Indoor Information Infrastructure: MAGUN

Indoor Positioning using NFC Tags

Indoor-Positionsbestimmung anhand von NFC-Tags

Bachelor Thesis

Bachelorarbeit

Johannes Bolz
Matriculation Number | Matrikelnummer: 754314
Email: johannes-bolz@gmx.net

Supervisor | Betreuer: Prof. Dr. Roland M. Wagner
Department III | Fachbereich III
Beuth Hochschule für Technik, Berlin, Germany
Course of Studies | Studiengang: Geoinformation

Berlin, July 11, 2011

Abstract

There is currently no standard technology for indoor positioning, which virtually limits location-based services to the outdoors. However, there are several attempts being investigated. This thesis evaluates the method of object based location referencing. Object based referencing determines the location from the proximity to an object that is recognized by the navigation device. In particular, NFC tags are evaluated for their use as such referencing objects.

This thesis focuses on the development of a demonstrator application for the Android OS and the necessary data models.

Two approaches are used: Using explicit references which are written to the tags and can be directly evaluated when read. The other method is recognizing a unique feature on the object, like the unique identifier of NFC tags, and an external reference model that links the unique feature to a location.

While the explicit referencing method doesn't require to maintain a separate reference store, the reference model approach allows for centralized data management.

While using NFC tags for positioning has some drawbacks in usability compared to common outdoor localization methods, it is a solution that is easy and cheap to install in buildings.

Table of Contents

Terminology

This is an overview of specific terminology and abbreviations used throughout this document.

Term:	**Definition:**
GNSS	Global Navigation Satellite System.
GPS	In this document specifically used for the Navstar Global Positioning System.
JVM	Java Virtual Machine.
NDEF	NFC Data Exchange Format.
NFC	Near Field Communication.
OS	Operating system.
QR code	Quick Response Code.
Reference Model	Data set that maps unique values to (data or location) references.
Tag	Collective term used for NFC transponders of any type as well as QR codes.
UID	Unique identifier.
VGI	Volunteered Geographic Information.
WLAN	Wireless Local Area Network.

1 Introduction

1.1 Motivation

During the past 20 years, personal navigation has seen huge growth and a rapidly progressing development. From what was originally intended as an exclusively military appliance during it's development – GPS – has since become a widespread mass-market technology, available to the average consumer[1].

In the early 1990s, GPS receiver units could merely display the current location as a coordinate pair, with the more advanced ones being able to plot a recorded track or a minimalistic map on the monochrome display.

The introduction of GPS-based car navigation systems on a broad scale ultimately made GPS available to the masses. The devices are not only capable of displaying a map, but also of route calculation based on digital street data[2].

Since the Apple iPhone was introduced in 2007, smartphones have become commonplace, at least in industrial nations[3]. Most of them not only include a GPS receiver module, but also 3G internet connectivity and a

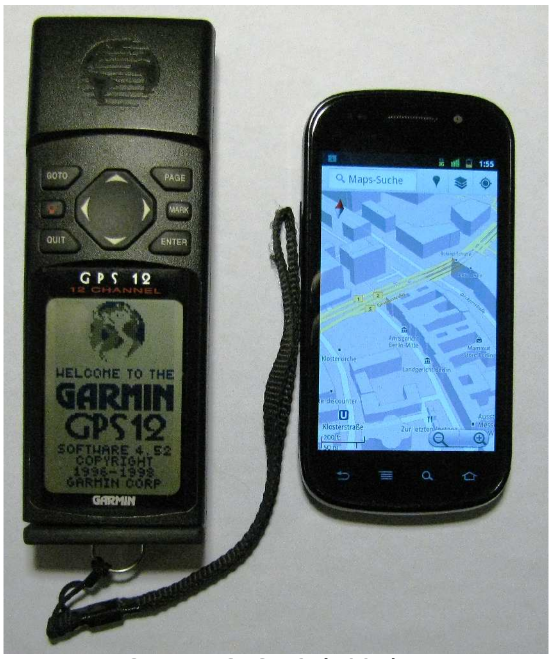


Figure 1: Garmin GPS 12 (1997), Samsung/Google Nexus S (2010)

number of sensors (like compass, accelerometers, gyroscopes). With the

1 http://www.time.com/time/magazine/article/0,9171,1901500,00.html [retrieved 2011-07-08]

2 http://hst250.history.msu.edu/wiki-online/index.php/The_History_of_the_Automotive_Navigation_System [retrieved 2011-07-08]

3 http://www.heise.de/newsticker/meldung/Smartphone-Verkaufszahlen-klettern-um-mehr-als-40-Prozent-1044272.html [retrieved 2011-07-08]

growing abundance of smartphones, a multitude of location-based service (LBS) applications have been deployed, making use of the positioning capability and internet connectivity[4].

