DELIVERABLE

D2.4 – UNCAP certification suite – alpha version

Project Acronym: UNCAP

Grant Agreement number: 643555

Project Title: Ubiquitous iNteroperable Care for Ageing People

Revision: 01

Authors: Andrei-Vlad Rad (FIDA); Fabio Roncato (TRILOGIS);

Project co-funded by the the Horizon 2020 Framework Programme of the European Union		
Dissemination Level		
P	Public	**X**
C	Confidential, only for members of the consortium and the Commission Services	

1. Revision history and statement of originality

1.1. Revision history

Rev	Date	Author	Organization	Description
0.1	27/06/2016	Andrei-Vlad Rad	FIDA	First version of deliverable
0.2	30/06/2016	Fabio Roncato	Trilogis	Final version
0.3	30/06/2016	Irene Facchin	Trilogis	Quality check

1.2. Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

2. List of references

Number	Full Reference
1	Project website: http://www.uncap.eu/

3. Executive Abstract

UNCAP certification suite is a part of the UNCAP products. This is a part of the UNCAP website (http://www.uncap.eu) that provides relative guidelines, to help further hardware or sensor manufacturers test compliancy of their solutions to the standards supported by UNCAP.

The **UNCAP certification suite** will be a key **business facilitator**, specifically thought to help European ICT companies in the health and care domain (especially SMEs) acquire a pivotal role in the support (and eventually extension) of current standards, thus contributing to **improve competitiveness of European ICT industry** in the care and health domain. Wide take-up of standards will help SMEs compete on equal grounds with low-cost Asian manufacturers and with major US corporations who may impose proprietary solutions in the consumer market.

The **UNCAP certification suite** (i.e., a conformance testing tool) against the SensorThings / SWE standards, to ensure that further technologies can be tested against UNCAP. This includes a compliance testing testbed officially defined in agreement with the OGC.

