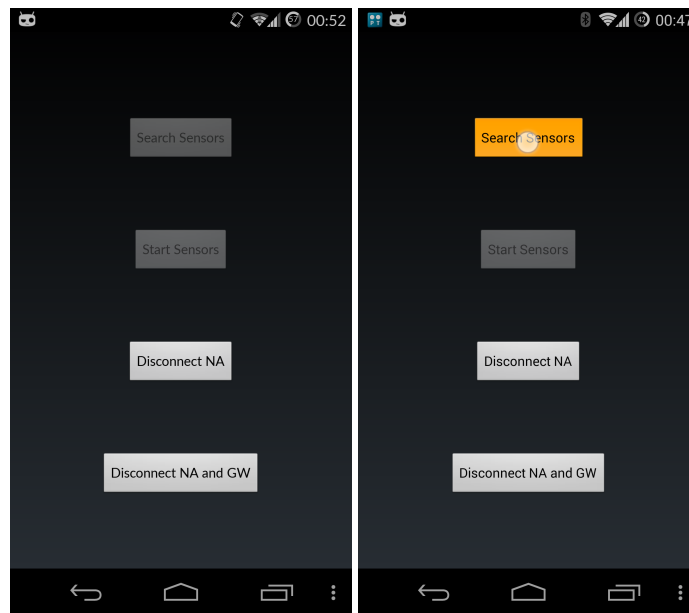


Figure 5.8: Network Application start screen.

After starting the Network Application, the commands interface appears. Initially, the search button is grayed out, as shown in Figure 5.9a, but after the NA is correctly registered in the NSCL, or it successfully retrieves the data from the storage, it becomes clickable, as shown in Figure 5.9b. It is irrelevant to the user if the local interface was chosen, since the Network Application works exactly the same with either mode, being fully transparent.

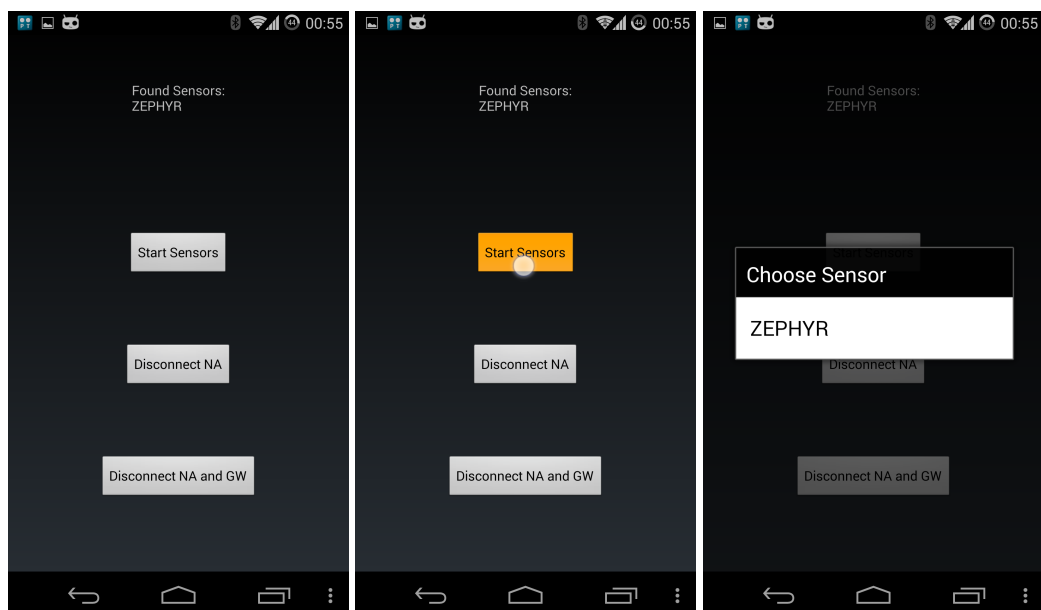
After pressing the search button, it disappears, giving place to the list of sensors found. Figure 5.10a shows this list with the ZEPHYR sensor shown, being found and registered by the Gateway. In Figure 5.10b the start sensor button is being pressed, triggering the dialog to choose from the available sensors is shown to the user, as illustrated in Figure 5.10c. In this test only



(a) Search button disabled (b) Search button pressed.

Figure 5.9: Network Application commands screen.

the Zephyr sensor was found, so it was the only one available on the list, but the implementation already supports multiple sensors.

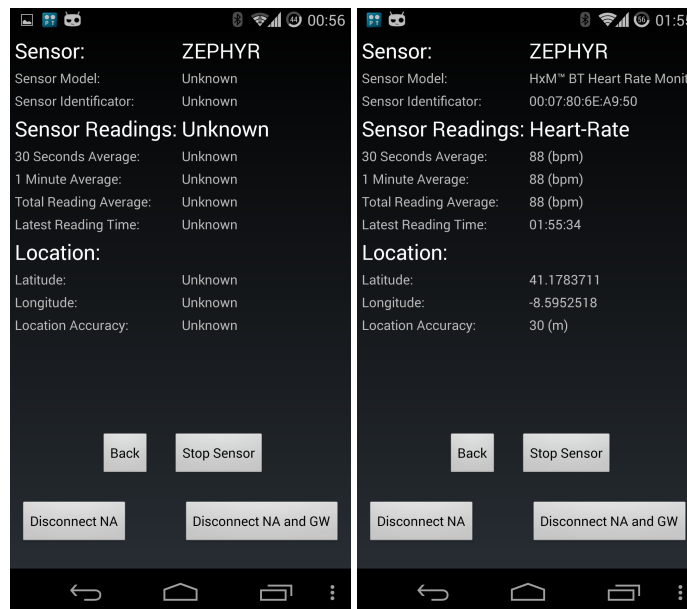


(a) Displaying found sensors. (b) Start Sensor button pressed. (c) Choose sensor dialog.

Figure 5.10: Network Application commands screen with received sensors.

After choosing the sensor from the dialog, the interface changes to the one depicted in Figure 5.11, which shows some statistics on the running sensor. Right after starting the sensor, when no information has been received yet, the interface is filled with "Unknown" strings everywhere

except the sensor name, which is the only thing known at the time, as shown in Figure 5.11a. Then, after the readings start being received and processed, the interface starts getting updated, as shown in Figure 5.11b. This was created as a demo interface, to show that the information was received properly, and to be shown at a live demo to be held in *PTIN* in the near future, where this dissertation's results will be shown.



(a) No data received yet. (b) After receiving some data.

Figure 5.11: Network Application measurements interface.

5.9.2 Communication

The messages exchanged between the mobile M2M NA and the NSCL, shown in Figure 5.12 are detailed below, with aid of the figure's numeration.

[1]: The first message sent by the mobile M2M NA is the POST request to create an *Application* resource on `[nscURI]/m2m/applications`. The *Id* attributed to mobile M2M NA is, just as with the mobile M2M GW, derived from the device model and serial number, assuring there are no two equal IDs.

[2 & 3]: If the mobile M2M NA on the smartphone was previously connected to the NSCL and the resource already exists, the NSCL returns the HTTP code 405 (Method not Allowed), specifying that it was a Bad Request. The mobile M2M GW then retrieves the data stored on the smartphone's memory (procedure previously explained in Section 5.7.2) allowing the communication to continue using the original resources, without issues.

[4]: If the mobile M2M NA had never connected to the NSCL before or the new uniquely generated *Id* was successfully recognized, the NSCL returns the HTTP code 200 (OK), with the

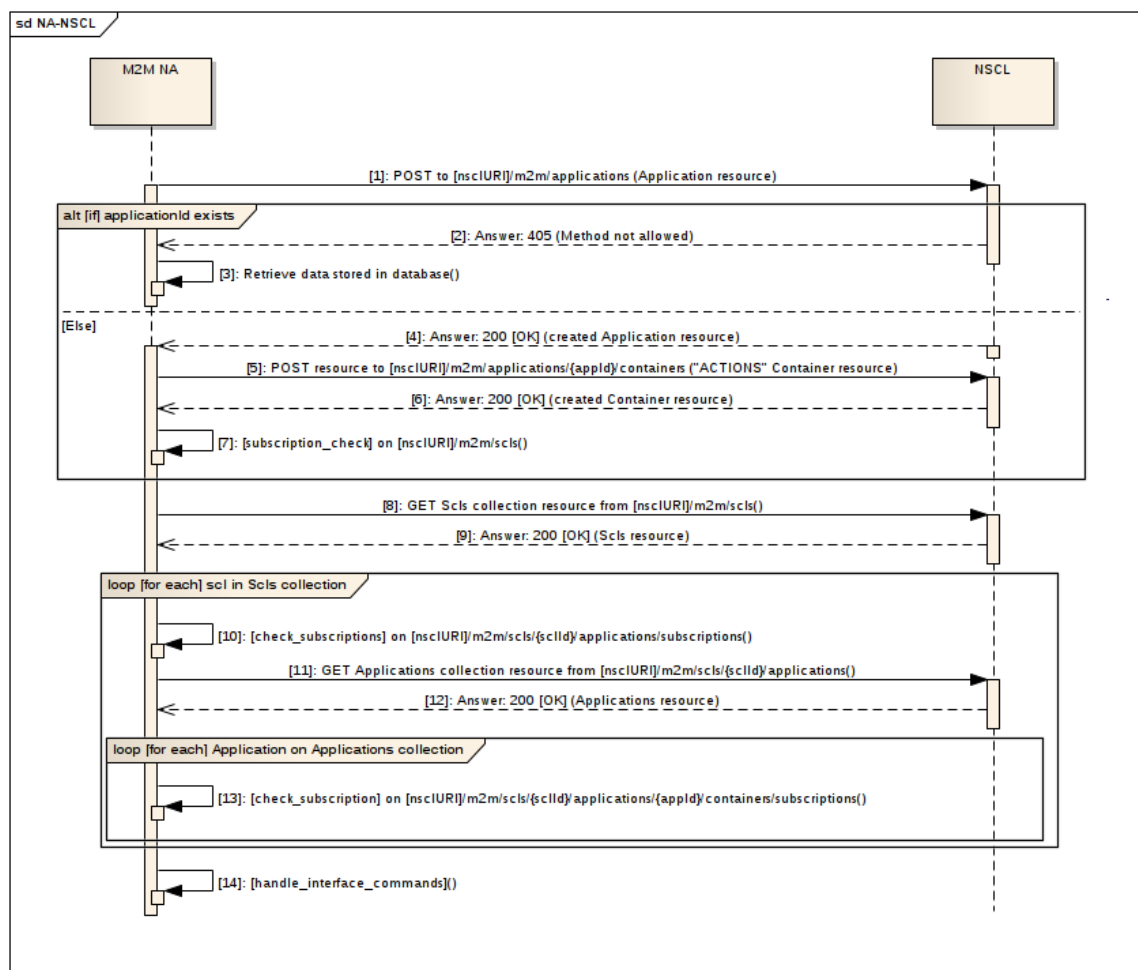


Figure 5.12: Sequence diagram of the communication held between the mobile M2M NA and the NSCL.

complete *Application* resource in the message body. The following messages are exchanged only if the resources have not yet been created in the NSCL, as show in Figure 5.12.

- [5]: The mobile M2M NA then registers its *ACTIONS Container* resource on the NSCL *Containers* collection, by sending a POST message to [nscIURI]/applications/[Id]/containers, with the attributes needed to successfully create the resource in the NSCL.
- [6]: The NSCL creates the *Container* resource locally, and generates its specific attributes, before returning the HTTP code 200 (OK) with the created resource marshalled in the message body.
- [7]: Then, the mobile M2M NA executes a *Subscription Check* procedure, checking if the current mobile M2M NA (with the same URI) has already subscribed to the [nscIURI]/m2m/scls collection, which holds the mobile M2M GW's *Scl* resources.
- [8 & 9]: The mobile M2M NA then retrieves the *Scls* collection by sending a GET request to [nscIURI]/m2m/scls. Communications posterior to this message yield the same behavior in

the mobile M2M NA, since it now has the needed resources active in memory. The NSCL returns the HTTP code 200 (OK) with *Scls* collection in the body, which contains the Id and URI of every stored *Scl* resource.

[10, 11 & 12]: The mobile M2M NA then executes the *Subscription Check* procedure on the *Applications* collection on of every *Scl* retrieved. It also performs a retrieve request by performing a GET call to [nscIURI]/m2m/scls/scIId/applications, to retrieve the *Applications* collection and make sure its containers are subscribed. The NSCL returns the HTTP code 200 (OK) with the *Applications* collection in the message body.

[13]: The mobile M2M NA executes the *Subscription Check* procedure on [nscIURI]/m2m/scls/scIId/applications/appId/containers/subscriptions for every *Application* present in the retrieved *Applications* collection.

[14]: The mobile M2M NA then awaits for the user input, processed from the graphical user interface, and explained below, in Section 5.9.3

5.9.3 Interface Commands

The mobile M2M NA user interface flow, explained in Section 5.9.1, is complemented in this section with the details behind its logic. Once the *ACTIONS Container* is properly registered, the mobile M2M NA awaits the user input, to execute the commands. The actionable commands are shown in Figure 5.13 and explained below. The user ultimately has three possible courses of actions, even though only one is active at each point in time. Each one produces the following effect on the communication:

Search Sensors When the user clicks the *search sensors* button, an object of the *ActionSearchSensors* (explained in Section 5.7.3) is created, requiring only the search of external sensors (as the only sensor integrated is external). This object is then marshalled into a *ContentInstance* resource and registered in the NSCL on the "Actions" *Container* previously created. After the mobile M2M GW's internal procedures to search sensors, it registers a *Container* resource, which is delivered to the mobile M2M NA in the body of a notification, triggered by the registration of the *ContentInstance*. The mobile M2M NA then uses the found sensors which are still valid (found less than 2 minutes ago) and shows them in the Commands interface.

Start Sensor Referring to Figure 5.10, after the user clicks the *start sensor* button he has the ability to choose an available sensor. After this choice, a *ActionStartSensor* class is created, and the sensor chosen is set as the attribute that signals the *Container* resource to start. Then this object is marshalled into a *ContentInstance* resource and, once again, registered under this mobile M2M NA's "Actions" *Container*. In addition to the delivery of the specific command, the mobile M2M NA also uses this opportunity to subscribe to the *ContentInstances* collection of the *Container* that was triggered to start. This happens so that the mobile

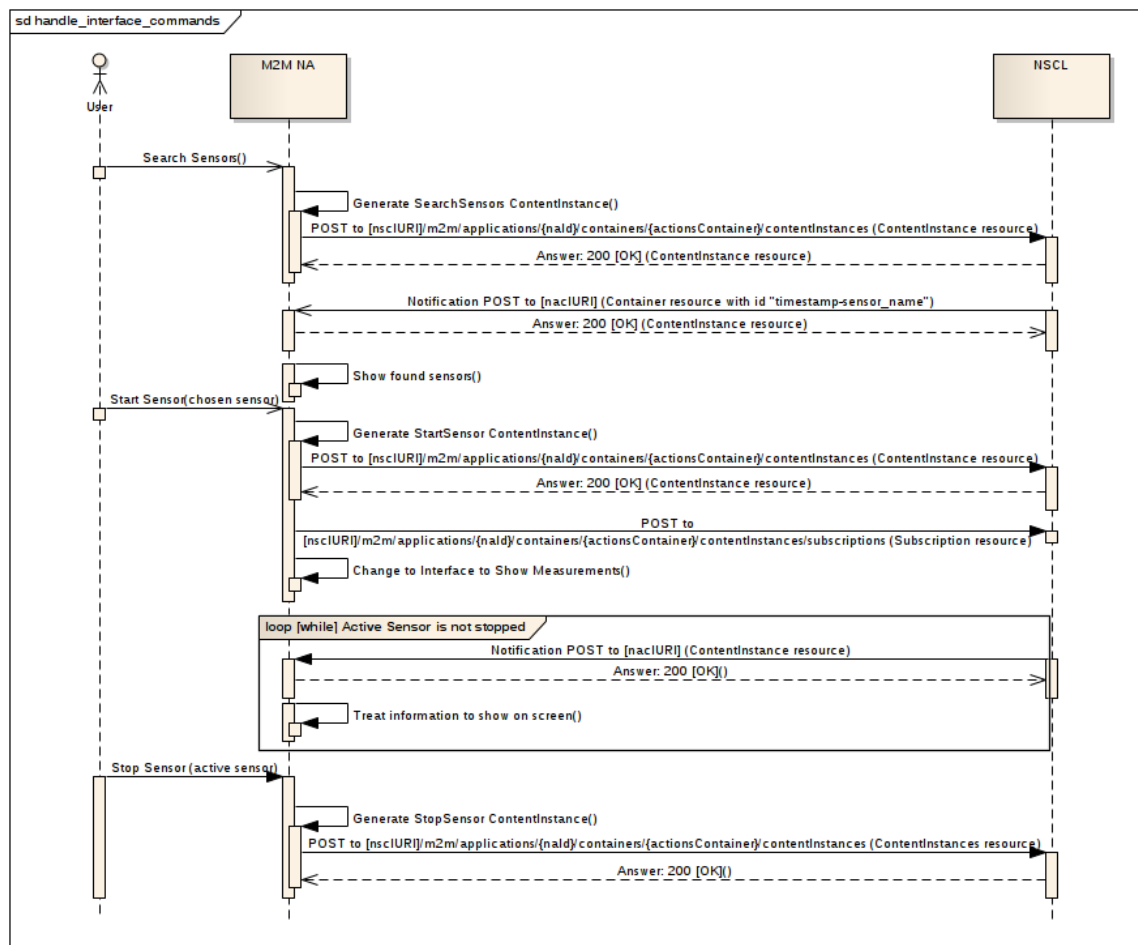


Figure 5.13: Sequence diagram of the mobile M2M NA commands.

M2M NA does not receive any sensor information ahead of time, before it was requested by its own command. The last procedure to do in this situation, is to change the active user interface to show the measurements received. The mobile M2M NA then awaits for the *Notification* POST messages with the sensor information, for further treatment to be shown in the interface. This process is repeated until the sensor is stopped or the mobile M2M GW and/or mobile M2M NA is shut down.

Stop Sensor After the user presses the *stop sensor* button, the active *Container*'s name is used to create the *ActionStopSensor* object. This is then marshalled into a *ContentInstance* resource and registered into the "Actions" *Container*. The mobile M2M NA's GUI re-enables the *search sensor* button to allow the user to re-search for sensors and possibly start a different one.

5.10 Local Interface

The local interface concept, explained in Section ??, aims to reduce the amount of traffic used when the mobile M2M GW is in the same smartphone as the mobile M2M NA, and this Section will detail this implementation, with aid of Figure 5.14. Given the presence of HTTP web-servers on both applications, the obvious way to communicate between them was to take advantage of that feature, using *localhost* as a *fake* NSCL URI. This communication is currently unencrypted since the web-servers are not working over TLS yet, for the reasons mentioned in Section 5.4. The local interface was built as shown in Figure 5.14, and its communication interactions are detailed below:

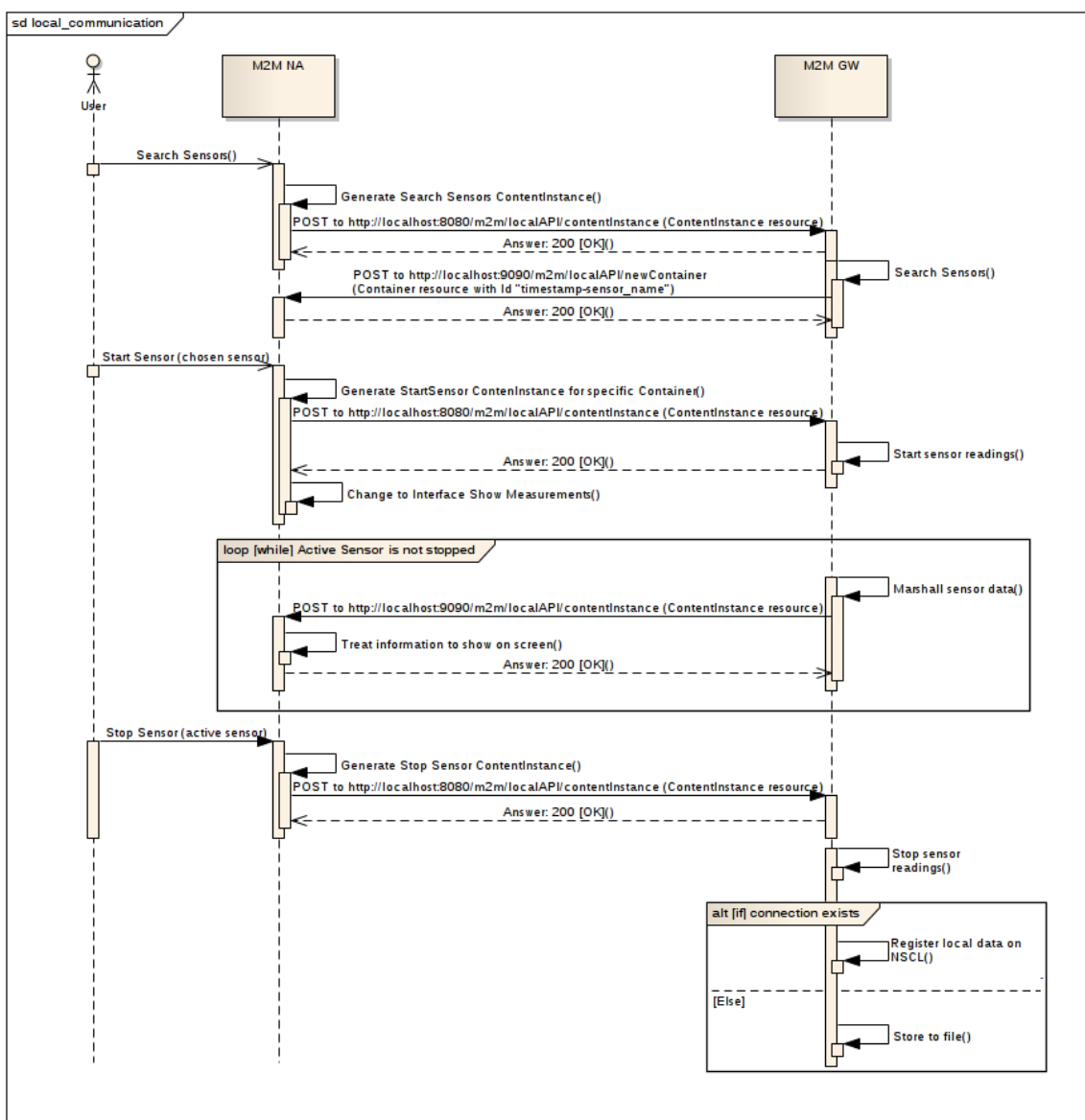


Figure 5.14: Sequence diagram of the communication between the mobile M2M GW and NA using the local interface.

Search Sensors When the user presses the *search sensors* button on the mobile M2M NA, it builds the *ActionSearchSensors ContentInstance* just as if it was going to communicate with the NSCL. But instead it sends a POST message to `http://localhost:[gwPort]/m2m/localAPI/contentInstance`. The mobile M2M GW returns the HTTP code 200 (OK) and starts the search sensors procedure. Then just as it would communicate with the NSCL, it registers a *Container* resource for each sensor found, with the same timestamp in the name as normally, on:

`http://localhost:[naPort]/m2m/localAPI/newContainer`. The mobile M2M NA returns the HTTP code 200 (OK) for each *Container* resource received, completing the cycle of internally searching for sensors without recurring to the NSCL.

Start Sensor When the user clicks the *start sensor* button in the mobile M2M NA's command interface, the same *ContentInstance* with the marshalled *ActionStartSensor* is created, and registered on `http://localhost:[gwPort]/m2m/localAPI/contentInstance`. The mobile M2M GW starts the procedures to start the measurements from that sensor, and returns the HTTP code 200 (OK) afterward. In the localAPI the mobile M2M NA does not need to subscribe to the *ContentInstances* collection since the delivery of the command acts as subscription. The mobile M2M NA then changes its interface to show the measurements, just as it would normally. The delivery of sensor data is executed on:

`http://localhost:[naPort]/m2m/localAPI/contentInstance` with the sensor data marshalled into a *ContentInstance* resource. The mobile M2M NA then treats the information to show on screen and answers the HTTP code 200 (OK) back to the mobile M2M GW. Once again, the sensor stays on until it is explicitly stopped.

Stop Sensor When the user clicks the *stop sensor* button, the mobile M2M NA registers a *ContentInstance* containing the marshalled *ActionStopSensor* object on:

`http://localhost:[gwPort]/m2m/localAPI/contentInstance`. Besides triggering the stoppage of sensor readings, this command over the local interface also triggers the delivery of all the sensor related resources registered by the mobile M2M GW (*Container* and *ContentInstance* resources) to be delivered to the NSCL, so that no important data is lost when it is used. If there is no Internet connection at that time, a local interface storage file (detailed below) is used to store that information, which will be delivered the next time the mobile M2M GW connects to the NSCL and erased from memory then.

