
# Development Of An Android Usage Study System

Master Thesis

Tobias Ruhe

September 2012

## Participants

### 1st examiner

Prof. Dr. rer. nat. Josef von Helden
Ricklinger Stadtweg 120
30459 Hannover

E-Mail: josef.vonhelden@fh-hannover.de
Tel.: +49 511 9296-1500

### 2nd examiner

Ingo Bente M.Sc.
Ricklinger Stadtweg 120
30459 Hannover

E-Mail: ingo.bente@fh-hannover.de
Tel.: +49 511 9296-1828

### Author

Tobias Ruhe
Wilhelm-Bluhm-Str. 36
30451 Hannover

E-Mail: tobias.ruhe@stud.fh-hannover.de

**Thesis Declaration**
I declare on oath that I completed this thesis on my own and that information which has been directly or indirectly taken from other sources has been noted as such. Neither this, nor a similar thesis, has been published or presented to an examination committee.

Hannover, September 30, 2012

Tobias Ruhe

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Over 400 millions smartphones were sold worldwide in the 2nd quarter[Gar12] 2012. At the time of writing, there are 500 millions activated Android devices, about one million devices are added every day[1]. The success of modern smartphones can be explained by regarding their capabilities. With lot of sensors built into the phone and the ability to upgrade the functionality by 3rd party applications, the device is not anymore limited to telephony and SMS messaging. Instead, the users can do a lot of more tasks with their phones. Meanwhile the users do also sensitive tasks like payment transactions with their smartphones. As a consequence, the mobile platforms are a worthwhile destination for malware programmers.
The *ESUKOM*[2] project deals with real-time network analysis of correlated meta-data. The emergence of smartphones in business enterprise networks introduces new security problems. One of the key challenge is to encounter new threats caused by smartphones. In order to recognize an anomaly in smartphone activity, it must be distinguish between normal and abnormal behavior. The IDS[3] used in the project has to be trained with real life data to enable realistic assessments about threats.
The goal of this paper is to find out how ordinary users use their smartphones. To achieve this objective an architecture for collecting, storing and analyzing data on the Android platform is introduced. After finishing the development, a field test is run with some volunteers in order to make statements with the collected data. Furthermore the data is used to train the IDS from *ESUKOM* project. A similar project like this thesis is the *Device Analyzer*[4] by the University of Cambridge, but the scope is different from this paper. The intention there is not security related. Instead, improving next generation smartphones on basis of user behavior is focused.

[1] http://news.cnet.com/8301-1035_3-57461870-94/android-activations-reach-1-million-per-day/
[2] http://www.esukom.de/
[3] Intrusion Detection System
[4] http://deviceanalyzer.cl.cam.ac.uk/

This paper is done within the *Trust@FHH*[5] research group and is related to the *ESUKOM* project.

## 1.2 Structure of paper

The paper is separated into the following sections:

**Smartphone usage and market share**
At first an overview of todays mobile market shares is given and the usage of todays smartphone users is discussed. Afterwards, the Android platform is examined.

**Development of an Android usage study system**
At first, the requirements for an Android usage study system are established. Some approaches to meet the requirements are discussed. Finally, the best solutions are chosen.

**Design and implementation of software components**
The software components required for developing an Android usage study system will be designed and implemented. This also includes the deployment of the software. Another object is the field test running with the developed software afterwards.

**Analysis of collected data**
With the data collected during field test, some exemplary analyses will be done.

**Conclusion**
A summary and the experiences from development and the field test are given. The paper finishes with regarding future prospects.

## 1.3 Typographic conventions

The following typographic conventions are used:

*italic:* Keywords, technologies
`teletype:` Script-, class- and file names

[5] https://trust.inform.fh-hannover.de/

# Chapter 2

# Smartphone usage and market share

## 2.1 Overview

In this chapter an overview of the smartphone market is given and the purposes user employ their devices for are examined. Afterwards a survey into the Android plattform is carried out. At the end of this chapter the use cases and requirements for an Android usage study system is discussed.

## 2.2 Smartphone sales market

The market share of smartphones with Android as the operating system is dominant: 64.1 % of all smartphones sales in Q2/2012 are shipped with Android preinstalled.
In comparison with the year before, the trend continues to rise upwards. While *iOS* has a gain of 0.6 percent points, the Android platform circulation has grown by 20.7 percent points. Android devices sell more than twice as much as *iOS* phones. Together, Android and iOS device make up 82.9 percent of all worldwide smartphone sales.
The huge spread of the Android platform and the willingness of the user to do sensitive tasks such as payment transactions or proccessing customer data on smartphones piques the interest of malware programmers. Considering the frequent malware reports from the two well known antivirus companies (F-Secure and Kaspersky Labs), in Q2/2012 more than 40 new families of malware were detected in the wild[Lab12b][Lab12a]. Starting in Q1/2011 with a detection rate of 10 new families and after reaching a peek of 52 in Q3/2011, the rate keeps growing continously in 2012 until now. The malware developer mostly focus on two operating systems (Android, iOS), since the infection rate is proportionaly large to the spread of the system and this leads to higher profits.
As the proverb goes: One man's meat is another man's poison. The rapid evolution

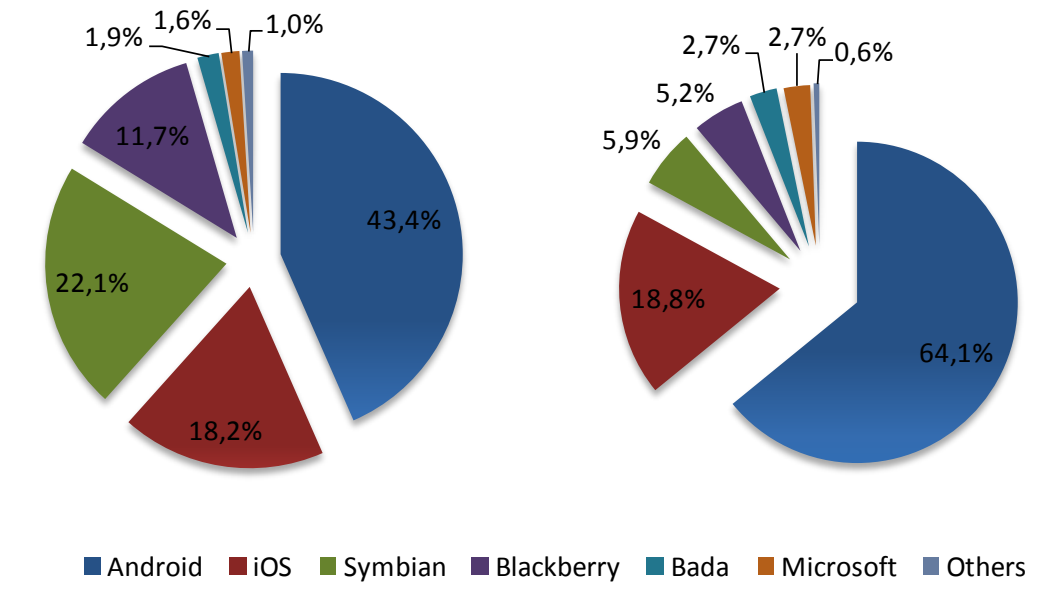



Figure 2.1: The smartphone marketshares for the years 2011 and 2012 (quarter 2)

in the smartphone market leads to new challenges in security. In order to be able to develop counter measures against these threats, a deeper look into common ways people use their smartphones today is needed.

## 2.3 Smartphone usage

Up until a few years ago, the capabilities of telephony devices were limited. Small displays, low CPU power and a small amount of memory restricted the phone to main use cases: telephony and SMS messaging. Today most devices come with a lot of features: a big display, multi core CPUs and a lot of memory and storage space. Also, a lot of additional hardware is built into modern devices, like cameras and GPS sensors. The system can be enhanced by installing additional applications from 3rd parties. Thus, devices are not limited to the classic usage anymore.
The *O2-Company* in the UK released their *All About You Report*[O212] study in 2012. The study makes statements about how much time smartphone users spend on different tasks. The result clearly places telephony and SMS writing at lower positions. With a big distance the users mostly employed their phones for other purposesw like browsing the internet or social networking.
On average, a smartphone user is spending two hours a day using the smartphone. Smartphones are not only used in private life. Also, mobile phone usage has become a big part of everyday business. The main uses are driving directions (GPS), accessing customer data and last minute internet research[WC12].

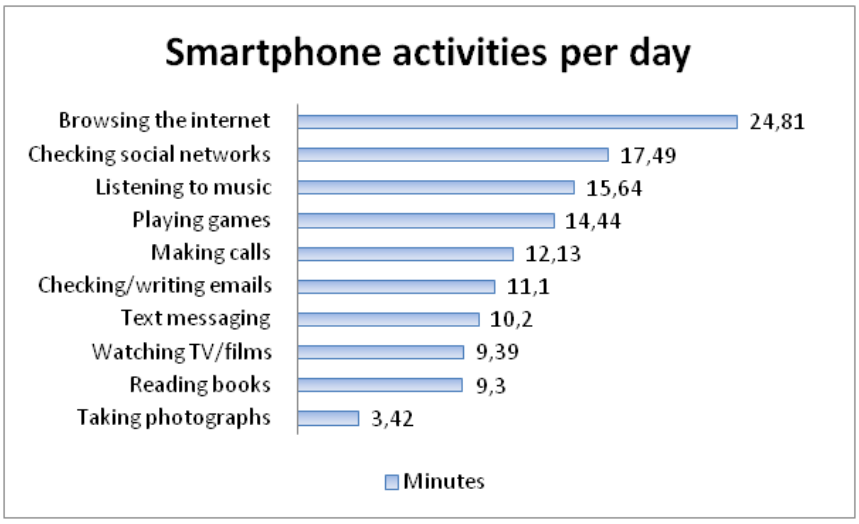



Figure 2.2: The top 10 use cases of smartphone usage. Calling and text messaging reside only on lower positions.

In summary, smartphones are a substantial part of modern people's life. Many tasks can easily done by non-technically-inclined people. Tasks that required workstation systems only years ago can nowadays be performed anywhere on mobile devices.

## 2.4 The Android plattform

Since the introduction of the Android platform, there have been a lot of new releases. Examining the version distribution of the Android operating system, there are almost all older versions are still in circulation. With more than 70 percent, the android 2.x platform is the top distibuted version, where the 2.3 platform (aka Gingerbread) with 57.3 percent is dominating[1]. The latest distributions (versions >= 4.x) are slowy gaining relevance, while old versions below 2.x are mostly irrelevant with a spread rate of 0.6 %.
There exist some other Android ports in the wild, which are developed by a community or 3rd party companies. One of the most popular derivative is *CyanogenMod*[2].

### 2.4.1 The software stack

The Android system is not only an operating system, but a whole software stack composed of different layers (2.4).

[1] http://developer.android.com/about/dashboards/index.html
[2] http://www.cyanogenmod.com/

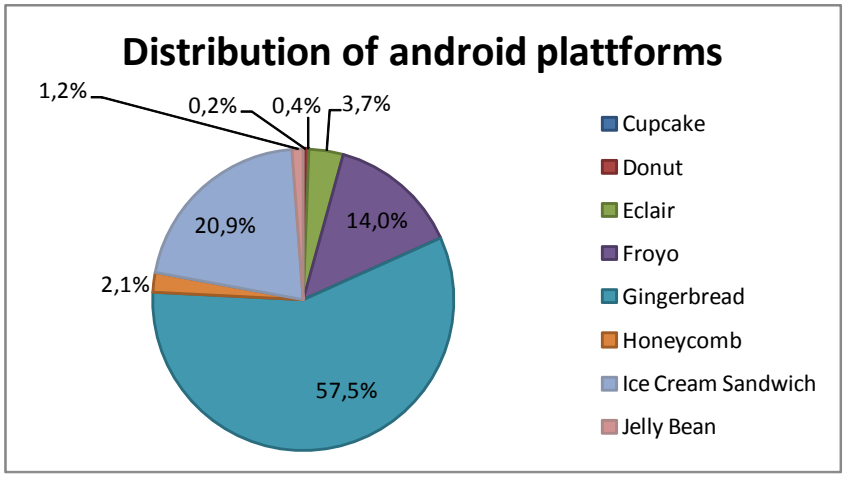



Figure 2.3: The distribution of different Android versions (September 2012).

| Version | Codename | API | Distribution (%) |
|---|---|---|---|
| 1.5 | Cupcake | 3 | 0.2 |
| 1.6 | Donut | 4 | 0.4 |
| 2.1 | Eclair | 7 | 3.7 |
| 2.2 | Froyo | 8 | 14 |
| 2.3 - 2.3.2 | Gingerbread | 9 | 0.3 |
| 2.3.3 - 2.3.7 | Gingerbread | 10 | 57.2 |
| 3.1 | Honeycomb | 12 | 0.5 |
| 3.2 | Honeycomb | 13 | 1.6 |
| 4.0 - 4.0.2 | Ice Cream Sandwich | 14 | 0.1 |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 20.8 |
| 4.1 | Jelly Bean | 16 | 1.2 |

Table 2.1: Versions and API level of Android platform

**Linux Kernel**
The underlying system of every Android version is the Linux operating system. The drivers for the device reside there. Scheduling, process and memory management and other tasks of an operating system are executed here.

**Libraries**
Android includes a lot of libraries known from other conventional Linux operating systems. This covers networking components such as SSL[3], SQLite (an embedded database system) or OpenGL.

[3] Secure Socket Layer

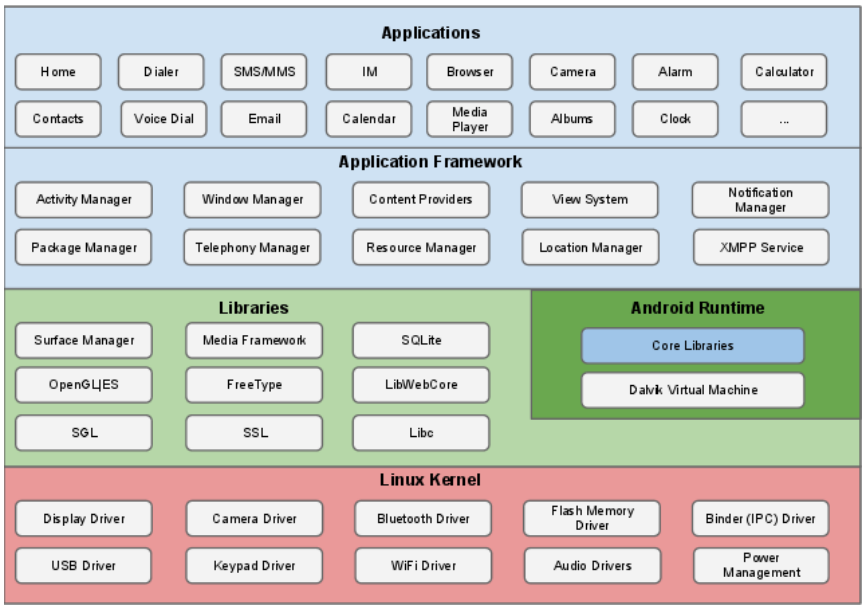



Figure 2.4: The android software stack. Source:[sou]

### Android runtime

The Java programming language is the choice of coding for Android applications. In order to run the apps, a virtual machine (VM) named *Dalvik Virtual machine* is included in Android. Before interpreting and executing the application, the Java byte code is translated to the *Dalvik Executable* file format (files are ending with *.dex*).

### Application framework

Above the libraries and runtime layers, an application framework is built. The developer uses the classes and methods from here to build applications.

### Applications

On top of the software stack, the applications are found. They use the provided libraries for accessing and working with the Android system. All applications are written in Java, but there can also be additional native libraries included, written in C.

Applications can be obtained from the offical Google Play Store[4]. In any Android system, there exists an application to accomplish this task. Some manufactures and 3rd parties have their own market infrastructure to provide additional software.

[4] https://play.google.com/

# Chapter 3

# Development of an Android usage study system

After introducing the Android platform in the previous chapter, the development of an Android usage study system is described in the following. At first, the requirements for the client application and the remote webservice are declared. Afterwards, possible approaches are discussed and the best solution is chosen.