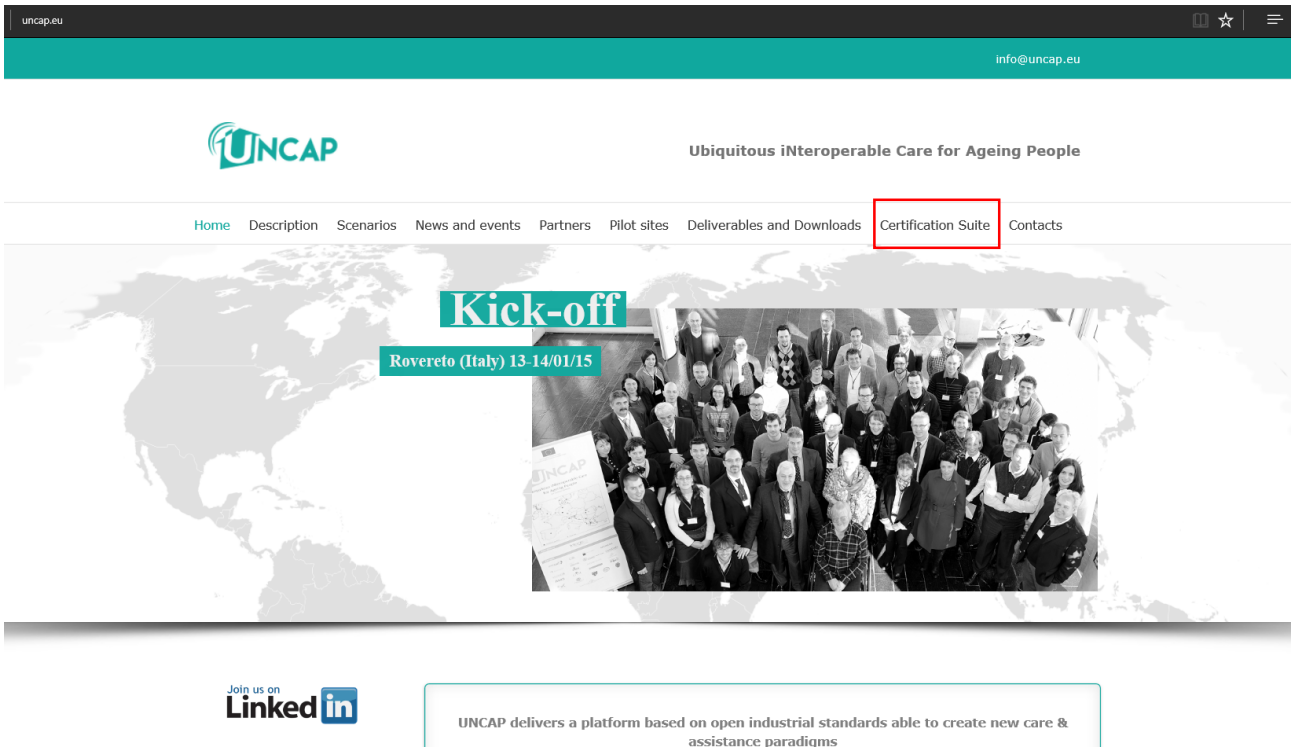
4. Table of Content

1. Revision history and statement of originality 2
1.1. Revision history 2
1.2. Statement of originality 2
2. List of references 3
3. Table of Acronyms ..... Error! Bookmark not defined.
4. Executive Abstract 4
5. Table of Content 5
6. UNCAP certification suite – alpha version 6
6.1. Introduction 6
6.2. Access to the Certification Suite 6
6.3. Available interaction with the UNCAP ecosystem 6
6.4. Connectable to the UNCAP box type of devices 7
6.5. Driver description 7
6.5.1. Drivers introduction 7
6.5.2. Driver catalog 8
6.5.3. Master Android app driver installation 8
6.5.4. Master Android app driver detection 8
6.5.5. Communication with the driver 9
6.5.5.1. When each option is used: 10
6.5.5.2. Option 1: HTML GUIs (for communication master → driver) 10
6.5.5.3. Option 2: Opening the driver app (master → driver) 11
6.5.5.4. Option 3: Communicating back the results (driver → master) 11
6.6. Contact details 12

5. UNCAP certification suite – alpha version

5.1. Introduction

The UNCAP certification suite is a public component of the UNCAP website that can be found at this address: http://www.uncap.eu/certification-suite/, or following the ***Certification Suite*** menu in the main page.



This suite is structured as a *Questions & Answers* page for a much easier reading. The content of the suite describes how developers can create hardware and software that works under the UNCAP standards; therefore, they are UNCAP compatible.

The suite provides detailed description of the interfaces that the UNCAP platform provides and also samples of drivers (for the hardware) and examples of connection (for the software).

The process of how a developer receives the certificate of compliance is also described.

5.2. Access to the Certification Suite

The certification suite is a public component of the UNCAP ecosystem that is accessible through the UNCAP website or any other social communication environment.

5.3. Available interaction with the UNCAP ecosystem

There are three types of available interaction with the UNCAP ecosystem:

- Hardware ->to-> UNCAP box (through drivers)
- Hardware ->to-> UNCAP (through RAPTOR connection for sending data)

- Software ->to-> UNCAP Interfaces (through iFrames)

These three interactions will be thoroughly described in the final version of the certification suite. For the current version (alpha version), only the compliance to the core Android master app will be presented.

5.4. Connectable to the UNCAP box type of devices

There is already a list of devices that are connecting to the UNCAP box and cloud that are gathering data and monitoring elder persons:

- Blood Pressure and Heart Rate Monitor
- Oximeter
- Scale
- Glucometer

These devices can connect to the UNCAP box that runs on an Android 4.4.4 (or later), or can connect directly to the UNCAP cloud (through the RaptorBox aggregator) to provide the data directly to the server.

A new type of device can be connected, if the driver for it is developed. The data for the device can then already be displayed, if it is in one of the standard, already supported formats. If this is a new type of data, a server-side visualization plugin will also have to be developed.

5.5. Driver description

In the UNCAP website, there is a document that guides the hardware developer into creating UNCAP compliant devices.