Local Interface Storage:

The local interface storage holds the three attributes to ensure that the information can be properly delivered to the NSCL once the mobile M2M GW restarts. It stores the *Container* resource created, a String holding the target URI of said *Container*, and an array of *ContentInstance* resources. Since the *ContentInstances* belong to the *Container*, only one URI is needed, because the URI to register these resources to is generated by adding `"/containerId/contentInstances"` to the end of it. Storing this data serialized guarantees that the important healthcare information always reaches the NSCL.

Chapter 6

Results

This chapter will present the results of the implementation presented in Chapter 5. To show the functionalities of the solution, Section 6.1 will detail the testing procedures executed, followed by the traffic measurements done to evaluate the performance of the solution with and without the local interface, in Section 6.2. Then, these results will be discussed in Section 6.2.3.

6.1 Proof of Concept

To prove that the implementation is functional, a few tests were executed and its results were gathered. These tests were executed on a LG Nexus 5 (codename hammerhead, as seen below in the logs), running Android version 4.4.4 on the custom ROM Paranoid Android, version 4.42. To execute the tests, a guide was created, to assure that the same procedure was followed when communicating through the NSCL or through the local interface. The guide is detailed below:

1. Install the mobile M2M GW and mobile M2M NA on an Android smartphone running at least version 4.2 of the OS.
2. Wear the Zephyr heart-rate monitor, and assure the smartphone has a Bluetooth pair with the device.
3. Turn on the mobile M2M NA. Since the test is done within FEUP network, the "Testing in FEUP" option is turned on, so that the mobile applications use the static IP address that has the configured routes. Choose to use (or not) the local interface.
4. Press the "Start M2M NA" button, which also starts the mobile M2M GW.
5. Await for the "Search Sensors" button to become active, and press it, to send the *search sensors* command.
6. Await for the found sensors list to arrive at the mobile M2M NA. Press *start sensor* and choose the Zephyr, which is the only available sensor. Choosing the sensor starts its measurements, which are then shown in a separate GUI. If using the local interface, during the sensor data exchange, turn the Wi-Fi off and back on, to prove that it works even without an active Internet connection.

7. Press the *stop sensor* button, stopping the measurements by delivering that command to the mobile M2M GW. After stopping the measurements, turn off both applications by pressing the button *Disconnect NA and GW*.

This testing procedure is executed twice, first communicating normally through the NSCL, and then using the local interface. The full length of the logs of the test executed through the NSCL are presented in Appendices B.1 and C.1, and for the test executed through the local interface in Appendices B.2 and C.2. To demonstrate that the communication is working, let us look at the registration of sensor data¹ (using both connections) and to the delivery of the *stop sensor* command:

- Sensor data through the NSCL:

```
— REGISTRATION ON THE MOBILE M2M GW —
(...)
07-14 02:47:25.578 8276-8305/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing
ZEPHYR.
07-14 02:47:25.628 8276-8309/pt.ptinovacao.mtom.gw D/CreateResources:
Registering contentInstance to /m2m/scIs/Demohammerhead-04506dfa25231287-
TesteFEUP/applications/HealthSensors/containers/1405302431976-ZEPHYR/
contentInstances .
07-14 02:47:25.728 8276-8333/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
(...)

— NOTIFICATION ON THE MOBILE M2M NA —
(...)
07-14 02:47:25.798 8058-8389/pt.ptinovacao.mtom.na D/TreatSensorData:
Measurement Updated. Array Size: 7
(...)
```

This example shows that almost immediately after the mobile M2M GW receives the answer that the *ContentInstance* was properly registered, the mobile M2M NA is already updating its User Interface with that information, after the parsing of the *ContentInstances* collection notification (the log signaling the receipt of this collection was omitted, since the measurement updates are sufficient as they are only triggered when that notification exists).

- Sensor data through the local interface:

```
— REGISTRATION ON THE MOBILE M2M GW —
(...)
07-14 02:51:36.298 10585-10615/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing
ZEPHYR.
07-14 02:51:36.348 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: No
Internet Connection, but local interface connected. Sending data.
07-14 02:51:36.348 10585-10614/pt.ptinovacao.mtom.gw D/CreateResources:
Registering contentInstance to /m2m/localAPI/contentInstance/1405302658817-
ZEPHYR .
(...)
07-14 02:51:36.478 10585-10641/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
(...)

— NOTIFICATION ON THE MOBILE M2M NA —
(...)
07-14 02:51:36.388 10482-10680/pt.ptinovacao.mtom.na D/HandleRequests: Local
connection: ContentInstance of Container: 1405302658817-ZEPHYR
```

¹The exchange of sensor data is only possible after the successful delivery of the *search sensors* and *start sensor* commands. For the full detail of the communications, see Appendices B and C

```
07-14 02:51:36.428 10482-10680/pt.ptinovacao.mtom.na D/TreatSensorData:
    Measurement Updated. Array Size: 25
(...)
```

This example shows that using the local interface, even when there is no Internet connection, the data is still successfully exchanged between the applications. Being an internal connection, it naturally was almost instant, and the message of a successful registration was already after the mobile M2M NA's GUI was updated.

- *Stop Sensor* through the NSCL:

```
— DELIVERY OF THE COMMAND BY THE MOBILE M2M NA —
(...)
07-14 02:47:44.688 8058-8310/pt.ptinovacao.mtom.na D/CreateResources:
    Registering contentInstance to /m2m/applications/HealthConsumerNA-hammerhead
    -04506dfa25231287-TesteFEUP/containers/ACTIONS/contentInstances .
(...)

— RECEIVAL OF THE COMMAND BY THE MOBILE M2M GW —
(...)
07-14 02:47:44.858 8276-8334/pt.ptinovacao.mtom.gw V/HandleRequests: Received
    ContentInstances collection notification.
07-14 02:47:44.878 8276-8334/pt.ptinovacao.mtom.gw V/HandleRequests: Valid
    Container to Stop command.
    Container: 1405302431976-ZEPHYR
07-14 02:47:44.878 8276-8276/pt.ptinovacao.mtom.gw D/ZephyrHandler: Stopping
    Zephyr...
(...)
```

- *Stop Sensor* through the local interface:

```
— DELIVERY OF THE COMMAND BY THE MOBILE M2M NA —
(...)
07-14 02:51:54.178 10482-10601/pt.ptinovacao.mtom.na D/CreateResources:
    Registering contentInstance to /m2m/localAPI/contentInstance .
07-14 02:51:54.628 10482-10680/pt.ptinovacao.mtom.na V/HandleRequests: Received
    Containers collection notification.
(...)
07-14 02:51:54.688 10482-10601/pt.ptinovacao.mtom.na I/SCL: Container
    1405302658817-ZEPHYR is now active.
(...)

— RECEIVAL OF THE COMMAND BY THE MOBILE M2M GW —
(...)
07-14 02:51:54.218 10585-10642/pt.ptinovacao.mtom.gw D/HandleRequests: Local
    connection. Received Command.
07-14 02:51:54.218 10585-10642/pt.ptinovacao.mtom.gw V/HandleRequests: Valid
    Container to Stop command.
    Container: 1405302658817-ZEPHYR
07-14 02:51:54.218 10585-10585/pt.ptinovacao.mtom.gw D/ZephyrHandler: Stopping
    Zephyr...
07-14 02:51:54.228 10585-10614/pt.ptinovacao.mtom.gw I/CreateResources:
    Registering container: 1405302658817-ZEPHYR to: /m2m/scls/hammerhead-04506
    dfa25231287-TesteFEUP/applications/HealthSensors/containers .
07-14 02:51:54.228 10585-10614/pt.ptinovacao.mtom.gw D/CreateResources:
    Registering contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
    applications/HealthSensors/containers/1405302658817-ZEPHYR/contentInstances .
07-14 02:51:54.228 10585-10614/pt.ptinovacao.mtom.gw D/CreateResources:
    Registering contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
    applications/HealthSensors/containers/1405302658817-ZEPHYR/contentInstances .
(...)
07-14 02:51:54.718 10585-10641/pt.ptinovacao.mtom.gw D/CreateResources: Content
    Instance successfully registered.
07-14 02:51:54.718 10585-10614/pt.ptinovacao.mtom.gw I/GSCL: Local Container
    1405302658817-ZEPHYR created successfully.
```

```
07-14 02:51:54.858 10585-10641/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
(...)
```

This excerpt of the logs, show that the delivery of the *stop sensor* command triggered the registration of the data created locally, and that the notification related to this registration was immediately received by the mobile M2M NA.

To evaluate the ability of the mobile M2M GW to properly use the designed databases for mobile scenarios where the connection drops, three more specific tests were designed:

1. When communicating through the NSCL, turn off the Wi-Fi connection while the sensor is actively sending data, checking if that data is properly stored. Then, turn the connection back on and check if the data was properly delivered.
2. When communicating through the NSCL, turn off the Wi-Fi connection while the sensor is actively sending data, but follow the connection drop by turning off both application. Then, turn the connectivity and the mobile M2M GW back on, in order to evaluate if the data is properly registered in the NSCL.
3. When using the local interface with the sensor running, shut down both application, causing the data sent locally to be stored in the mobile M2M GW's database. Then, turn the connectivity and the mobile M2M GW application back on, in order to evaluate if the data is properly registered in the NSCL.

Tests 1. and 2. were executed sequentially in a "lost connectivity" test, and their results are shown in Appendices B.3 and C.3, while the results for the local interface connectivity test are presented in Appendices B.4 and C.4. To demonstrate that these mobile scenarios are working, let us look at the logs for each case:

- *Scenario 1.:*

```
— CONNECTIVITY LOST, ON THE MOBILE M2M GW —
(...)
07-14 03:57:19.428 32257-32257/pt.ptinovacao.mtom.gw W/ConnectionListener: Network
connectivity change
07-14 03:57:19.438 32257-32257/pt.ptinovacao.mtom.gw D/ConnectionChecker: No
network connectivity
07-14 03:57:19.438 32257-32257/pt.ptinovacao.mtom.gw D/ProtocolManager: No
connection. Resetting stored IPV4 address.
(...)
07-14 03:57:23.378 32257-32276/pt.ptinovacao.mtom.gw I/ProtocolManager: No
Internet Connection: Can't send data. Saving to Database.
(...)

— CONNECTIVITY REGAINED, ON THE MOBILE M2M GW —
(...)
07-14 03:57:44.698 32257-32257/pt.ptinovacao.mtom.gw W/ConnectionListener: Network
connectivity change
07-14 03:57:44.708 32257-32257/pt.ptinovacao.mtom.gw I/ConnectionChecker: Network
WIFI connected
07-14 03:57:44.708 32257-32276/pt.ptinovacao.mtom.gw I/ProtocolManager: Connection
returned, reading stored data... Size: 2386
07-14 03:57:44.708 32257-32276/pt.ptinovacao.mtom.gw V/ProtocolManager: Reached
end of file.
```

```

07-14 03:57:44.718 32257-32276/pt.ptinovacao.mtom.gw V/ProtocolManager: File
    successfully erased.
07-14 03:57:44.718 32257-32279/pt.ptinovacao.mtom.gw D/CreateResources:
    Registering contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
    applications/HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .
07-14 03:57:44.718 32257-32279/pt.ptinovacao.mtom.gw D/CreateResources:
    Registering contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
    applications/HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .
07-14 03:57:44.828 32257-32348/pt.ptinovacao.mtom.gw D/CreateResources: Content
    Instance successfully registered.
07-14 03:57:44.958 32257-32348/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC
    freed 308K, 20% free 3469K/4328K, paused 10ms, total 10ms
07-14 03:57:44.958 32257-32348/pt.ptinovacao.mtom.gw D/CreateResources: Content
    Instance successfully registered.
(...)

--- CONNECTIVITY REGAINED, ON THE MOBILE M2M NA ---
(...)
07-14 03:57:44.688 32041-32041/pt.ptinovacao.mtom.na W/ConnectionListener: Network
    connectivity change
07-14 03:57:44.698 32041-32041/pt.ptinovacao.mtom.na I/ConnectionChecker: Network
    WIFI connected
07-14 03:57:44.698 32041-32272/pt.ptinovacao.mtom.na D/InternetManager:NA_SCL
    already running.
07-14 03:57:44.908 32041-32449/pt.ptinovacao.mtom.na D/TreatSensorData:
    Measurement Updated. Array Size: 25
07-14 03:57:45.118 32041-32449/pt.ptinovacao.mtom.na D/TreatSensorData:
    Measurement Updated. Array Size: 34
07-14 03:57:53.558 32041-32449/pt.ptinovacao.mtom.na D/TreatSensorData:
    Measurement Updated. Array Size: 44
(...)

```

These logs show that when the connectivity is lost, the mobile M2M GW stores the sensor information on the local database, and that when it is regained, the data is properly retrieved and registered on the NSCL.

- *Scenario 2.:*

```

--- MOBILE M2M GW LOSING CONNECTIVITY AND BEING TURNED OFF ---
(...)
07-14 03:57:59.078 32257-32257/pt.ptinovacao.mtom.gw W/ConnectionListener: Network
    connectivity change
07-14 03:57:59.088 32257-32257/pt.ptinovacao.mtom.gw D/ConnectionChecker: No
    network connectivity
07-14 03:57:59.088 32257-32257/pt.ptinovacao.mtom.gw D/ProtocolManager: No
    connection. Resetting stored IPV4 address.
07-14 03:58:03.368 32257-32280/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing
    ZEPHYR.
07-14 03:58:03.398 32257-32276/pt.ptinovacao.mtom.gw I/ProtocolManager: No
    Internet Connection: Can't send data. Saving to Database.
(...)
07-14 03:58:13.378 32257-32280/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing
    ZEPHYR.
07-14 03:58:13.458 32257-32276/pt.ptinovacao.mtom.gw I/ProtocolManager: No
    Internet Connection: Can't send data. Saving to Database.
(...)
07-14 03:58:14.698 32257-32257/pt.ptinovacao.mtom.gw D/MainService: M2M Gateway
    Stopping
(...)

--- MOBILE M2M GW RESTART AFTER LOSING CONNECTIVITY ---
(...)
07-14 03:59:12.938 1155-1890/pt.ptinovacao.mtom.gw I/ProtocolManager: Reading
    stored data... Size: 2395
07-14 03:59:12.988 1155-1890/pt.ptinovacao.mtom.gw V/ProtocolManager: Reached
    end of file.

```

```

07-14 03:59:12.998    1155-1890/pt.ptinovacao.mtom.gw V/ProtocolManager: File
    successfully erased.
(...)
07-14 03:59:13.298    1155-1890/pt.ptinovacao.mtom.gw D/CreateResources:
    Registering contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
    applications/HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .
07-14 03:59:13.308    1155-1890/pt.ptinovacao.mtom.gw D/CreateResources:
    Registering contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
    applications/HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .
(...)
07-14 03:59:14.178    1155-2126/pt.ptinovacao.mtom.gw D/CreateResources: Content
    Instance successfully registered.
(...)
07-14 03:59:14.378    1155-2126/pt.ptinovacao.mtom.gw D/CreateResources: Content
    Instance successfully registered.
(...)

```

These logs show that the mobile M2M GW properly stores the data that is not able to deliver after connectivity is lost, and that when it is restarted, and has proper connectivity, that data is read and successfully delivered to the NSCL.

- *Scenario 3.:*

```

— MOBILE M2M GW BEING DISCONNECTED WITH LOCAL API AND RUNNING SENSOR —
(...)
07-14 03:34:38.218    25740-25769/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing
    ZEPHYR.
07-14 03:34:38.238    25740-25773/pt.ptinovacao.mtom.gw D/CreateResources:
    Registering contentInstance to /m2m/localAPI/contentInstance/1405305242955-
    ZEPHYR .
07-14 03:34:38.348    25740-25797/pt.ptinovacao.mtom.gw D/CreateResources: Content
    Instance successfully registered.
(...)
07-14 03:34:42.448    25740-25740/pt.ptinovacao.mtom.gw D/MainService: M2M Gateway
    Stopping
(...)
07-14 03:34:42.448    25740-25740/pt.ptinovacao.mtom.gw D/GSCL: GSCL Stopping
07-14 03:34:42.528    25740-25740/pt.ptinovacao.mtom.gw D/GSCL: Successfully saved
    state to storage.
07-14 03:34:42.578    25740-25740/pt.ptinovacao.mtom.gw D/GSCL: Successfully saved
    local interface data to storage.
(...)

— MOBILE M2M GW RESTART AFTER DISCONNECTION —
07-14 03:36:20.948    26363-26763/pt.ptinovacao.mtom.gw D/GSCL: Memory database
    successfully read!
07-14 03:36:21.008    26363-26763/pt.ptinovacao.mtom.gw V/GSCL: Local Interface file
    successfully erased.
(...)
07-14 03:36:21.358    26363-26763/pt.ptinovacao.mtom.gw I/CreateResources:
    Registering container: 1405305242955-ZEPHYR to: /m2m/scls/hammerhead-04506
    dfa25231287-TesteFEUP/applications/HealthSensors/containers .
07-14 03:36:21.528    26363-26763/pt.ptinovacao.mtom.gw D/CreateResources:
    Registering contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
    applications/HealthSensors/containers/1405305242955-ZEPHYR/contentInstances .
07-14 03:36:21.548    26363-26763/pt.ptinovacao.mtom.gw D/CreateResources:
    Registering contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
    applications/HealthSensors/containers/1405305242955-ZEPHYR/contentInstances .
(...)
07-14 03:36:22.468    26363-26829/pt.ptinovacao.mtom.gw D/CreateResources: Content
    Instance successfully registered.
07-14 03:36:22.558    26363-26763/pt.ptinovacao.mtom.gw I/GSCL: Local Container
    1405305242955-ZEPHYR created successfully.
07-14 03:36:22.668    26363-26829/pt.ptinovacao.mtom.gw D/CreateResources: Content
    Instance successfully registered.
(...)

```

These logs show that when the local interface is being used, and the sensor is not properly stopped before the application is turned off, the data is properly stored in the database. After the application is restarted, that data is then read and successfully registered, assuring that no data is lost when using the local interface.

The size and performance parameters of the applications is detailed in Table 6.1. The application size refers to the memory occupied by the application's installation on the smartphone, and the maximum and average memory used refer to the RAM memory. This RAM memory information was gathered from the logs in Appendix B and C, since the Java programming calls the garbage collector (GC) when memory is not being used, showing the current memory in use as well as the maximum memory up to that time.

Type	Mobile M2M GW	Mobile M2M NA
Application Size	4350Kb	3600Kb
Maximum Memory Used	4328Kb	8872Kb
Average Memory Used	3424Kb	4846Kb

Table 6.1: Mobile M2M applications footprint.

6.2 Traffic Measurements

To test the bandwidth used by both applications, when communicating through the NSCL or through the local interface, some traffic measurements were executed. Given the complexity of executing battery tests, associated with the shortage of time to do them, a direct evaluation was not possible. However, given the fact that these are mobile applications, the sheer use of the 3G communications uses a great deal of power, as shown in Figure 6.1². Even though the tests executed to test the concept were done through Wi-Fi, in real scenarios, 3G/4G data would possibly play a bigger role in mobile M2M scenarios, making traffic savings significant in terms of battery.

To perform this measurements, a traffic monitor Android applications was first tried. However, all of them rely on Android's *TrafficStats* API, which counts localhost as regular network traffic, not fitting the requirements for this dissertation's measurements. A Wireshark [84] examination was also attempted, but since the communication is encrypted end-to-end, there was no way to distinguish which data was being sent by which application. This led to the programmatically creation of counters on the HTTP client and server classes on both the mobile M2M GW and the mobile M2M NA, differentiating when traffic is inbound or outbound, as well as internal and external. These counters were done by counting the characters of the outgoing and incoming

²Abbreviations:

3G = Third Generation;

GSM = Groupe Special Mobile;

Wi-Fi + SA = Wi-Fi with scan and transfer.

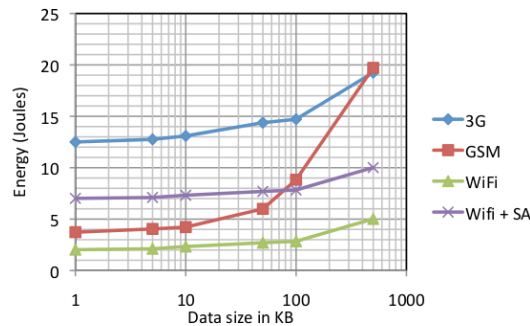


Figure 6.1: Battery test studies. From [9].

HTTP traffic, as shown below. These cannot account for the headers and information created by the Apache HTTP library used, but a very close estimate is acquired.

```

— POST SIZE —
postSize += post.getEntity().getContentLength();
postSize += post.getRequestLine().toString().length();
postSize += post.getEntity().getContentType().toString().length();

— HEADERS SIZE —
for(int i = 0; i<headArray.length;i++){
    headerSize += (long) headArray[i].toString().getBytes().length;
}

```

Then, there were two important tests to be performed. Firstly, there was the need to evaluate the amount of traffic generated during the resource creation, Section 6.2.1 addresses this case, as well as evaluate the performance of the local interface in the amount of data saved, which is detailed in Section 6.2.2.

6.2.1 Resource Registration

During the implementation, the increase in traffic, when new resources were created, was evident. Accordingly, a study where the traffic was measured when simulating several smartphones connecting was important. To do so, the unique *sclId* of the mobile M2M Gateway, as well as the *applId* of the mobile M2M NA, were purposefully altered. This aimed to simulate the first time each device connects to the NSCL, creating its resources, and subscribing the existing resources if needed, as explained in the *Subscription Check* procedure.

The test executed consisted simply of turning both applications on without the local interface flag, so that they would connect to the NSCL, and perform their registration and subscriptions as described in this chapter. It was ran several times, to calculate the traffic increase caused by each new resource. The average increase measurements were calculated according to the following formulas³:

³Abbreviations:
Out = Outbound;
mX = Measurement number;
In = Inbound.

$$\text{Outbound_Average_}\delta = \frac{(\text{Out_m4}-\text{Out_m3})+(\text{Out_m3}-\text{Out_m2})+(\text{Out_m2}-\text{Out_m1})}{3}$$

$$\text{Inbound_Average_}\delta = \frac{(\text{In_m4}-\text{In_m3})+(\text{In_m3}-\text{In_m2})+(\text{In_m2}-\text{In_m1})}{3}$$

The results were then gathered into Table 6.2. All the measurements are displayed in bytes.

Measurement	Traffic Type	Mobile M2M GW	Mobile M2M NA	Total
1	Outbound	10233	7556	45079
	Inbound	12516	14774	
2	Outbound	10908	8301	49188
	Inbound	13424	16555	
3	Outbound	11579	9046	53289
	Inbound	14328	18336	
4	Outbound	12258	9791	57406
	Inbound	15240	20117	
Outbound Average Δ		675	745	$\Delta = 4109$
Inbound Average Δ		908	1781	
Total Δ		1583	2526	

Table 6.2: Resource Registration Traffic

6.2.2 Local Interface vs NSCL

To compare the traffic performance of both applications running normally through the NSCL and through the local interface, a timer was programmatically added to assure that both would run for exactly the same time, chosen arbitrarily to be 10 minutes. The timer was set to start when the *start sensor* button was pressed, and both applications had already been registered, so that they would get their resources from the database. To execute these tests, the *Subscription Check* procedures were also turned off, so that the measurements were exclusively from the sensor data. The results yielded are present in Table 6.3⁴. All the measurements displayed are in bytes.

Connection	Traffic Type	Mobile M2M GW		Mobile M2M NA		Total	
		Int	Ext	Int	Ext	Int	Ext
NSCL	Inbound	0	110746	0	277168	0	743727
	Outbound	0	95522	0	260291		
Local Interface	Inbound	87541	4552	93057	4553	355143	10192
	Outbound	91181	481	83364	606		

Table 6.3: Network traffic comparison table

⁴Abbreviations:

Int = Internal traffic, i.e. localhost;

Ext = External traffic, i.e. through NSCL.

6.2.3 Discussion

Starting with the analysis of the results from Table 6.2, it becomes clear that there is a significant increase in traffic due to the *Check Subscription* procedure, which performs retrieve requests for each resource found, in order to check for the subscription. The average outbound increase rate for the mobile M2M GW is 675 bytes, while the mobile M2M NA has a 745 bytes increase. The small discrepancy is due to the different resource sizes, since *Scl*'s have more attributes. As for incoming traffic, the mobile M2M GW average delta was 908 bytes, while the mobile M2M NA's was 1781 bytes. This greater discrepancy is double on the NA, because the NA has to retrieve double the resources as it needs to check the `/m2m/scls/[sclId]/applications/subscriptions` as well as the `/m2m/scls/[sclId]/applications/appId/containers/subscriptions` collections while the GW only has to check the `/m2m/applications/subscriptions` collection. Still from column *Total* of Table 6.2 we can check the significant amount of data exchanged during this process when adding both application's traffic. The total data exchanged increases 4109 bytes with each new registration on the NSCL, which is quite significant in the constrained scenario of this dissertation. The local storage tackles this issue, by storing the created resources and using them at the next connection, saving a great deal of traffic.