## 3.1 Requirements

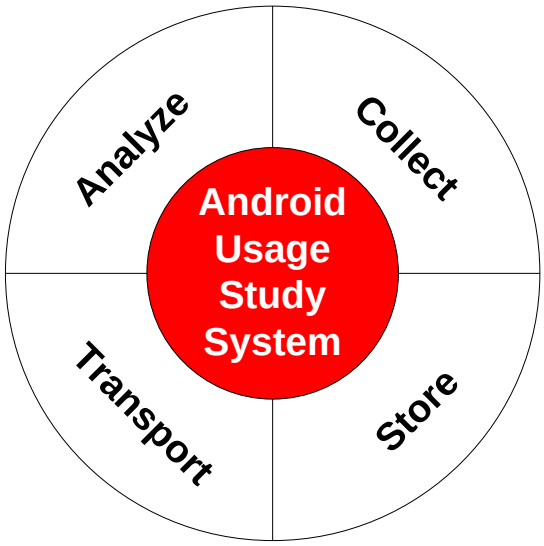



Figure 3.1: The assignments of an Android usage study system

The Android usage study system can roughly be divided into four assignments: Collecting, storing, transporting and analyzing the data. Each use case can potentially

affect both the client and the server. Some combinations will not be considered (eg. analyzing data on client side), because they are not in the focus of this paper.

### 3.1.1 Requirements for the client application

In the following the requirements for the Android client are determined.

**Collecting**

- The app should measure as much information as possible, to cover the most use cases. The app must collect all necessary information that is available for the used API level. That means, if a device does not support a specific feature, all other available features are still collected.
- Data collection should be possible at any time. Features can be measured once or repeatedly at a given time or period. Some features are collected if specific events occur (eg. the user receives an incoming phone call).
- The users do not have access to the internet at all times. Collected data can't be sent at any time, so storing it temporarily on the device is a must.
- The measurement and upload times should be adjustable.

**Storing**

- An appropriate data model for storing, processing and analyzing the data must be introduced. It should be easy to query, navigate and serialize the data. Processing these tasks should be finished in appropriate time.
- The data model must be able to handle different kinds of measurement data.
- As mentioned in the beginning (1.1) of this paper. One goal of this paper is to obtain data from real persons to train an IDS. There, anomalies in smartphone usage should be detected. It should be considered to use the same model, if possible, to avoid the transformation from other data models.
- Because a smartphone has limited capabilities in storage, the space the collected data uses should be as small as possible. After successful upload, the data should be erased on the device.
- Information should be continuously collected by as many participants as possible. Some analyses may require constant measurement of specific features. To achieve usable results for analysis, appropriate measurement intervals should be chosen.
- The user can make a copy of all currently stored features to a chosen path, without removing the original database.

**Transport**

- All data must be transmitted over an IP-networked environment.
- The upload process can be automatically performed by the application or can be manually triggered by the user. If the collected data is not successfully transmitted, it is not erased on client side.

- The transmission protocol must be encrypted and secure cryptographic standards must be used.
- The user can set up new network credentials.

**System**

- The application should run on the most popular Android platforms. Versions with low distribution rates can be ignored.
- The application should not decrease the smartphone's performance significantly. Also the battery power consumption should be considered.
- After rebooting, the app should start without the interaction of the user and continue its work in the background.
- The application should run without the intervention of user. This includes collecting and the automatic upload of data.

**Privacy**

- The identity of the user must be protected. To ensure that the user cannot be tracked by analyzing the data. All sensitive data (eg. the IMEI or telephone numbers) must be anonymized before storing. For cryptographic operations such as hashing or encrypting, well known algorithms with sufficient security must be used.
- The data must not be accessible to other apps running on the smartphone.
- The user has to have full control about which features are collected and when the upload mechanism is triggered. The specific preferences should be easily accessible and intuitive to use.
- The collected data will never be given to other people than members of *Trust@FHH*. After the project has ended, the data will completely erased.

**User interface**
The user should have the ability to

- see what data is being collected
- enable/disable specific features
- change preferences of all measurement and upload intervals. Also network settings for upload host and the paths for local storage must be configurable.
- manually upload data and save all available features to local storage.

Additionally the application should display a boot screen to indicate that the app is starting. The user should see that the application is up and running in background through a notification. The client only sends features to the server, but is never retrieving features.

### 3.1.2 Requirements of webservice

Some requirements made in 3.1.1 affect the webservice as well (eg. the transport part).

- The remote interface should be easily accessible. To achieve a small contact surface for remote attackers, working with a limited set of offered functionality and parameters is a must.
- The availability must be guaranteed. The participant users can upload their data at any time. If possible, the host should be up at all times.
- The development and integration should be as easy and fast as possible. Well known technologies such as application servers and frameworks could be used to achieve this purpose.
- Incoming data must be validated and the payload (database) must be checked for consistency.
- The service must accept data from multiple clients and has to ensure that a previously uploaded database can not be overwritten by a new upload.

## 3.2 Approaches for an Android usage study system

In this chapter approaches to solve the requirements compiled in 3.1.1 and 3.1.2 are discussed. At first, a definition of which data can be collected is made and a proper data metamodel for handling and storing is introduced. Afterwards, the architecture for the client and the server is developed. It is defined how data is collected and stored on the Android device and measurement times are selected. The transport mechanism and webservice interfaces are introduced. To make analysis on the collected data easy, an approach to store the received data in an appropriate format is established. The user's security is fundamental, privacy and encryption methods used in the applications are examined. Finally, the requirements from the previous chapter are compared to the developed concept.

### 3.2.1 Data selection

As mentioned in 2.3, users accomplish a lot of different tasks with their device. In order to achieve a good coverage of these tasks, it is required to define which data should be collected. The following set was developed:

**Basic Information** Overall information about the device and the system. Covers device and system characteristics such as model identifier or kernel version. System preferences should also be considered.

**Network stats** The network activity. This includes WiFi, 3G and bluetooth connections, if such an adapter is available on the device. The amount of connections and incoming and outgoing traffic could be analyzed. For WiFi adapters especially, information about seen and connected access points could be considered.

**Telephony** The amount and duration of incoming and outgoing calls should be collected. Additionally, the change of a GSM cell could be observed.

**Messaging** The messaging activity. Mostly this means incoming and outgoing SMS messages, but this could also be email messages or other types of messaging (eg. Skype).

**Application** The applications which are installed on the device are identified. The required and requested permissions are checked. The installation, update and removal of apps is noted.

**Sensors** The usage of sensors should be recognized. This covers cameras, bluetooth and WiFi adapters, GPS sensors and other types.

**Storage** Information about free and used space on in- and external storage. The insertion and removal of external SD-cards should also be considered.

**CPU and memory** The CPU load and memory usage on the system is observed. Statistics for the whole system and individual apps are considered.

**Power** The power consumption behavior. The battery status and plugged in chargers should be recognized.

The available feature set depends on device capabilities and the used Android API version.

### 3.2.2 Data model

A deeper look into the used metamodel for the correlation engine[WSW+12] is given to decide if it is appropriate.

#### Abstract metamodel

The data model is a subset of the abstract model evaluated in the research project *ESUKOM*[WSW+12]. The original model consists of five component groups: Core, Context-related, signature, anomaly detection and policy components. Only the core and the context-related components are of special interest; the other three components are used only by the correlation engine. In the following, the first two are discussed. With this abstract metamodel, model instances are developed to match the given scenario.

#### Core components

This group consists of components which can be used to describe arbitrary metadata. In our use case the phone data that will be collected is set up with this model.

#### Feature

A feature describes one characteristic for an arbitrary domain model. Depending on the scenario, features could map different purposes. For example, following our scenario, this could be basic system information (kernel-version, screen properties,

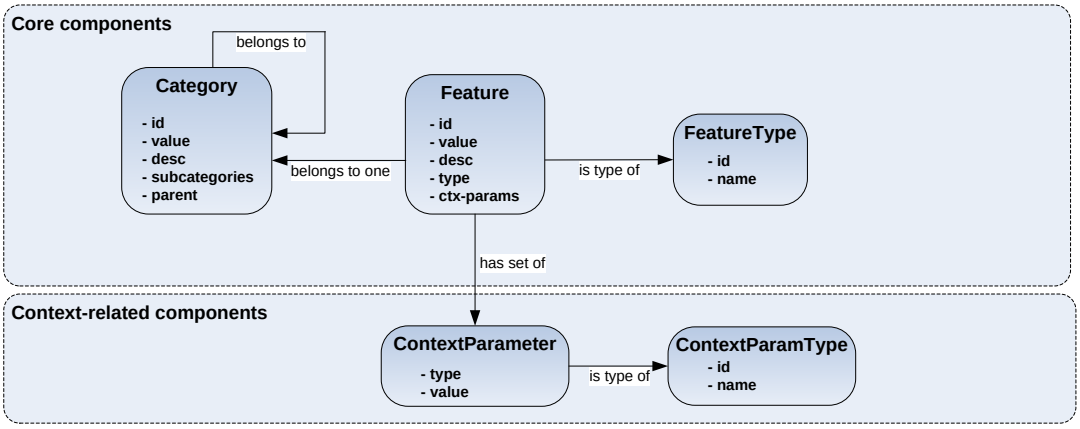



Figure 3.2: Subset of the domain independent abstract metamodel

installed apps) or something like battery and ip-traffic statistics. One feature consists of a single key/value pair, a qualified type and belongs to at least one category. The fields have the following meaning:

**id** An identifier describing the feature

**type** Defines the type of the value. The value can be *quantitative* (an integer), *qualified* (one value from a predefined pool of values) or *arbitrary* (email-address).

**value** A value with respect to the type definition

**category** Each feature belongs to one category

**ctx-params** Optional set of parameters. They can be used to describe arbitrary metadata, which does not directly belong to the feature (such as timestamp or GPS position).

### Category

Each category encapsulates a set of features, which belong semantically to each other. Categories can have subcategories and a parent category. With the ability to nest categories in each other, a tree structure can be modeled. There, the categories are represented by inner nodes, while the features are leaf nodes (figure 3.3). The fields are:

**id** An unique identifier for a specific domain

**subCategories** Optional set of subcategories

**parent** Optional parent category

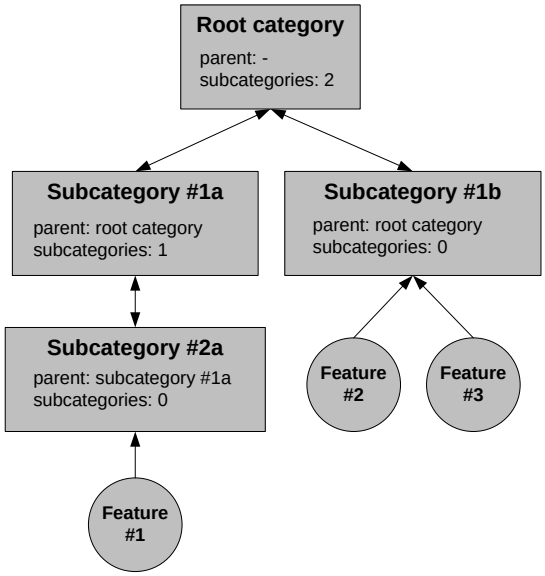



Figure 3.3: The abstract metamodel can be used to model a tree hierarchy. The bidirectional relation symbolizes that a category knows his parent and subcategories. The unidirectional relation between a feature and his category symbolizes that categories have no reference to their features. However, a feature knows his category.

**Context related components**
Defines context relevant metadata, which does not directly belong to the measured feature.

**Context-Parameter**
A single value object with a type definition. The fields are:

**value** a single value

**type** the type of the value (domain specific)

As an example, a Context-Parameter could be a timestamp or a GPS-position.

**Benefits**
There are some benefits to using such an abstract metamodel:

- The same metamodel can be used to model different domains.
- The model does not depend on a certain technology. There are no dependencies to external systems.
- The model can easily be adapted to match new types of metadata.

The data model fulfills the requirements made before. It is easily navigable and can handle different kinds of data. It can easily be adopted, so there is no need to develop a new model.

### 3.2.3 Supported platforms

Higher API levels of the Android system support a richer feature set than lower versions. Supporting lower APIs means to omit some possible measurements; working with higher APIs locks out older devices. A good compromise is to support the versions most prevalent.

### 3.2.4 Transport of data

A secure channel over IP-network is considered. Working with a webservice means the HTTP[1] protocol is involved. This protocol in combination with SSL can be used as a secure and standardized way to transmit data. Using raw sockets on top of TCP/IP requires the development of an exchange mechanism, which can be problematic and time consuming. APIs for HTTP-handling are included in Android, even in lower versions.

### 3.2.5 Storage of data

The mapping of the data model depends on the specific model. For complex data structures working with object oriented databases is considered. For simpler data structures, a flat file or relational database is a better approach. If possible the same storage format is used on client and server side. For the object oriented approach, the db4o[2] can be used, as it is also used by the *ESUKOM* project. A flat file could be in CSV*Comma seperated values* format.

#### Database for objects - introducing db4o

When it comes to serialisation of complex datastructures, the use of an object oriented database management system (OODBMS) should be considered. In contrast to relational database systems (RDBMS), the developer does not need to care about assembling and dissasembling of objects. No manual mapping is needed, the application's data scheme is also the database scheme. This saves coding time resulting in fewer lines of code and the (de-)serialisation of objects is faster. As a consequence less CPU time is used. Also, the navigation of complex graph structures is easier and more intuitive than in relational databases. There are some disadvantages too: the database structure is more complex than in relational tables. Object oriented databases are not widespread, resulting in less support of good user tools. One of the oldest companies in the OODBMS-market is the *Versant Company*[3]. Their major product is the `db4o Object Database`, a leightweight open source OODBMS-engine for Java and .NET platforms. The product is dual licensed: GPL[4] for non-commercial use and a license for commercial use is available. Some advantages:

[1] Hyper Text Transfer Protocol
[2] Database for Objects
[3] http://www.versant.com/
[4] http://www.gnu.org/copyleft/gpl.html

- Each database is stored in one single file.
- The library is build modularly with a small footprint and it is embeddable. The usage in restricted environments with limited resources such as smartphones can be accomplished. db4o supports the Android platform.
- All configurations are done in application code. This makes the database more portable.
- Querying and storing are done with a few lines of code.

In the following an example for using the db4o database is given. The complete db4o reference documentation, API and tutorials can be found at *Versant's* community pages[5].

**Examples**

To demonstrate the capabilities of db4o, some examples with the Java library are shown. For storing and querying the database, a simple entity class `Person` with two member variables and corresponding getter and setter methods is used

```
1 class Person {
2   private String name;
3   private String age;
4
5   Person(String n) {
6     this.name = n;
7   }
8 }
```

Listing 3.1: A simple entity class used for examples

**Prequisites / Installing**

A Java SDK 1.1.x to 1.6 is required to run a db4o database. Before developing an application with db4o, the libraries[6] have to be unpacked and the included `db4o-x-y.jar` file must be extracted (the x in filename is replaced with the db4o version number and y names the module. With module `all`, all components from db4o are included.). The application has to include this JAR-file in the class path.

**Configuration**

As mentioned above, the configuration is completly done in application code.

```
1 EmbeddedConfiguration conf = Db4oEmbedded.newConfiguration();
2 conf.common().add(new AndroidSupport());
```

Listing 3.2: A simple configuration for db4o

A new configuration object from type `EmbeddedConfiguration` is created through a static method in `Db4oEmbedded`. Android support is added to the configuration

[5] http://community.versant.com/documentation.aspx
[6] http://community.versant.com/Downloads/db4o.aspx

by using the add method in `common()`.