5.5.1. Drivers introduction

UNCAP drivers are standalone Android applications. Each application has the following metadata:

- package id (string: com.uncap.driver.glucose.vpd2in1)
- service id (string: com.uncap.driver.glucose.vpd2in1.GlucometerService)
- driverType: "foreground" or "background"
- description: "Human readable description for catalog, etc."
- masterAppButtonLabel: "Blood glucose meter"
- version: (int: 1)
- lastUpdateDatetime: "2015-01-01 11:11:11"
- apkUrl: https://uncap.eu/drivers/driverGlucose.apk
- barCodeId: "00000001"
- hasSettings: true
- hasMeasurement: true
- [more to be added if necessary]

5.5.2. Driver catalog

All of this is packed into a JSON array, and stored on the server. This serves as source data for the UNCAP driver catalog, and for the Android application to learn about new drivers and their capabilities.

```
{
 "drivers": [
   {
     "packageId": "com.uncap.driver.mydriver.mycompany",
     "serviceId": "com.uncap.driver.mydriver.mycompany.HeartrateService",
     "driverType": "background",
     "description": "This is the driver for heartrate",
     "masterAppButtonLabel": "HeartRate",
     "version": 1,
     "lastUpdateDatetime": "2015-25-11",
     "apkUrl": "https://dl.dropboxusercontent.com/u/79979620/hr.apk",
     "barCodeId": "00000003",
     "hasSettings": true,
     "hasMeasurement": true
   },
   {
     ...
   },
   {
     ...
   }
 ]
}
```

Current version of the catalog hosted by TRI: http://www.uncap.eu/driver-apk/driver-list.json

The idea is that if this file is updated regularly, it can power both the UNCAP catalog online, as well as all of the UNCAP Android apps.

5.5.3. Master Android app driver installation

Users can add (install) driver apps in three ways:

- **[The preferred way]** By scanning the barcode / QR code (or manually entering the ID) through the UNCAP master app, which automatically downloads the driver app and prompts user to install it.
- By finding the link in the online UNCAP catalog, which takes the user to the APK.
- By manually finding the app in the Google Play store and installing it.

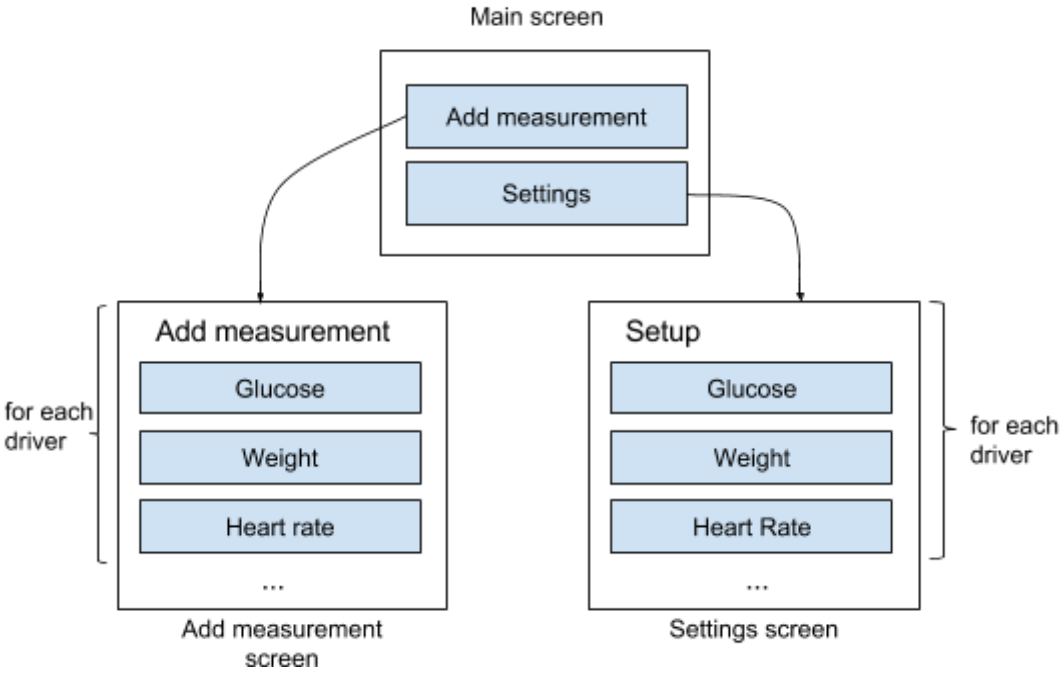
5.5.4. Master Android app driver detection

The master app:

- Regularly fetches the driver catalog (JSON) and stores the last version locally
- Based on the packageId, the master app checks if the packageId is installed
    - This happens when user presses [Settings] or [Add measurement] buttons in the master app
- When user presses the [Settings] button:

- The master app loops through JSON, checking which driver apps are installed
  - For each installed app, it shows the settings button
  - This only happens, if the driver has "hasSettings" flag set to true
  - The button that appears, has the text specified in "masterAppButtonLabel"

- When user presses the [Add measurement] button:
  - The master app loops through JSON, checking which driver apps are installed
    - For each installed app it shows the "Add measurement" button
    - This only happens, if the driver has "hasMeasurement" flag set to true
    - The button that appears, has the text specified in "masterAppButtonLabel"



5.5.5. Communication with the driver

Communication with the driver happens in three ways

- by opening a HTML-based interface in the webview of the master app
  - actions in this interface are triggered using JS, and relayed back to the driver using intents
- by opening the driver application directly from the master app

- this is meant primarily for media-rich and interactive applications (featuring video, animations, or apps that cannot be easily ported to HTML GUI)

- and with the driver sending the data to master app in the background using intents

5.5.5.1. When each option is used:

- For settings, option 1 is always used.
- For measurement, either option 1 or option 2 is used
  - this is based on the "driverType" field in the JSON metatada
  - "driverType": "background" means option 1 will be used
  - "driverType": "foreground" means option 2 will be used
- For communicating data from driver app back to master app, android intents are always used.

Each option is described in more detail below.

5.5.5.2. Option 1: HTML GUIs (for communication master → driver)

Each driver comes with a handful of static HTML (CSS, JS) files in the `assets` directory. On first launch, it copies the files to a folder `(sdcard)/UncapFiles/`.

- For this each driver creates a subdirectory with the name of its package id
  - example: (sdcard)/UncapFiles/com.uncap.driver.glucose.vpd2in1

Each driver comes with two main HTML files (both residing in that directory); both files can include additional CSS or JS files, or have the CSS/JS code inline.

- settings.html → here a settings GUI is defined
- measurement.html → a measurement GUI