Then, Table 6.3 shows the comparison of traffic between the communication that flows through the NSCL, and the communication that flows through the local interface. First thing to notice is that in both cases, the inbound and outbound flows are almost the same, with slight advantage to the inbound, and this occurs because the ETSI standard requires the objects to flow back and forth, even when there is no change to them, and because of some requests executed by the applications. The big difference here is that for an outbound data of 95522 bytes from the mobile M2M GW, the mobile M2M NA has a 277168 bytes inbound register, which is almost three times more. This happens because the packet size of the *ContentInstances* collection, adding headers and actual payload, is roughly 3 times the size of the *ContentInstance* packet sent by the mobile M2M GW. This was proven by further testing, where the average HTTP *ContentInstance* packet containing the Zephyr data leaving the mobile M2M GW was ≈ 1500 bytes, and the notification HTTP packet containing the *ContentInstances* collection had ≈ 4200 bytes arriving at the mobile M2M NA, hence the discrepancy. The sum of all external traffic communicating through the NSCL is then ≈ 743 Kb, which is very high for a mobile scenario on constrained devices. Comparing these with the local interface traffic measurements, the first thing to notice is that the external communication is nearly non-existent, caused only by the Bootstrap procedures and the initial attempt to register the unique resources, which then leads to the retrieval from the database. Then, since the communication is mostly executed locally, the outbound traffic of the mobile M2M GW should be almost exactly the same as the inbound traffic from the mobile M2M NA. However, there is a discrepancy between the outbound and inbound measurements, which occurs due to the fact that the Apache HTTP class adds some HTTP standard headers when executing the request, that are not reachable programmatically from the sender side, thus not being counted in this study. Examining Table 6.3 it is clear that using the local interface for the communication saves a significant amount of exter-

nal traffic ($\approx 355\text{Kb}$ using the NSCL VS $\approx 10\text{Kb}$ using the local interface), when ran for the same amount of time than the communication relying on the NSCL. According to these measurements, the external traffic generated using the local interface is $\approx 1,4\%$ of the generated by the normal communications, being a positive result for this dissertation's study.

Even so, the local interface generates significantly less traffic than the communications through the NSCL, and considering that most of them are local ($\approx 355\text{Kb}$) as opposed to external ($\approx 10\text{Kb}$). The amount of traffic generated using the local interface is then $\approx 1,4\%$ of the generated by the normal communications, being a positive result for this dissertation's study.

6.2.3.1 Evaluation Discussion

According to the evaluation metrics proposed in Section 4.3, Table 6.4 summarizes which ones could or could not be fulfilled. The mobile M2M NA only partially implements the commands, as there was no time to develop a GUI that the user could use to change the active sensors configurations, that was already implemented in the mobile M2M GW.

Target Application	Priority	What to Evaluate	Status
<u>Basic Use Case</u>			
Both	High	Communicate securely with the NSCL using at least one supported protocol	Completed
Both	High	Register itself and its resources in the NSCL	Completed
Both	High	Support a web-server that is able to receive subscriptions and messages.	
Mobile M2M GW	High	Support at least one medical sensor	Completed
Mobile M2M GW	High	Have the ability to receive and process commands according to the Use Case scenario	Completed
Mobile M2M NA	High	Have the ability to send commands to the Gateway and process its answers.	Partially Completed
Mobile M2M GW	Low	Support more medical sensors, widening the test base	Not completed
<u>Traffic Reduction Use Case</u>			
Both	High	Implement the local interface	Completed
Both	High	Compare communication traffic through the NSCL and through the local interface	Completed
Mobile M2M GW	Medium	Use a local storage to safeguard the sensor data when the connection drops	Completed

Table 6.4: Evaluation metrics

Chapter 7

Conclusions and Future Work

This Chapter provides an overview of this dissertation's work, its contributions, and the conclusions drawn. Then, some guidelines for future work are presented.

7.1 Conclusions

The main objective of this dissertation was to use a mobile system using M2M communications on a specific healthcare scenario, as a proof-of-concept for other mobile M2M scenarios. To create this solution, the following steps were taken, specifying this dissertation's contributions:

- First of all, the use cases were developed, in order to have concrete objectives in what the user should be able to do, and which were the important abilities of an M2M enabled healthcare scenario, keeping in mind that a mobile device has limited resources (traffic and battery).
- Then, in order to fulfill the use cases, M2M standards were procured, and ETSI's was chosen given that the project's partners were already on the way to support it, and since it fit the requirements of this dissertation's project.
- Since the ETSI standard is very versatile, the first step was to map its resources into tangible concepts, in order to focus the approach on the healthcare scenario at hand. Together with the project use cases, the mobile M2M GW architecture started to be designed in a modular way, aiming to allow the easier addition of protocols, sensors and functionalities. This map of the ETSI resources executed for this dissertation, is one of its contributions, showing a possible approach for future M2M scenarios.
- The Zephyr sensor was then integrated to the mobile M2M GW, in order to test the ETSI mId connection to the NSCL, and to also prove that the modular architecture that it supported was functional. The connection was implemented and tested successfully, and mobile aware features such as a local database for when the connection drops, were added.

- Afterwards, the mobile M2M NA architecture was designed based on the working mobile M2M GW, implementing ETSI's mIa interface to communicate with the NSCL. After this connection was successfully tested, the first step was to get the information flowing between the two applications: The mobile M2M NA should be able to send commands to the mobile M2M GW, trigger actions, and receive the information corresponding to the command sent. To execute this, there was the need to determine the best way, within ETSI's scope, to deliver the commands to the mobile M2M GW. After the decision to create a mobile M2M NA *Container* that the mobile M2M GW would subscribe, this approach was implemented and successfully tested. This mechanism to flow information between the mobile M2M GW and NA is also a contribution of this dissertation, taking the map designed to create a bi-directional data flow.
- With the communication flowing properly through the NSCL, attention shifted into the implementation of the local interface, to allow the system to be functional even without an Internet connection. The projected local interface was implemented and successfully tested, with the information correctly flowing between the two applications, either communicating through the NSCL, or through the local interface. This is also a valid contribution of this dissertation's work, since it is an alternative and allows for new local scenarios.
- Finally, to evaluate the real efficiency of this local interface, some bandwidth tests were executed, which showed that communicating through the local interface saved traffic, proving the concept behind its creation. This local interface is a contribution of this dissertation's work, that can be adapted for future M2M scenarios that have similar efficiency concerns.

The solution presented in this dissertation was functional, and it communicates successfully with the NSCL developed by *PTIN*. This proved that the interpretation and map of the ETSI standard is valid, and that this kind of implementation is fairly straight-forward, when the standard starts to be understood. The local interface proved be useful when attached to an M2M system like the one described and implemented, where both applications are running on the same smartphone. The communication mechanisms created for the information to properly flow between the two applications can be adapted to other areas of M2M, mobile or not, extending the reach beyond the healthcare scenario studied in this dissertation.

Using web-servers on a mobile system such as the one studied in this dissertation, may not be the best approach, because of the reachability issues discussed in Section 5.4. As pointed out, the ETSI standard has a *long-polling* mechanism that can provide the solution to this problem, allowing the subscriptions and subsequent notifications to be delivered to the mobile M2M GW and NA without the use of a web-server, which could improve the scalability of the solution.

Working with the *PTIN* partners was not always easy because of schedule differences and other issues mentioned in Chapter 5, but ultimately it was a positive experience, since all the hurdles only drove me to work harder and try and complete the work I had set out to do. It naturally gave me a better view of the corporate world, being a preparation for the (near) future.

7.2 Future Work

Despite the success of the solution presented in this dissertation, there is still room for improvement in several aspects of the system. This section divides those improvements into three categories, one regarding solely to the mobile M2M GW, another to the mobile M2M NA, while the final concerning the system as a whole.

The improvements for the mobile M2M GW could be:

- **Additional Sensors:** More healthcare sensors could be integrated with the mobile M2M GW in order to improve its functionalities in healthcare scenarios.
- **Minimal GUI:** The mobile M2M GW could benefit from a minimalistic popup-based interface, to control if it is running or not, for situations when the NA is not present in the same smartphone.

The improvement for the mobile M2M NA could be:

- **GUI Improvements:** The mobile M2M NA can show real-time graphs with data related to which sensor being used, for easier understanding.
- **Sensor configurations:** The mobile M2M NA needs to have a GUI that allows the user to change the active sensor's configuration parameters.

The improvements for the mobile M2M system could be:

- **Long-Polling:** As mentioned in the previous section, the ETSI *long-polling* mechanism could be implemented into both applications, in an attempt to replace the web-server, to create a significantly more scalable mobile solution.
- **Implement more commands:** The system should support the exchange of more commands, such as: Sensor configurations, to change the intervals between data deliveries, or even the maximum buffer size; Which measurements to store, given the case that for a specific sensor, not all data is interesting to be stored. These would mean adding some features on the GUI of the mobile M2M NA, as well adding support for those commands on both mobile applications.
- **Security:** Fully implement TLS on the mobile applications web-servers for secure communication with the NSCL. For security in the local interface, new certificates could be used, or a list of authorized *Scl* and *Application* could be stored in the NSCL so that only allowed parties could send messages.
- **Connection to healthcare service and data privacy:** The current system can be extended by a different M2M NA (not necessarily mobile), that could connect the NSCL to PTIN's healthcare platform, that would then manage the user's medical records. The *Access Rights*

feature has been disregarded so far, but it is essential in an M2M system like the one described in this dissertation, to ensure that each user has only access to its own data, and that it is hidden from other users querying the NSCL. When processing the data and introducing it into the healthcare service, it should be deleted from the NSCL, in order to safeguard the user's interests. A connection could then be created between this healthcare service and the mobile M2M NA, so that the user can have access to its medical records from within the application.

Appendix A

ETSI classes changes

This appendix documents the changes executed on the classes automatically generated from the ETSI standard schemas. These changes can difficult the adoption of new classes when that the standard schemas change, which is why it is relevant to document which ones were altered.

Firstly, as mentioned in Section 5.5, all the classes were stripped of the *"javax.xml.*"* imports, as Android OS does not implement them. Then, three different scopes of changes had to be executed, and these will be explained below and summarized in Table A.1¹.

Jackson Annotations: On some cases (specially when there were JSON Arrays), the removal of the XML imports, caused errors during serialization an de-serialization, because the Class variable names were different than they should. To fix this, some Jackson annotations had to be done, most of them to force the variable name into the specified in the documents. The classes that now have Jackson annotations are:

Custom Jackson Serializers and De-Serializers: Even with the annotations, some classes could only be fully compliant with the ETSI specifications after they had a custom Jackson serializer and/or de-serializer made for them. Only the class *SearchStrings* needed a custom serializer. The following list enumerates the ones that needed a custom de-serializer:

These custom made classes are located in the common module of the project, under "pt.ptinovacao.mtom.common.marshallers".

Java Serialization Extend: To create the databases holding the current state of the Gateway and the Network Application as well as the local API storage, the following ETSI classes had to be slightly altered to allow Java Serialization:

¹Table Abbreviations:

JA: Jackson Annotations

CS: Custom Serializer

CDS: Custom De-Serializer

JSE: Java Serialization Extend

Java Class Name	JA¹	CS¹	CDS¹	JSE¹
AnyURIList	Yes	No	No	Yes
APocHandling	Yes	No	No	No
Application	Yes	No	Yes	Yes
Applications	Yes	No	No	No
Container	Yes	No	Yes	Yes
Content	No	No	No	Yes
ContentInstanceCollection	Yes	No	No	No
ContentTypes	Yes	No	Yes	Yes
CreateRequestIndication	Yes	No	No	No
IntegrityValResults	No	No	No	Yes
MgmtPtotocolType	No	No	No	Yes
NamedReferenceCollection	Yes	No	No	No
OnlineStatus	No	No	No	Yes
Scl	Yes	No	Yes	Yes
SearchStrings	Yes	Yes	Yes	Yes
Subscription	Yes	No	Yes	Yes
SubscriptionType	Yes	No	No	No

Table A.1: Summary of affected classes

Appendix B

Mobile M2M GW Logs

This appendix shows the logs of the mobile M2M GW running and communicating through the NSCL (in Section B.1) and through the local API (in Section B.2).

B.1 NSCL

```
07-14 02:46:40.558 8276-8300/pt.ptinovacao.mtom.gw D/MainService: Thread Started:
8300
07-14 02:46:40.578 8276-8300/pt.ptinovacao.mtom.gw I/ProtocolManager: Storage file
created successfully!
07-14 02:46:40.578 8276-8304/pt.ptinovacao.mtom.gw D/Sensor Handler: Thread Started:
8304
07-14 02:46:40.578 8276-8303/pt.ptinovacao.mtom.gw D/ProtocolManager: Thread Started:
8303
07-14 02:46:40.578 8276-8307/pt.ptinovacao.mtom.gw D/BluetoothManager: Thread Started
: 8307
07-14 02:46:40.588 8276-8305/pt.ptinovacao.mtom.gw D/MemoryManager: Thread Started:
8305
07-14 02:46:40.588 8276-8276/pt.ptinovacao.mtom.gw I/ConnectionChecker: Network WIFI
connected
07-14 02:46:40.588 8276-8276/pt.ptinovacao.mtom.gw V/ConnectionListener:
BroadcastReceiver started successfully.
07-14 02:46:40.588 8276-8309/pt.ptinovacao.mtom.gw D/GSCL: Thread Started: 8309
07-14 02:46:40.588 8276-8308/pt.ptinovacao.mtom.gw D/GPS: Thread Started: 8308
07-14 02:46:40.588 8276-8303/pt.ptinovacao.mtom.gw D/ProtocolManager: GSCL already
running.
07-14 02:46:40.588 8276-8303/pt.ptinovacao.mtom.gw D/ProtocolManager: Storage file
empty. Nothing to flush.
07-14 02:46:40.598 8276-8308/pt.ptinovacao.mtom.gw I/GPS: Provider network has been
selected.
07-14 02:46:40.628 8276-8309/pt.ptinovacao.mtom.gw V/GSCL: GSCL Root URI: http://
mobilelab.fe.up.pt:8080
07-14 02:46:40.778 8276-8309/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 278
K, 19% free 2872K/3516K, paused 9ms, total 9ms
07-14 02:46:41.088 8276-8309/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 313
K, 19% free 3070K/3752K, paused 17ms, total 18ms
07-14 02:46:41.108 8276-8309/pt.ptinovacao.mtom.gw D/M2M-BootStrap: GET https://
phonegw.nscl.m2m.ptinovacao.pt:8443/bootstrapParamSet HTTP/1.1
07-14 02:46:41.108 8276-8309/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap GET
Size: 74
07-14 02:46:41.158 8276-8309/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 02:46:41.158 8276-8309/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nscl.
m2m.ptinovacao.pt
```

```

07-14 02:46:41.698      8276-8309/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
connection with phonegw.nscl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 02:46:41.908      8276-8309/pt.ptinovacao.mtom.gw D/M2M-BootStrap: RESPONSE HTTP/1.1
200 OK
07-14 02:46:41.918      8276-8309/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap Answer
Size: 4307
07-14 02:46:41.928      8276-8309/pt.ptinovacao.mtom.gw D/M2M-BootStrap: Certificate
Validity: Fri Jan 30 01:46:42 WET 2015
07-14 02:46:42.048      8276-8309/pt.ptinovacao.mtom.gw D/CreateResources: Registering SCL
.
07-14 02:46:42.048      8276-8334/pt.ptinovacao.mtom.gw D/HTTPServer: Thread Started: 8334
07-14 02:46:42.078      8276-8309/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 391
K, 20% free 3189K/3948K, paused 9ms, total 9ms
07-14 02:46:42.118      8276-8309/pt.ptinovacao.mtom.gw I/CreateResources: Registering scl
: hammerhead-04506dfa25231287-TesteFEUP to: /m2m/scls .
07-14 02:46:42.118      8276-8309/pt.ptinovacao.mtom.gw I/GSCL: Local API file created
successfully!
07-14 02:46:42.248      8276-8333/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 02:46:42.248      8276-8333/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nscl.
m2m.ptinovacao.pt
07-14 02:46:42.618      8276-8333/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
connection with phonegw.nscl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 02:46:42.818      8276-8333/pt.ptinovacao.mtom.gw W/HTTPClient: RESPONSE HTTP/1.1
405 Method Not Allowed
07-14 02:46:42.978      8276-8309/pt.ptinovacao.mtom.gw D/GSCL: SCL already exists.
Retrieving Scls.
07-14 02:46:42.978      8276-8309/pt.ptinovacao.mtom.gw D/GSCL: Memory database
successfully read!
07-14 02:46:42.978      8276-8309/pt.ptinovacao.mtom.gw W/GSCL: Stored SclStorage is null.
Re-creating.
07-14 02:46:42.978      8276-8309/pt.ptinovacao.mtom.gw I/RetrieveResources: Retrieving
Container: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP
07-14 02:46:42.978      8276-8309/pt.ptinovacao.mtom.gw I/RetrieveResources: Retrieving
Container: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
HealthSensors
07-14 02:46:42.978      8276-8333/pt.ptinovacao.mtom.gw I/HTTPClient: GET sent to domain:
/m2m/scls/hammerhead-04506dfa25231287-TesteFEUP
07-14 02:46:43.358      8276-8333/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 436
K, 20% free 3265K/4068K, paused 10ms, total 11ms
07-14 02:46:43.358      8276-8333/pt.ptinovacao.mtom.gw I/HTTPClient: GET sent to domain:
/m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/HealthSensors
07-14 02:46:43.388      8276-8309/pt.ptinovacao.mtom.gw I/GSCL: SCL stored successfully.
07-14 02:46:43.878      8276-8309/pt.ptinovacao.mtom.gw I/GSCL: Local Application
HealthSensors stored successfully.
07-14 02:46:44.038      8276-8308/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.1784305; Longitude: -8.5953201; Accuracy: 43.871.
07-14 02:46:53.848      8276-8334/pt.ptinovacao.mtom.gw V/HandleRequests: Received
ContentInstances collection notification.
07-14 02:46:53.918      8276-8334/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 416
K, 19% free 3360K/4144K, paused 11ms, total 11ms
07-14 02:46:53.998      8276-8334/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 494
K, 21% free 3378K/4240K, paused 10ms, total 10ms
07-14 02:46:54.008      8276-8334/pt.ptinovacao.mtom.gw V/HandleRequests: Valid Search
Sensors Command.
Bluetooth: trueSearch Internal: true
07-14 02:46:54.008      8276-8276/pt.ptinovacao.mtom.gw I/MainService: MainService Started
with Intent SEARCH_SENSORS
07-14 02:46:56.988      8276-8276/pt.ptinovacao.mtom.gw D/BluetoothManager: Found Zephyr
BT device: HXM017399
07-14 02:46:56.988      8276-8276/pt.ptinovacao.mtom.gw V/BluetoothManager: Zephyr bonded.
Starting ZephyrHandler...
07-14 02:46:56.998      8276-8276/pt.ptinovacao.mtom.gw V/BluetoothManager: RSSI: -49
07-14 02:47:11.998      8276-8309/pt.ptinovacao.mtom.gw I/CreateResources: Registering
container: 1405302431976-ZEPHYR to: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
applications/HealthSensors/containers .
07-14 02:47:12.218      8276-8309/pt.ptinovacao.mtom.gw I/GSCL: Local Container
1405302431976-ZEPHYR created successfully.

```

```

07-14 02:47:15.558 8276-8334/pt.ptinovacao.mtom.gw V/HandleRequests: Received
ContentInstances collection notification .
07-14 02:47:15.568 8276-8334/pt.ptinovacao.mtom.gw V/HandleRequests: Valid Container
to Start command.
Container: 1405302431976-ZEPHYR
07-14 02:47:15.568 8276-8276/pt.ptinovacao.mtom.gw I/MainService: Starting
1405302431976-ZEPHYR sensor .
07-14 02:47:15.568 8276-8276/pt.ptinovacao.mtom.gw D/ZephyrHandler: Zephyr MAC
Address: 00:07:80:5A:3C:63
07-14 02:47:15.578 8276-8276/pt.ptinovacao.mtom.gw D/BluetoothSocket: connect() ,
SocketState: INIT , mPfd: {ParcelFileDescriptor: FileDescriptor[74]}
07-14 02:47:16.978 8276-8276/pt.ptinovacao.mtom.gw I/ZephyrHandler: Zephyr connected
07-14 02:47:16.988 8276-8605/pt.ptinovacao.mtom.gw I/ZEPHYR: Connected to BioHarness
HXM017399 .
07-14 02:47:16.998 8276-8605/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 455
K, 20% free 3433K/4256K, paused 10ms, total 10ms
07-14 02:47:17.668 8276-8308/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.1784549; Longitude: -8.5953409; Accuracy: 42.588 .
07-14 02:47:17.798 8276-8608/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet .
07-14 02:47:18.048 8276-8276/pt.ptinovacao.mtom.gw W/ZephyrHandler: Timestamp empty .
07-14 02:47:20.798 8276-8613/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet .
07-14 02:47:23.798 8276-8619/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet .
07-14 02:47:25.578 8276-8305/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR .
07-14 02:47:25.628 8276-8309/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications /
HealthSensors / containers/1405302431976-ZEPHYR / contentInstances .
07-14 02:47:25.728 8276-8333/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered .
07-14 02:47:26.818 8276-8645/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet .
07-14 02:47:29.828 8276-8654/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet .
07-14 02:47:32.848 8276-8655/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet .
07-14 02:47:35.588 8276-8305/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR .
07-14 02:47:35.608 8276-8309/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications /
HealthSensors / containers/1405302431976-ZEPHYR / contentInstances .
07-14 02:47:35.728 8276-8333/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered .
07-14 02:47:35.848 8276-8657/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet .
07-14 02:47:38.858 8276-8682/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet .
07-14 02:47:41.878 8276-8690/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet .
07-14 02:47:44.858 8276-8334/pt.ptinovacao.mtom.gw V/HandleRequests: Received
ContentInstances collection notification .
07-14 02:47:44.878 8276-8334/pt.ptinovacao.mtom.gw V/HandleRequests: Valid Container
to Stop command.
Container: 1405302431976-ZEPHYR
07-14 02:47:44.878 8276-8276/pt.ptinovacao.mtom.gw D/ZephyrHandler: Stopping Zephyr
...
07-14 02:47:48.048 8276-8276/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 456
K, 20% free 3459K/4288K, paused 19ms, total 20ms
07-14 02:47:48.048 8276-8276/pt.ptinovacao.mtom.gw I/dalvikvm-heap: Grow heap (frag
case) to 3.812MB for 82384-byte allocation
07-14 02:47:48.068 8276-8285/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 4K,
20% free 3534K/4372K, paused 21ms, total 21ms
07-14 02:47:48.178 8276-8276/pt.ptinovacao.mtom.gw D/MainService: M2M Gateway
Stopping
07-14 02:47:48.188 8276-8276/pt.ptinovacao.mtom.gw D/ProtocolManager: ProtocolManager
Service Stopping
07-14 02:47:48.188 8276-8276/pt.ptinovacao.mtom.gw D/GSCL: GSCL Stopping

```

```

07-14 02:47:48.268 8276-8276/pt.ptinovacao.mtom.gw D/GSCL: Successfully saved state
to storage.
07-14 02:47:48.268 8276-8276/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound External
Traffic: 4841. Inbound External Traffic: 6951
07-14 02:47:48.268 8276-8276/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound Internal
Traffic: 0. Inbound Internal Traffic: 0
07-14 02:47:48.268 8276-8333/pt.ptinovacao.mtom.gw D/HTTPClient: HTTP Client Shut
Down
07-14 02:47:48.268 8276-8276/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound External
Traffic: 7566. Inbound External Traffic: 8061
07-14 02:47:48.268 8276-8276/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound Internal
Traffic: 0. Inbound Internal Traffic: 0
07-14 02:47:48.268 8276-8276/pt.ptinovacao.mtom.gw I/ProtocolManager: Temporary
storage file successfully deleted.
07-14 02:47:48.268 8276-8276/pt.ptinovacao.mtom.gw D/GPS: GPS stopping
07-14 02:47:48.268 8276-8276/pt.ptinovacao.mtom.gw D/Sensor Handler: Stopping Sensor
Handler.
07-14 02:47:48.268 8276-8276/pt.ptinovacao.mtom.gw D/BluetoothManager: BT Manager
Stopping..
07-14 02:47:48.278 8276-8276/pt.ptinovacao.mtom.gw D/MemoryManager: MemoryManager
stopping
07-14 02:47:48.288 8276-8276/pt.ptinovacao.mtom.gw I/MainService: Stopping Service.
Goodbye!