**Creating / Opening**

To open a database, the `Db4oEmbedded.openFile()` method can be used. A configuration object and the target file are used as parameters. The database resides in one file only. If the file does not exist in the file system, a new one is created.

```
ObjectContainer db = Db4oEmbedded.openFile(conf,
    "database.db4o");
```

Listing 3.3: Opening or creating the database

**Storing**

Storing is done in a single line of code. Here, objects from type `Person` are saved to the database.

```
db.store(new Person("Alice"));
db.store(new Person("Bob"));
```

Listing 3.4: Opening or creating the database

**Querying**

For querying the database, three existing possibilities can be used in db4o:

- Queries by example
- Named Queries
- Native Queries

These methods are different in complexity and performance.

```
Person alice = new Person("Alice");
alice.setAge(33);
ObjectSet<Person> persons = db.queryByExample(alice);
// Iterate through all found instances
for(Person p : persons) {
  // do something with person object
}
```

Listing 3.5: Query by Example. The result set contains all objects named *Alice* with *age* equal to 33

If using *Queries by example* an example object must be created earlier. The db4o engine searches for objects that look like the sample object. The engine checks all fields from the source object against the non-null fields of the objects in the database. If an object has exactly the same value(s), it is added to the search result set.

```
ObjectSet<Pilot> persons = db.query(new Predicate<Person>() {
    @Override
    public boolean match(Person person) {
        return (person.getName().equals("Alice") &&
          person.getAge() == 33);
    }
});
```

```
// Iterate through all found instances
for(Person p : persons) {
  // do something with person object
}
```

Listing 3.6: Native query

*Native queries* are written directly in the host language. They are type save and are checked during compile time. The query interface does not rely on string literals. With this type of query language a `Predicate` is used to set the query parameters. The user must overwrite the `match()` method to accomplish this task.

```
Query query = db.query();
query.constrain(Person.class);
query.descend("age").constrain(33)
   .and(query.descend("name").constrain("Alice"));
ObjectSet<Object> persons = query.execute();
// Iterate through all found instances
for(Person p : persons) {
  // do something with person object
}
```

Listing 3.7: SODA Query API

The*SODA Query API* is a low level API for operating directly on nodes of the query graph. Strings are used for identifying fields, so no type safety nor compiler checks are possible. Db4o also uses the *SODA Query API* as the internal query mechanism. As a result, all queries from other methods are transformed to *SODA*-queries by db4o. It is the most flexible mechanism for querying db4o databases. At first an instance of a `Query`-object is created. Afterwards the constraints can be set by navigating through the hierachy.
The benefits of db4o are:

- No object transformation is required before storing. The entity objects with their references to other entities are stored transparently, without the need to parse out the data.
- The correlation engine from *ESUKOM* works with db4o to analyze the data. If data is stored in db4o from the beginning, there is no need to transform the data again.
- db4o is compatible with the Android platform. The available libraries for Java work fine on the Android platform[7] as well.
- The database offers multiple facilities to query the object hierarchy programmatically: *Named Queries*, *Native Queries* and *Queries by example*. More information about these techniques is available in the official db4o documentation. There also exist plugins for the Eclipse IDE and Visual Studio 2010 to build queries and browse the object hierachy from within the IDE. These plugins are called *Object Manager Enterprise* (OME).

[7] http://www.db4o.com/Android/default.aspx

## 3.3 Solution

In this section decisions for the Android usage study system are made. As an appropriate data model, the abstract metamodel from *ESUKOM* project is adopted (3.2.2). Normally, collected data must be transformed to match the engines data model. This step can be difficult and time expensive if the data structure is sufficiently complex. Instead, if using the same model, the exchange should be easily accomplished. This makes the data migration simpler. It meets all the requirements: Different kinds of measurement data can be modeled and the resulting data set can easily be navigated. The minimal supported Android platform are 2.2 and above, as this are the most popular versions.
For storing on client and server the object oriented approach is chosen and the same datastructure as in the correlation engine is used (db4o). The db4o database engine can be used with ordinary systems (workstations or server systems) and is also applicable for the Android platform.
For the transport, the clients send the data through a SSL secured HTTP connection. The webservice accepts POST requests with the raw binary data as payload. This seams the simplest solution for transmitting data over an IP network. Android comes with good support to handle HTTP sessions encrypted with SSL. This type of werbservice can be easily developed with the Java platform and Netbeans IDE.

### 3.3.1 Domain-specific metamodel

Based on the abstract metamodel established in the previous section, we develop a model for our domain specific scenario. The requirements can be find in the analysis section. All features, categories and Context-parameter types are defined to match our scenario.

#### Naming conventions

Categories are written in lowercase. Subcategories are named with the full path up to the root. For example, supposing the parent of *battery* is *smartphone*. The ID for this category is *smartphone.battery*. Each category can be separated with a dot (.) or colon (:). A dot is used, if the subcategory is static and there will be only one instance existing, eg. `smartphone.android`. If a subcategory is separated with a colon (:), there can be multiple instances for a parent category. For example, the category *smartphone.android.app* can have multiple instance groups for each application. All features which normally belongs to *smartphone.android.app* are stored into a subcategory. Thus, the apps can be distinguished. A category can have multiple group instances. To clearly identify a feature and to avoid conflicts with category IDs, all features will be written in UPPERCASE.

#### Context-Parameter

To match our scenario we need some additional context information. For any collected feature we need the time when the feature is collected. Every feature must include one instance of this type. Some features have phone numbers involved (eg. receiving a SMS); such information is indicated through a Context-Parameter. For each GPS coordinate, two parameters (longitude and latitude) are used. The feature here will be the distance between the current and the last position. At last, we use a generic Context-Parameter which can include arbitrary data.
For this purpose, the following Context-Parameters are introduced:

#### Ctx-P-Timestamp

- **description:**The time when the feature was measured
- **value:** A timestamp. Can have two representations: The first one is in unix time format (milliseconds since 1970) or in *xsd:dateTime* format (eg. "yyyy-MM-dd'T'HH:mm:ssZ"). More information about time formats are available in Java API[8].
- **type:** timestamp

#### Ctx-P-Phonenumber

- **description:**A phone number
- **value:** A number which represents a phone number.
- **type:** phone number

#### Ctx-P-Longitude

- **description:**The longitude coordinate from GPS position
- **value:** A float number
- **type:** longitude

#### Ctx-P-Latitude

- **description:**The latitude coordinate from GPS position
- **value:** A float number
- **type:** latitude

Except for *Ctx-P-Timestamp*, the parameters are optional and are not essential.

#### Domain-specific features

In this section a model instance for our scenario is introduced. We need to match the requirements defined in 3.2.1. Figure 3.4 shows an overview of all categories which will be developed. All features include at least the `Ctx-P-Timestamp` parameter. Some features use additional Context-Parameters, which will be explained in the list.

#### C: smartphone

[8]http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html

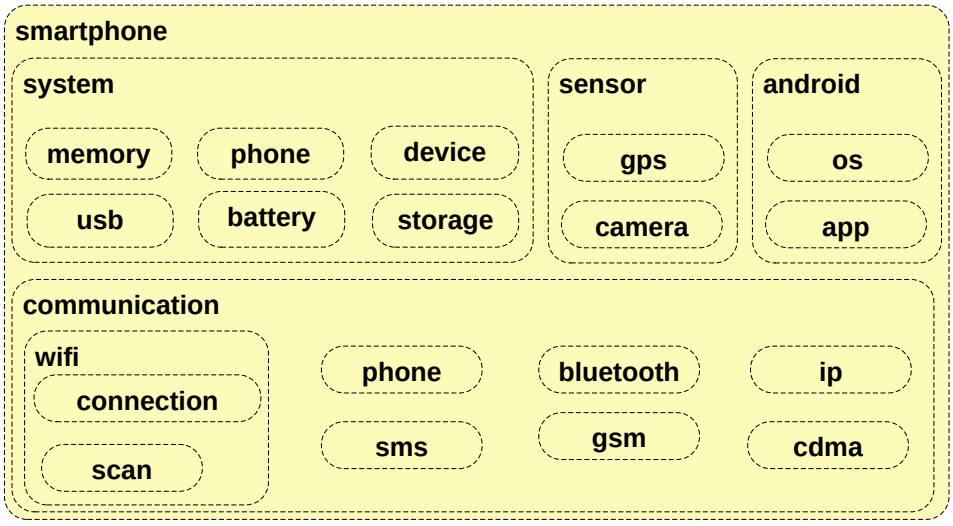



Figure 3.4: Domain-specific categories for Android Usage Study scenario. Each feature resists in one category.

- **id:** smartphone
- **desc:** The root category holding all other categories and features for our scenario.

**C: smartphone.system**

- **id:** smartphone.system
- **desc:** Groups all system specific features like basic device characteristics or information about the underlying Linux system. Also configuration settings, process- and memory information resist here.

**F: IMEI**

- **id:** smartphone.system.IMEI
- **desc:** Each smartphone is branded with an unique device identifier string called IMEI (International Mobile Equipment Identity)[9]. Because the IMEI is (mostly) unchangeable, the users device can clearly identified with this string. To preserve user privacy the value must be hashed before storing.
- **value:** An arbitrary string.
- **type:** arbitrary

**F: IMSI**

- **id:** smartphone.system.IMSI

[9] http://en.wikipedia.org/wiki/International_Mobile_Equipment_Identity

- **desc:** The IMSI has the same characteristics like the IMEI, but it is not identifying the device. It is used to clearly identify the SIM-Card[10]. The value must be hashed too.
- **value:** An arbitrary string.
- **type:** arbitrary

**F: SYSTEM_BOOT**

- **id:** smartphone.system.SYSTEM_BOOT
- **desc:** The last time the smartphone booted
- **value:** date time
- **type:** date time

**F: SYSTEM_SHUTDOWN**

- **id:** smartphone.system.SYSTEM_SHUTDOWN
- **desc:** The Android system is shutting down
- **value:** date time
- **type:** date time

**F: ADB_ENABLED**

- **id:** smartphone.system.ADB_ENABLED
- **desc:** Is the *Android Debug Bridge* (ADB) enabled[11]. Used for Android debugging.
- **value:** true or false
- **type:** qualified

**F: DATA_ROAMING_ENABLED**

- **id:** smartphone.system.DATA_ROAMING_ENABLED
- **desc:** The state of the data roaming feature.
- **value:** true or false
- **type:** qualified

**F: DISPLAY_BRIGHTNESS**

- **id:** smartphone.system.DISPLAY_BRIGHTNESS
- **desc:** The brightness of the display.
- **value:** An integer between 0 and 254
- **type:** quantity

**F: DISPLAY_WIDTH**

- **id:** smartphone.system.DISPLAY_WIDTH
- **desc:** The width of the display.

[10] http://en.wikipedia.org/wiki/International_Mobile_Subscriber_Identity
[11] http://developer.android.com/tools/help/adb.html

- **value:** An integer in px.
- **type:** quantity

**F: DISPLAY_HEIGHT**

- **id:** smartphone.system.DISPLAY_HEIGHT
- **desc:** The height of the display.
- **value:** An integer in px.
- **type:** quantity

**F: NON_MARKET_INSTALL**

- **id:** smartphone.system.NON_MARKET_INSTALL
- **desc:** The possibility to install applications from other marketplaces than Google's Play Store.
- **value:** true or false
- **type:** qualified

**F: MODEL**

- **id:** smartphone.system.MODEL
- **desc:** The model identifier.
- **value:** An arbitrary string
- **type:** arbitrary

**F: FIRMWARE_VERSION**

- **id:** smartphone.system.FIRMWARE_VERSION
- **desc:** The device firmware version.
- **value:** An arbitrary string
- **type:** arbitrary

**F: PRODUCT**

- **id:** smartphone.system.PRODUCT
- **desc:** An identifier describing the phone.
- **value:** An arbitrary string
- **type:** arbitrary

**F: KERNEL_VERSION**

- **id:** smartphone.system.KERNEL_VERSION
- **desc:** The kernel version for the underlying Linux system.
- **value:** An arbitrary string
- **type:** arbitrary

**F: BUILD_NUMBER**

- **id:** smartphone.system.BUILD_NUMBER

- **desc:** The kernel build number.
- **value:** An arbitrary string
- **type:** arbitrary

**F: OS**

- **id:** smartphone.system.OS
- **desc:** The operating systems version number
- **value:** An arbitrary string
- **type:** arbitrary

**F: SDK**

- **id:** smartphone.system.SDK
- **desc:** The SDK version of the Android system
- **value:** Integer
- **type:** qualified

**F: BASEBAND_VERSION**

- **id:** smartphone.system.BASEBAND_VERSION
- **desc:** The smartphones baseband version.
- **value:** An arbitrary string or *unknown*
- **type:** arbitrary

**F: MANUFACTURER**

- **id:** smartphone.system.MANUFACTURER
- **desc:** The smartphones manufacturer.
- **value:** An arbitrary string
- **type:** arbitrary

**F: MAC**

- **id:** smartphone.system.MAC
- **desc:** The MAC-address of WiFi adapter, if available.
- **value:** 6-byte hexadecimal string with (-) as separator or *unknown* if adapter is unavailable.
- **type:** arbitrary

**F: TETHERING_ENABLED**

- **id:** smartphone.system.TETHERING_ENABLED
- **desc:** Indicates if the user has tethering enabled.[12]
- **value:** *true* or *false*

[12]With tethering enabled, the mobile internet connection can be shared to other devices via Wifi. The smartphone acts as a mobile hotspot and routes packets through clients and the connection adapter.

- **type:** qualified

**F: USB_MASS_STORAGE_ENABLED**

- **id:** smartphone.system.USB_MASS_STORAGE_ENABLED
- **desc:** Indicates if USB mass storage is enabled.
- **value:** *true* or *false*
- **type:** qualified

**F: PROCESS_COUNT**

- **id:** smartphone.system.PROCESS_COUNT
- **desc:** The amount of active processes.
- **value:** An integer >0
- **type:** quantity

**F: SIM_STATE**

- **id:** smartphone.system.SIM_STATE
- **desc:** The state of the smartphones SIM card.
- **value:**
  - *SIM_STATE_ABSENT* no SIM card inserted
  - *SIM_STATE_NETWORK_LOCKED*
  - *SIM_STATE_PIN_REQUIRED* a PIN number is required to unlock the card
  - *SIM_STATE_PUK_REQUIRED* a PUK number is required to unlock the card
  - *SIM_STATE_READY* the SIM card is unlocked and ready for work
  - *SIM_STATE_UNKNOWN* the state is unknown
- **type:** qualified

**F: PHONE_TYPE**

- **id:** smartphone.system.PHONE_TYPE
- **desc:** Describes the phone type
- **value:**
  - *PHONE_TYPE_GSM* a smartphone with GSM[13] support
  - *PHONE_TYPE_CDMA* a smartphone with CDMA[14] support
  - *PHONE_TYPE_NONE* a smartphone with neither GSM nor CDMA support.
  - *PHONE_TYPE_UNKNOWN* the device type is unknown
- **type:** qualified

[13] http://en.wikipedia.org/wiki/Global_System_for_Mobile_Communications
[14] http://en.wikipedia.org/wiki/CDMA2000

**F: AIRPLANE_MODE**

- **id:** smartphone.system.AIRPLANE_MODE
- **desc:** When the phone is in airplane mode, all network adapters are turned off.
- **value:** *true* or *false*
- **type:** quantity

**C: smartphone.system.phone**

- **id:** smartphone.system.phone
- **desc:** Groups all phone specific features

**F: SERVICE_STATE**

- **id:** smartphone.system.SERVICE_STATE
- **desc:** The phones service state
- **value:**
  - *STATE_IN_SERVICE* The phone is fully working. It is registered with an operator. The phone could be registered with the home network or roaming.
  - *STATE_OUT_OF_SERVICE* Phone is not operating at current time. There are different reasons, why this situation can appear: The phone is actually searching for an operator and/or is not registered to any. This also happens, if the radio signal is unavailable or the registration to an operator is denied.
  - *STATE_EMERGENCY_ONLY* Only emergency calls can be made (the phone is regularly registered and locked).
  - *STATE_POWER_OFF* The radio or telephony is turned off.
  - *SIM_STATE_UNKNOWN* The phone state is unknown.
- **type:** qualified

**C: smartphone.system.memory**

- **id:** smartphone.system.memory
- **desc:** All memory specific features resists here.

**F: MEMORY_AVAILABLE**

- **id:** smartphone.memory.MEMORY_AVAILABLE
- **desc:** Indicates the available system memory in bytes.
- **value:** An integer >=0
- **type:** quantity

**F: MEMORY_LOW**

- **id:** smartphone.memory.MEMORY_LOW
- **desc:** Indicates that the system considered to be in a low memory state.

- **value:** *true* or *false*
- **type:** qualified

**F: MEMORY_TRESHOLD**

- **id:** smartphone.memory.MEMORY_TRESHOLD
- **desc:** Indicates which threshold (in bytes) has to be reached, before the system is in low memory state and start killing processes.
- **value:** An integer >=0
- **type:** quantity

**C: smartphone.system.battery**

- **id:** smartphone.system.battery
- **desc:** All power related features resists here.