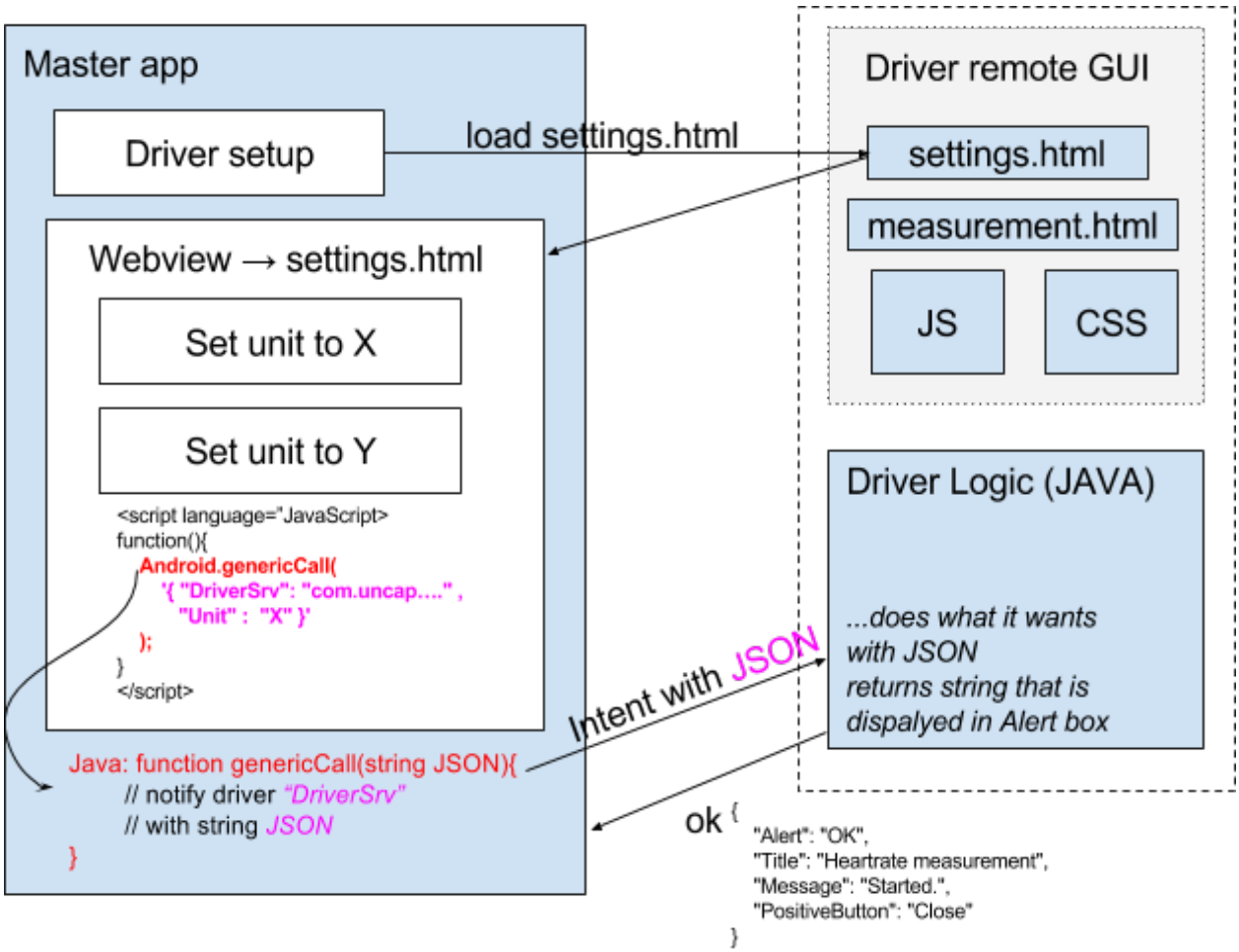
HTML interface guidelines

Each of both HTML files represents a GUI; two main GUIs are foreseen:

- Configuration GUI (settings.html)
- Measurement GUI (measurement.html)

Each interface packs all the data it wishes to communicate back to the driver (= back to itself) in JSON format. Then, it sends the payload to the method *genericCall()* defined in the master app. The payload is then relayed by the master app back to the driver, which unpacks it and parses it.

For an example, see figure below.



5.5.5.3. Option 2: Opening the driver app (master → driver)

If app has the "driverType" == "foreground", the actual driver app is opened in the foreground. When measurements are done, the user is returned back to the master app.

5.5.5.4. Option 3: Communicating back the results (driver → master)

When the driver app obtains the results from the device, it sends the data in JSON form by using intents.

To do this, driver binds to **com.uncap.box.main.COM**.

Four methods are exposed for sending data to the master application:

- pushMeasure is used for sending measurement data.
    - The driver is expected to pack the measurement data as an openmhealth payload, with added measurement type (openmhealth schema id, e.g. `omh:heart-rate:1.0`), like this:

```
{
    "type": "omh:heart-rate:1.0",
    "payload": {
```

```
        "effective_time_frame": {
            "date_time": "2015-12-08T16:37:25Z"
        },
        "heart_rate": {
            "value": 104,
            "unit": "beats\/min"
        }
    }
}
```

- the received JSON string is equipped with user_id (collected from login details), raptor stream name, raptor device id, and is pushed to the message bus
- all data that appears on message bus is picked by the sync module of the master app
- Sync module forwards the data to the server; if there is no connectivity, data is stored and sent later, when connectivity is back
- Raptor receives this and relays it to chino, CEP, etc.

- pushAlarm is used for raising an alarm
- pushPosition
- pushPhr

5.6. Contact details

A dedicated contact point will be responsible with the interactions between the UNCAP community and the developers that need clarifications regarding anything related to the compliance of their devices with the UNCAP ecosystem.

The UNCAP consortium

www.uncap.eu

Coordinator: Trilogis srl

Via F. Zeni 8, 38068 Rovereto, ITALY

Phone: +39 0461 1788032

Email: helpdesk@trilogis.it

Web: www.trilogis.it

5.7. Getting the certificate

The developers must create a new driver intended to work with then UNCAP Android application. First of all, this driver should be able to simulate data – rather than getting the data from the device.

A form should be filled in and sent along with the drivers (simulating ones) to the certification suite contact (see 6.6).

Once it is confirmed that the driver works fine with the UNCAP application in the Driver Catalog (see 6.5.2.), the ServiceID of the driver will be inserted and all the UNCAP devices will now to look for that application.

What the developer has to do next is to spread the product and teach the users to install the driver also. The UNCAP application will receive measurements from the new driver.