```

B.2 Local API

```

07-14 02:50:34.278 10585-10600/pt.ptinovacao.mtom.gw D/MainService: Thread Started:
10600
07-14 02:50:34.318 10585-10600/pt.ptinovacao.mtom.gw I/ProtocolManager: Storage file
created successfully!
07-14 02:50:34.328 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: Thread Started:
10612
07-14 02:50:34.328 10585-10614/pt.ptinovacao.mtom.gw D/GSCL: Thread Started: 10614
07-14 02:50:34.338 10585-10613/pt.ptinovacao.mtom.gw D/Sensor Handler: Thread Started:
10613
07-14 02:50:34.338 10585-10585/pt.ptinovacao.mtom.gw I/ConnectionChecker: Network WIFI
connected
07-14 02:50:34.338 10585-10585/pt.ptinovacao.mtom.gw V/ConnectionListener:
BroadcastReceiver started successfully.
07-14 02:50:34.348 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: GSCL already
running.
07-14 02:50:34.348 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: Storage file
empty. Nothing to flush.
07-14 02:50:34.348 10585-10615/pt.ptinovacao.mtom.gw D/MemoryManager: Thread Started:
10615
07-14 02:50:34.348 10585-10618/pt.ptinovacao.mtom.gw D/GPS: Thread Started: 10618
07-14 02:50:34.358 10585-10616/pt.ptinovacao.mtom.gw D/BluetoothManager: Thread Started
: 10616
07-14 02:50:34.358 10585-10618/pt.ptinovacao.mtom.gw I/GPS: Provider network has been
selected.
07-14 02:50:34.368 10585-10614/pt.ptinovacao.mtom.gw V/GSCL: GSCL Root URI: http://
mobilelab.fe.up.pt:8080
07-14 02:50:34.468 10585-10614/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 296
K, 19% free 2854K/3516K, paused 9ms, total 9ms
07-14 02:50:34.698 10585-10614/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 302
K, 18% free 3063K/3732K, paused 9ms, total 9ms
07-14 02:50:34.768 10585-10614/pt.ptinovacao.mtom.gw D/M2M-BootStrap: GET https://
phonegw.nsl.m2m.ptinovacao.pt:8443/bootstrapParamSet HTTP/1.1
07-14 02:50:34.768 10585-10614/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap GET
Size: 74
07-14 02:50:34.858 10585-10614/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 02:50:34.858 10585-10614/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 02:50:35.128 10585-10614/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5

```

```

07-14 02:50:35.498 10585-10614/pt.ptinovacao.mtom.gw D/M2M-BootStrap: RESPONSE HTTP/1.1
200 OK
07-14 02:50:35.498 10585-10614/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap Answer
Size: 4299
07-14 02:50:35.508 10585-10614/pt.ptinovacao.mtom.gw D/M2M-BootStrap: Certificate
Validity: Fri Jan 30 01:50:35 WET 2015
07-14 02:50:35.598 10585-10614/pt.ptinovacao.mtom.gw D/CreateResources: Registering SCL
.
07-14 02:50:35.608 10585-10642/pt.ptinovacao.mtom.gw D/HTTPServer: Thread Started:
10642
07-14 02:50:35.628 10585-10614/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 389
K, 20% free 3187K/3944K, paused 9ms, total 9ms
07-14 02:50:35.678 10585-10614/pt.ptinovacao.mtom.gw I/CreateResources: Registering scl
: hammerhead-04506dfa25231287-TesteFEUP to: /m2m/scls .
07-14 02:50:35.698 10585-10641/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 02:50:35.698 10585-10641/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nssl.
m2m.ptinovacao.pt
07-14 02:50:36.028 10585-10641/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
connection with phonegw.nssl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 02:50:36.208 10585-10641/pt.ptinovacao.mtom.gw W/HTTPClient: RESPONSE HTTP/1.1
405 Method Not Allowed
07-14 02:50:36.388 10585-10614/pt.ptinovacao.mtom.gw D/GSCL: SCL already exists.
Retrieving Scls.
07-14 02:50:36.428 10585-10614/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 426
K, 20% free 3275K/4068K, paused 11ms, total 11ms
07-14 02:50:36.448 10585-10614/pt.ptinovacao.mtom.gw D/GSCL: Memory database
successfully read!
07-14 02:50:37.738 10585-10618/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.1784581; Longitude: -8.5953115; Accuracy: 44.656.
07-14 02:50:45.528 10585-10642/pt.ptinovacao.mtom.gw D/HandleRequests: Local connection
. Received Command.
07-14 02:50:45.628 10585-10642/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 426
K, 20% free 3361K/4156K, paused 9ms, total 9ms
07-14 02:50:45.638 10585-10642/pt.ptinovacao.mtom.gw V/HandleRequests: Valid Search
Sensors Command.
Bluetooth: trueSearch Internal: true
07-14 02:50:45.638 10585-10585/pt.ptinovacao.mtom.gw I/MainService: MainService Started
with Intent SEARCH_SENSORS
07-14 02:50:46.868 10585-10585/pt.ptinovacao.mtom.gw D/BluetoothManager: Found Zephyr
BT device: HXM017399
07-14 02:50:46.868 10585-10585/pt.ptinovacao.mtom.gw V/BluetoothManager: Zephyr bonded.
Starting ZephyrHandler...
07-14 02:50:46.868 10585-10585/pt.ptinovacao.mtom.gw V/BluetoothManager: RSSI: -56
07-14 02:50:52.868 10585-10585/pt.ptinovacao.mtom.gw W/ConnectionListener: Network
connectivity change
07-14 02:50:52.868 10585-10585/pt.ptinovacao.mtom.gw I/ConnectionChecker: Network WIFI
connected
07-14 02:50:52.878 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: GSCL already
running.
07-14 02:50:52.878 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: Storage file
empty. Nothing to flush.
07-14 02:50:58.848 10585-10614/pt.ptinovacao.mtom.gw I/CreateResources: Registering
container: 1405302658817-ZEPHYR to: /m2m/localAPI/newContainer/1405302658817-ZEPHYR
.
07-14 02:51:06.248 10585-10642/pt.ptinovacao.mtom.gw D/HandleRequests: Local connection
. Received Command.
07-14 02:51:06.268 10585-10642/pt.ptinovacao.mtom.gw V/HandleRequests: Valid Container
to Start command.
Container: 1405302658817-ZEPHYR
07-14 02:51:06.268 10585-10585/pt.ptinovacao.mtom.gw I/MainService: Starting
1405302658817-ZEPHYR sensor.
07-14 02:51:06.268 10585-10585/pt.ptinovacao.mtom.gw D/ZephyrHandler: Zephyr MAC
Address: 00:07:80:5A:3C:63
07-14 02:51:06.278 10585-10585/pt.ptinovacao.mtom.gw D/BluetoothSocket: connect(),
SocketState: INIT, mPfd: {ParcelFileDescriptor: FileDescriptor[66]}
07-14 02:51:07.368 10585-10585/pt.ptinovacao.mtom.gw I/ZephyrHandler: Zephyr connected
07-14 02:51:07.388 10585-11060/pt.ptinovacao.mtom.gw I/ZEPHYR: Connected to BioHarness
HXM017399.

```

```

07-14 02:51:08.188 10585-11063/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:08.468 10585-10585/pt.ptinovacao.mtom.gw W/ZephyrHandler: Timestamp empty.
07-14 02:51:11.188 10585-11069/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:11.238 10585-10618/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.1784354; Longitude: -8.5953272; Accuracy: 43.4.
07-14 02:51:14.218 10585-11077/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:16.278 10585-10615/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 02:51:16.318 10585-10612/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 438
K, 19% free 3434K/4240K, paused 20ms, total 21ms
07-14 02:51:16.338 10585-10614/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/localAPI/contentInstance/1405302658817-ZEPHYR .
07-14 02:51:16.718 10585-10641/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 02:51:17.218 10585-11105/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:20.228 10585-11147/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:22.058 10585-10585/pt.ptinovacao.mtom.gw W/ConnectionListener: Network
connectivity change
07-14 02:51:22.058 10585-10585/pt.ptinovacao.mtom.gw D/ConnectionChecker: No network
connectivity
07-14 02:51:22.058 10585-10585/pt.ptinovacao.mtom.gw D/ProtocolManager: No connection.
Resetting stored IPV4 address.
07-14 02:51:23.238 10585-11222/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:26.248 10585-11251/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:26.288 10585-10615/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 02:51:26.338 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: No Internet
Connection, but local API connected. Sending data.
07-14 02:51:26.338 10585-10614/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/localAPI/contentInstance/1405302658817-ZEPHYR .
07-14 02:51:26.448 10585-10641/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 02:51:29.258 10585-11262/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:32.268 10585-11272/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:35.288 10585-11317/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:36.298 10585-10615/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 02:51:36.348 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: No Internet
Connection, but local API connected. Sending data.
07-14 02:51:36.348 10585-10614/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/localAPI/contentInstance/1405302658817-ZEPHYR .
07-14 02:51:36.478 10585-10641/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 457
K, 20% free 3439K/4296K, paused 25ms, total 26ms
07-14 02:51:36.478 10585-10641/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 02:51:38.298 10585-11319/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:41.308 10585-11329/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:41.368 10585-10618/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.1784444; Longitude: -8.5953291; Accuracy: 43.322.
07-14 02:51:44.308 10585-11347/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:46.308 10585-10615/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 02:51:46.358 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: No Internet
Connection, but local API connected. Sending data.
07-14 02:51:46.358 10585-10614/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/localAPI/contentInstance/1405302658817-ZEPHYR .
07-14 02:51:46.518 10585-10641/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.

```



```

07-14 02:51:47.308 10585-11385/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:50.328 10585-11450/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:50.508 10585-10585/pt.ptinovacao.mtom.gw W/ConnectionListener: Network
connectivity change
07-14 02:51:50.508 10585-10585/pt.ptinovacao.mtom.gw I/ConnectionChecker: Network WIFI
connected
07-14 02:51:50.528 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: GSCL already
running.
07-14 02:51:50.528 10585-10612/pt.ptinovacao.mtom.gw D/ProtocolManager: Storage file
empty. Nothing to flush.
07-14 02:51:53.328 10585-11490/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 02:51:54.218 10585-10642/pt.ptinovacao.mtom.gw D/HandleRequests: Local connection
. Received Command.
07-14 02:51:54.218 10585-10642/pt.ptinovacao.mtom.gw V/HandleRequests: Valid Container
to Stop command.
Container: 1405302658817-ZEPHYR
07-14 02:51:54.218 10585-10585/pt.ptinovacao.mtom.gw D/ZephyrHandler: Stopping Zephyr
...
07-14 02:51:54.228 10585-10614/pt.ptinovacao.mtom.gw I/CreateResources: Registering
container: 1405302658817-ZEPHYR to: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
applications/HealthSensors/containers .
07-14 02:51:54.228 10585-10614/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
HealthSensors/containers/1405302658817-ZEPHYR/contentInstances .
07-14 02:51:54.228 10585-10614/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
HealthSensors/containers/1405302658817-ZEPHYR/contentInstances .
07-14 02:51:54.278 10585-10641/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 02:51:54.278 10585-10641/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nsc1.
m2m.ptinovacao.pt
07-14 02:51:54.378 10585-10641/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsc1.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 02:51:54.718 10585-10641/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 469
K, 20% free 3475K/4316K, paused 18ms, total 19ms
07-14 02:51:54.718 10585-10641/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 02:51:54.718 10585-10614/pt.ptinovacao.mtom.gw I/GSCL: Local Container
1405302658817-ZEPHYR created successfully.
07-14 02:51:54.858 10585-10641/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 02:52:02.808 10585-10585/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 159
K, 21% free 3443K/4316K, paused 12ms, total 12ms
07-14 02:52:02.818 10585-10585/pt.ptinovacao.mtom.gw I/dalvikvm-heap: Grow heap (frag
case) to 3.796MB for 82384-byte allocation
07-14 02:52:02.828 10585-10595/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed <1K
, 20% free 3523K/4400K, paused 15ms, total 15ms
07-14 02:52:02.858 10585-10585/pt.ptinovacao.mtom.gw D/MainService: M2M Gateway
Stopping
07-14 02:52:02.858 10585-10585/pt.ptinovacao.mtom.gw D/ProtocolManager: ProtocolManager
Service Stopping
07-14 02:52:02.858 10585-10585/pt.ptinovacao.mtom.gw D/GSCL: GSCL Stopping
07-14 02:52:02.928 10585-10585/pt.ptinovacao.mtom.gw D/GSCL: Successfully saved state
to storage.
07-14 02:52:02.958 10585-10585/pt.ptinovacao.mtom.gw D/GSCL: Successfully saved state
to storage.
07-14 02:52:02.958 10585-10585/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound External
Traffic: 4313. Inbound External Traffic: 4563
07-14 02:52:02.958 10585-10585/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound Internal
Traffic: 6650. Inbound Internal Traffic: 6322
07-14 02:52:02.958 10585-10641/pt.ptinovacao.mtom.gw D/HTTPClient: HTTP Client Shut
Down
07-14 02:52:02.968 10585-10585/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound External
Traffic: 0. Inbound External Traffic: 0
07-14 02:52:02.968 10585-10585/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound Internal
Traffic: 1457. Inbound Internal Traffic: 2072

```

```

07-14 02:52:02.968 10585-10585/pt.ptinovacao.mtom.gw I/ProtocolManager: Temporary
storage file successfully deleted.
07-14 02:52:02.968 10585-10585/pt.ptinovacao.mtom.gw D/GPS: GPS stopping
07-14 02:52:02.968 10585-10585/pt.ptinovacao.mtom.gw D/Sensor Handler: Stopping Sensor
Handler.
07-14 02:52:02.968 10585-10585/pt.ptinovacao.mtom.gw D/BluetoothManager: BT Manager
Stopping ..
07-14 02:52:02.968 10585-10585/pt.ptinovacao.mtom.gw D/MemoryManager: MemoryManager
stopping
07-14 02:52:02.968 10585-10585/pt.ptinovacao.mtom.gw I/MainService: Stopping Service.
Goodbye!
07-14 02:52:02.968 10585-10585/pt.ptinovacao.mtom.gw I/Process: Sending signal. PID:
10585 SIG: 9

```

B.3 Connectivity Lost Test

```

07-14 03:56:27.818 32257-32275/pt.ptinovacao.mtom.gw D/MainService: Thread Started:
32275
07-14 03:56:27.828 32257-32275/pt.ptinovacao.mtom.gw I/ProtocolManager: Storage file
created successfully!
07-14 03:56:27.838 32257-32276/pt.ptinovacao.mtom.gw D/ProtocolManager: Thread Started:
32276
07-14 03:56:27.848 32257-32280/pt.ptinovacao.mtom.gw D/MemoryManager: Thread Started:
32280
07-14 03:56:27.848 32257-32281/pt.ptinovacao.mtom.gw D/GPS: Thread Started: 32281
07-14 03:56:27.848 32257-32257/pt.ptinovacao.mtom.gw I/ConnectionChecker: Network WIFI
connected
07-14 03:56:27.848 32257-32257/pt.ptinovacao.mtom.gw V/ConnectionListener:
BroadcastReceiver started successfully.
07-14 03:56:27.848 32257-32277/pt.ptinovacao.mtom.gw D/Sensor Handler: Thread Started:
32277
07-14 03:56:27.848 32257-32282/pt.ptinovacao.mtom.gw D/BluetoothManager: Thread Started
: 32282
07-14 03:56:27.858 32257-32281/pt.ptinovacao.mtom.gw I/GPS: Provider network has been
selected.
07-14 03:56:27.858 32257-32276/pt.ptinovacao.mtom.gw D/ProtocolManager: Storage file
empty. Nothing to flush.
07-14 03:56:27.858 32257-32279/pt.ptinovacao.mtom.gw D/GSCL: Thread Started: 32279
07-14 03:56:27.888 32257-32279/pt.ptinovacao.mtom.gw V/GSCL: GSCL Root URI: http://
mobilelab.fe.up.pt:8080
07-14 03:56:28.058 32257-32279/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 286
K, 19% free 2864K/3516K, paused 29ms, total 29ms
07-14 03:56:28.298 32257-32279/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 309
K, 19% free 3067K/3744K, paused 10ms, total 10ms
07-14 03:56:28.308 32257-32279/pt.ptinovacao.mtom.gw D/M2M-BootStrap: GET https://
phonegw.nsl.m2m.ptinovacao.pt:8443/bootstrapParamSet HTTP/1.1
07-14 03:56:28.318 32257-32279/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap GET
Size: 74
07-14 03:56:28.398 32257-32279/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 03:56:28.398 32257-32279/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 03:56:28.768 32257-32279/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:56:28.968 32257-32279/pt.ptinovacao.mtom.gw D/M2M-BootStrap: RESPONSE HTTP/1.1
200 OK
07-14 03:56:28.968 32257-32279/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap Answer
Size: 4299
07-14 03:56:28.978 32257-32279/pt.ptinovacao.mtom.gw D/M2M-BootStrap: Certificate
Validity: Fri Jan 30 02:56:29 WET 2015
07-14 03:56:29.078 32257-32349/pt.ptinovacao.mtom.gw D/HTTPServer: Thread Started:
32349
07-14 03:56:29.078 32257-32279/pt.ptinovacao.mtom.gw D/CreateResources: Registering SCL
.
07-14 03:56:29.118 32257-32279/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 388
K, 20% free 3193K/3948K, paused 8ms, total 8ms

```

```

07-14 03:56:29.168 32257-32279/pt.ptinovacao.mtom.gw I/CreateResources: Registering scl
: hammerhead-04506dfa25231287-TesteFEUP to: /m2m/scls .
07-14 03:56:29.318 32257-32348/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 03:56:29.318 32257-32348/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nsc1.
m2m.ptinovacao.pt
07-14 03:56:29.858 32257-32348/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsc1.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:56:30.218 32257-32348/pt.ptinovacao.mtom.gw W/HTTPClient: RESPONSE HTTP/1.1
405 Method Not Allowed
07-14 03:56:30.218 32257-32279/pt.ptinovacao.mtom.gw D/GSCL: SCL already exists.
Retrieving Scls.
07-14 03:56:31.238 32257-32279/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 425
K, 20% free 3279K/4072K, paused 12ms, total 12ms
07-14 03:56:30.278 32257-32279/pt.ptinovacao.mtom.gw D/GSCL: Memory database
successfully read!
07-14 03:56:31.238 32257-32281/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.1784321; Longitude: -8.5952674; Accuracy: 20.0.
07-14 03:56:44.988 32257-32349/pt.ptinovacao.mtom.gw V/HandleRequests: Received
ContentInstances collection notification.
07-14 03:56:45.088 32257-32349/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 419
K, 19% free 3372K/4160K, paused 10ms, total 10ms
07-14 03:56:45.148 32257-32349/pt.ptinovacao.mtom.gw V/HandleRequests: Valid Search
Sensors Command.
Bluetooth: trueSearch Internal: true
07-14 03:56:45.158 32257-32257/pt.ptinovacao.mtom.gw I/MainService: MainService Started
with Intent SEARCH_SENSORS
07-14 03:56:47.868 32257-32257/pt.ptinovacao.mtom.gw D/BluetoothManager: Found Zephyr
BT device: HXM017399
07-14 03:56:47.868 32257-32257/pt.ptinovacao.mtom.gw V/BluetoothManager: Zephyr bonded.
Starting ZephyrHandler...
07-14 03:56:47.868 32257-32257/pt.ptinovacao.mtom.gw V/BluetoothManager: RSSI: -54
07-14 03:57:01.048 32257-32279/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 471
K, 20% free 3415K/4252K, paused 13ms, total 13ms
07-14 03:57:01.078 32257-32279/pt.ptinovacao.mtom.gw I/CreateResources: Registering
container: 1405306620999-ZEPHYR to: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
applications/HealthSensors/containers .
07-14 03:57:01.308 32257-32279/pt.ptinovacao.mtom.gw I/GSCL: Local Container
1405306620999-ZEPHYR created successfully.
07-14 03:57:03.288 32257-32349/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 244
K, 20% free 3448K/4284K, paused 11ms, total 11ms
07-14 03:57:03.298 32257-32349/pt.ptinovacao.mtom.gw V/HandleRequests: Received
ContentInstances collection notification.
07-14 03:57:03.308 32257-32349/pt.ptinovacao.mtom.gw V/HandleRequests: Valid Container
to Start command.
Container: 1405306620999-ZEPHYR
07-14 03:57:03.308 32257-32257/pt.ptinovacao.mtom.gw I/MainService: Starting
1405306620999-ZEPHYR sensor.
07-14 03:57:03.308 32257-32257/pt.ptinovacao.mtom.gw D/ZephyrHandler: Zephyr MAC
Address: 00:07:80:5A:3C:63
07-14 03:57:03.308 32257-32257/pt.ptinovacao.mtom.gw W/BluetoothAdapter:
getBluetoothService() called with no BluetoothManagerCallback
07-14 03:57:03.318 32257-32257/pt.ptinovacao.mtom.gw D/BluetoothSocket: connect(),
SocketState: INIT, mPfd: {ParcelFileDescriptor: FileDescriptor[74]}
07-14 03:57:04.278 32257-32257/pt.ptinovacao.mtom.gw I/ZephyrHandler: Zephyr connected
07-14 03:57:04.288 32257-32688/pt.ptinovacao.mtom.gw I/ZEPHYR: Connected to BioHarness
HXM017399.
07-14 03:57:04.828 32257-32281/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.1784336; Longitude: -8.5952627; Accuracy: 20.0.
07-14 03:57:05.098 32257-32691/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:05.338 32257-32257/pt.ptinovacao.mtom.gw W/ZephyrHandler: Timestamp empty.
07-14 03:57:08.098 32257-32706/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:11.118 32257-32712/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:13.318 32257-32280/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.

```

```

07-14 03:57:13.358 32257-32279/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .
07-14 03:57:13.448 32257-32348/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 03:57:14.118 32257-32738/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:17.128 32257-32747/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:19.428 32257-32257/pt.ptinovacao.mtom.gw W/ConnectionListener: Network
connectivity change
07-14 03:57:19.438 32257-32257/pt.ptinovacao.mtom.gw D/ConnectionChecker: No network
connectivity
07-14 03:57:19.438 32257-32257/pt.ptinovacao.mtom.gw D/ProtocolManager: No connection.
Resetting stored IPV4 address.
07-14 03:57:20.128 32257-357/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:23.128 32257-392/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:23.328 32257-32280/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 03:57:23.378 32257-32276/pt.ptinovacao.mtom.gw I/ProtocolManager: No Internet
Connection: Can't send data. Saving to Database.
07-14 03:57:26.128 32257-426/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:29.158 32257-437/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:32.168 32257-438/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:33.338 32257-32280/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 03:57:33.358 32257-32276/pt.ptinovacao.mtom.gw I/ProtocolManager: No Internet
Connection: Can't send data. Saving to Database.
07-14 03:57:35.038 32257-32281/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.1784385; Longitude: -8.5952659; Accuracy: 20.0.
07-14 03:57:35.178 32257-440/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:36.328 32257-32257/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 506
K, 21% free 3454K/4328K, paused 18ms, total 19ms
07-14 03:57:38.178 32257-480/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:41.188 32257-531/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:43.348 32257-32280/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 03:57:43.378 32257-32279/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .
07-14 03:57:43.428 32257-32348/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 03:57:43.428 32257-32348/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nscl.
m2m.ptinovacao.pt
07-14 03:57:43.558 32257-32348/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
connection with phonegw.nscl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:57:43.618 32257-32348/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 03:57:44.188 32257-546/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:44.698 32257-32257/pt.ptinovacao.mtom.gw W/ConnectionListener: Network
connectivity change
07-14 03:57:44.708 32257-32257/pt.ptinovacao.mtom.gw I/ConnectionChecker: Network WIFI
connected
07-14 03:57:44.708 32257-32276/pt.ptinovacao.mtom.gw I/ProtocolManager: Connection
returned, reading stored data... Size: 2386
07-14 03:57:44.708 32257-32276/pt.ptinovacao.mtom.gw V/ProtocolManager: Reached end of
file.
07-14 03:57:44.718 32257-32276/pt.ptinovacao.mtom.gw V/ProtocolManager: File
successfully erased.
07-14 03:57:44.718 32257-32279/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .

```

```

07-14 03:57:44.718 32257-32279/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .
07-14 03:57:44.828 32257-32348/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 03:57:44.958 32257-32348/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 308
K, 20% free 3469K/4328K, paused 10ms, total 10ms
07-14 03:57:44.958 32257-32348/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 03:57:47.198 32257-677/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:50.198 32257-759/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:53.198 32257-784/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:53.358 32257-32280/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 03:57:53.368 32257-32279/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .
07-14 03:57:53.448 32257-32348/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 03:57:56.198 32257-848/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:57:59.078 32257-32257/pt.ptinovacao.mtom.gw W/ConnectionListener: Network
connectivity change
07-14 03:57:59.088 32257-32257/pt.ptinovacao.mtom.gw D/ConnectionChecker: No network
connectivity
07-14 03:57:59.088 32257-32257/pt.ptinovacao.mtom.gw D/ProtocolManager: No connection.
Resetting stored IPV4 address.
07-14 03:57:59.208 32257-880/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:58:02.208 32257-913/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:58:03.368 32257-32280/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 03:58:03.398 32257-32276/pt.ptinovacao.mtom.gw I/ProtocolManager: No Internet
Connection: Can't send data. Saving to Database.
07-14 03:58:05.208 32257-945/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:58:05.318 32257-32281/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.1784154; Longitude: -8.5953212; Accuracy: 43.754.
07-14 03:58:08.208 32257-962/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:58:11.208 32257-977/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:58:13.378 32257-32280/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 03:58:13.458 32257-32276/pt.ptinovacao.mtom.gw I/ProtocolManager: No Internet
Connection: Can't send data. Saving to Database.
07-14 03:58:14.208 32257-982/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:58:14.658 32257-32257/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 362
K, 21% free 3460K/4328K, paused 16ms, total 17ms
07-14 03:58:14.658 32257-32257/pt.ptinovacao.mtom.gw I/dalvikvm-heap: Grow heap (frag
case) to 3.813MB for 82384-byte allocation
07-14 03:58:14.698 32257-32268/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed <1K
, 20% free 3540K/4412K, paused 33ms, total 33ms
07-14 03:58:14.698 32257-32257/pt.ptinovacao.mtom.gw D/MainService: M2M Gateway
Stopping
07-14 03:58:14.708 32257-32257/pt.ptinovacao.mtom.gw D/ProtocolManager: ProtocolManager
Service Stopping
07-14 03:58:14.708 32257-32257/pt.ptinovacao.mtom.gw D/ProtocolManager: Temporary
storage file has information. Not being deleted.
07-14 03:58:14.708 32257-32257/pt.ptinovacao.mtom.gw D/GSCL: GSCL Stopping
07-14 03:58:14.788 32257-32257/pt.ptinovacao.mtom.gw D/GSCL: Successfully saved state
to storage.
07-14 03:58:14.788 32257-32257/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound External
Traffic: 9551. Inbound External Traffic: 9680
07-14 03:58:14.788 32257-32257/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound Internal
Traffic: 0. Inbound Internal Traffic: 0

```

```

07-14 03:58:14.788 32257-32348/pt.ptinovacao.mtom.gw D/HTTPClient: HTTP Client Shut
Down
07-14 03:58:14.788 32257-32257/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound External
Traffic: 5048. Inbound External Traffic: 5378
07-14 03:58:14.788 32257-32257/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound Internal
Traffic: 0. Inbound Internal Traffic: 0
07-14 03:58:14.788 32257-32257/pt.ptinovacao.mtom.gw D/GPS: GPS stopping
07-14 03:58:14.788 32257-32257/pt.ptinovacao.mtom.gw D/Sensor Handler: Stopping Sensor
Handler.
07-14 03:58:14.788 32257-32257/pt.ptinovacao.mtom.gw D/ZephyrHandler: Stopping Zephyr
...
07-14 03:58:14.788 32257-32690/pt.ptinovacao.mtom.gw W/System.err: java.io.IOException:
bt socket closed, read return: -1
07-14 03:58:14.788 32257-32690/pt.ptinovacao.mtom.gw W/System.err: at android.bluetooth
.BluetoothSocket.read(BluetoothSocket.java:429)
07-14 03:58:14.788 32257-32690/pt.ptinovacao.mtom.gw W/System.err: at android.bluetooth
.BluetoothInputStream.read(BluetoothInputStream.java:96)
07-14 03:58:14.788 32257-32690/pt.ptinovacao.mtom.gw W/System.err: at java.io.
InputStream.read(InputStream.java:162)
07-14 03:58:14.788 32257-32690/pt.ptinovacao.mtom.gw W/System.err: at zephyr.android.
HxMBT.BTComms.run(BTComms.java:103)
07-14 03:58:14.788 32257-32257/pt.ptinovacao.mtom.gw D/BluetoothManager: BT Manager
Stopping..
07-14 03:58:14.788 32257-32257/pt.ptinovacao.mtom.gw D/MemoryManager: MemoryManager
stopping

```

— APPLICATION RESTART —

```

07-14 03:58:56.428 1155-1155/pt.ptinovacao.mtom.gw D/dalvikvm: Late-enabling CheckJNI
07-14 03:58:56.578 1155-1155/pt.ptinovacao.mtom.gw I/System.out: Sending WAIT chunk
07-14 03:58:56.578 1155-1155/pt.ptinovacao.mtom.gw W/ActivityThread: Application pt.
ptinovacao.mtom.gw is waiting for the debugger on port 8100...
07-14 03:58:56.588 1155-1161/pt.ptinovacao.mtom.gw I/dalvikvm: Debugger is active
07-14 03:58:56.778 1155-1155/pt.ptinovacao.mtom.gw I/System.out: Debugger has
connected
07-14 03:58:56.778 1155-1155/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle ...
07-14 03:58:56.978 1155-1155/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle ...
07-14 03:58:57.178 1155-1155/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle ...
07-14 03:58:57.378 1155-1155/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle ...
07-14 03:58:57.578 1155-1155/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle ...
07-14 03:58:57.778 1155-1155/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle ...
07-14 03:58:57.988 1155-1155/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle ...
07-14 03:58:58.188 1155-1155/pt.ptinovacao.mtom.gw I/System.out: debugger has settled
(1434)
07-14 03:58:58.538 1155-1439/pt.ptinovacao.mtom.gw D/MainService: Thread Started:
1439
07-14 03:58:58.588 1155-1457/pt.ptinovacao.mtom.gw D/ProtocolManager: Thread Started:
1457
07-14 03:58:58.598 1155-1458/pt.ptinovacao.mtom.gw D/Sensor Handler: Thread Started:
1458
07-14 03:58:58.598 1155-1477/pt.ptinovacao.mtom.gw D/BluetoothManager: Thread Started
: 1477
07-14 03:58:58.608 1155-1476/pt.ptinovacao.mtom.gw D/MemoryManager: Thread Started:
1476
07-14 03:58:58.608 1155-1155/pt.ptinovacao.mtom.gw D/ConnectionChecker: No network
connectivity
07-14 03:58:58.618 1155-1478/pt.ptinovacao.mtom.gw D/GPS: Thread Started: 1478
07-14 03:58:58.618 1155-1155/pt.ptinovacao.mtom.gw D/ProtocolManager: No connection.
Resetting stored IPV4 address.
07-14 03:58:58.618 1155-1155/pt.ptinovacao.mtom.gw V/ConnectionListener:
BroadcastReceiver started successfully.

```

```

07-14 03:58:58.628    1155-1478/pt.ptinovacao.mtom.gw I/GPS: Provider network has been
      selected.
07-14 03:59:06.748    1155-1155/pt.ptinovacao.mtom.gw W/ConnectionListener: Network
      connectivity change
07-14 03:59:06.748    1155-1155/pt.ptinovacao.mtom.gw I/ConnectionChecker: Network WIFI
      connected
07-14 03:59:06.758    1155-1890/pt.ptinovacao.mtom.gw D/GSCL: Thread Started: 1890
07-14 03:59:06.838    1155-1890/pt.ptinovacao.mtom.gw V/GSCL: GSCL Root URI: http://
      mobilelab.fe.up.pt:8080
07-14 03:59:08.128    1155-1890/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 271
      K, 19% free 2879K/3516K, paused 9ms, total 9ms
07-14 03:59:09.638    1155-1890/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 327
      K, 19% free 3065K/3760K, paused 11ms, total 12ms
07-14 03:59:09.698    1155-1890/pt.ptinovacao.mtom.gw D/M2M-BootStrap: GET https://
      phonegw.nsc1.m2m.ptinovacao.pt:8443/bootstrapParamSet HTTP/1.1
07-14 03:59:09.698    1155-1890/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap GET
      Size: 74
07-14 03:59:09.798    1155-1890/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 03:59:09.798    1155-1890/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nsc1.
      m2m.ptinovacao.pt
07-14 03:59:10.068    1155-1890/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
      connection with phonegw.nsc1.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:59:10.338    1155-1890/pt.ptinovacao.mtom.gw D/M2M-BootStrap: RESPONSE HTTP/1.1
      200 OK
07-14 03:59:10.348    1155-1890/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap Answer
      Size: 4307
07-14 03:59:10.428    1155-1890/pt.ptinovacao.mtom.gw D/M2M-BootStrap: Certificate
      Validity: Fri Jan 30 02:59:10 WET 2015
07-14 03:59:12.568    1155-1890/pt.ptinovacao.mtom.gw D/CreateResources: Registering SCL
      .
07-14 03:59:12.578    1155-2127/pt.ptinovacao.mtom.gw D/HTTPServer: Thread Started: 2127
07-14 03:59:12.678    1155-1890/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 373
      K, 19% free 3203K/3944K, paused 9ms, total 15ms
07-14 03:59:12.928    1155-1890/pt.ptinovacao.mtom.gw I/CreateResources: Registering scl
      : hammerhead-04506dfa25231287-TesteFEUP to: /m2m/scls .
07-14 03:59:12.938    1155-1890/pt.ptinovacao.mtom.gw I/ProtocolManager: Reading stored
      data ... Size: 2395
07-14 03:59:12.988    1155-1890/pt.ptinovacao.mtom.gw V/ProtocolManager: Reached end of
      file .
07-14 03:59:12.998    1155-1890/pt.ptinovacao.mtom.gw V/ProtocolManager: File
      successfully erased.
07-14 03:59:13.038    1155-2126/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 03:59:13.038    1155-2126/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nsc1.
      m2m.ptinovacao.pt
07-14 03:59:13.298    1155-1890/pt.ptinovacao.mtom.gw D/CreateResources: Registering
      contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
      HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .
07-14 03:59:13.308    1155-1890/pt.ptinovacao.mtom.gw D/CreateResources: Registering
      contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
      HealthSensors/containers/1405306620999-ZEPHYR/contentInstances .
07-14 03:59:13.498    1155-2126/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 437
      K, 20% free 3280K/4084K, paused 9ms, total 9ms
07-14 03:59:13.778    1155-2126/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
      connection with phonegw.nsc1.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:59:13.998    1155-2126/pt.ptinovacao.mtom.gw W/HttpClient: RESPONSE HTTP/1.1
      405 Method Not Allowed
07-14 03:59:14.008    1155-1890/pt.ptinovacao.mtom.gw D/GSCL: SCL already exists.
      Retrieving Scls.
07-14 03:59:14.178    1155-2126/pt.ptinovacao.mtom.gw D/CreateResources: Content
      Instance successfully registered.
07-14 03:59:14.278    1155-1890/pt.ptinovacao.mtom.gw D/GSCL: Memory database
      successfully read!
07-14 03:59:14.378    1155-2126/pt.ptinovacao.mtom.gw D/CreateResources: Content
      Instance successfully registered.
07-14 03:59:20.188    1155-1155/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 365
      K, 20% free 3338K/4144K, paused 26ms, total 27ms
07-14 03:59:20.188    1155-1155/pt.ptinovacao.mtom.gw I/dalvikvm-heap: Grow heap (frag
      case) to 3.694MB for 82384-byte allocation

```

```

07-14 03:59:20.208    1155-1164/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 0K,
    20% free 3419K/4228K, paused 17ms, total 17ms
07-14 03:59:20.228    1155-1155/pt.ptinovacao.mtom.gw D/MainService: M2M Gateway
    Stopping
07-14 03:59:20.238    1155-1155/pt.ptinovacao.mtom.gw D/ProtocolManager: ProtocolManager
    Service Stopping
07-14 03:59:20.238    1155-1155/pt.ptinovacao.mtom.gw I/ProtocolManager: Temporary
    storage file successfully deleted.
07-14 03:59:20.238    1155-1155/pt.ptinovacao.mtom.gw D/GSCL: GSCL Stopping
07-14 03:59:20.558    1155-1155/pt.ptinovacao.mtom.gw D/GSCL: Successfully saved state
    to storage.
07-14 03:59:20.568    1155-1155/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound External
    Traffic: 3890. Inbound External Traffic: 3658
07-14 03:59:20.568    1155-1155/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound Internal
    Traffic: 0. Inbound Internal Traffic: 0
07-14 03:59:20.568    1155-1155/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound External
    Traffic: 0. Inbound External Traffic: 0
07-14 03:59:20.568    1155-1155/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound Internal
    Traffic: 0. Inbound Internal Traffic: 0
07-14 03:59:20.568    1155-2126/pt.ptinovacao.mtom.gw D/HTTPClient: HTTP Client Shut
    Down
07-14 03:59:20.568    1155-1155/pt.ptinovacao.mtom.gw D/GPS: GPS stopping
07-14 03:59:20.568    1155-1155/pt.ptinovacao.mtom.gw D/Sensor Handler: Stopping Sensor
    Handler.
07-14 03:59:20.568    1155-1155/pt.ptinovacao.mtom.gw D/BluetoothManager: BT Manager
    Stopping..
07-14 03:59:20.568    1155-1155/pt.ptinovacao.mtom.gw D/MemoryManager: MemoryManager
    stopping
07-14 03:59:20.608    1155-1155/pt.ptinovacao.mtom.gw I/MainService: Stopping Service.
    Goodbye!
07-14 03:59:20.608    1155-1155/pt.ptinovacao.mtom.gw I/Process: Sending signal. PID:
    1155 SIG: 9

```

B.4 Disconnection after using local API

```

07-14 03:33:35.698    25740-25755/pt.ptinovacao.mtom.gw D/MainService: Thread Started:
    25755
07-14 03:33:35.718    25740-25755/pt.ptinovacao.mtom.gw I/ProtocolManager: Storage file
    created successfully!
07-14 03:33:35.728    25740-25767/pt.ptinovacao.mtom.gw D/ProtocolManager: Thread Started:
    25767
07-14 03:33:35.738    25740-25769/pt.ptinovacao.mtom.gw D/MemoryManager: Thread Started:
    25769
07-14 03:33:35.738    25740-25768/pt.ptinovacao.mtom.gw D/Sensor Handler: Thread Started:
    25768
07-14 03:33:35.738    25740-25772/pt.ptinovacao.mtom.gw D/BluetoothManager: Thread Started
    : 25772
07-14 03:33:35.748    25740-25770/pt.ptinovacao.mtom.gw D/GPS: Thread Started: 25770
07-14 03:33:35.758    25740-25770/pt.ptinovacao.mtom.gw I/GPS: Provider network has been
    selected.
07-14 03:33:35.758    25740-25773/pt.ptinovacao.mtom.gw D/GSCL: Thread Started: 25773
07-14 03:33:35.758    25740-25740/pt.ptinovacao.mtom.gw I/ConnectionChecker: Network WIFI
    connected
07-14 03:33:35.758    25740-25740/pt.ptinovacao.mtom.gw V/ConnectionListener:
    BroadcastReceiver started successfully.
07-14 03:33:35.758    25740-25767/pt.ptinovacao.mtom.gw D/ProtocolManager: Storage file
    empty. Nothing to flush.
07-14 03:33:35.858    25740-25773/pt.ptinovacao.mtom.gw V/GSCL: GSCL Root URI: http://
    mobilelab.fe.up.pt:8080
07-14 03:33:35.948    25740-25773/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 294
    K, 19% free 2856K/3516K, paused 9ms, total 9ms
07-14 03:33:36.178    25740-25773/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 311
    K, 19% free 3057K/3736K, paused 10ms, total 10ms
07-14 03:33:36.288    25740-25773/pt.ptinovacao.mtom.gw D/M2M-BootStrap: GET https://
    phonegw.nsc1.m2m.ptinovacao.pt:8443/bootstrapParamSet HTTP/1.1

```



```

07-14 03:33:36.288 25740-25773/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap GET
    Size: 74
07-14 03:33:36.498 25740-25773/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 03:33:36.498 25740-25773/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nsc1.
    m2m.ptinovacao.pt
07-14 03:33:37.038 25740-25773/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
    connection with phonegw.nsc1.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:33:37.408 25740-25773/pt.ptinovacao.mtom.gw D/M2M-BootStrap: RESPONSE HTTP/1.1
    200 OK
07-14 03:33:37.418 25740-25773/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap Answer
    Size: 4307
07-14 03:33:37.428 25740-25773/pt.ptinovacao.mtom.gw D/M2M-BootStrap: Certificate
    Validity: Fri Jan 30 02:33:37 WET 2015
07-14 03:33:37.548 25740-25773/pt.ptinovacao.mtom.gw D/CreateResources: Registering SCL
    .
07-14 03:33:37.568 25740-25773/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 391
    K, 20% free 3177K/3936K, paused 9ms, total 9ms
07-14 03:33:37.568 25740-25798/pt.ptinovacao.mtom.gw D/HTTPServer: Thread Started:
    25798
07-14 03:33:37.608 25740-25773/pt.ptinovacao.mtom.gw I/CreateResources: Registering scl
    : hammerhead-04506dfa25231287-TesteFEUP to: /m2m/scls .
07-14 03:33:37.608 25740-25773/pt.ptinovacao.mtom.gw I/GSCL: Local API file created
    successfully!
07-14 03:33:37.758 25740-25797/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 03:33:37.758 25740-25797/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nsc1.
    m2m.ptinovacao.pt
07-14 03:33:38.308 25740-25797/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
    connection with phonegw.nsc1.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:33:38.488 25740-25797/pt.ptinovacao.mtom.gw W/HTTPClient: RESPONSE HTTP/1.1
    405 Method Not Allowed
07-14 03:33:38.488 25740-25773/pt.ptinovacao.mtom.gw D/GSCL: SCL already exists.
    Retrieving Scls.
07-14 03:33:38.488 25740-25773/pt.ptinovacao.mtom.gw D/GSCL: Memory database
    successfully read!
07-14 03:33:38.498 25740-25773/pt.ptinovacao.mtom.gw W/GSCL: Stored SclStorage is null.
    Re-creating.
07-14 03:33:38.498 25740-25773/pt.ptinovacao.mtom.gw I/RetrieveResources: Retrieving
    Container: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP
07-14 03:33:38.498 25740-25773/pt.ptinovacao.mtom.gw I/RetrieveResources: Retrieving
    Container: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
    HealthSensors
07-14 03:33:38.498 25740-25797/pt.ptinovacao.mtom.gw I/HTTPClient: GET sent to domain:
    /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP
07-14 03:33:39.048 25740-25797/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 417
    K, 20% free 3263K/4048K, paused 14ms, total 14ms
07-14 03:33:39.048 25740-25797/pt.ptinovacao.mtom.gw I/HTTPClient: GET sent to domain:
    /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/HealthSensors
07-14 03:33:39.108 25740-25773/pt.ptinovacao.mtom.gw I/GSCL: SCL stored successfully.
07-14 03:33:39.118 25740-25770/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
    41.1784629; Longitude: -8.5953346; Accuracy: 43.104.
07-14 03:33:39.468 25740-25773/pt.ptinovacao.mtom.gw I/GSCL: Local Application
    HealthSensors stored successfully.
07-14 03:33:49.278 25740-25798/pt.ptinovacao.mtom.gw D/HandleRequests: Local connection
    . Received Command.
07-14 03:33:49.338 25740-25798/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 432
    K, 20% free 3340K/4140K, paused 11ms, total 11ms
07-14 03:33:49.378 25740-25798/pt.ptinovacao.mtom.gw V/HandleRequests: Valid Search
    Sensors Command.
    Bluetooth: trueSearch Internal: true
07-14 03:33:49.378 25740-25740/pt.ptinovacao.mtom.gw I/MainService: MainService Started
    with Intent SEARCH_SENSORS
07-14 03:33:51.378 25740-25740/pt.ptinovacao.mtom.gw D/BluetoothManager: Found Zephyr
    BT device: HXM017399
07-14 03:33:51.378 25740-25740/pt.ptinovacao.mtom.gw V/BluetoothManager: Zephyr bonded.
    Starting ZephyrHandler...
07-14 03:33:51.378 25740-25740/pt.ptinovacao.mtom.gw V/BluetoothManager: RSSI: -45

```

```

07-14 03:34:03.018 25740-25773/pt.ptinovacao.mtom.gw I/CreateResources: Registering
container: 1405305242955-ZEPHYR to: /m2m/localAPI/newContainer/1405305242955-ZEPHYR
.
07-14 03:34:03.068 25740-25797/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 453
K, 20% free 3385K/4212K, paused 22ms, total 22ms
07-14 03:34:08.188 25740-25798/pt.ptinovacao.mtom.gw D/HandleRequests: Local connection
. Received Command.
07-14 03:34:08.188 25740-25798/pt.ptinovacao.mtom.gw V/HandleRequests: Valid Container
to Start command.
Container: 1405305242955-ZEPHYR
07-14 03:34:08.188 25740-25740/pt.ptinovacao.mtom.gw I/MainService: Starting
1405305242955-ZEPHYR sensor.
07-14 03:34:08.188 25740-25740/pt.ptinovacao.mtom.gw D/ZephyrHandler: Zephyr MAC
Address: 00:07:80:5A:3C:63
07-14 03:34:08.198 25740-25740/pt.ptinovacao.mtom.gw D/BluetoothSocket: connect(),
SocketState: INIT, mPfd: {ParcelFileDescriptor: FileDescriptor[71]}
07-14 03:34:10.198 25740-25740/pt.ptinovacao.mtom.gw I/ZephyrHandler: Zephyr connected
07-14 03:34:10.208 25740-25997/pt.ptinovacao.mtom.gw I/ZEPHYR: Connected to BioHarness
HXM017399.
07-14 03:34:11.008 25740-26000/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:11.088 25740-25740/pt.ptinovacao.mtom.gw W/ZephyrHandler: Timestamp empty.
07-14 03:34:12.788 25740-25770/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.178454; Longitude: -8.5953345; Accuracy: 43.024.
07-14 03:34:14.028 25740-26005/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:17.038 25740-26025/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:18.198 25740-25769/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 03:34:18.248 25740-25773/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/localAPI/contentInstance/1405305242955-ZEPHYR .
07-14 03:34:18.698 25740-25797/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 03:34:20.058 25740-26035/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:23.078 25740-26046/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:26.078 25740-26066/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:28.208 25740-25769/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 03:34:28.268 25740-25773/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/localAPI/contentInstance/1405305242955-ZEPHYR .
07-14 03:34:28.298 25740-25797/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 426
K, 20% free 3436K/4256K, paused 12ms, total 13ms
07-14 03:34:28.388 25740-25797/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 03:34:29.078 25740-26096/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:32.088 25740-26134/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:35.108 25740-26136/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:38.148 25740-26137/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:38.218 25740-25769/pt.ptinovacao.mtom.gw D/MemoryManager: Flushing ZEPHYR.
07-14 03:34:38.238 25740-25773/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/localAPI/contentInstance/1405305242955-ZEPHYR .
07-14 03:34:38.348 25740-25797/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 03:34:41.158 25740-26138/pt.ptinovacao.mtom.gw I/System.out: Sending life sign
packet.
07-14 03:34:42.408 25740-25740/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 258
K, 20% free 3427K/4256K, paused 16ms, total 16ms
07-14 03:34:42.408 25740-25740/pt.ptinovacao.mtom.gw I/dalvikvm-heap: Grow heap (frag
case) to 3.781MB for 82384-byte allocation
07-14 03:34:42.428 25740-25750/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed <1K
, 20% free 3507K/4340K, paused 24ms, total 24ms