**F: POWER**

- **id:** smartphone.system.battery.POWER
- **desc:** Indicates that the phone is connected to a power supply.
- **value:** *CONNECTED* or *DISCONNECTED*
- **type:** qualified

**F: LEVEL**

- **id:** smartphone.system.battery.LEVEL
- **desc:** The current power level.
- **value:** 0-100 percent.
- **type:** quantity

**F: STATUS**

- **id:** smartphone.system.STATUS
- **desc:** The actual battery status.
- **value:**
  - *BATTERY_STATUS_CHARGING* the phone is charging its battery
  - *BATTERY_STATUS_DISCHARGING* the batteries phone is discharging
  - *BATTERY_STATUS_FULL* the battery is fully charged.
  - *BATTERY_STATUS_NOT_CHARGING* the phone is currently not charging
  - *BATTERY_STATUS_UNKNOWN* the battery state is unknown
- **type:** qualified

**F: VOLTAGE**

- **id:** smartphone.system.battery.VOLTAGE
- **desc:** The current voltage.

- **value:** An integer >=0
- **type:** quantity

**C: smartphone.system.storage**

- **id:** smartphone.system.storage
- **desc:** All features that belongs to storage, resist here.

**F: MEDIA_STATE**

- **id:** smartphone.system.storage.MEDIA_STATE
- **desc:** The state of primary external storage media
- **value:**
  - *MEDIA_BAD_REMOVAL* a bad removal
  - *MEDIA_CHECKING* the storage is currently checked
  - *MEDIA_EJECT* the storage is ejected
  - *MEDIA_MOUNTED* the storage is mounted
  - *MEDIA_REMOVED* the storage is removed
  - *MEDIA_SHARED* the storage is shared
  - *MEDIA_UNMOUNTABLE* the storage is not mountable
  - *MEDIA_UNMOUNTED* the storage is currently unmounted, but is mountable
  - *MEDIA_STATE_UNKNOWN* the storage state is unknown
- **type:** qualified

**F: MEDIA_INTERNAL_SIZE**

- **id:** smartphone.system.storage.MEDIA_INTERNAL_SIZE
- **desc:** The total size of internal storage in bytes.
- **value:** An integer >=0
- **type:** quantity

**F: MEDIA_INTERNAL_FREE**

- **id:** smartphone.system.storage.MEDIA_INTERNAL_FREE
- **desc:** The amount of free internal storage in bytes.
- **value:** An integer >=0
- **type:** quantity

**F: MEDIA_EXTERNAL_SIZE**

- **id:** smartphone.system.storage.MEDIA_EXTERNAL_SIZE
- **desc:** The total size of external storage in bytes.
- **value:** An integer >=0
- **type:** quantity

**F: MEDIA_EXTERNAL_FREE**

- **id:** smartphone.system.storage.MEDIA_EXTERNAL_FREE
- **desc:** The amount of free external storage in bytes.
- **value:** An integer >=0
- **type:** quantity

**C: smartphone.system.usb**

- **id:** smartphone.system.usb
- **desc:** Groups all USB related features

**F: STATE**

- **id:**smartphone.system.usb.STATE
- **desc:** Indicates the state of USB connection.
- **value:** *CONNECTED* or *DISCONNECTED*
- **type:** qualified

**C: smartphone.sensor**

- **id:** smartphone.sensor
- **desc:** Sensor related features resist here.

**C: smartphone.sensor.gps**

- **id:** smartphone.sensor.gps
- **desc:** Information about the GPS sensor.

**F: LOCATION_DISTANCE**

- **id:** smartphone.sensor.gps.LOCATION_DISTANCE
- **desc:** The distance between the actual and last known GPS[15] position.
- **value:** An integer in meter.
- **type:** quantity
- **ctx-params:** There are two additional parameters: the longitude and latitude of current position (*ContextParamTypes* are LONGITUDE an LATITUDE).

**C: smartphone.sensor.camera**

- **id:** smartphone.sensor.camera
- **desc:** All information about camera usage.

**F: NEW_PICTURE**

- **id:** smartphone.sensor.camera.NEW_PICTURE
- **desc:** Indicates that a new picture is taken with a camera.
- **value:** always true
- **type:** qualified

[15] Global Positioning System

**C: smartphone.communication**

- **id:** smartphone.communication
- **desc:** Groups all information about network activities.

**C: smartphone.communication.ip**

- **id:** smartphone.communication.ip
- **desc:** All IP related features resist here.

**F: RX_3G**

- **id:** smartphone.communication.ip.RX_3G
- **desc:** The amount of bytes received through the mobile interface.
- **value:** The traffic in bytes >=0
- **type:** quantity

**F: TX_3G**

- **id:** smartphone.communication.ip.TX_3G
- **desc:** The amount of bytes send through the mobile interface.
- **value:** The traffic in bytes >=0
- **type:** quantity

**F: RX_OTHER**

- **id:** smartphone.communication.ip.RX_OTHER
- **desc:** The amount of bytes received through other interfaces (without 3g).
- **value:** The traffic in bytes >=0
- **type:** quantity

**F: TX_OTHER**

- **id:** smartphone.communication.ip.TX_OTHER
- **desc:** The amount of bytes send through other interfaces (without 3g).
- **value:** The traffic in bytes >=0
- **type:** quantity

**C: smartphone.communication.gsm**

- **id:** smartphone.communication.gsm
- **desc:** Information related to GSM

**F: CELL_ID**

- **id:** smartphone.communication.gsm.CELL_ID
- **desc:** The ID of an GSM cell.
- **value:** an arbitrary string
- **type:** arbitrary

**F: CELL_LAC**

- **id:** smartphone.communication.gsm.CELL_LAC
- **desc:** The local area code of the GSM cell.
- **value:** an arbitrary string
- **type:** arbitrary

**C: smartphone.communication.cdma**

- **id:** smartphone.communication.cdma
- **desc:** Groups all CDMA specific features

**F: CELL_BASESTATION_ID**

- **id:** smartphone.communication.cdma.CELL_BASESTATION_ID
- **desc:** The base station identification number for a CDMA cell.
- **value:** an integer
- **type:** quantity

**F: CELL_NETWORK_ID**

- **id:** smartphone.communication.cdma.CELL_NETWORK_ID
- **desc:** The network identification number for a CDMA cell.
- **value:** an integer
- **type:** quantity

**F: CELL_SYSTEM_ID**

- **id:** smartphone.communication.cdma.CELL_SYSTEM_ID
- **desc:** The system identification number for a CDMA cell.
- **value:** an integer
- **type:** quantity

**F: CELL_LATITUDE**

- **id:** smartphone.communication.cdma.CELL_LATITUDE
- **desc:** The GPS latitude value for a CDMA cell.
- **value:** an integer
- **type:** arbitrary

**F: CELL_LONGITUDE**

- **id:** smartphone.communication.cdma.CELL_LONGITUDE
- **desc:** The GPS longitude value for a CDMA cell.
- **value:** an integer
- **type:** arbitrary

**C: smartphone.communication.wifi**

- **id:** smartphone.communication.wifi
- **desc:** Information bout WiFi related features

**C: smartphone.communication.wifi.connection.#**

- **id:** smartphone.communication.wifi.connection.#
- **desc:** For each different access point a client is connected to, a subcategory is created. An 8-digit hash code of the access-points BSSID is used as name.

**F: BSSID**

- **id:** smartphone.communication.wifi.connection.#.BSSID
- **desc:** The access point's BSSID. Mostly this is the access point's MAC-address, but can also be a random string.
- **value:** an arbitrary string
- **type:** arbitrary

**F: SSID**

- **id:** smartphone.communication.wifi.connection.#.SSID
- **desc:** The access point's SSID.
- **value:** an arbitrary string
- **type:** arbitrary

**F: SSID_HIDDEN**

- **id:** smartphone.communication.wifi.connection.#.SSID_HIDDEN
- **desc:** Indicates if the access point hides its SSID.
- **value:** true or false
- **type:** qualified

**F: LINK_SPEED**

- **id:** smartphone.communication.wifi.connection.#.LINK_SPEED
- **desc:** The speed of established connection in Mbps.
- **value:** integer
- **type:** quantity

**F: IP_ADDRESS**

- **id:** smartphone.communication.wifi.connection.#.IP_ADDRESS
- **desc:** The client's IP-address.
- **value:** integer
- **type:** quantity

**F: MAC_ADDRESS**

- **id:** smartphone.communication.wifi.connection.#.MAC_ADDRESS
- **desc:** The access point's MAC-address.

- **value:** A valid MAC-address.
- **type:** arbitrary

**F: WIFI_STATE**

- **id:** smartphone.communication.wifi.connection.#.WIFI_STATE
- **desc:** The state of the WiFi adapter.
- **value:**
  - *CONNECTED* the smartphone has successfully joined a WiFi network
  - *DISCONNECTED* the smartphone has disconnected from access point
  - *CONNECTING* the smartphone is trying to establish a WiFi connection
  - *DISCONNECTING* the smartphone is disconnecting from access point
  - *SUSPENDED* the string is possible, but the definition is unknown. The API does not describe this value.
  - *UNKNOWN* the WiFi state is unknown
- **type:** qualified

**C: smartphone.communication.wifi.scan.#**

- **id:** smartphone.communication.wifi.scan.#
- **desc:** Information about scanned access points. An 8-digit hash code of the access point's BSSID is used as name.

- **id:** smartphone.communication.wifi.connection.#.BSSID
- **desc:** The access point's BSSID. Mostly this is the access points MAC-address, but can also be a random string.
- **value:** an arbitrary string
- **type:** arbitrary

**F: SSID**

- **id:** smartphone.communication.wifi.connection.#.SSID
- **desc:** The access point's SSID.
- **value:** an arbitrary string
- **type:** arbitrary

**F: FREQUENCY**

- **id:** smartphone.communication.wifi.connection.#.FREQUENCY
- **desc:** The access point's channel frequency in Mhz the client is communicating through.
- **value:** integer
- **type:** quantity

**F: LEVEL**

- **id:** smartphone.communication.wifi.connection.#.LEVEL

- **desc:** The detected signal level of access point in dBm.
- **value:** integer
- **type:** quantity

**F: CAPABILITIES**

- **id:** smartphone.communication.wifi.connection.#.CAPABILITIES
- **desc:** The supported security capabilities (authentication, key management and encryption schemes).
- **value:** a string with special format. Every supported scheme appears in []-Brackets and is parted into maximum three parts for authentication, key management and encryption schemes (in this order). A colon - is used as separator. If a scheme is not supported, it is missing in output. Multiple supported schemes appear in the same line after a closing bracket (]) from previous scheme. If there is support for multiple encryption algorithms, a plus sign (+) is used as separator. Examples:

```
[WPA-PSK-TKIP+CCMP][WPA2-PSK-TKIP+CCMP][WPS]
[WPA2-PSK-TKIP]
[WEP]
```

- **type:** arbitrary

**C: smartphone.communication.bluetooth**

- **id:** smartphone.communication.bluetooth
- **desc:** Information about the bluetooth state.

**F: STATE**

- **id:** smartphone.communication.bluetooth.STATE
- **desc:** Indicates the state of the bluetooth adapter.
- **value:**
  - *STATE_ ON* the bluetooth adapter is turned on
  - *STATE_ OFF* the bluetooth adapter is turned off
  - *STATE_ TURNING_ ON* the bluetooth adapter is going to be turned on
  - *STATE_ TURNING_ OFF* the bluetooth adapter is going to be turned off
- **type:** qualified

**F: SCANMODE**

- **id:** smartphone.communication.bluetooth.SCANMODE
- **desc:** Indicates the scan mode for the bluetooth adapter.
- **value:**
  - *SCAN_ MODE_ CONNECTABLE* other clients can connect to the smartphone bluetooth adapter, but the device is invisible.

- *SCAN_ MODE_ CONNECTABLE_ DISCOVERABLE* other clients can connect to the smartphone bluetooth adapter and the device is visible.
- *SCAN_ MODE_ NONE* its not possible to connect to the bluetooth adapter and the device is not visible
- *SCAN_ MODE_ UNKNOWN* the scanmode cannot be determined

- **type:** qualified

**C: smartphone.communication.bluetooth.#**

- **id:** smartphone.communication.bluetooth.#
- **desc:** Information about bluetooth connections. An 8-digit hash code of the ADDRESS from the target bluetooth adapter is used.

**F: NAME**

- **id:** smartphone.communication.bluetooth.#.NAME
- **desc:** The name of the remote device.
- **value:** an arbitrary string
- **type:** arbitrary

**F: ADDRESS**

- **id:** smartphone.communication.bluetooth.#.ADDRESS
- **desc:** The address of the remote device.
- **value:** an arbitrary string
- **type:** arbitrary

**F: CONNECTION_STATE**

- **id:** smartphone.communication.bluetooth.#.CONNECTION_STATE
- **desc:** The connection state.
- **value:** CONNECTED or DISCONNECTED
- **type:** arbitrary

**F: BOND_STATE**

- **id:** smartphone.communication.bluetooth.#.BOND_STATE
- **desc:** Indicates the bond state for the bluetooth adapter.
- **value:**
  - *BOND_ BONDED* bluetooth adapter is bonded to a remote device
  - *BOND_ BONDING* bluetooth adapter is trying to bond to a remote device
  - *BOND_ NONE* bluetooth adapter is not bonded to a remote device
  - *BOND_ UNKNOWN* bluetooth adapter bonding state is unknown
- **type:** qualified

**C: smartphone.android**

- **id:** smartphone.android
- **desc:** All Android specific features which are not related to the underlying kernel system.

**F: MEDIA_IMAGE_COUNT**

- **id:** smartphone.android.MEDIA_IMAGE_COUNT
- **desc:** The amount of images found on device.
- **value:** An integer >=0
- **type:** quantity

**F: MEDIA_VIDEO_COUNT**

- **id:** smartphone.android.MEDIA_VIDEO_COUNT
- **desc:** The amount of videos found on device.
- **value:** An integer >=0
- **type:** quantity

**F: MEDIA_AUDIO_COUNT**

- **id:** smartphone.android.MEDIA_AUDIO_COUNT
- **desc:** The amount of audio tracks found on device.
- **value:** An integer >=0
- **type:** quantity

**F: RINGMODE**

- **id:** smartphone.android.RINGMODE
- **desc:** The ringmode setting
- **value:**
  - *RINGER_MODE_NORMAL* the ringer mode is normal
  - *RINGER_MODE_SILENT* the ringer mode is silent.
  - *RINGER_MODE_VIBRATE* the ringer mode is set to vibrate
- **type:** qualified

**F: VIBRATE_TYPE**

- **id:** smartphone.android.VIBRATE_TYPE
- **desc:** The vibrate mode. Can be either ringer (the phone is ringing) or noiseless notification.
- **value:**
  - *VIBRATE_TYPE_RINGER* the vibrate mode is ringing
  - *VIBRATE_TYPE_NOTIFICATION* the vibrate mode is notification
- **type:** qualified

**F: VIBRATE_SETTING**

- **id:** smartphone.android.VIBRATE_SETTING
- **desc:** The settings for vibrate mode.
- **value:**
  - *VIBRATE_SETTING_ON* vibration is enabled
  - *VIBRATE_SETTING_OFF* vibration is disabled
  - *VIBRATE_SETTING_ONLY_SILENT* vibration is only enabled in silent mode
- **type:** qualified

**F: SCREEN**

- **id:** smartphone.android.SCREEN
- **desc:** Indicates if the screen is turned on or off.
- **value:** true or false
- **type:** qualified

The subcategory # is named as the underlying feature MAC-ADDRESS.

**C: smartphone.android.app.#**

- **id:** smartphone.android.app
- **desc:** Each individual app resides under this category. Every # subcategory represents one app. A 8-digit hash code, which is generated from the app's unique package name is used.