```

```

07-14 03:34:42.448 25740-25740/pt.ptinovacao.mtom.gw D/MainService: M2M Gateway
Stopping
07-14 03:34:42.448 25740-25740/pt.ptinovacao.mtom.gw D/ProtocolManager: ProtocolManager
Service Stopping
07-14 03:34:42.448 25740-25740/pt.ptinovacao.mtom.gw I/ProtocolManager: Temporary
storage file successfully deleted.
07-14 03:34:42.448 25740-25740/pt.ptinovacao.mtom.gw D/GSCL: GSCL Stopping
07-14 03:34:42.528 25740-25740/pt.ptinovacao.mtom.gw D/GSCL: Successfully saved state
to storage.
07-14 03:34:42.578 25740-25740/pt.ptinovacao.mtom.gw D/GSCL: Successfully saved local
API data to storage.
07-14 03:34:42.578 25740-25740/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound External
Traffic: 926. Inbound External Traffic: 2634
07-14 03:34:42.578 25740-25740/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound Internal
Traffic: 5079. Inbound Internal Traffic: 4819
07-14 03:34:42.588 25740-25797/pt.ptinovacao.mtom.gw D/HTTPClient: HTTP Client Shut
Down
07-14 03:34:42.588 25740-25740/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound External
Traffic: 0. Inbound External Traffic: 0
07-14 03:34:42.588 25740-25740/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound Internal
Traffic: 974. Inbound Internal Traffic: 1384
07-14 03:34:42.588 25740-25740/pt.ptinovacao.mtom.gw D/GPS: GPS stopping
07-14 03:34:42.588 25740-25740/pt.ptinovacao.mtom.gw D/Sensor Handler: Stopping Sensor
Handler.
07-14 03:34:42.588 25740-25740/pt.ptinovacao.mtom.gw D/ZephyrHandler: Stopping Zephyr
...
07-14 03:34:42.588 25740-25740/pt.ptinovacao.mtom.gw D/BluetoothManager: BT Manager
Stopping..
07-14 03:34:42.588 25740-25740/pt.ptinovacao.mtom.gw D/MemoryManager: MemoryManager
stopping
07-14 03:34:42.598 25740-25740/pt.ptinovacao.mtom.gw I/MainService: Stopping Service.
Goodbye!
07-14 03:34:42.598 25740-25740/pt.ptinovacao.mtom.gw I/Process: Sending signal. PID:
25740 SIG: 9

```

— APPLICATION RESTART —

```

07-14 03:36:06.278 26363-26363/pt.ptinovacao.mtom.gw D/dalvikvm: Late-enabling CheckJNI
07-14 03:36:06.578 26363-26363/pt.ptinovacao.mtom.gw I/System.out: Sending WAIT chunk
07-14 03:36:06.578 26363-26363/pt.ptinovacao.mtom.gw W/ActivityThread: Application pt.
ptinovacao.mtom.gw is waiting for the debugger on port 8100...
07-14 03:36:06.588 26363-26369/pt.ptinovacao.mtom.gw I/dalvikvm: Debugger is active
07-14 03:36:06.778 26363-26363/pt.ptinovacao.mtom.gw I/System.out: Debugger has
connected
07-14 03:36:06.778 26363-26363/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle...
07-14 03:36:06.978 26363-26363/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle...
07-14 03:36:07.178 26363-26363/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle...
07-14 03:36:07.378 26363-26363/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle...
07-14 03:36:07.578 26363-26363/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle...
07-14 03:36:07.778 26363-26363/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle...
07-14 03:36:07.978 26363-26363/pt.ptinovacao.mtom.gw I/System.out: waiting for debugger
to settle...
07-14 03:36:08.178 26363-26363/pt.ptinovacao.mtom.gw I/System.out: debugger has settled
(1400)
07-14 03:36:08.538 26363-26511/pt.ptinovacao.mtom.gw D/MainService: Thread Started:
26511
07-14 03:36:13.068 26363-26511/pt.ptinovacao.mtom.gw I/ProtocolManager: Storage file
created successfully!
07-14 03:36:13.088 26363-26729/pt.ptinovacao.mtom.gw D/ProtocolManager: Thread Started:
26729
07-14 03:36:13.098 26363-26730/pt.ptinovacao.mtom.gw D/Sensor Handler: Thread Started:
26730

```

```

07-14 03:36:13.098 26363-26732/pt.ptinovacao.mtom.gw D/BluetoothManager: Thread Started
: 26732
07-14 03:36:13.108 26363-26363/pt.ptinovacao.mtom.gw I/ConnectionChecker: Network WIFI
connected
07-14 03:36:13.108 26363-26363/pt.ptinovacao.mtom.gw V/ConnectionListener:
BroadcastReceiver started successfully.
07-14 03:36:13.108 26363-26731/pt.ptinovacao.mtom.gw D/MemoryManager: Thread Started:
26731
07-14 03:36:13.118 26363-26733/pt.ptinovacao.mtom.gw D/GPS: Thread Started: 26733
07-14 03:36:13.128 26363-26733/pt.ptinovacao.mtom.gw I/GPS: Provider network has been
selected.
07-14 03:36:14.958 26363-26729/pt.ptinovacao.mtom.gw D/ProtocolManager: Storage file
empty. Nothing to flush.
07-14 03:36:14.968 26363-26763/pt.ptinovacao.mtom.gw D/GSCL: Thread Started: 26763
07-14 03:36:15.018 26363-26763/pt.ptinovacao.mtom.gw V/GSCL: GSCL Root URI: http://
mobilelab.fe.up.pt:8080
07-14 03:36:15.318 26363-26763/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 298
K, 19% free 2851K/3516K, paused 9ms, total 10ms
07-14 03:36:16.468 26363-26733/pt.ptinovacao.mtom.gw V/GPS: Location updated! Latitude:
41.1784473; Longitude: -8.5953513; Accuracy: 41.801.
07-14 03:36:16.988 26363-26763/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 308
K, 19% free 3056K/3732K, paused 7ms, total 7ms
07-14 03:36:17.848 26363-26763/pt.ptinovacao.mtom.gw D/M2M-BootStrap: GET https://
phonegw.nsl.m2m.ptinovacao.pt:8443/bootstrapParamSet HTTP/1.1
07-14 03:36:17.848 26363-26763/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap GET
Size: 74
07-14 03:36:18.018 26363-26763/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 03:36:18.028 26363-26763/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 03:36:18.588 26363-26763/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:36:18.968 26363-26763/pt.ptinovacao.mtom.gw D/M2M-BootStrap: RESPONSE HTTP/1.1
200 OK
07-14 03:36:18.978 26363-26763/pt.ptinovacao.mtom.gw V/M2M-BootStrap: Bootstrap Answer
Size: 4307
07-14 03:36:19.048 26363-26763/pt.ptinovacao.mtom.gw D/M2M-BootStrap: Certificate
Validity: Fri Jan 30 02:36:19 WET 2015
07-14 03:36:20.488 26363-26763/pt.ptinovacao.mtom.gw D/CreateResources: Registering SCL
.
07-14 03:36:20.508 26363-26830/pt.ptinovacao.mtom.gw D/HTTPServer: Thread Started:
26830
07-14 03:36:20.528 26363-26763/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 390
K, 20% free 3178K/3936K, paused 9ms, total 10ms
07-14 03:36:20.798 26363-26763/pt.ptinovacao.mtom.gw I/CreateResources: Registering scl
: hammerhead-04506dfa25231287-TesteFEUP to: /m2m/scls .
07-14 03:36:20.868 26363-26829/pt.ptinovacao.mtom.gw I/TLS_SNI: Setting SNI hostname
07-14 03:36:20.868 26363-26829/pt.ptinovacao.mtom.gw D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 03:36:20.948 26363-26763/pt.ptinovacao.mtom.gw D/GSCL: Memory database
successfully read!
07-14 03:36:21.008 26363-26763/pt.ptinovacao.mtom.gw V/GSCL: Local API file
successfully erased.
07-14 03:36:21.098 26363-26763/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 426
K, 20% free 3264K/4056K, paused 10ms, total 11ms
07-14 03:36:21.358 26363-26763/pt.ptinovacao.mtom.gw I/CreateResources: Registering
container: 1405305242955-ZEPHYR to: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/
applications/HealthSensors/containers .
07-14 03:36:21.528 26363-26763/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
HealthSensors/containers/1405305242955-ZEPHYR/contentInstances .
07-14 03:36:21.548 26363-26763/pt.ptinovacao.mtom.gw D/CreateResources: Registering
contentInstance to /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/
HealthSensors/containers/1405305242955-ZEPHYR/contentInstances .
07-14 03:36:21.718 26363-26829/pt.ptinovacao.mtom.gw I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:36:21.978 26363-26829/pt.ptinovacao.mtom.gw W/HTTPClient: RESPONSE HTTP/1.1
405 Method Not Allowed

```

```
07-14 03:36:21.998 26363-26763/pt.ptinovacao.mtom.gw D/GSCL: SCL already exists.
Retrieving Scls.
07-14 03:36:22.148 26363-26763/pt.ptinovacao.mtom.gw D/GSCL: Memory database
successfully read!
07-14 03:36:22.348 26363-26829/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 408
K, 19% free 3366K/4144K, paused 15ms, total 15ms
07-14 03:36:22.468 26363-26829/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 03:36:22.558 26363-26763/pt.ptinovacao.mtom.gw I/GSCL: Local Container
1405305242955-ZEPHYR created successfully.
07-14 03:36:22.668 26363-26829/pt.ptinovacao.mtom.gw D/CreateResources: Content
Instance successfully registered.
07-14 03:36:32.678 26363-26363/pt.ptinovacao.mtom.gw D/MainService: M2M Gateway
Stopping
07-14 03:36:32.678 26363-26363/pt.ptinovacao.mtom.gw D/ProtocolManager: ProtocolManager
Service Stopping
07-14 03:36:32.678 26363-26363/pt.ptinovacao.mtom.gw I/ProtocolManager: Temporary
storage file successfully deleted.
07-14 03:36:32.678 26363-26363/pt.ptinovacao.mtom.gw D/GSCL: GSCL Stopping
07-14 03:36:32.928 26363-26363/pt.ptinovacao.mtom.gw D/dalvikvm: GC_FOR_ALLOC freed 385
K, 18% free 3486K/4244K, paused 12ms, total 12ms
07-14 03:36:32.948 26363-26363/pt.ptinovacao.mtom.gw D/GSCL: Successfully saved state
to storage.
07-14 03:36:32.958 26363-26363/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound External
Traffic: 4329. Inbound External Traffic: 4579
07-14 03:36:32.958 26363-26363/pt.ptinovacao.mtom.gw V/HTTPClient: Outbound Internal
Traffic: 0. Inbound Internal Traffic: 0
07-14 03:36:32.958 26363-26363/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound External
Traffic: 0. Inbound External Traffic: 0
07-14 03:36:32.958 26363-26363/pt.ptinovacao.mtom.gw V/ServerTraffic: Outbound Internal
Traffic: 0. Inbound Internal Traffic: 0
07-14 03:36:32.958 26363-26363/pt.ptinovacao.mtom.gw D/GPS: GPS stopping
07-14 03:36:32.958 26363-26363/pt.ptinovacao.mtom.gw D/Sensor Handler: Stopping Sensor
Handler.
07-14 03:36:32.958 26363-26829/pt.ptinovacao.mtom.gw D/HTTPClient: HTTP Client Shut
Down
07-14 03:36:32.958 26363-26363/pt.ptinovacao.mtom.gw D/BluetoothManager: BT Manager
Stopping..
07-14 03:36:32.958 26363-26363/pt.ptinovacao.mtom.gw D/MemoryManager: MemoryManager
stopping
```


Appendix C

Mobile M2M NA Logs

This appendix shows the logs of the mobile M2M NA running and communicating through the NSCL (in Section C.1) and through the local API (in Section C.2).

C.1 NSCL

```
07-14 02:46:33.818 8058-8058/pt.ptinovacao.mtom.na D/dalvikvm: Late-enabling CheckJNI
07-14 02:46:33.908 8058-8058/pt.ptinovacao.mtom.na I/System.out: Sending WAIT chunk
07-14 02:46:33.908 8058-8058/pt.ptinovacao.mtom.na W/ActivityThread: Application pt.
ptinovacao.mtom.na is waiting for the debugger on port 8100...
07-14 02:46:33.918 8058-8064/pt.ptinovacao.mtom.na I/dalvikvm: Debugger is active
07-14 02:46:34.108 8058-8058/pt.ptinovacao.mtom.na I/System.out: Debugger has
connected
07-14 02:46:34.108 8058-8058/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle...
07-14 02:46:34.308 8058-8058/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle...
07-14 02:46:34.508 8058-8058/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle...
07-14 02:46:34.708 8058-8058/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle...
07-14 02:46:34.908 8058-8058/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle...
07-14 02:46:35.108 8058-8058/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle...
07-14 02:46:35.308 8058-8058/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle...
07-14 02:46:35.508 8058-8058/pt.ptinovacao.mtom.na I/System.out: debugger has settled
(1435)
07-14 02:46:35.628 8058-8058/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 115
K, 15% free 2975K/3460K, paused 11ms, total 11ms
07-14 02:46:35.658 8058-8058/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 4K,
14% free 3297K/3800K, paused 7ms, total 7ms
07-14 02:46:35.738 8058-8058/pt.ptinovacao.mtom.na I/Adreno-EGL: <
qeglDrvAPI_eglInitialize:320>: EGL 1.4 QUALCOMM Build:
10404c4692afb8623f95c43aeb6d5e13ed4b30ddbDate: 11/06/13
07-14 02:46:35.758 8058-8058/pt.ptinovacao.mtom.na D/OpenGLRenderer: Enabling debug
mode 0
07-14 02:46:40.488 8058-8058/pt.ptinovacao.mtom.na I/StartInterface: Started Mobile
Gateway.
07-14 02:46:40.528 8058-8287/pt.ptinovacao.mtom.na D/MainService: Thread Started:
8287
07-14 02:46:40.538 8058-8290/pt.ptinovacao.mtom.na D/InternetManager: Thread Started:
8290
```

```

07-14 02:46:40.608      8058-8310/pt.ptinovacao.mtom.na D/SCL: Thread Started: 8310
07-14 02:46:40.828      8058-8310/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 114
K, 12% free 3696K/4176K, paused 9ms, total 9ms
07-14 02:46:40.838      8058-8310/pt.ptinovacao.mtom.na V/SCL:NA_SCL Root URI: http://
mobilelab.fe.up.pt:9090
07-14 02:46:40.928      8058-8058/pt.ptinovacao.mtom.na I/ConnectionChecker: Network WIFI
connected
07-14 02:46:40.938      8058-8058/pt.ptinovacao.mtom.na V/ConnectionListener:
BroadcastReceiver started successfully.
07-14 02:46:40.948      8058-8290/pt.ptinovacao.mtom.na D/InternetManager:NA_SCL already
running.
07-14 02:46:43.718      8058-8310/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 247
K, 14% free 3956K/4572K, paused 8ms, total 8ms
07-14 02:46:45.038      8058-8310/pt.ptinovacao.mtom.na D/M2M-BootStrap: GET https://
phonegw.nsl.m2m.ptinovacao.pt:8443/bootstrapParamSet HTTP/1.1
07-14 02:46:45.038      8058-8310/pt.ptinovacao.mtom.na V/M2M-BootStrap: Bootstrap GET
Size: 74
07-14 02:46:45.148      8058-8310/pt.ptinovacao.mtom.na I/TLS_SNI: Setting SNI hostname
07-14 02:46:45.148      8058-8310/pt.ptinovacao.mtom.na D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 02:46:45.438      8058-8310/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 366
K, 16% free 4103K/4836K, paused 9ms, total 10ms
07-14 02:46:45.718      8058-8310/pt.ptinovacao.mtom.na I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 02:46:46.078      8058-8310/pt.ptinovacao.mtom.na D/M2M-BootStrap: RESPONSE HTTP/1.1
200 OK
07-14 02:46:46.098      8058-8310/pt.ptinovacao.mtom.na V/M2M-BootStrap: Bootstrap Answer
Size: 4299
07-14 02:46:46.168      8058-8310/pt.ptinovacao.mtom.na D/M2M-BootStrap: Certificate
Validity: Fri Jan 30 01:46:46 WET 2015
07-14 02:46:47.518      8058-8389/pt.ptinovacao.mtom.na D/HTTPServer: Thread Started: 8389
07-14 02:46:48.768      8058-8310/pt.ptinovacao.mtom.na I/CreateResources: Registering
application: HealthConsumerNA-hammerhead-04506dfa25231287-TesteFEUP to: /m2m/
applications.
07-14 02:46:48.858      8058-8388/pt.ptinovacao.mtom.na I/TLS_SNI: Setting SNI hostname
07-14 02:46:48.858      8058-8388/pt.ptinovacao.mtom.na D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 02:46:49.118      8058-8388/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 349
K, 15% free 4267K/4984K, paused 10ms, total 10ms
07-14 02:46:49.528      8058-8388/pt.ptinovacao.mtom.na I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 02:46:49.908      8058-8388/pt.ptinovacao.mtom.na W/HTTPClient: RESPONSE HTTP/1.1
405 Method Not Allowed
07-14 02:46:49.928      8058-8310/pt.ptinovacao.mtom.na D/SCL: Application already exists.
Restoring from storage...
07-14 02:46:50.088      8058-8310/pt.ptinovacao.mtom.na D/SCL: Memory database
successfully read!
07-14 02:46:50.748      8058-8310/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 586
K, 19% free 4225K/5180K, paused 12ms, total 12ms
07-14 02:46:50.878      8058-8310/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving
Applications: /m2m/scls/TesteTrafego2hammerhead-04506dfa25231287-TesteFEUP/
applications
07-14 02:46:50.888      8058-8310/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving
Applications: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications
07-14 02:46:50.898      8058-8310/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving
Applications: /m2m/scls/hammerhead-04506dfa25231287-AssignedIP/applications
07-14 02:46:52.818      8058-8388/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 507
K, 18% free 4250K/5180K, paused 13ms, total 13ms
07-14 02:46:53.178      8058-8310/pt.ptinovacao.mtom.na D/SCL: All resources checked and
subscription is present.
07-14 02:46:53.388      8058-8310/pt.ptinovacao.mtom.na D/SCL: All resources checked and
subscription is present.
07-14 02:46:53.628      8058-8310/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 495
K, 18% free 4295K/5180K, paused 11ms, total 11ms
07-14 02:46:53.678      8058-8310/pt.ptinovacao.mtom.na D/CreateResources: Registering
contentInstance to /m2m/applications/HealthConsumerNA-hammerhead-04506dfa25231287-
TesteFEUP/containers/ACTIONS/contentInstances.

```



```

07-14 02:47:12.248 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Received
Containers collection notification.
07-14 02:47:12.348 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.348 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.348 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.348 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.348 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.358 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.358 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.358 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.358 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.358 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.358 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.368 8058-8389/pt.ptinovacao.mtom.na V/HandleRequests: Container
created more than two minutes ago. Discarding.
07-14 02:47:12.368 8058-8310/pt.ptinovacao.mtom.na I/SCL: Container 1405302431976-
ZEPHYR is now active.
07-14 02:47:13.968 8058-8058/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 248
K, 17% free 4341K/5180K, paused 11ms, total 11ms
07-14 02:47:13.968 8058-8058/pt.ptinovacao.mtom.na I/dalvikvm-heap: Grow heap (frag
case) to 4.810MB for 225520-byte allocation
07-14 02:47:13.978 8058-8067/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 2K,
16% free 4559K/5404K, paused 11ms, total 11ms
07-14 02:47:13.988 8058-8058/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed <1K
, 16% free 4559K/5404K, paused 12ms, total 12ms
07-14 02:47:13.988 8058-8058/pt.ptinovacao.mtom.na I/dalvikvm-heap: Grow heap (frag
case) to 7.228MB for 2536936-byte allocation
07-14 02:47:14.008 8058-8067/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 0K,
11% free 7037K/7884K, paused 18ms, total 18ms
07-14 02:47:14.998 8058-8058/pt.ptinovacao.mtom.na I/CommandsActivity: Sensor ZEPHYR
chosen to start.
07-14 02:47:15.128 8058-8310/pt.ptinovacao.mtom.na D/CreateResources: Registering
contentInstance to /m2m/applications/HealthConsumerNA-hammerhead-04506dfa25231287-
TesteFEUP/containers/ACTIONS/contentInstances.
07-14 02:47:16.438 8058-8310/pt.ptinovacao.mtom.na V/SCL: Resource still has no
subscriptions. Subscribing...
07-14 02:47:16.508 8058-8310/pt.ptinovacao.mtom.na D/CreateResources: Subscribing to
/m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/HealthSensors/
containers/1405302431976-ZEPHYR/contentInstances/subscriptions
07-14 02:47:16.808 8058-8388/pt.ptinovacao.mtom.na I/CreateResources: Subscription to
/m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/HealthSensors/
containers/1405302431976-ZEPHYR/contentInstances/subscriptions successful.
07-14 02:47:16.868 8058-8389/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 341
K, 8% free 8165K/8872K, paused 12ms, total 12ms
07-14 02:47:25.798 8058-8389/pt.ptinovacao.mtom.na D/TreatSensorData: Measurement
Updated. Array Size: 7
07-14 02:47:35.848 8058-8389/pt.ptinovacao.mtom.na D/TreatSensorData: Measurement
Updated. Array Size: 16
07-14 02:47:44.688 8058-8310/pt.ptinovacao.mtom.na D/CreateResources: Registering
contentInstance to /m2m/applications/HealthConsumerNA-hammerhead-04506dfa25231287-
TesteFEUP/containers/ACTIONS/contentInstances.
07-14 02:47:48.028 8058-8058/pt.ptinovacao.mtom.na I/ShowMeasurements: Sent intent to
stop Mobile Gateway.
07-14 02:47:48.048 8058-8058/pt.ptinovacao.mtom.na D/MainService: M2M NA Stopping
07-14 02:47:48.168 8058-8058/pt.ptinovacao.mtom.na D/InternetManager: InternetManager
Service Stopping
07-14 02:47:48.178 8058-8058/pt.ptinovacao.mtom.na D/SCL:NA_SCL Stopping

```

```

07-14 02:47:48.438 8058-8058/pt.ptinovacao.mtom.na D/SCL: Successfully saved state to
storage.
07-14 02:47:48.438 8058-8058/pt.ptinovacao.mtom.na V/HttpClient: Outbound External
Traffic: 9890. Inbound External Traffic: 11362
07-14 02:47:48.438 8058-8058/pt.ptinovacao.mtom.na V/HttpClient: Outbound Internal
Traffic: 0. Inbound Internal Traffic: 0
07-14 02:47:48.438 8058-8388/pt.ptinovacao.mtom.na D/HttpClient: HTTP Client Shut
Down
07-14 02:47:48.448 8058-8058/pt.ptinovacao.mtom.na V/ServerTraffic: Outbound External
Traffic: 15126. Inbound External Traffic: 15786
07-14 02:47:48.448 8058-8058/pt.ptinovacao.mtom.na V/ServerTraffic: Outbound Internal
Traffic: 0. Inbound Internal Traffic: 0
07-14 02:47:48.488 8058-8058/pt.ptinovacao.mtom.na I/MainService: Stopping Service.
Goodbye!

```

C.2 Local API

```

07-14 02:50:21.088 10482-10482/pt.ptinovacao.mtom.na D/dalvikvm: Late-enabling CheckJNI
07-14 02:50:21.378 10482-10482/pt.ptinovacao.mtom.na I/System.out: Sending WAIT chunk
07-14 02:50:21.378 10482-10488/pt.ptinovacao.mtom.na I/dalvikvm: Debugger is active
07-14 02:50:21.378 10482-10482/pt.ptinovacao.mtom.na W/ActivityThread: Application pt.
ptinovacao.mtom.na is waiting for the debugger on port 8100...
07-14 02:50:21.578 10482-10482/pt.ptinovacao.mtom.na I/System.out: Debugger has
connected
07-14 02:50:21.578 10482-10482/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle ...
07-14 02:50:21.778 10482-10482/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle ...
07-14 02:50:21.978 10482-10482/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle ...
07-14 02:50:22.178 10482-10482/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle ...
07-14 02:50:22.378 10482-10482/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle ...
07-14 02:50:22.578 10482-10482/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle ...
07-14 02:50:22.778 10482-10482/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle ...
07-14 02:50:22.978 10482-10482/pt.ptinovacao.mtom.na I/System.out: debugger has settled
(1419)
07-14 02:50:23.098 10482-10482/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 100
K, 14% free 3040K/3516K, paused 13ms, total 13ms
07-14 02:50:23.128 10482-10482/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 4K,
13% free 3362K/3856K, paused 10ms, total 10ms
07-14 02:50:23.238 10482-10482/pt.ptinovacao.mtom.na I/Adreno-EGL: <
qeglDrvAPI_eglInitialize:320>: EGL 1.4 QUALCOMM Build:
I0404c4692afb8623f95c43aeb6d5e13ed4b30ddbDate: 11/06/13
07-14 02:50:23.258 10482-10482/pt.ptinovacao.mtom.na D/OpenGLRenderer: Enabling debug
mode 0
07-14 02:50:34.218 10482-10482/pt.ptinovacao.mtom.na I/StartInterface: Started Mobile
Gateway.
07-14 02:50:34.228 10482-10591/pt.ptinovacao.mtom.na D/MainService: Thread Started:
10591
07-14 02:50:34.258 10482-10597/pt.ptinovacao.mtom.na D/InternetManager: Thread Started:
10597
07-14 02:50:34.288 10482-10601/pt.ptinovacao.mtom.na D/SCL: Thread Started: 10601
07-14 02:50:34.458 10482-10601/pt.ptinovacao.mtom.na V/SCL:NA_SCL Root URI: http://
mobilelab.fe.up.pt:9090
07-14 02:50:34.498 10482-10482/pt.ptinovacao.mtom.na I/ConnectionChecker: Network WIFI
connected
07-14 02:50:34.508 10482-10482/pt.ptinovacao.mtom.na V/ConnectionListener:
BroadcastReceiver started successfully.
07-14 02:50:34.508 10482-10597/pt.ptinovacao.mtom.na D/InternetManager:NA_SCL already
running.

```

```

07-14 02:50:34.618 10482-10601/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 151
K, 13% free 3722K/4240K, paused 17ms, total 19ms
07-14 02:50:36.248 10482-10601/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 236
K, 14% free 3998K/4600K, paused 9ms, total 10ms
07-14 02:50:37.238 10482-10601/pt.ptinovacao.mtom.na D/M2M-BootStrap: GET https://
phonegw.nsl.m2m.ptinovacao.pt:8443/bootstrapParamSet HTTP/1.1
07-14 02:50:37.238 10482-10601/pt.ptinovacao.mtom.na V/M2M-BootStrap: Bootstrap GET
Size: 74
07-14 02:50:37.318 10482-10601/pt.ptinovacao.mtom.na I/TLS_SNI: Setting SNI hostname
07-14 02:50:37.318 10482-10601/pt.ptinovacao.mtom.na D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 02:50:37.748 10482-10601/pt.ptinovacao.mtom.na I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 02:50:38.118 10482-10601/pt.ptinovacao.mtom.na D/M2M-BootStrap: RESPONSE HTTP/1.1
200 OK
07-14 02:50:38.128 10482-10601/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 374
K, 16% free 4134K/4876K, paused 12ms, total 12ms
07-14 02:50:38.138 10482-10601/pt.ptinovacao.mtom.na V/M2M-BootStrap: Bootstrap Answer
Size: 4307
07-14 02:50:38.218 10482-10601/pt.ptinovacao.mtom.na D/M2M-BootStrap: Certificate
Validity: Fri Jan 30 01:50:38 WET 2015
07-14 02:50:39.718 10482-10680/pt.ptinovacao.mtom.na D/HTTPServer: Thread Started:
10680
07-14 02:50:40.938 10482-10601/pt.ptinovacao.mtom.na I/CreateResources: Registering
application: HealthConsumerNA-hammerhead-04506dfa25231287-TesteFEUP to: /m2m/
applications .
07-14 02:50:41.068 10482-10679/pt.ptinovacao.mtom.na I/TLS_SNI: Setting SNI hostname
07-14 02:50:41.068 10482-10679/pt.ptinovacao.mtom.na D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 02:50:41.628 10482-10679/pt.ptinovacao.mtom.na I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 02:50:42.048 10482-10679/pt.ptinovacao.mtom.na W/HttpClient: RESPONSE HTTP/1.1
405 Method Not Allowed
07-14 02:50:42.078 10482-10679/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 335
K, 15% free 4300K/5008K, paused 25ms, total 27ms
07-14 02:50:42.088 10482-10601/pt.ptinovacao.mtom.na D/SCL: Application already exists.
Restoring from storage ...
07-14 02:50:42.268 10482-10601/pt.ptinovacao.mtom.na D/SCL: Memory database
successfully read!
07-14 02:50:43.148 10482-10601/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 628
K, 20% free 4228K/5224K, paused 10ms, total 11ms
07-14 02:50:43.178 10482-10601/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving
Applications: /m2m/scls/TesteTrafego2hammerhead-04506dfa25231287-TesteFEUP/
applications
07-14 02:50:43.188 10482-10601/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving
Applications: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications
07-14 02:50:43.198 10482-10601/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving
Applications: /m2m/scls/hammerhead-04506dfa25231287-AssignedIP/applications
07-14 02:50:44.668 10482-10601/pt.ptinovacao.mtom.na D/CreateResources: Registering
contentInstance to /m2m/localAPI/contentInstance .
07-14 02:50:44.938 10482-10679/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 458
K, 18% free 4290K/5224K, paused 17ms, total 17ms
07-14 02:50:45.768 10482-10679/pt.ptinovacao.mtom.na I/TLS_SNI: Setting SNI hostname
07-14 02:50:45.768 10482-10679/pt.ptinovacao.mtom.na D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 02:50:45.868 10482-10679/pt.ptinovacao.mtom.na I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 02:50:46.098 10482-10679/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 470
K, 18% free 4297K/5224K, paused 14ms, total 14ms
07-14 02:50:46.128 10482-10601/pt.ptinovacao.mtom.na D/SCL: All resources checked and
subscription is present.
07-14 02:50:46.398 10482-10601/pt.ptinovacao.mtom.na D/SCL: All resources checked and
subscription is present.
07-14 02:50:52.858 10482-10482/pt.ptinovacao.mtom.na W/ConnectionListener: Network
connectivity change
07-14 02:50:52.868 10482-10482/pt.ptinovacao.mtom.na I/ConnectionChecker: Network WIFI
connected

```



```

07-14 02:51:54.688 10482-10680/pt.ptinovacao.mtom.na V/HandleRequests: Container
    created more than two minutes ago. Discarding.
07-14 02:51:54.688 10482-10680/pt.ptinovacao.mtom.na V/HandleRequests: Container
    created more than two minutes ago. Discarding.
07-14 02:51:54.688 10482-10680/pt.ptinovacao.mtom.na V/HandleRequests: Container
    created more than two minutes ago. Discarding.
07-14 02:51:54.688 10482-10680/pt.ptinovacao.mtom.na V/HandleRequests: Container
    created more than two minutes ago. Discarding.
07-14 02:51:54.688 10482-10680/pt.ptinovacao.mtom.na V/HandleRequests: Container
    created more than two minutes ago. Discarding.
07-14 02:51:54.688 10482-10680/pt.ptinovacao.mtom.na V/HandleRequests: Container
    created more than two minutes ago. Discarding.
07-14 02:51:54.688 10482-10601/pt.ptinovacao.mtom.na I/SCL: Container 1405302658817-
    ZEPHYR is now active.
07-14 02:52:02.798 10482-10482/pt.ptinovacao.mtom.na I/ShowMeasurements: Sent intent to
    stop Mobile Gateway.
07-14 02:52:02.828 10482-10482/pt.ptinovacao.mtom.na D/MainService: M2M NA Stopping
07-14 02:52:02.848 10482-10482/pt.ptinovacao.mtom.na D/InternetManager: InternetManager
    Service Stopping
07-14 02:52:02.848 10482-10482/pt.ptinovacao.mtom.na D/SCL:NA_SCL Stopping
07-14 02:52:03.098 10482-10482/pt.ptinovacao.mtom.na D/SCL: Successfully saved state to
    storage.
07-14 02:52:03.098 10482-10482/pt.ptinovacao.mtom.na V/HTTPClient: Outbound External
    Traffic: 6105. Inbound External Traffic: 8224
07-14 02:52:03.098 10482-10482/pt.ptinovacao.mtom.na V/HTTPClient: Outbound Internal
    Traffic: 1925. Inbound Internal Traffic: 1658
07-14 02:52:03.098 10482-10679/pt.ptinovacao.mtom.na D/HTTPClient: HTTP Client Shut
    Down
07-14 02:52:03.098 10482-10482/pt.ptinovacao.mtom.na V/ServerTraffic: Outbound External
    Traffic: 5867. Inbound External Traffic: 6032
07-14 02:52:03.098 10482-10482/pt.ptinovacao.mtom.na V/ServerTraffic: Outbound Internal
    Traffic: 5983. Inbound Internal Traffic: 6797
07-14 02:52:03.178 10482-10482/pt.ptinovacao.mtom.na I/MainService: Stopping Service.
    Goodbye!
07-14 02:52:03.178 10482-10482/pt.ptinovacao.mtom.na I/Process: Sending signal. PID:
    10482 SIG: 9

```

C.3 Connectivity Lost Test

```

07-14 03:56:14.398 32041-32041/pt.ptinovacao.mtom.na D/dalvikvm: Late-enabling CheckJNI
07-14 03:56:14.428 32041-32047/pt.ptinovacao.mtom.na E/jdwp: Failed sending reply to
    debugger: Broken pipe
07-14 03:56:14.428 32041-32047/pt.ptinovacao.mtom.na D/dalvikvm: Debugger has detached;
    object registry had 1 entries
07-14 03:56:14.448 32041-32041/pt.ptinovacao.mtom.na I/System.out: Sending WAIT chunk
07-14 03:56:14.448 32041-32041/pt.ptinovacao.mtom.na W/ActivityThread: Application pt.
    ptinovacao.mtom.na is waiting for the debugger on port 8100...
07-14 03:56:15.478 32041-32047/pt.ptinovacao.mtom.na I/dalvikvm: Debugger is active
07-14 03:56:15.648 32041-32041/pt.ptinovacao.mtom.na I/System.out: Debugger has
    connected
07-14 03:56:15.658 32041-32041/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
    to settle ...
07-14 03:56:15.858 32041-32041/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
    to settle ...
07-14 03:56:16.058 32041-32041/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
    to settle ...
07-14 03:56:16.258 32041-32041/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
    to settle ...
07-14 03:56:16.458 32041-32041/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
    to settle ...
07-14 03:56:16.668 32041-32041/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
    to settle ...

```

```

07-14 03:56:16.868 32041-32041/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
to settle ...
07-14 03:56:17.068 32041-32041/pt.ptinovacao.mtom.na I/System.out: debugger has settled
(1351)
07-14 03:56:17.188 32041-32041/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 81K
, 14% free 3040K/3496K, paused 12ms, total 14ms
07-14 03:56:17.218 32041-32041/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 4K,
13% free 3427K/3904K, paused 11ms, total 11ms
07-14 03:56:17.338 32041-32041/pt.ptinovacao.mtom.na I/Adreno-EGL: <
qeglDrvAPI_eglInitialize:320>: EGL 1.4 QUALCOMM Build:
I0404c4692afb8623f95c43aeb6d5e13ed4b30ddbDate: 11/06/13
07-14 03:56:17.388 32041-32041/pt.ptinovacao.mtom.na D/OpenGLRenderer: Enabling debug
mode 0
07-14 03:56:27.748 32041-32041/pt.ptinovacao.mtom.na I/StartInterface: Started Mobile
Gateway.
07-14 03:56:27.748 32041-32265/pt.ptinovacao.mtom.na D/MainService: Thread Started:
32265
07-14 03:56:27.788 32041-32272/pt.ptinovacao.mtom.na D/InternetManager: Thread Started:
32272
07-14 03:56:27.848 32041-32278/pt.ptinovacao.mtom.na D/SCL: Thread Started: 32278
07-14 03:56:28.028 32041-32278/pt.ptinovacao.mtom.na V/SCL: NA_SCL Root URI: http://
mobilelab.fe.up.pt:9090
07-14 03:56:28.308 32041-32041/pt.ptinovacao.mtom.na I/ConnectionChecker: Network WIFI
connected
07-14 03:56:28.308 32041-32041/pt.ptinovacao.mtom.na V/ConnectionListener:
BroadcastReceiver started successfully.
07-14 03:56:28.318 32041-32278/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 181
K, 13% free 3757K/4304K, paused 10ms, total 12ms
07-14 03:56:28.328 32041-32272/pt.ptinovacao.mtom.na D/InternetManager:NA_SCL already
running.
07-14 03:56:29.978 32041-32278/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 245
K, 14% free 4024K/4636K, paused 8ms, total 8ms
07-14 03:56:30.868 32041-32278/pt.ptinovacao.mtom.na D/M2M-BootStrap: GET https://
phonegw.nsl.m2m.ptinovacao.pt:8443/bootstrapParamSet HTTP/1.1
07-14 03:56:30.868 32041-32278/pt.ptinovacao.mtom.na V/M2M-BootStrap: Bootstrap GET
Size: 74
07-14 03:56:30.948 32041-32278/pt.ptinovacao.mtom.na I/TLS_SNI: Setting SNI hostname
07-14 03:56:30.948 32041-32278/pt.ptinovacao.mtom.na D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 03:56:31.458 32041-32278/pt.ptinovacao.mtom.na I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:56:31.828 32041-32278/pt.ptinovacao.mtom.na D/M2M-BootStrap: RESPONSE HTTP/1.1
200 OK
07-14 03:56:31.838 32041-32278/pt.ptinovacao.mtom.na V/M2M-BootStrap: Bootstrap Answer
Size: 4307
07-14 03:56:31.858 32041-32278/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 371
K, 16% free 4158K/4904K, paused 12ms, total 13ms
07-14 03:56:31.928 32041-32278/pt.ptinovacao.mtom.na D/M2M-BootStrap: Certificate
Validity: Fri Jan 30 02:56:32 WET 2015
07-14 03:56:34.158 32041-32449/pt.ptinovacao.mtom.na D/HTTPServer: Thread Started:
32449
07-14 03:56:35.378 32041-32278/pt.ptinovacao.mtom.na I/CreateResources: Registering
application: HealthConsumerNA-hammerhead-04506dfa25231287-TesteFEUP to: /m2m/
applications.
07-14 03:56:35.458 32041-32448/pt.ptinovacao.mtom.na I/TLS_SNI: Setting SNI hostname
07-14 03:56:35.458 32041-32448/pt.ptinovacao.mtom.na D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 03:56:35.988 32041-32448/pt.ptinovacao.mtom.na I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:56:36.438 32041-32448/pt.ptinovacao.mtom.na W/HTTPClient: RESPONSE HTTP/1.1
405 Method Not Allowed
07-14 03:56:36.458 32041-32278/pt.ptinovacao.mtom.na D/SCL: Application already exists.
Restoring from storage ...
07-14 03:56:36.528 32041-32278/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 358
K, 15% free 4311K/5036K, paused 10ms, total 12ms
07-14 03:56:36.678 32041-32278/pt.ptinovacao.mtom.na D/SCL: Memory database
successfully read!

```

07-14 03:56:37.388 32041-32278/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving Applications: /m2m/scls/TesteTrafego2hammerhead-04506dfa25231287-TesteFEUP/applications

07-14 03:56:37.398 32041-32278/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving Applications: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications

07-14 03:56:37.418 32041-32278/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving Applications: /m2m/scls/hammerhead-04506dfa25231287-AssignedIP/applications

07-14 03:56:37.558 32041-32278/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 637 K, 20% free 4236K/5240K, paused 20ms, total 20ms

07-14 03:56:40.418 32041-32448/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 492 K, 19% free 4279K/5240K, paused 20ms, total 20ms

07-14 03:56:40.598 32041-32278/pt.ptinovacao.mtom.na D/SCL: All resources checked and subscription is present.

07-14 03:56:40.838 32041-32278/pt.ptinovacao.mtom.na D/SCL: All resources checked and subscription is present.

07-14 03:56:44.808 32041-32278/pt.ptinovacao.mtom.na D/CreateResources: Registering contentInstance to /m2m/applications/HealthConsumerNA-hammerhead-04506dfa25231287-TesteFEUP/containers/ACTIONS/contentInstances .

07-14 03:57:01.328 32041-32449/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 502 K, 18% free 4327K/5240K, paused 15ms, total 15ms

07-14 03:57:01.338 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Received Containers collection notification.

07-14 03:57:01.408 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.408 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.408 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.408 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.408 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.408 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.418 32041-32449/pt.ptinovacao.mtom.na V/HandleRequests: Container created more than two minutes ago. Discarding.

07-14 03:57:01.438 32041-32278/pt.ptinovacao.mtom.na I/SCL: Container 1405306620999-ZEPHYR is now active.

07-14 03:57:01.998 32041-32041/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 131 K, 18% free 4342K/5240K, paused 13ms, total 14ms

07-14 03:57:01.998 32041-32041/pt.ptinovacao.mtom.na I/dalvikvm-heap: Grow heap (frag case) to 4.811MB for 225520-byte allocation

07-14 03:57:02.008 32041-32050/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 4K, 17% free 4558K/5464K, paused 16ms, total 16ms

```

07-14 03:57:02.028 32041-32041/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed <1K
, 17% free 4558K/5464K, paused 12ms, total 12ms
07-14 03:57:02.028 32041-32041/pt.ptinovacao.mtom.na I/dalvikvm-heap: Grow heap (frag
case) to 7.227MB for 2536936-byte allocation
07-14 03:57:02.048 32041-32050/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 0K,
12% free 7036K/7944K, paused 15ms, total 15ms
07-14 03:57:02.918 32041-32041/pt.ptinovacao.mtom.na I/CommandsActivity: Sensor ZEPHYR
chosen to start.
07-14 03:57:03.048 32041-32278/pt.ptinovacao.mtom.na D/CreateResources: Registering
contentInstance to /m2m/applications/HealthConsumerNA-hammerhead-04506dfa25231287-
TesteFEUP/containers/ACTIONS/contentInstances .
07-14 03:57:03.898 32041-32278/pt.ptinovacao.mtom.na V/SCL: Resource still has no
subscriptions. Subscribing...
07-14 03:57:03.978 32041-32278/pt.ptinovacao.mtom.na D/CreateResources: Subscribing to
/m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/HealthSensors/
containers/1405306620999-ZEPHYR/contentInstances/subscriptions
07-14 03:57:04.248 32041-32448/pt.ptinovacao.mtom.na I/CreateResources: Subscription to
/m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications/HealthSensors/
containers/1405306620999-ZEPHYR/contentInstances/subscriptions successful.
07-14 03:57:04.338 32041-32449/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 343
K, 8% free 8163K/8872K, paused 11ms, total 11ms
07-14 03:57:13.528 32041-32449/pt.ptinovacao.mtom.na D/TreatSensorData: Measurement
Updated. Array Size: 7
07-14 03:57:19.428 32041-32041/pt.ptinovacao.mtom.na W/ConnectionListener: Network
connectivity change
07-14 03:57:19.448 32041-32041/pt.ptinovacao.mtom.na D/ConnectionChecker: No network
connectivity
07-14 03:57:19.448 32041-32041/pt.ptinovacao.mtom.na D/InternetManager: No connection.
Resetting stored IPV4 address.
07-14 03:57:43.718 32041-32449/pt.ptinovacao.mtom.na D/TreatSensorData: Measurement
Updated. Array Size: 16
07-14 03:57:44.688 32041-32041/pt.ptinovacao.mtom.na W/ConnectionListener: Network
connectivity change
07-14 03:57:44.698 32041-32041/pt.ptinovacao.mtom.na I/ConnectionChecker: Network WIFI
connected
07-14 03:57:44.698 32041-32272/pt.ptinovacao.mtom.na D/InternetManager:NA_SCL already
running.
07-14 03:57:44.908 32041-32449/pt.ptinovacao.mtom.na D/TreatSensorData: Measurement
Updated. Array Size: 25
07-14 03:57:45.118 32041-32449/pt.ptinovacao.mtom.na D/TreatSensorData: Measurement
Updated. Array Size: 34
07-14 03:57:53.558 32041-32449/pt.ptinovacao.mtom.na D/TreatSensorData: Measurement
Updated. Array Size: 44
07-14 03:57:59.078 32041-32041/pt.ptinovacao.mtom.na W/ConnectionListener: Network
connectivity change
07-14 03:57:59.098 32041-32041/pt.ptinovacao.mtom.na D/ConnectionChecker: No network
connectivity
07-14 03:57:59.098 32041-32041/pt.ptinovacao.mtom.na D/InternetManager: No connection.
Resetting stored IPV4 address.
07-14 03:58:14.638 32041-32041/pt.ptinovacao.mtom.na I/ShowMeasurements: Sent intent to
stop Mobile Gateway.
07-14 03:58:14.668 32041-32041/pt.ptinovacao.mtom.na D/MainService: M2M NA Stopping
07-14 03:58:14.678 32041-32041/pt.ptinovacao.mtom.na D/InternetManager: InternetManager
Service Stopping
07-14 03:58:14.678 32041-32041/pt.ptinovacao.mtom.na D/SCL:NA_SCL Stopping
07-14 03:58:14.918 32041-32041/pt.ptinovacao.mtom.na D/SCL: Successfully saved state to
storage.
07-14 03:58:14.918 32041-32041/pt.ptinovacao.mtom.na V/HTTPClient: Outbound External
Traffic: 9068. Inbound External Traffic: 10627
07-14 03:58:14.918 32041-32041/pt.ptinovacao.mtom.na V/HTTPClient: Outbound Internal
Traffic: 0. Inbound Internal Traffic: 0
07-14 03:58:14.918 32041-32041/pt.ptinovacao.mtom.na V/ServerTraffic: Outbound External
Traffic: 30417. Inbound External Traffic: 31572
07-14 03:58:14.918 32041-32041/pt.ptinovacao.mtom.na V/ServerTraffic: Outbound Internal
Traffic: 0. Inbound Internal Traffic: 0
07-14 03:58:14.928 32041-32448/pt.ptinovacao.mtom.na D/HTTPClient: HTTP Client Shut
Down

```


C.4 Disconnection after using local API

```

07-14 03:32:15.628 25330-25330/pt.ptinovacao.mtom.na D/dalvikvm: Late-enabling CheckJNI
07-14 03:32:15.648 25330-25336/pt.ptinovacao.mtom.na E/jdwp: Failed sending reply to
  debugger: Broken pipe
07-14 03:32:15.648 25330-25336/pt.ptinovacao.mtom.na D/dalvikvm: Debugger has detached;
  object registry had 1 entries
07-14 03:32:15.748 25330-25330/pt.ptinovacao.mtom.na I/System.out: Sending WAIT chunk
07-14 03:32:15.748 25330-25330/pt.ptinovacao.mtom.na W/ActivityThread: Application pt.
  ptinovacao.mtom.na is waiting for the debugger on port 8100...
07-14 03:32:16.708 25330-25336/pt.ptinovacao.mtom.na I/dalvikvm: Debugger is active
07-14 03:32:16.748 25330-25330/pt.ptinovacao.mtom.na I/System.out: Debugger has
  connected
07-14 03:32:16.748 25330-25330/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
  to settle...
07-14 03:32:16.958 25330-25330/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
  to settle...
07-14 03:32:17.158 25330-25330/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
  to settle...
07-14 03:32:17.358 25330-25330/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
  to settle...
07-14 03:32:17.558 25330-25330/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
  to settle...
07-14 03:32:17.758 25330-25330/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
  to settle...
07-14 03:32:17.958 25330-25330/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
  to settle...
07-14 03:32:18.158 25330-25330/pt.ptinovacao.mtom.na I/System.out: waiting for debugger
  to settle...
07-14 03:32:18.358 25330-25330/pt.ptinovacao.mtom.na I/System.out: debugger has settled
  (1455)
07-14 03:32:18.498 25330-25330/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 77K
  , 13% free 3040K/3492K, paused 8ms, total 15ms
07-14 03:32:18.528 25330-25330/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 4K,
  13% free 3427K/3900K, paused 10ms, total 11ms
07-14 03:32:18.638 25330-25330/pt.ptinovacao.mtom.na I/Adreno-EGL: <
  qeglDrvAPI_eglInitialize:320>: EGL 1.4 QUALCOMM Build:
  I0404c4692afb8623f95c43aeb6d5e13ed4b30ddbDate: 11/06/13
07-14 03:32:18.668 25330-25330/pt.ptinovacao.mtom.na D/OpenGLRenderer: Enabling debug
  mode 0
07-14 03:33:35.628 25330-25330/pt.ptinovacao.mtom.na I/StartInterface: Started Mobile
  Gateway.
07-14 03:33:35.638 25330-25745/pt.ptinovacao.mtom.na D/MainService: Thread Started:
  25745
07-14 03:33:35.668 25330-25752/pt.ptinovacao.mtom.na D/InternetManager: Thread Started:
  25752
07-14 03:33:35.708 25330-25764/pt.ptinovacao.mtom.na D/SCL: Thread Started: 25764
07-14 03:33:35.858 25330-25764/pt.ptinovacao.mtom.na V/SCL:NA_SCL Root URI: http://
  mobilelab.fe.up.pt:9090
07-14 03:33:35.988 25330-25330/pt.ptinovacao.mtom.na I/ConnectionChecker: Network WIFI
  connected
07-14 03:33:35.988 25330-25330/pt.ptinovacao.mtom.na V/ConnectionListener:
  BroadcastReceiver started successfully.
07-14 03:33:35.998 25330-25752/pt.ptinovacao.mtom.na D/InternetManager:NA_SCL already
  running.
07-14 03:33:36.278 25330-25764/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 182
  K, 13% free 3755K/4304K, paused 13ms, total 14ms
07-14 03:33:39.208 25330-25764/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 252
  K, 14% free 4017K/4636K, paused 8ms, total 8ms
07-14 03:33:40.318 25330-25764/pt.ptinovacao.mtom.na D/M2M-BootStrap: GET https://
  phonegw.nsc1.m2m.ptinovacao.pt:8443/bootstrapParamSet HTTP/1.1
07-14 03:33:40.318 25330-25764/pt.ptinovacao.mtom.na V/M2M-BootStrap: Bootstrap GET
  Size: 74
07-14 03:33:40.438 25330-25764/pt.ptinovacao.mtom.na I/TLS_SNI: Setting SNI hostname
07-14 03:33:40.438 25330-25764/pt.ptinovacao.mtom.na D/TLS_SNI: Hostname: phonegw.nsc1.
  m2m.ptinovacao.pt