**F: NAME**

- **id:** smartphone.android.app.#.NAME
- **desc:** The applications name
- **value:** an arbitrary string
- **type:** arbitrary

**F: PACKAGE_NAME**

- **id:** smartphone.android.app.#.PACKAGE_NAME
- **desc:** The applications package name. This value is unique in system and is also used as process name on system level; it can be used to clearly identify a running app.
- **value:** an arbitrary string
- **type:** arbitrary

**F: INSTALLER**

- **id:** smartphone.android.app.#.INSTALLER

- **desc:** The package name of the installer which installs this app. If the app is installed from official Google market, the value is `com.android.vending` or `com.google.android.feedback`. Node: in future Android versions this can change and an other namespace is used. If the value is UNKNOWN or null, the application seems to be installed from other sources (eg. from external SD-card).
- **value:** an arbitrary string
- **type:** arbitrary

**F: VERSION_NAME**

- **id:** smartphone.android.app.#.VERSION_NAME
- **desc:** The applications version name.
- **value:** an arbitrary string
- **type:** arbitrary

**F: VERSION_CODE**

- **id:** smartphone.android.app.#.VERSION_CODE
- **desc:** The applications version code.
- **value:** an integer
- **type:** Integer

**F: MIN_SDK**

- **id:** smartphone.android.app.#.MIN_SDK
- **desc:** The minimum required SDK version in order to run this app.
- **value:** an integer $>0$
- **type:** Integer

**F: INSTALLED**

- **id:** smartphone.android.app.#.INSTALLED
- **desc:** The installation time.
- **value:** date time
- **type:** date time

**F: LAST_UPDATE**

- **id:** smartphone.android.app.#.LAST_UPDATE
- **desc:** The time the app is last updated.
- **value:** date time
- **type:** date time

**F: CPU_LOAD**

- **id:** smartphone.android.app.#.CPU_LOAD

- **desc:** The app's CPU usage. The value is a float between 0.0 and 100.0 in percent.
- **value:** float
- **type:** quantity

**F: REPLACED**

- **id:** smartphone.android.app.#.REPLACED
- **desc:** Indicates if the app is replaced.
- **value:** true or false
- **type:** qualified

**F: REMOVED**

- **id:** smartphone.android.app.#.REMOVED
- **desc:** Indicates if the app is removed.
- **value:** true or false
- **type:** qualified

**C: smartphone.android.app.#.permission:#**

- **id:** smartphone.android.app.#.permission:#
- **desc:** Information about required and requested permissions for a specific app. For every permission a unique subcategory is created, separated by a colon :.

**F: PERMISSION_REQUIRED**

- **id:** smartphone.android.app.#.permission:#.PERMISSION_REQUIRED
- **desc:** Indicates a required permission for this app.
- **value:** an arbitrary string
- **type:** arbitrary

**F: PERMISSION_REQUESTED**

- **id:** smartphone.android.app.#.permission:#.PERMISSION_REQUESTED
- **desc:** Indicates a requested permission for this app.
- **value:** an arbitrary string
- **type:** arbitrary

**F: PROTECTION_LEVEL**

- **id:** smartphone.android.app.#.permission:#.PROTECTION_LEVEL
- **desc:** Indicates a the protection level of a required permission.
- **value:** can be normal (0), dangerous (1), signature (2) and signature or system (3). See the Android developer pages for more information about protection levels.[16]

[16] http://developer.android.com/reference/android/content/pm/PermissionInfo.html

- **type:** integer

**C: smartphone.communication.phone**

- **id:** smartphone.communication.phone
- **desc:** Information about incoming and outgoing calls.

**F: OUTGOING_CALL**

- **id:** smartphone.communication.phone.OUTGOING_CALL
- **desc:** An outgoing call is made.
- **value:** the phone number from called phone. This number will be hashed for hiding the identity.
- **type:** phone number

**F: STATE**

- **id:** smartphone.communication.phone.STATE
- **desc:** The phones calling state. Changes if a call is received or made by the device.
- **value:**
  - *IDLE* the phone is in idle state, no call is active or hold.
  - *RINGING* the phone is in ringing state.
  - *OFFHOOK* the phone is dialing, activating or holding a call.
- **type:** qualified
- **ctx-params:** If the state is *STATE_RINGING* and the caller did not suppress his phone number, the incoming number is included here.

**C: smartphone.communication.sms**

- **id:** smartphone.communication.sms
- **desc:** Information about incoming and outgoing SMS messages.

**F: OUTGOING_SMS**

- **id:** smartphone.communication.sms.OUTGOING_SMS
- **desc:** An outgoing SMS is detected.
- **value:** The receivers phone number. This number will be hashed for hiding the identity.
- **type:** phone number
- **ctx-params:** The length of the SMS (the character count of the payload).

**F: INCOMING_SMS**

- **id:** smartphone.communication.sms.INCOMING_SMS
- **desc:** An incoming SMS is detected.
- **value:** The senders phone number. This number will be hashed for hiding the identity.

- **type:** phone number
- **ctx-params:** Has one additional context parameter: the character count of the received SMS.

In A.1 an overview of all features and categories is given. The measurement times are discussed in the following section.

Table 3.1: Overview of meta model instances

| **Feature / Category** | **Type** | **Interval** |
|---|---|---|
| **C: smartphone** | | |
| **C: smartphone.system** | | |
| F: IMEI | String | once-time |
| F: IMSI | String | once-time |
| F: MODEL | String | once-time |
| F: FIRMWARE_VERSION | String | once-time |
| F: PRODUCT | String | once-time |
| F: MANUFACTURER | String | once-time |
| F: BASEBAND_VERSION | String | once-time |
| F: KERNEL_VERSION | String | once-time |
| F: BUILD_NUMBER | String | once-time |
| F: OS | String | once-time |
| F: SDK | Integer | once-time |
| F: MAC | String | once-time |
| F: SYSTEM_BOOT | Date time | event-based |
| F: SYSTEM_SHUTDOWN | Date time | event-based |
| F: ADB_ENABLED | Boolean | event-based |
| F: TETHERING_ENABLED | Boolean | event-based |
| F: NON_MARKET_INSTALL | Boolean | event-based |
| F: DATA_ROAMING_ENABLED | Boolean | event-based |
| F: USB_MASS_STORAGE_ENABLED | Boolean | event-based |
| F: AIRPLANE_MODE | Boolean | event-based |
| F: SIM_STATE | String | event-based |
| F: SERVICE_STATE | String | event-based |
| F: PHONE_TYPE | String | once-time |
| F: DISPLAY_BRIGHTNESS | Integer | event-based |
| F: DISPLAY_WIDTH | Integer | once-time |
| F: DISPLAY_HEIGHT | Integer | once-time |
| F: PROCESS_COUNT | Integer | periodic |
| **C: smartphone.system.memory** | | |
| F: MEMORY_AVAILABLE | Integer | periodic |
| F: MEMORY_LOW | Boolean | periodic |
| F: MEMORY_TRESHOLD | Integer | periodic |

| | | |
|---|---|---|
| **C: smartphone.system.battery** | | |
| F: POWER | Boolean | event-based |
| F: LEVEL | Integer | periodic |
| F: VOLTAGE | Integer | periodic |
| F: STATUS | String | periodic |
| **C: smartphone.system.storage** | | |
| F: MEDIA_STATE | String | event-based |
| F: MEDIA_INTERNAL_SIZE | Integer | periodic |
| F: MEDIA_INTERNAL_FREE | Integer | periodic |
| F: MEDIA_EXERNAL_SIZE | Integer | periodic |
| F: MEDIA_EXERNAL_FREE | Integer | periodic |
| **C: smartphone.system.usb** | | |
| F: STATE | String | event-based |
| **C: smartphone.sensor** | | |
| **C: smartphone.sensor.gps** | | |
| F: LOCATION_DISTANCE | Integer | periodic |
| **C: smartphone.sensor.camera** | | |
| F: NEW_PICTURE | Integer | event-base |
| **C: smartphone.communication** | | |
| **C: smartphone.communication.ip** | | |
| F: RX_3G | Integer | periodic |
| F: TX_3G | Integer | periodic |
| F: RX_OTHER | Integer | periodic |
| F: TX_OTHER | Integer | periodic |
| **C: smartphone.communication.gsm** | | |
| F: CELL_ID | String | event-based |
| F: CELL_LAC | String | event-based |
| **C: smartphone.communication.cdma** | | |
| F: CELL_BASESTATION_ID | Integer | event-based |
| F: CELL_NETWORK_ID | Integer | event-based |
| F: CELL_SYSTEM_ID | Integer | event-based |
| F: CELL_LATITUDE | Integer | event-based |
| F: CELL_LONGITUDE | Integer | event-based |
| F: CELL_NETWORK_ID | Integer | event-based |
| **C: smartphone.communication.wifi** | | |
| **C: smartphone.communication.wifi.connection.#** | | |
| F: BSSID | String | event-based |
| F: SSID | String | event-based |
| F: SSID_HIDDEN | Boolean | event-based |
| F: LINK_SPEED | Integer | event-based |
| F: IP_ADDRESS | Integer | event-based |

| | | |
|---|---|---|
| F: MAC_ADDRESS | String | event-based |
| F: WIFI_STATE | String | event-based |
| **C: smartphone.communication.wifi.scan.#** | | |
| F: BSSID | String | event-based |
| F: SSID | String | event-based |
| F: FREQUENCY | Integer | event-based |
| F: LEVEL | Integer | event-based |
| F: CAPABILITIES | String | event-based |
| **C: smartphone.communication.bluetooth** | | |
| F: STATE | String | event-based |
| F: SCAN_MODE | String | event-based |
| **C: smartphone.communication.bluetooth.#** | | |
| F: NAME | String | event-based |
| F: ADDRESS | String | event-based |
| F: CONNECTION_STATE | String | event-based |
| F: BOND_STATE | String | event-based |
| **C: smartphone.android** | | |
| F: MEDIA_IMAGE_COUNT | Integer | periodic |
| F: MEDIA_VIDEO_COUNT | Integer | periodic |
| F: MEDIA_AUDIO_COUNT | Integer | periodic |
| F: RINGMODE | Integer | event-based |
| F: VIBRATE_SETTING | Integer | event-based |
| F: SCREEN | String | event-based |
| **C: smartphone.android.app.#** | | |
| F: NAME | String | event-based |
| F: PACKAGE_NAME | String | event-based |
| F: INSTALLER | String | event-based |
| F: VERSION_NAME | String | event-based |
| F: VERSION_CODE | Integer | event-based |
| F: MIN_SDK | Integer | event-based |
| F: INSTALLED | Date time | event-based |
| F: LAST_UPDATE | Date time | event-based |
| F: CPU_LOAD | Integer | event-based |
| F: REPLACED | Boolean | event-based |
| F: REMOVED | Boolean | event-based |
| **C: smartphone.android.app.#.permission:#** | | |
| F: PERMISSION_REQUIRED | String | event-based |
| F: PERMISSION_REQUESTED | String | event-based |
| F: PROTECTION_LEVEL | Integer | event-based |
| **C: smartphone.communication.phone** | | |
| F: STATE | String | event-based |

| | | |
|---|---|---|
| F: OUTGOING_CALL | phone number | event-based |
| **C: smartphone.communication.sms** | | |
| F: OUTGOING_SMS | phone number | event-based |
| F: INCOMING_SMS | phone number | event-based |

**Measurement times**

The measurement times can differ through the whole feature set. Some statements only make sense if the needed features are gathered continuously over a period of time and an adequate amount of data is collected. The reasons for missing features are

- the phone is turned off over a long period
- the is no hardware support (eg. camera is missing)
- the user has disabled the collecting of features
- the app is stopped or has crashed
- the user uninstalled the app before uploading the gathered data

Also, the measurement intervals must be chosen intelligently. If too short, the CPU and power consumption may increase inappropriately. If too long, some important features are missing. Looking at the feature set in 3.3.1, the features can be grouped based on three measurement times:

- One time
- Periodic
- Event based

One time features are features like the IMEI or model name. They will be measured only once at the first start of the application. They will probably not change during the lifetime of a smartphone. If a user upgrades the operating system, the application will (hopefully) be reinstalled, and the information is collected again. Some features must be measured periodically, eg. the battery status or scans for WiFi access points. The intervals should range between one minute and up to one day. The client comes with default measurement times, but these can be changed or adjusted at any time. Event based features occur only sometimes at an unspecific time. Such features are incoming and outgoing calls or new picture being taken with the camera.

# Chapter 4

# Design and implementation of software components

In this chapter the previously developed concept for an Android usage study system is implemented.

## 4.1 Architecture of the Android usage study system

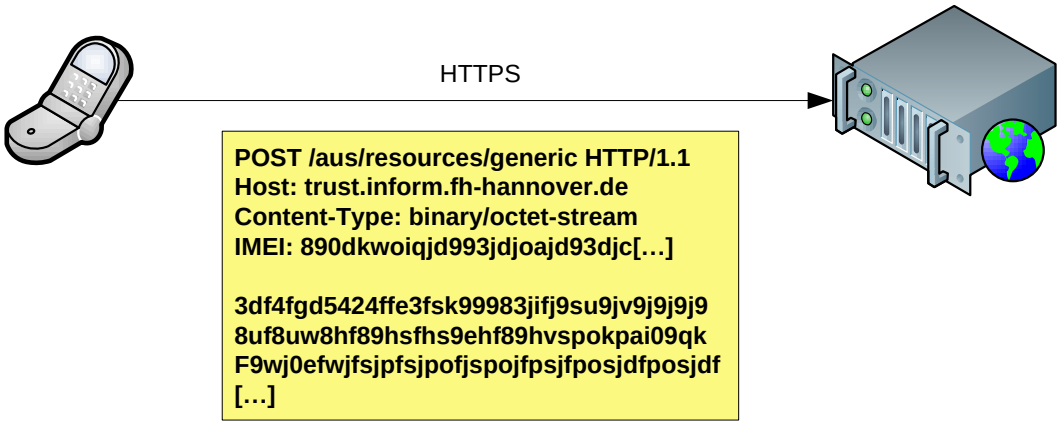



Figure 4.1: The client uploads their data to a server instance over the internet. A HTTPS POST-request with the data as payload is used.

As shown in 4.1, the data is transmitted via a simple HTTP-request including the binary db4o database file. The request type is POST and it's payload is the base64 encoded db4o file. There is an additional custom header field: IMEI. It contains the hashed IMEI of a smartphone user. It is used for saving the file with this hashed prefix in order to keep the different devices apart. If the header does not exist, a default file name is chosen. The webservice is available via SSL and offers one interface to

the user. This interface only accepts POST data with Content-Type 'binary/octet-stream'. All other types are rejected. If the input validation of the IMEI-header fails, the request is no longer processed. The service also checks the payload for a correct db4o database.

## 4.2 Client application

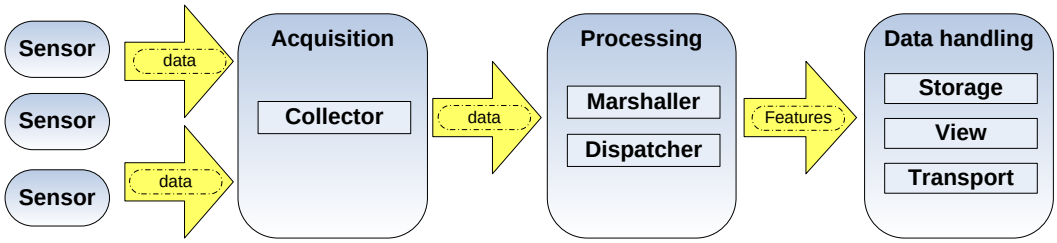



Figure 4.2: The clients components

The app can be divided into three components: Acquisition, processing and handling of data.
The acquisition component handles all incoming data collected by sensors. Sensors collect the data in background. Each sensor has different characteristics in type and quantity of the collected data. Some are observers and listen to system changes and some are only triggered on user action. Others will repeadly be activated at a specific time. The data the sensors collect can have different representations, so transformation to a common format is necessary. Incorrect data is rejected, the rest is relayed to the processing component. The processing component is responsible for marshaling the collected data to the metamodel introduced in 3.3.1. After transformation, the features are relayed to the next component. The data handling component acts as a dispatcher and chooses what to do with the incoming features. The data could be stored, viewed or transported. The application is implemented with the following package structure:

**de.fhhannover.inform.trust**
The metamodel classes shared from the *Trust@FHH* team. This package is included from an external JAR-archive.

**com.db4o.* / EDU.purdue.cs.bloat.***
The db4o package for storing and queryng the data. This package is included from an external JAR-archive.