```

```

07-14 03:33:40.998 25330-25764/pt.ptinovacao.mtom.na I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:33:41.388 25330-25764/pt.ptinovacao.mtom.na D/M2M-BootStrap: RESPONSE HTTP/1.1
200 OK
07-14 03:33:41.478 25330-25764/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 354
K, 15% free 4167K/4888K, paused 41ms, total 45ms
07-14 03:33:41.488 25330-25764/pt.ptinovacao.mtom.na V/M2M-BootStrap: Bootstrap Answer
Size: 4307
07-14 03:33:41.568 25330-25764/pt.ptinovacao.mtom.na D/M2M-BootStrap: Certificate
Validity: Fri Jan 30 02:33:41 WET 2015
07-14 03:33:43.198 25330-25883/pt.ptinovacao.mtom.na D/HTTPServer: Thread Started:
25883
07-14 03:33:44.428 25330-25764/pt.ptinovacao.mtom.na I/CreateResources: Registering
application: HealthConsumerNA-hammerhead-04506dfa25231287-TesteFEUP to: /m2m/
applications .
07-14 03:33:44.488 25330-25882/pt.ptinovacao.mtom.na I/TLS_SNI: Setting SNI hostname
07-14 03:33:44.488 25330-25882/pt.ptinovacao.mtom.na D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 03:33:45.058 25330-25882/pt.ptinovacao.mtom.na I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:33:45.508 25330-25882/pt.ptinovacao.mtom.na W/HTTPClient: RESPONSE HTTP/1.1
405 Method Not Allowed
07-14 03:33:45.528 25330-25764/pt.ptinovacao.mtom.na D/SCL: Application already exists.
Restoring from storage ...
07-14 03:33:45.568 25330-25764/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 377
K, 15% free 4304K/5048K, paused 14ms, total 16ms
07-14 03:33:45.738 25330-25764/pt.ptinovacao.mtom.na D/SCL: Memory database
successfully read!
07-14 03:33:46.498 25330-25764/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving
Applications: /m2m/scls/TesteTrafego2hammerhead-04506dfa25231287-TesteFEUP/
applications
07-14 03:33:46.518 25330-25764/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving
Applications: /m2m/scls/hammerhead-04506dfa25231287-TesteFEUP/applications
07-14 03:33:46.528 25330-25764/pt.ptinovacao.mtom.na I/RetrieveResources: Retrieving
Applications: /m2m/scls/hammerhead-04506dfa25231287-AssignedIP/applications
07-14 03:33:46.628 25330-25882/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 618
K, 19% free 4237K/5224K, paused 13ms, total 13ms
07-14 03:33:47.398 25330-25764/pt.ptinovacao.mtom.na D/CreateResources: Registering
contentInstance to /m2m/localAPI/contentInstance .
07-14 03:33:48.868 25330-25882/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 473
K, 18% free 4299K/5224K, paused 15ms, total 15ms
07-14 03:33:49.498 25330-25882/pt.ptinovacao.mtom.na I/TLS_SNI: Setting SNI hostname
07-14 03:33:49.498 25330-25882/pt.ptinovacao.mtom.na D/TLS_SNI: Hostname: phonegw.nsl.
m2m.ptinovacao.pt
07-14 03:33:49.638 25330-25882/pt.ptinovacao.mtom.na I/TLS_SNI: Established TLSv1.2
connection with phonegw.nsl.m2m.ptinovacao.pt using SSL_RSA_WITH_RC4_128_MD5
07-14 03:33:50.268 25330-25764/pt.ptinovacao.mtom.na D/SCL: All resources checked and
subscription is present.
07-14 03:33:50.648 25330-25882/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 526
K, 18% free 4303K/5224K, paused 13ms, total 13ms
07-14 03:33:50.688 25330-25764/pt.ptinovacao.mtom.na D/SCL: All resources checked and
subscription is present.
07-14 03:34:03.088 25330-25883/pt.ptinovacao.mtom.na D/HandleRequests: Local connection
: New Container: 1405305242955-ZEPHYR
07-14 03:34:03.088 25330-25883/pt.ptinovacao.mtom.na I/SCL: Container 1405305242955-
ZEPHYR is now active.
07-14 03:34:06.988 25330-25330/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 70K
, 18% free 4335K/5224K, paused 14ms, total 14ms
07-14 03:34:06.988 25330-25330/pt.ptinovacao.mtom.na I/dalvikvm-heap: Grow heap (frag
case) to 4.804MB for 225520-byte allocation
07-14 03:34:07.018 25330-25339/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed <1K
, 17% free 4554K/5448K, paused 25ms, total 25ms
07-14 03:34:07.038 25330-25330/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed <1K
, 17% free 4555K/5448K, paused 14ms, total 15ms
07-14 03:34:07.038 25330-25330/pt.ptinovacao.mtom.na I/dalvikvm-heap: Grow heap (frag
case) to 7.223MB for 2536936-byte allocation
07-14 03:34:07.058 25330-25339/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 0K,
12% free 7033K/7928K, paused 20ms, total 20ms

```

```
07-14 03:34:07.968 25330-25330/pt.ptinovacao.mtom.na I/CommandsActivity: Sensor ZEPHYR
chosen to start.
07-14 03:34:08.098 25330-25764/pt.ptinovacao.mtom.na D/CreateResources: Registering
contentInstance to /m2m/localAPI/contentInstance .
07-14 03:34:18.278 25330-25883/pt.ptinovacao.mtom.na D/HandleRequests: Local connection
: ContentInstance of Container: 1405305242955-ZEPHYR
07-14 03:34:18.558 25330-25883/pt.ptinovacao.mtom.na D/dalvikvm: GC_FOR_ALLOC freed 350
K, 9% free 8151K/8868K, paused 10ms, total 12ms
07-14 03:34:18.668 25330-25883/pt.ptinovacao.mtom.na D/TreatSensorData: Measurement
Updated. Array Size: 7
07-14 03:34:28.308 25330-25883/pt.ptinovacao.mtom.na D/HandleRequests: Local connection
: ContentInstance of Container: 1405305242955-ZEPHYR
07-14 03:34:28.358 25330-25883/pt.ptinovacao.mtom.na D/TreatSensorData: Measurement
Updated. Array Size: 16
07-14 03:34:38.278 25330-25883/pt.ptinovacao.mtom.na D/HandleRequests: Local connection
: ContentInstance of Container: 1405305242955-ZEPHYR
07-14 03:34:38.318 25330-25883/pt.ptinovacao.mtom.na D/TreatSensorData: Measurement
Updated. Array Size: 25
07-14 03:34:42.388 25330-25330/pt.ptinovacao.mtom.na I/ShowMeasurements: Sent intent to
stop Mobile Gateway.
07-14 03:34:42.418 25330-25330/pt.ptinovacao.mtom.na D/MainService: M2M NA Stopping
07-14 03:34:42.438 25330-25330/pt.ptinovacao.mtom.na D/InternetManager: InternetManager
Service Stopping
07-14 03:34:42.438 25330-25330/pt.ptinovacao.mtom.na D/SCL: NA_SCL Stopping
07-14 03:34:42.718 25330-25330/pt.ptinovacao.mtom.na D/SCL: Successfully saved state to
storage.
07-14 03:34:42.718 25330-25330/pt.ptinovacao.mtom.na V/HTTPClient: Outbound External
Traffic: 6105. Inbound External Traffic: 8224
07-14 03:34:42.718 25330-25330/pt.ptinovacao.mtom.na V/HTTPClient: Outbound Internal
Traffic: 1286. Inbound Internal Traffic: 1108
07-14 03:34:42.718 25330-25330/pt.ptinovacao.mtom.na V/ServerTraffic: Outbound External
Traffic: 0. Inbound External Traffic: 0
07-14 03:34:42.718 25330-25330/pt.ptinovacao.mtom.na V/ServerTraffic: Outbound Internal
Traffic: 4548. Inbound Internal Traffic: 5197
07-14 03:34:42.718 25330-25882/pt.ptinovacao.mtom.na D/HTTPClient: HTTP Client Shut
Down
07-14 03:34:42.778 25330-25330/pt.ptinovacao.mtom.na I/MainService: Stopping Service.
Goodbye
```


Bibliography

- [1] Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013. <http://www.gartner.com/newsroom/id/2665715>. Online; accessed 20-06-2014.
- [2] *ETSI TS 102 690 V2.1.1 (2013-10) Machine-to-Machine communications (M2M); Functional architecture*, 2013.
- [3] Focus Group on M2M Service Layer. D2.1 – Version 1.0: M2M service layer: Requirements and architectural framework. Technical report, ITU-T, April 2014.
- [4] Guenter Klas (Vodafone), Friedhelm Rodermund (Vodafone), Zach Shelby (ARM), Sandeep Akhouri (Ericsson), and Jan Höller (Ericsson). *White Paper: "LightweightM2M": Enabling Devicemanagement and applications for the Internet of Things*, March 2014.
- [5] *Lightweight Internet protocols for web enablement of sensors using constrained gateway devices*, 2013 International Conference on Computing, Networking and Communications (ICNC 2013). IEEE, 2013.
- [6] RabbitMQ. Amqp 0.9.1 model explained. <https://www.rabbitmq.com/tutorials/amqp-concepts.html>. Online; accessed 11-02-2014.
- [7] Malcolm Clarke, Joost de Folter, Charles Palmer, and Vivek Verma. Building point of care health technologies on the IEEE 11073 health device standards. In *2013 IEEE Point-of-Care Healthcare Technologies (PHT)*, pages 117–119. IEEE, January 2013. URL: <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6461298>, doi:10.1109/PHT.2013.6461298.
- [8] Biomedical Engineering - ECG Assignment. <http://eleceng.dit.ie/tburke/biomed/assignment1.html>. Online; accessed 20-06-2014.
- [9] Niranjana Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *ACM SIGCOMM Internet Measurement Conference, IMC*, pages 280–293. ACM Press, 2009.

- [10] Aeronautics and National Research Council Space Engineering Board. *The Global Positioning System: A Shared National Asset*. The National Academies Press, 1995. URL: http://www.nap.edu/openbook.php?record_id=4920.
- [11] Rongxing Lu, Xu Li, Xiaohui Liang, Xuemin Shen, and Xiaodong Lin. GRS: the green, reliability, and security of emerging machine to machine communications. *IEEE Communications Magazine*, 49(4):28–35, April 2011. URL: <http://dx.doi.org/10.1109/MCOM.2011.5741143>, doi:10.1109/MCOM.2011.5741143.
- [12] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, 2010. URL: <http://dx.doi.org/10.1016/j.comnet.2010.05.010>, doi:10.1016/j.comnet.2010.05.010.
- [13] ETSI TR 102 935 V2.1.1 (2012-09) *Machine-to-Machine communications (M2M); Applicability of M2M architecture to Smart Grid Networks; Impact of Smart Grids on M2M platform*, 2012.
- [14] ETSI TR 102 691 V1.1.1 (2010-05) *Machine-to-Machine communications (M2M); Smart Metering Use Cases*, 2010.
- [15] Min Chen, Jiafu Wan, Sergio Gonzalez, Xiaofei Liao, and Victor C.M. Leung. A Survey of Recent Developments in Home M2M Networks. *IEEE Communications Surveys & Tutorials*, pages 1–17, 2014. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84888162287&partnerID=tZ0tx3ylhttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6674156>, doi:10.1109/SURV.2013.110113.00249.
- [16] Android Operating System. <http://www.android.com/>. Online; accessed 29-06-2014.
- [17] GE Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), 1965. URL: http://web.eng.fiu.edu/npala/EEE5425/Gordon_Moore_1965_Article.pdf.
- [18] IBM. Mqtt v3.1 protocol specification. http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/MQTT_V3.1_Protocol_Specific.pdf, 2010. Online; accessed 27-06-2014.
- [19] Jon Tong-Seng Quah and Guey Long Lim. Push selling—Multicast messages to wireless devices based on the publish/subscribe model. *Electronic Commerce Research and Applications*, 1(3-4):235–246, September 2002.
- [20] 3GPP. Service requirements for Machine-Type Communications (MTC); stage 1, release 12. Technical Report TS 22.368 V12.3.0, 3GPP, 2013.

- [21] 3GPP. Technical Specification Group Services and System Aspects; Study on Facilitating Machine to Machine Communication in 3GPP Systems. Technical Report TR 22.868, 3GPP, 2007.
- [22] *ETSI TS 102 689 V2.1.1 (2013-07) Machine-to-Machine communications (M2M); M2M service requirements*, 2013.
- [23] Focus Group on M2M Service Layer. D1.1 – Version 1.0: M2M use cases: e-health. Technical report, ITU-T, April 2014.
- [24] Focus Group on M2M Service Layer. D0.1 – Version 1.0: M2M standardization activities and gap analysis: e-health. Technical report, ITU-T, April 2014.
- [25] Focus Group on M2M Service Layer. D0.2 – Version 1.0: M2M enabled ecosystems: e-health. Technical report, ITU-T, April 2014.
- [26] Focus Group on M2M Service Layer. *Y.2060: Overview of the Internet of things*, June 2012.
- [27] Focus Group on M2M Service Layer. D3.1 – Version 1.0: M2M service layer: APIs and protocols overview. Technical report, ITU-T, April 2014.
- [28] OneM2M. <http://www.onem2m.org/>. Online; accessed 24-06-2014.
- [29] Specification detail: Technical realization of the Short Message Service (SMS). <http://www.3gpp.org/DynaReport/23040.htm>. Online; accessed 05-07-2014.
- [30] ZigBee. <http://www.zigbee.org/>. Online; accessed 05-07-2014.
- [31] N. Kushalnagar, Intel Corp, G. Montenegro, and Microsoft Corporation and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, Danfoss A/S, August 2007. URL: <http://www.rfc-editor.org/rfc/pdf/rfc4919.txt.pdf>.
- [32] IEEE 802.15.4: Low Rate WPAN. <http://www.ieee802.org/15/pub/TG4.html>. Online; accessed 05-07-2014.
- [33] *Lightweight Machine to Machine Technical Specification [Version 1.0]*, December 2013.
- [34] Roy T. Fielding. Architectural styles and the design of network-based software architectures. http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf, 2000. Online; accessed 26-06-2014.
- [35] T. Berners-Lee, MIT/LCS, R. Fielding, U.C. Irvine, L. Masinter, and Xerox Corporation. Transmission Control Protocol. RFC 2396, Network Working Group, August 1998. URL: <http://www.ietf.org/rfc/rfc2396.txt>.

- [36] R. Fielding, UC Irvine, J. Gettys, Compaq/W3C, J. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft, T. Berners-Lee, and W3C/MIT. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Network Working Group, June 1999. URL: <http://tools.ietf.org/pdf/rfc2616.pdf>.
- [37] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The World-Wide Web. *Communications of the ACM*, 37(8):76–82, August 1994. URL: <http://dl.acm.org/citation.cfm?id=179606.179671>, doi:10.1145/179606.179671.
- [38] T. Dierks, Certicom, and C. Allen. The TLS Protocol, Version 1.0. RFC 2246, Network Working Group, January 1999. URL: <http://www.ietf.org/rfc/rfc2246.txt>.
- [39] R. Fielding, UC Irvine, J. Gettys, J. Mogul, DEC, H. Frystyk, T. Berners-Lee, and MIT/LCS. Hypertext Transfer Protocol – HTTP/1.1. RFC 2068, Network Working Group, January 1997. URL: <http://tools.ietf.org/pdf/rfc2068.pdf>.
- [40] Jon Postel. Transmission Control Protocol. RFC 793, Information Sciences Institute, University of Southern California, September 1981. URL: <http://www.ietf.org/rfc/rfc793.txt>.
- [41] J Postel. User Datagram Protocol. RFC 768, Information Sciences Institute, University of Southern California, August 1980. URL: <http://tools.ietf.org/pdf/rfc768.pdf>.
- [42] Z. Shelby, ARM, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252, Internet Engineering Task Force, June 2014. URL: <http://tools.ietf.org/html/rfc7252>.
- [43] ETSI, IPSO Alliance, and OMA. CoAP 3 & OMA Lightweight M2M Plugtest. <http://www.etsi.org/news-events/past-events/693-coap-oma-lightweight-m2m>, November 2013. Online; accessed 07-02-2014.
- [44] Carsten Bormann, Angelo P. Castellani, and Zach Shelby. CoAP: An Application Protocol for Billions of Tiny Internet Nodes. *IEEE Internet Computing*, 16(2):62–67, March 2012. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6159216>, doi:10.1109/MIC.2012.29.
- [45] J Postel and ISI. User Datagram Protocol. RFC 768, ISI, August 1980. URL: <http://tools.ietf.org/pdf/rfc768.pdf>.
- [46] Suresh Krishnan and Sheila Frankel. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071, IETF, February 2011. URL: <http://tools.ietf.org/html/rfc6071>.

- [47] Eric Rescorla and Nagendra Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347, IETF, January 2012. URL: <http://tools.ietf.org/search/rfc6347>.
- [48] Jon Postel. Transmission Control Protocol. RFC 793, Information Sciences Institute, University of Southern California, September 1981. URL: <http://www.ietf.org/rfc/rfc793.txt>.
- [49] OASIS Standard. Oasis advanced message queuing protocol (amqp) version 1.0. <http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf>. Online; accessed 11-02-2014.
- [50] Ed. A. Melnikov, Isode Limited, Ed. K. Zeilenga, and OpenLDAP Foundation. Simple Authentication and Security Layer (SASL). RFC 4422, Network Working Group, June 2006. URL: <http://www.ietf.org/rfc/rfc4422.txt>.
- [51] S. Blake-Wilson, BCI, M. Nystrom, RSA Security, D. Hopwood, Independent Consultant, J. Mikkelsen, Transactionware, T. Wright, and Vodafone. Transport Layer Security (TLS) Extensions. RFC 4366, Network Working Group, June 2006. URL: <http://www.ietf.org/rfc/rfc4366.txt>.
- [52] World Health Organization - E-Health. <http://www.who.int/trade/glossary/story021/en/>. Online; accessed 29-06-2014.
- [53] Elina Mattila, Juha Pärkkä, Marion Hermersdorf, Jussi Kaasinen, Janne Vainio, Kai Samposalo, Juho Merilahti, Juha Kolari, Minna Kulju, Raimo Lappalainen, and Ilkka Korhonen. Mobile diary for wellness management—results on usage and usability in two user studies. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, 12(4):501–12, July 2008. URL: <http://www.ncbi.nlm.nih.gov/pubmed/18632330>, doi:10.1109/TITB.2007.908237.
- [54] Joseph J Oresko, Heather Duschl, and Allen C Cheng. A wearable smartphone-based platform for real-time cardiovascular disease detection via electrocardiogram processing. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, 14(3):734–40, May 2010. URL: <http://www.ncbi.nlm.nih.gov/pubmed/20388600>, doi:10.1109/TITB.2010.2047865.
- [55] Charles Worringham, Amanda Rojek, and Ian Stewart. Development and feasibility of a smartphone, ECG and GPS based system for remotely monitoring exercise in cardiac rehabilitation. *PloS one*, 6(2):e14669, January 2011. URL: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3036581&tool=pmcentrez&rendertype=abstract>, doi:10.1371/journal.pone.0014669.

- [56] Christopher C. Tsai, Gunny Lee, Fred Raab, Gregory J. Norman, Timothy Sohn, William G. Griswold, and Kevin Patrick. Usability and Feasibility of PmEB: A Mobile Phone Application for Monitoring Real Time Caloric Balance. *Mobile Networks and Applications*, 12(2-3):173–184, July 2007. URL: <http://link.springer.com/10.1007/s11036-007-0014-4>, doi:10.1007/s11036-007-0014-4.
- [57] Jianchu Yao and Steve Warren. Applying the ISO/IEEE 11073 standards to wearable home health monitoring systems. *Journal of clinical monitoring and computing*, 19(6):427–36, December 2005. URL: <http://www.ncbi.nlm.nih.gov/pubmed/16437294>, doi:10.1007/s10877-005-2033-7.
- [58] Malcolm Clarke, Douglas Bogia, Kai Hassing, Lars Steubesand, Tony Chan, and Deepak Ayyagari. Developing a standard for personal health devices based on 11073. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2007(Dim):6175–7, January 2007. URL: <http://www.ncbi.nlm.nih.gov/pubmed/18003430>, doi:10.1109/IEMBS.2007.4353764.
- [59] ISO/IEC. Information technology - open systems interconnection - basic reference model: The basic model. International Standard 7498-1, ISO/IEC, 1994.
- [60] Google Inc. Android BluetoothHealth API. <http://developer.android.com/reference/android/bluetooth/BluetoothHealth.html>.
- [61] Continua Health Alliance. www.continuaalliance.org/.
- [62] ETSI TS 102 921 V2.1.1 (2013-12) Machine-to-Machine communications (M2M); *m1a, d1a and m1d interfaces*, 2013.
- [63] V. Ryan, S. Seligman, R. Lee, and Inc. Sun Microsystems. The Base16, Base32, and Base64 Data Encodings. RFC 4648, Network Working Group, October 2006. URL: <http://tools.ietf.org/pdf/rfc4648.pdf>.
- [64] Eclipse Foundation OpenM2M (OM2M). <http://eclipse.org/om2m/>. Online; accessed 20-06-2014.
- [65] Actility. Cocoon™- ETSI M2M Gateway Open Source Implementation. <http://cocoon.actility.com/>. Online; accessed 08-07-2014.
- [66] V. Ryan, S. Seligman, R. Lee, and Inc. Sun Microsystems. Schema for Representing Java(tm) Objects in an LDAP Directory. RFC 2713, Network Working Group, October 1999. URL: <http://tools.ietf.org/pdf/rfc2713.pdf>.
- [67] W3C. Extensible markup language (xml). <http://www.w3.org/TR/WD-xml-961114.html>, November 1996. Online; accessed 31-01-2014.

- [68] ISO 8879. Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML). ISO Standard, 1986.
- [69] M. Murata E. Whitehead, UC Irvine. XML Media Types. RFC 2376, Fuji Xerox Info. Systems, July 1998. URL: <http://tools.ietf.org/html/rfc2376>.
- [70] Douglas Crockford. The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627, JSON.org, July 2006. URL: <http://tools.ietf.org/html/rfc4627>.
- [71] William Stallings. *Cryptography and Network Security*. Prentice Hall, 5 edition, 2010.
- [72] B. Kaliski and J. Staddon. PKCS 1: RSA Cryptography Specifications Version 2.0. RFC 2437, RSA Laboratories, October 1998. URL: <http://www.ietf.org/rfc/rfc2437.txt>.
- [73] D. Cooper, S. Santesson, Microsoft, S. Farrell, Trinity College Dublin, S. Boeyen, Entrust, R. Housley, Vigil Security, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, NIST, May 2008. URL: <http://tools.ietf.org/pdf/rfc5280.pdf>.
- [74] Zephyr HxM BT Heart Rate Monitor. <http://zephyranywhere.com/products/hxm-bluetooth-heart-rate-monitor/>. Online; accessed 28-06-2014.
- [75] Get the Android SDK. http://developer.android.com/sdk/index.html?utm_source=weibolife. Online; accessed 09-06-2014.
- [76] JetBrains IntelliJ IDEA. <http://www.jetbrains.com/idea/>. Online; accessed 09-06-2014.
- [77] Apache Maven. <http://maven.apache.org/>. Online; accessed 09-06-2014.
- [78] Apache Subversion. <http://subversion.apache.org/>. Online; accessed 09-06-2014.
- [79] Californium (Cf) CoAP framework. <http://people.inf.ethz.ch/mkovatsc/californium.php>. Online; accessed 09-06-2014.
- [80] Eclipse Paho MQTT. <http://www.eclipse.org/paho/>. Online; accessed 09-06-2014.
- [81] Android Dashboard - Version Statistics. <https://developer.android.com/about/dashboards/index.html>. Online; accessed 09-06-2014.
- [82] NanoHttpd Library. <https://github.com/NanoHttpd/nanohttpd>. Online; accessed 09-06-2014.

- [83] Performance Comparison of JSON frameworks for Android OS. <https://github.com/martinadamek/json-android-compare>. Online; accessed 09-06-2014.
- [84] Wireshark. <http://www.wireshark.org/>. Online; accessed 21-06-2014.