**de.fhh.inform.trust.aus.activity**
All views resists here.

**de.fhh.inform.trust.aus.alarm**
Helper classes for repeadly executed tasks.

**de.fhh.inform.trust.aus.metadata**
Helper classes for working with meta data.

**de.fhh.inform.trust.aus.processor**
Components for marshalling the data.

**de.fhh.inform.trust.aus.receiver**
Sensor components which trigger on system or user events.

**de.fhh.inform.trust.aus.service**
Sensor components which will repeadly be activated at a specific time.

**de.fhh.inform.trust.aus.storage**
Classes that deals with the storage of data.

**de.fhh.inform.trust.aus.util**
Many other utility classes.

### 4.2.1 User interface

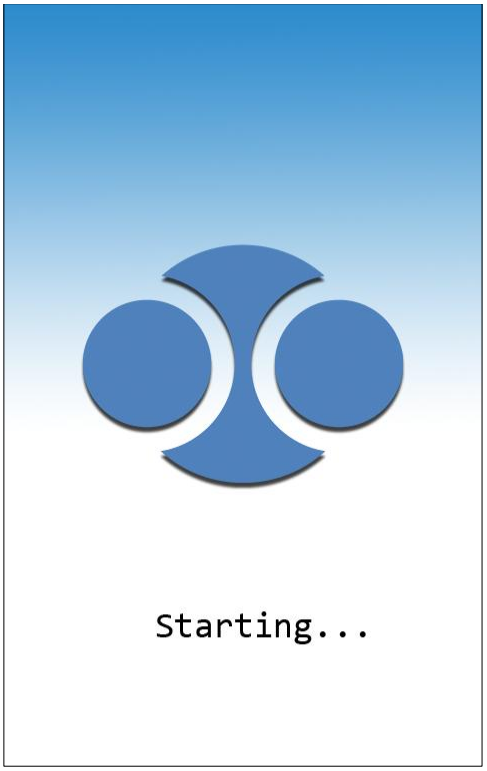
The user interface allows the user to

- see what data is being collected
- enable/disable collection of specific features
- set measurement intervals
- manually upload data

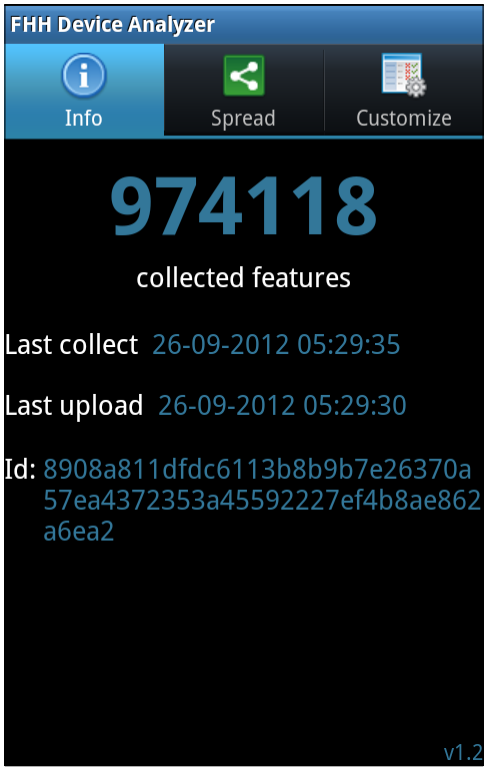
To achieve this, four screens are introduced (Figures 4.3 and 4.4. The *Boot*-View is shown on start of the application and is hided when initialisation is finished. The *Info*-View shows the overall user statistics such as the amount of collected features or the last upload date.
The *Spread*-View gives the user the ability to upload the data or store it in the file system.
The *Customize*-View lets the user change the preferences of the `FHH Device Analyzer`. The user can enable or disable groups of features. Settings for nearly every category and upload automation exist. The *Log*-view shows detailed information about every collected feature.

Figure 4.3: FHH Device Analyzer screenshots #1

(a) Boot screen




(b) Information about collected data




### 4.2.2 Developement enviroment

The *FHH Device Analyzer* is developed with the Eclipse[1] IDE and the Android SDK[2].

### 4.2.3 Delivering and installing

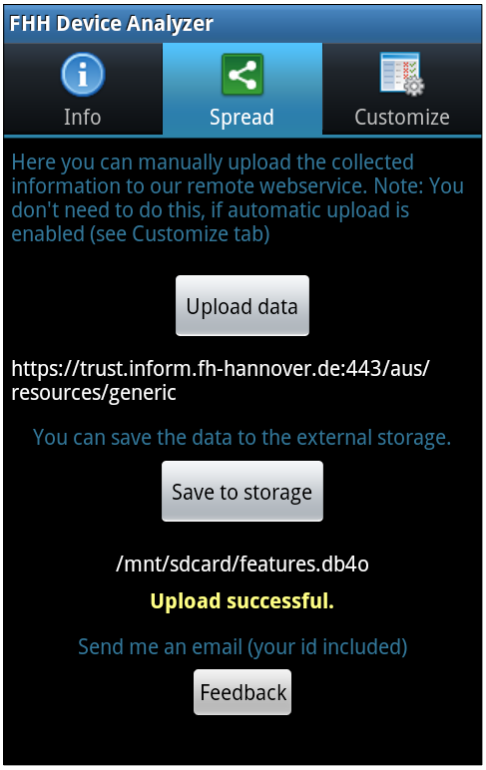
The *APK* file format is used for packaging and distributing applications. The file postfix is *.apk*. The app is named `FHH Device Analyzer` and can be obtained from the Wiki pages [3] at FHH. The user can download the *.apk* archive manually or let the smartphone install it via QR-Code[4]. The app requires the following Android permissions to run:

```
android.permission.INTERNET
android.permission.ACCESS_NETWORK_STATE
```

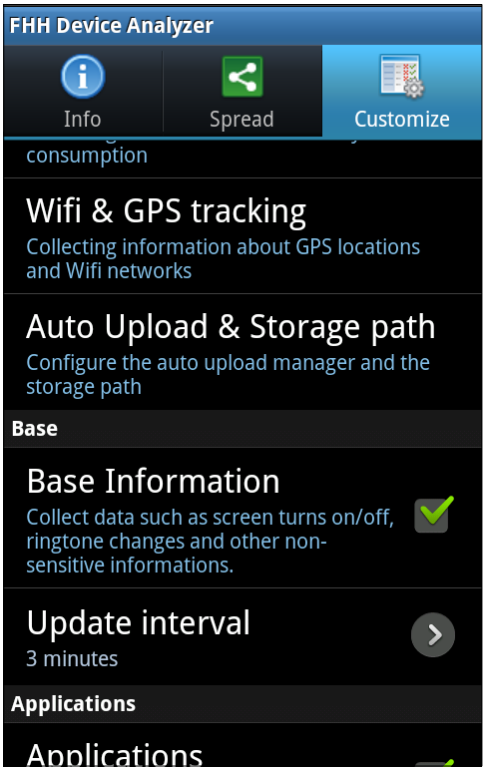
[1] https://www.eclipse.org/
[2] http://developer.android.com/sdk/index.html
[3] https://trust.inform.fh-hannover.de/trust_redmine/projects/android-usage-study-2012
[4] Quick response code

Figure 4.4: FHH Device Analyzer screenshots #2

(a) Upload and storage

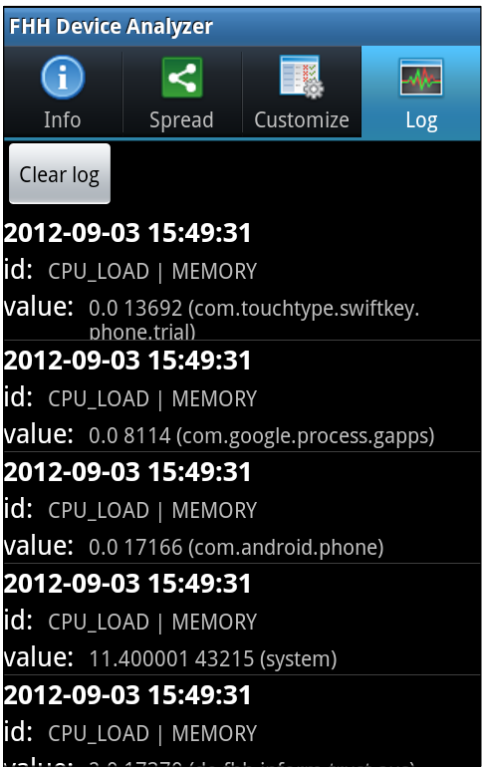



(b) Preferences




```
android.permission.ACCESS_WIFI_STATE
android.permission.READ_PHONE_STATE
android.permission.PROCESS_OUTGOING_CALLS
android.permission.RECEIVE_SMS
android.permission.BROADCAST_PACKAGE_REMOVED
android.permission.BROADCAST_PACKAGE_REMOVED
android.permission.READ_SMS
android.permission.WRITE_EXTERNAL_STORAGE
android.permission.ACCESS_FINE_LOCATION
android.permission.ACCESS_COARSE_LOCATION
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
android.permission.GET_TASKS
android.permission.BROADCAST_STICKY
android.permission.WAKE_LOCK
android.permission.CHANGE_WIFI_STATE
```

Figure 4.5: FHH Device Analyzer Log view




## 4.3 Webservice

The webservice part is developed with Netbeans[5] IDE. For hosting the webservice, the Glassfish[6] 3.x application server is used (which is already shipped with Netbeans). The chosen language is Java 1.6 with it's webservice stack JAX-WS (present since Java 1.5). The API offers a powerful set of functions for easily building and deploying RESTful[7] webservices.

To implement the service, the user must implement one method and using Java annotations for declaring the service interface.

```
1  @POST
2  @Consumes("binary/octet-stream")
3  @Produces(MediaType.TEXT_HTML)
4  public String postData(byte[] data) {
5      ...
6  }
```

Listing 4.1: The webservice method with annotations

[5] http://netbeans.org/
[6] http://glassfish.java.net/
[7] http://en.wikipedia.org/wiki/Representational_State_Transfer

The explanations of the fields are:

- **@POST** Set the interface to POST-request
- **@Consumes("binary/octet-stream")** Only accept POST-data of the given type
- **@Produces(MediaType.TEXT_HTML)** The content-type we return

With this declarations, the webservice accepts POST-requests with Content-Type *binary/octet-stream* as payload and its return type is text or HTML. Every time a client uploads data, this method is called. The whole webservice source code is available in the appendixB.

### 4.3.1 Deploying

With Netbeans, the webservice is exported as WAR[8] archive. This file includes all files to run on any Java certificated application server. On the Glassfish platform, the webservice can be deployed via web interface or commandline tool.

### 4.3.2 Receiving data

All sessions are secured with SSL. For the fieldtest, the Glassfish server is running in a virtual machine hosted at *trust.inform.fh-hannover.de*. To establish a secure channel, the public certificate is delivered with the client application. If a client sends data to the webservice and the Content-Type value is other than *binary/octet-stream*, the request is rejected with an error page. Because the IMEI value in header is used for naming the payload in file system, an attacker may try some directory traversal tricks. Thus the value is validated before using. If the IMEI header is absend, the value *unknown* is used instead.

### 4.3.3 Storing the files

Before storing the payload, a consistence check is done before. The db4o library includes some methods to accomplish this task. If the check is successful, the data is stored in the *upload* directory in the Glassfish installation folder with the following name convention:

**IMEI_YYYYMMDDHHMMSS.db4o**

The IMEI part is a 32-bit alphanumeric value, the result of a SHA[9]-256 one way trap function. It only contains letters and digits, no special characters are used. The second part is the upload date with the precision of seconds. If the check fails, the payload is stored anaway for later examination. The postfix is always *.db4o*, or if the database is corrupted *.corrupt*.

[8] http://en.wikipedia.org/wiki/WAR_file_format_(Sun)
[9] Secure hash algorithm

# Chapter 5

# Analysis of collected data

To show what information can be obtained from the collected data, some exemplary statements are done in this chapter.

## 5.1 Preliminary action

Before the data is analyzed, some sanitizing may be necessary:

- Merging the database files
- Droping out useless data sets
- Repairing broken hierarchy

### 5.1.1 Merging databases

The clients upload their data from time to time, sometimes at several times a day. At the end of the field test, the Glassfish's *upload/* directory contains multiple files for each participant. To be able to query the data efficiently, we have to merge the parts to one database for each smartphone user. Also, all databases should be merged to one single database. The user can choose which variant is better suited for his purpose. Some commandline tools have been written to accomplish these tasks. These tools begin with a `db4o_`-prefix. Most of them accept two parameters, pointing to a file or directory path.

```
db4o_merge_multiple SOURCE_DIR TARGET_DIR
db4o_merge_multiple_to_one SOURCE_DIR TARGET_FILE
db4o_merge_one_to_one SOURCE_FILE TARGET_FILE PREFIX
```

The first one merges all files with `.db4o`-postfix found in *SOURCE_DIR* to *TARGET_DIR*. For each client, a separate database with its hashed IMEI-prefix is created. The second variant behaves similarly, except that all databases result in one single *TARGET_FILE*.

With the third variant, all files with `.db4o`-postfix and *PREFIX* as the beginning of the filenames are merged.

### 5.1.2 Droping out useless data sets

After merging, the useless data sets are removed. During the field test, some people reported trouble with the *FHH Device Analyzer* running on their smartphone. Sometimes the application crashed after a while or there were problems with the upload mechanism. Some participants only ran the app temporarily or removed it after a short period. On some devices manufactured by HTC[1] it stopped working at all. As result, some data sets cannot be used for some analysing purposes. For example, to make statements about CPU usage over time, there is a need to have enough measurement data. These databases are excluded from next the step.

### 5.1.3 Repairing broken hierarchies

This step may become necessary, as bugs in the app are discovered: Some features are sorted into the wrong category. Also, the tree hierarchy can be broken, especially when it comes to namespaces with colons (:). There can also be duplicates, which have to be removed for some statements (basic information like the IMEI can be published repeatedly, eg. if the user reinstalles the app).

However, for most statements, the hierarchy is intact and no further preperation is required.

## 5.2 Exemplary analysis

### 5.2.1 Overall stats

- 15 participants
- 2,867,127 million collected features

The main field test is run for about 2 weeks, but some devices has been run for more than 1 month. In the following, the databases with the most promising amount of demanded features are chosen.

### 5.2.2 Traffic statistics from WiFi adapter

This analysis deals with statistics about incoming and outgoing network traffic. To make statements about the amount of data, two features are of special interest:

**smartphone.communication.ip.TX_OTHER**
**smartphone.communication.ip.RX_OTHER**

[1] http://www.htc.com/

These features describe the traffic from the WiFi adapter. The measurement is done in 15 minute intervals over a period of around two weeks. Five devices are selected for comparison. In figure 5.1 the outgoing traffic is shown. The x-axis denotes the different devices (1-5), while the y-axis shows the amount of transmitted data. It is important to note the y-axis is log scaled. It is discovered, the median is located at

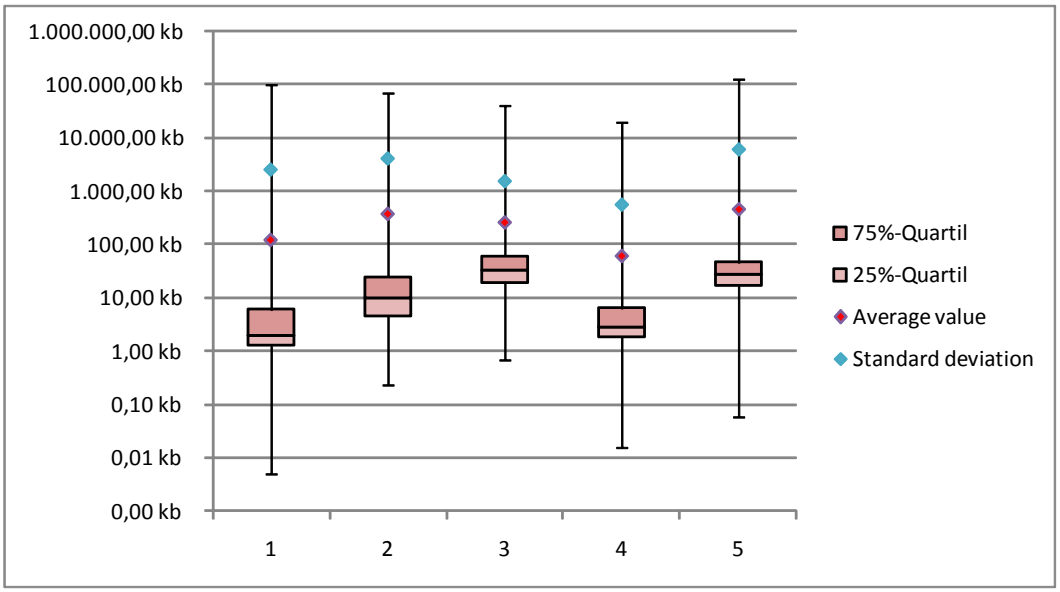



Figure 5.1: Outgoing traffic for WiFi connections

<50 kb or even <10 kb per 15 minutes. In total, the most traffic amount is between 1 kb and 100 kb. The outlier above the 75%-quartil can be interpreted as a side effect of the FHH Device Analyzer app. The data upload can produce a significant increase of outgoing traffic.
When examining the outgoing traffic in figure 5.2, the overall median is located at around the 100 kb mark. The devices 1 and 4 clearly remain under this mark (10 kb and 5 kb). These devices may rarely be used for internet surfing. On all devices, the outlier is located over 10 Mb and it's maximum is higher than 100 Mb. These values can be reached by installing new or updating old software. Also watching videos or video chatting can increase the incoming traffic significantly. In comparison the incoming traffic is 10 times higher than the outgoing traffic.

### 5.2.3 Scanned WiFi access points

The next statements are about WiFi access points and their capabilities in supporting security mechanisms. The *FHH Device Analyzer* stores information about all scanned access points under a category below `smartphone.communication.wifi.scan.#`. Each subcategory of `smartphone.communication.wifi.scan` represents one

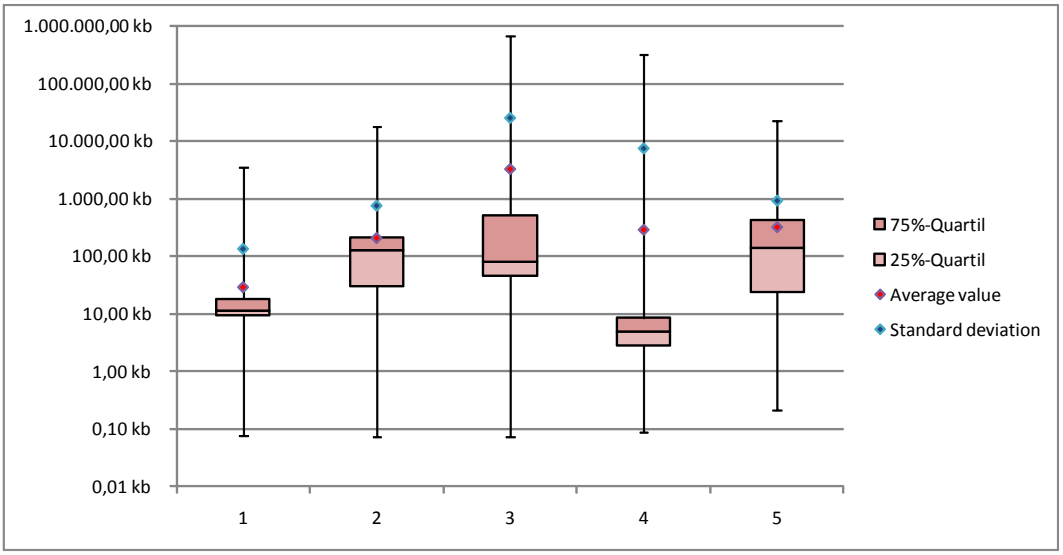



Figure 5.2: Incoming traffic for WiFi connections

access point. Every scanned device reports it's capabilities in supported authentication, key management and encryption schemes (in this order) and is mapped by the `smartphone.communication.wifi.connection.#.CAPABILITIES` feature (3.3.1). Five devices are selected for comparison. The days of measurement differ from device to device. As visualized in figure 5.3, the amount of access points is increased nearly proportional with the time. Only device 185 deviates here, having seen nearly two thousand access points in eight days. Some possible reasons are

- The participant resides in an area of high population
- The participant travels a lot
- There are areas where the amount of active access points is extremely high (eg. a computer security exposition) and the participant remained there.

On average every device saw 45-69 access points a day. The outlier saw 242 devices a day, this is nearly 3-5 time higher than other participants. Checking figure 5.5 for the authentication scheme mostly used, the *WPA/WPA2*[2] protocols clearly dominate at 91.5% in average. With 5.4% the insecure and outdated[TWP07] *WEP*[3] authentification scheme is used. The *OPEN*-networks mostly do their authentification on different layers after a successful WiFi connection with the client. This types of access point are often seen at train stations.
Another interesting fact is the wide spread of the WPS[4] protocol. This protocol created by the *Wi-Fi Alliance* is used to make the adjustment of access point clients

[2] Wi-Fi Protected Access
[3] Wired Equivalent Privacy
[4] Wi-Fi Protected Setup

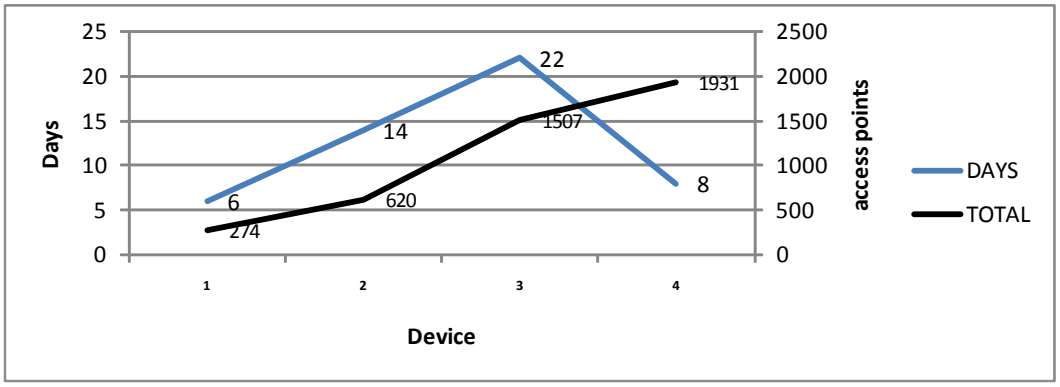



Figure 5.3: The measurement days

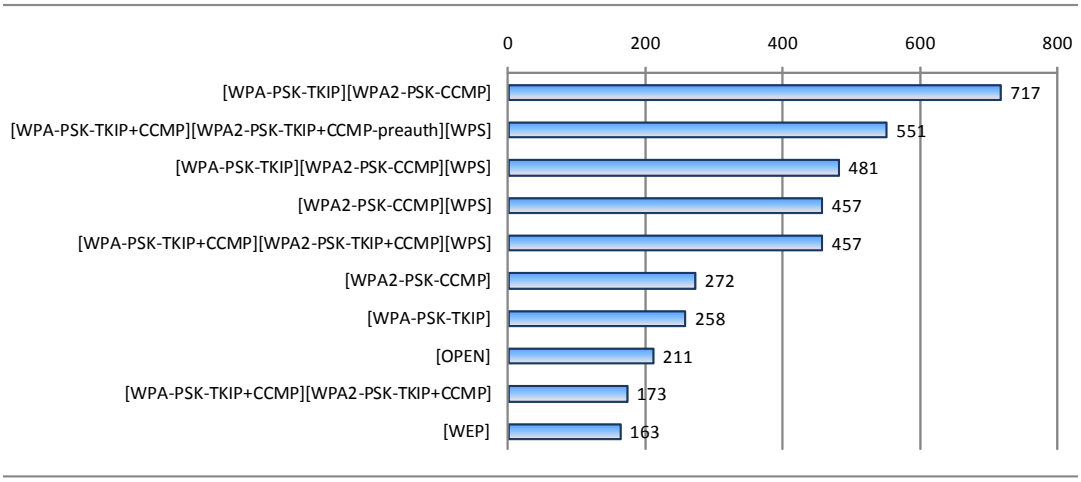



Figure 5.4: Top 10 capabilities strings of seen access points

more practical. Users with little or no knowledge of wireless security can easily setup their devices without knowing something about authentication schemes or typing long pass phrases. The protocol specifies multiple methods to achieve this. One of them allows the user to enter an eight digit numerical PIN[5] to trigger the access point to transmit the pre-shared keys needed to connect. The PIN is set to default factory settings or can be chosen by the owner. To get the device to be certified as *WPS* conform, all manufactures have to support this type of method. Most models come with *WPS* enabled by default.

Unfortunately, the protocol is vulnerable to brute-force attacks[Vie11]. Due to design flaws, the attempts an attacker has to untertake before guessing the right number, can be dramatically decreased. The maximum attempts to guess the right digit is 11,000.

[5]Personal identification number

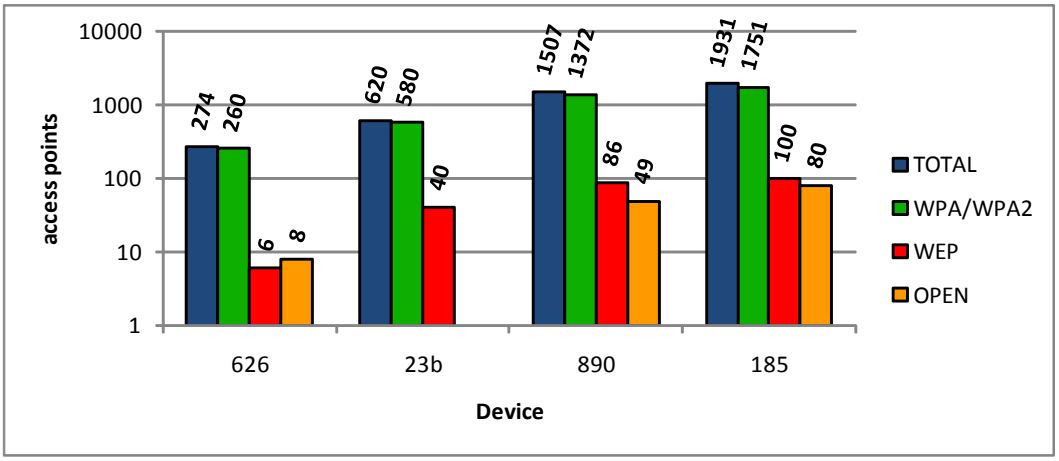



Figure 5.5: Overview of capabilities per device

Some routers may limit the amount of attempts in a period of time. The maximum guessing time in a worst-case scenario (blocking after five attempts for one hour) is 91.83 days, best-case is (with no lock down time) 0.17 days.
More than half (53.6%) of all seen access points (4332) have this feature enabled. The count of *WPS*-enabled devices increases proportional to the amount of seen access points on every smartphone5.6.

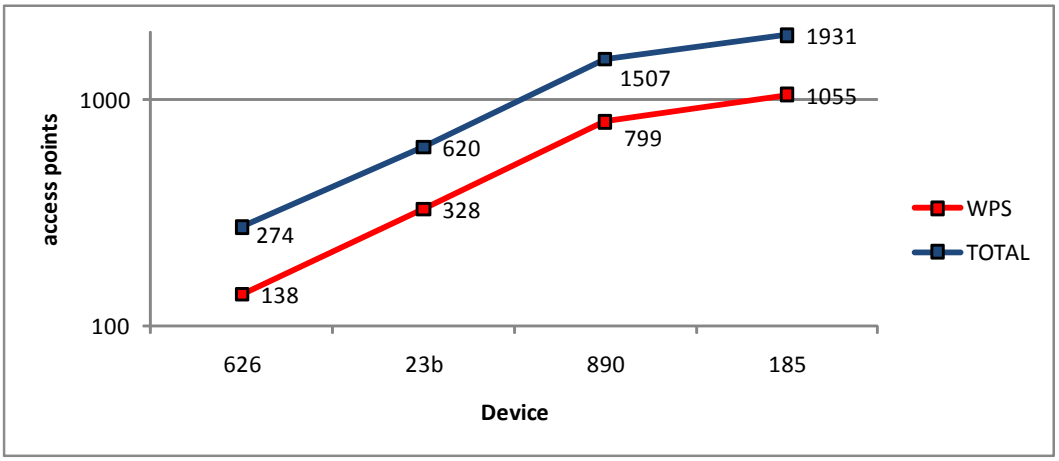



Figure 5.6: WPS enabled and total amount of devices

In summary, most access points are secured with a good authentication scheme. But this type of security highly depends on the used device model and firmware version (some vendors already introduced patches for limiting guessing attempts for *WPS*). If *WPS* is enabled and no blocking is done, the protection is authentication protocol is useless.

### 5.2.4 App usage

In this analysis, the apps installed by the user and their requested permissions are inspected. As in figure 5.7 seen, the most requested protection level is *dangerous* with

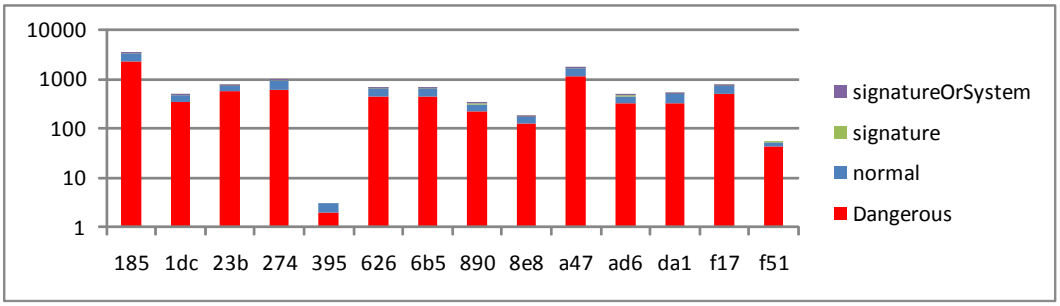



Figure 5.7: Overview of all protection levels used by any application on each smartphone. The most used protection level is clearly *dangerous*

nearly 70 percent in average. The *normal* level follows with 29.7 percent in average. The other types of protection level are rarely requested.
Figure 5.8 shows the most installed applications.

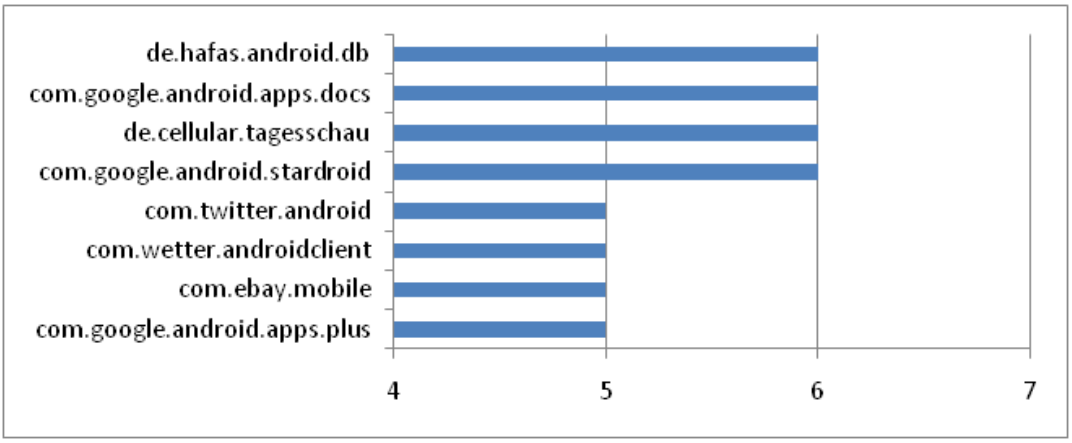



Figure 5.8: The most installed applications

# Chapter 6

# Conclusion

In the following a short summary about the reached goals is given and the experiences made during development are communicated. Additionaly an outlook to future work is given.

## 6.1 Summary

With this thesis the following tasks has been finished:

- An overview of the actual smartphone market
- The spread of the Android platform and its architecture
- The development of an Android usage study system
- Running a field test with voluntary participants to collect data
- Making some exemplary analysis on collected data

## 6.2 Experiences

At this point, the experiences made during development are described. This includes development experiences with the Android platform, the field test and the analyses afterwards.

- For developers, Google offers a rich set of free tools for the Android platform. One of them is the *Virtual Device Manager*, which allows to create a virtual device for testing applications. This makes the development and deployment easy, without the need of a real device.
- Coding for mobile platforms is different to other platforms (eg. workstations, server). The developer has to consider the limited capabilities of a smartphone such as storage space and CPU usage. As a result, the design of the application must consider this facts. During development of the *FHH Device Analyzer*, some mistakes are made. The most challenging parts are those with high CPU consumption. At the beginning, the app raises the CPU load very high, if

collection was in process. The implementation has to be changed, in order to run the app more smoothly.
- The analyzes of collected data was not as easy as imagined. Querying objects in the db4o database is very slow, if a high amount of features resist in database. The reason for this is how object oriented databases work. Every object that is considered internally by db4o during querying, an instance is created. If this object has additional references to other objects, they may be instantiated too. This leads to a high amount of memory usage and slows down the query.

## 6.3 Future work

- In order to fix some outstanding bugs and to fine tuning the performance, the client application should be reworked.
- There may be more measurement features available on newer Android versions. The app should be updated to collect these new features.
- The amount of participants can be increased in order to get more data for analyses.
- The application could be ported to other platforms such as *iOS* to make measurements on this devices available too.
- The webservice security should be enhanced. Authentication with credentials should be considered in order to make it harder to attack.

# Bibliography


[BP10] Becker, A. ; Pant, M.: *Android 2: Grundlagen und Programmierung*. dpunkt-Verlag, 2010. – ISBN 9783898646772

[Gar12] Gartner: *Gartner Says Worldwide Sales of Mobile Phones Declined 2.3 Percent in Second Quarter of 2012*. 2012. – http://www.gartner.com/it/page.jsp?id=2120015

[HKM10] Hashimi, S. ; Komatineni, S. ; MacLean, D.: *Pro Android 2*. Apress, 2010 (Apresspod Series Nr. 2). – ISBN 9781430226598

[Lab12a] Lab, Kaspersky: *IT Threat Evolution: Q2 2012*. 2012. – http://www.securelist.com/en/analysis/204792239/IT_Threat_Evolution_Q2_2012

[Lab12b] Labs, F-Secure: *Mobile Threat Report Q2 2012*. 2012. – http://www.f-secure.com/weblog/archives/MobileThreatReport_Q2_2012.pdf

[O212] O2: *All About You Report*. June 2012. – http://news.o2.co.uk/?press-release=Making-calls-has-become-fifth-most-frequent-use-for-a-Smartphone-for-newly-networked-generation-of-users

[sou] source.android.com: *Android Security Overview*. – http://source.android.com/tech/security/index.html

[TWP07] Tews, Erik ; Weinmann, Ralf-Philipp ; Pyshkin, Andrei: *Breaking 104 bit WEP in less than 60 seconds*, 2007. – http://eprint.iacr.org/2007/120.pdf

[Vie11] Viehböck, Stefan: *Brute forcing Wi-Fi Protected Setup*. December 2011. – http://sviehb.files.wordpress.com/2011/12/viehboeck_wps.pdf

[WC12] Wendy Connick, About.com G.: *7 Business Uses for a Smartphone*. August 2012. – http://sales.about.com/od/salesbasics/tp/7-Business-Uses-For-A-Smartphone.htm

[WSW+12] Westerkamp, Jürgen ; Starrach, Mark ; Winkelvos, Timo ; Heuser, Stephan ; Dunekacke, Dennis ; Scheuermann, Dirk ; Bente, Ingo ; Lucius, Jens ; Jahnke, Marcel: *Bericht AP4: Metadaten-Modell*, 2012

# List of Figures

# Listings

# Appendix A

# Android system permissions and protection levels

Table A.1: Permissions and their protection levels provided by the android system. API version 4.1 is used for generation of list

| **Permission** | **Protection level** |
|---|---|
| **android.permission** | |
| SEND_SMS | dangerous |
| CALL_PHONE | dangerous |
| RECEIVE_SMS | dangerous |
| RECEIVE_MMS | dangerous |
| READ_SMS | dangerous |
| WRITE_SMS | dangerous |
| RECEIVE_WAP_PUSH | dangerous |
| READ_CONTACTS | dangerous |
| WRITE_CONTACTS | dangerous |
| READ_CALENDAR | dangerous |
| WRITE_CALENDAR | dangerous |
| READ_USER_DICTIONARY | dangerous |
| WRITE_USER_DICTIONARY | normal |
| ACCESS_FINE_LOCATION | dangerous |
| ACCESS_COARSE_LOCATION | dangerous |
| ACCESS_MOCK_LOCATION | dangerous |
| ACCESS_LOCATION_EXTRA_COMMANDS | normal |
| INSTALL_LOCATION_PROVIDER | signatureOrSystem |
| INTERNET | dangerous |
| ACCESS_NETWORK_STATE | normal |
| ACCESS_WIFI_STATE | normal |

| | |
|---|---|
| ACCESS_WIMAX_STATE | normal |
| BLUETOOTH | dangerous |
| NFC | dangerous |
| USE_SIP | dangerous |
| ACCOUNT_MANAGER | signature |
| GET_ACCOUNTS | normal |
| AUTHENTICATE_ACCOUNTS | dangerous |
| USE_CREDENTIALS | dangerous |
| MANAGE_ACCOUNTS | dangerous |
| MODIFY_AUDIO_SETTINGS | dangerous |
| RECORD_AUDIO | dangerous |
| CAMERA | dangerous |
| VIBRATE | normal |
| FLASHLIGHT | normal |
| MANAGE_USB | signatureOrSystem |
| HARDWARE_TEST | signature |
| PROCESS_OUTGOING_CALLS | dangerous |
| MODIFY_PHONE_STATE | signatureOrSystem |
| READ_PHONE_STATE | dangerous |
| WRITE_EXTERNAL_STORAGE | dangerous |
| WRITE_SETTINGS | dangerous |
| WRITE_SECURE_SETTINGS | signatureOrSystem |
| WRITE_GSERVICES | signatureOrSystem |
| EXPAND_STATUS_BAR | normal |
| GET_TASKS | dangerous |
| REORDER_TASKS | dangerous |
| CHANGE_CONFIGURATION | dangerous |
| RESTART_PACKAGES | normal |
| KILL_BACKGROUND_PROCESSES | normal |
| FORCE_STOP_PACKAGES | signature |
| DUMP | signatureOrSystem |
| SYSTEM_ALERT_WINDOW | dangerous |
| SET_ANIMATION_SCALE | dangerous |
| PERSISTENT_ACTIVITY | dangerous |
| GET_PACKAGE_SIZE | normal |
| SET_PREFERRED_APPLICATIONS | signature |
| RECEIVE_BOOT_COMPLETED | normal |
| BROADCAST_STICKY | normal |
| WAKE_LOCK | dangerous |
| SET_WALLPAPER | normal |
| SET_WALLPAPER_HINTS | normal |

| | |
|---|---|
| SET_TIME | signatureOrSystem |
| SET_TIME_ZONE | dangerous |
| MOUNT_UNMOUNT_FILESYSTEMS | dangerous |
| MOUNT_FORMAT_FILESYSTEMS | dangerous |
| ASEC_ACCESS | signature |
| ASEC_CREATE | signature |
| ASEC_DESTROY | signature |
| ASEC_MOUNT_UNMOUNT | signature |
| ASEC_RENAME | signature |
| DISABLE_KEYGUARD | dangerous |
| READ_SYNC_SETTINGS | normal |
| WRITE_SYNC_SETTINGS | dangerous |
| READ_SYNC_STATS | normal |
| WRITE_APN_SETTINGS | dangerous |
| SUBSCRIBED_FEEDS_READ | normal |
| SUBSCRIBED_FEEDS_WRITE | dangerous |
| CHANGE_NETWORK_STATE | dangerous |
| CHANGE_WIFI_STATE | dangerous |
| CHANGE_WIMAX_STATE | dangerous |
| CHANGE_WIFI_MULTICAST_STATE | dangerous |
| BLUETOOTH_ADMIN | dangerous |
| CLEAR_APP_CACHE | dangerous |
| READ_LOGS | dangerous |
| SET_DEBUG_APP | dangerous |
| SET_PROCESS_LIMIT | dangerous |
| SET_ALWAYS_FINISH | dangerous |
| SIGNAL_PERSISTENT_PROCESSES | dangerous |
| DIAGNOSTIC | signature |
| STATUS_BAR | signatureOrSystem |
| STATUS_BAR_SERVICE | signature |
| FORCE_BACK | signature |
| UPDATE_DEVICE_STATS | signatureOrSystem |
| INTERNAL_SYSTEM_WINDOW | signature |
| MANAGE_APP_TOKENS | signature |
| INJECT_EVENTS | signature |
| SET_ACTIVITY_WATCHER | signature |
| SHUTDOWN | signature |
| STOP_APP_SWITCHES | signatureOrSystem |
| READ_INPUT_STATE | signature |
| BIND_INPUT_METHOD | signature |
| BIND_WALLPAPER | signatureOrSystem |

| | |
|---|---|
| BIND_DEVICE_ADMIN | signature |
| SET_ORIENTATION | signature |
| INSTALL_PACKAGES | signatureOrSystem |
| CLEAR_APP_USER_DATA | signature |
| DELETE_CACHE_FILES | signatureOrSystem |
| DELETE_PACKAGES | signatureOrSystem |
| MOVE_PACKAGE | signatureOrSystem |
| CHANGE_COMPONENT_ENABLED_STATE | signatureOrSystem |
| ACCESS_SURFACE_FLINGER | signature |
| READ_FRAME_BUFFER | signature |
| BRICK | signature |
| REBOOT | signatureOrSystem |
| DEVICE_POWER | signature |
| FACTORY_TEST | signature |
| BROADCAST_PACKAGE_REMOVED | signature |
| BROADCAST_SMS | signature |
| BROADCAST_WAP_PUSH | signature |
| MASTER_CLEAR | signatureOrSystem |
| CALL_PRIVILEGED | signatureOrSystem |
| PERFORM_CDMA_PROVISIONING | signatureOrSystem |
| CONTROL_LOCATION_UPDATES | signatureOrSystem |
| ACCESS_CHECKIN_PROPERTIES | signatureOrSystem |
| PACKAGE_USAGE_STATS | signature |
| BATTERY_STATS | normal |
| BACKUP | signatureOrSystem |
| BIND_APPWIDGET | signatureOrSystem |
| CHANGE_BACKGROUND_DATA_SETTING | signature |
| GLOBAL_SEARCH | signatureOrSystem |
| GLOBAL_SEARCH_CONTROL | signature |
| SET_WALLPAPER_COMPONENT | signatureOrSystem |
| ACCESS_CACHE_FILESYSTEM | signatureOrSystem |
| COPY_PROTECTED_DATA | signature |
| **android.intent.category** | |
| MASTER_CLEAR.permission.C2D_MESSAGE | signature |
| **com.android.browser.permission** | |
| READ_HISTORY_BOOKMARKS | dangerous |
| WRITE_HISTORY_BOOKMARKS | dangerous |
| SET_ALARM | normal |

# Appendix B

# Webservice for an Android usage study system

The following is the listing of the webservice.

```
1  /**
2   * File: FeatureWebService.java
3   *
4   * Copyright (C) 2012 Hochschule Hannover
5   *
6   * Ricklinger Stadtweg 118, 30459 Hannover, Germany
7   *
8   * Licensed under the Apache License, Version 2.0 (the
        "License"); you may not
9   * use this file except in compliance with the License. You
        may obtain a copy of
10  * the License at
11  *
12  * http://www.apache.org/licenses/LICENSE-2.0
13  *
14  * Unless required by applicable law or agreed to in
        writing, software
15  * distributed under the License is distributed on an "AS
        IS" BASIS, WITHOUT
16  * WARRANTIES OR CONDITIONS OF ANY KIND, either express or
        implied. See the
17  * License for the specific language governing permissions
        and limitations under
18  * the License.
19  *
20  */
21 package de.fhh.inform.trust.aus.server;
22
23 import com.db4o.Db4oEmbedded;
24 import com.db4o.EmbeddedObjectContainer;
25 import com.db4o.config.AndroidSupport;
26 import com.db4o.config.EmbeddedConfiguration;
27 import com.db4o.config.UuidSupport;
28 import com.db4o.consistency.ConsistencyChecker;
```

```
29 import com.db4o.consistency.ConsistencyReport;
30 import com.db4o.ext.Db4oException;
31 import java.io.*;
32 import java.text.DateFormat;
33 import java.text.SimpleDateFormat;
34 import java.util.Date;
35 import javax.ws.rs.Consumes;
36 import javax.ws.rs.POST;
37 import javax.ws.rs.Path;
38 import javax.ws.rs.Produces;
39 import javax.ws.rs.core.Context;
40 import javax.ws.rs.core.HttpHeaders;
41 import javax.ws.rs.core.MediaType;
42 import javax.ws.rs.core.MultivaluedMap;
43
44 @Path("generic")
45 public class FeatureWebService {
46
47     @Context
48     private javax.ws.rs.core.HttpHeaders context;
49     private DateFormat dateFormat = new
           SimpleDateFormat("yyyyMMdd_HHmmss");
50     private String[] nonValidCharcters = new String[]{".",
           "'", "_", "/", "\\"};
51     private String uploadDir =
           System.getProperty("com.sun.aas.instanceRoot") +
           "/uploads";
52     private String imei = null;
53
54     public FeatureWebService() {
55     }
56
57     @POST
58     @Consumes("binary/octet-stream")
59     @Produces(MediaType.TEXT_HTML)
60     public String postData(byte[] data) {
61         if (data != null && data.length > 0) {
62             MultivaluedMap<String, String> map =
                   context.getRequestHeaders();
63             imei = map.getFirst("imei");
64             if (imei == null) {
65                 imei = "unknown";
66             } else {
67                 // validate
68                 if (imei.trim().length() != 64) {
69                     return "Something goes wrong.";
70                 }
71
72                 for (String no : nonValidCharcters) {
73                     if (imei.contains(no)) {
74                         return "Something goes wrong.";
75                     }
76                 }
77             }
```

```
78              try {
79                  new File(uploadDir).mkdirs();
80                  String filename = uploadDir + '/' + imei +
                        "_" + dateFormat.format(new Date()) +
                        ".db4o";
81                  FileOutputStream out = new
                        FileOutputStream(filename);
82                  out.write(data);
83                  out.close();
84                  if (checkDb4o(filename, context)) {
85                      return "Upload successful!";
86                  }
87              } catch (FileNotFoundException ex) {
88              } catch (IOException ex) {
89              }
90          }
91          return "Something goes wrong.";
92      }
93
94      private boolean checkDb4o(String filename, HttpHeaders
            context) {
95          EmbeddedObjectContainer mDb4o = null;
96          try {
97
98              EmbeddedConfiguration conf =
                    Db4oEmbedded.newConfiguration();
99              conf.common().add(new UuidSupport());
100             conf.common().add(new AndroidSupport());
101             mDb4o = Db4oEmbedded.openFile(conf, filename);
102             ConsistencyChecker checker = new
                    ConsistencyChecker(mDb4o);
103             ConsistencyReport report =
                    checker.checkSlotConsistency();
104             if (!report.consistent()) {
105                 throw new Db4oException();
106             }
107             mDb4o.close();
108             return true;
109         } catch (Db4oException err) {
110             if (mDb4o != null) {
111                 mDb4o.close();
112             }
113             final Writer result = new StringWriter();
114             final PrintWriter printWriter = new
                    PrintWriter(result);
115             err.printStackTrace(printWriter);
116             writeLog("db4o_err.log",
                    result.toString().getBytes());
117             new File(filename).renameTo(new File(filename +
                    ".corrupt"));
118         }
119         return false;
120     }
```

```
121 |
122 |     private void writeLog(String path, byte[] data) {
123 |         FileOutputStream fp;
124 |         try {
125 |             fp = new FileOutputStream(path);
126 |             fp.write(data);
127 |             fp.close();
128 |         } catch (Exception ex) {
129 |         }
130 |     }
131 | }
```

Listing 1: Webservice accepting POST-Data and stores files in system after validating