Also, smartphones allow navigation "on a smaller scale", like pedestrian navigation, which requires more accurate localization capabilities (e.g. to determine the correct side of the road) and also finer-grained street data.

However, all of these services are virtually limited to the outdoors, particularly due to the lack of proper indoor positioning methods. There is usually no GPS reception indoors, and other localization technologies like WLAN fingerprinting simply lack accuracy for indoor usage.

With the indoors being mostly a white spot on digital maps, there is a demand for indoor location based services and thus indoor localization[5]. Some go as far as calling indoor localization "the holy grail for navigation technology vendors"[6].

There is a number of approaches concerning indoor localization being trialed, but at the time of writing, there is no standard technology or even a widely available one. Thus, this thesis addresses this lack of localization mechanism by evaluating the use of real-world objects for indoor positioning. More specifically, near field communication (NFC) transponders serve as (machine-readable and recognizable) objects in this piece of work. The NFC technology is currently being implemented in newly introduced devices, and thus might become a standard smartphone technology in the near future.

All current approaches to indoor positioning require a technological infrastructure that must be implemented in the respective building. NFC may turn out to be a cost-effective solution that is easy to implement.

Other than the location itself, information about the location may be referenced. This is a field of research addressed by the MAGUN project[7], and also partly in the framework of this thesis (see chapter 2.1: Integration with the MAGUN Project, p.12f.).

1.2 Scope

The goal of this work is to examine the feasibility of using NFC technology for linking location and information associated with it. The core part of this thesis covers the development of a demonstrator application running on the Android OS. The application gets location information by reading NFC tags and may also geo-reference new tags. For demonstration purposes, a exemplary set of tags is installed in Beuth University's Bauwesen building.

4 http://www.nytimes.com/2010/11/15/technology/15iht-navigate.html [retrieved 2011-07-08]

5 http://www.directionsmag.com/articles/indoor-positioning-and-navigation-the-next-lbs-frontier/129925 [retrieved 2011-07-08]

6 http://www.vicchi.org/2010/11/23/a-first-step-towards-indoor-navigation-literally/ [retrieved 2011-07-08]

7 http://magun.beuth-hochschule.de [retrieved 2011-07-08]

The main objectives of the demonstrator are:

- Demonstrating possible solutions as how to link location and location information using objects.
- Providing a convenient way to read and write NFC transponders (at the time of writing the technology is still new to the Android platform, as it was introduced in late 2010[8]).

1.3 Current State of Research

As mentioned above, there are several indoor positioning solutions being studied. This chapter provides a short overview of the most important technologies:

- **Tag-based solutions:** Apart from this project, there are other projects using tags (objects) to determine location, like RFID transponders, QR codes, or infrared beacons[9]. There are generally two ways to implement tag-based localization systems: Active systems (the user reads passive tags with a device which computes the location) and passive ones (the user carries a passive tag, which is read by fixed readers). The active systems, irrespective of the technology they actually use, follow the same principle: They read information from a tag and use it to determine the location. For example, there is a paper from the ISIK University of Istanbul that also proposes the use of NFC tags to link location information[10]. What sets this thesis' work apart from the aforementioned methods is the fact that other than merely providing location data, it is also intended to integrate with an indoor information infrastructure that is being developed in the MAGUN project. Also, internet-independence is a key consideration within MAGUN and this piece of work.

8 http://www.nearfieldcommunicationsworld.com/2010/12/07/35385/google-unveils-first-android-nfc-phone-but-nexus-s-is-limited-to-tag-reading-only-for-now/ [retrieved 2011-07-08]

9 http://www.geomatik-hamburg.de/forum-geomatik/2010/pdf/4-Indoor-Navigation_Hoenniger.pdf, slide 14 [retrieved 2011-07-08]

10 http://nfclab.isikun.edu.tr/papers/NFCInternal.pdf [retrieved 2011-07-08]

- **Inertial and sensor-based localization:** Another possibility is to

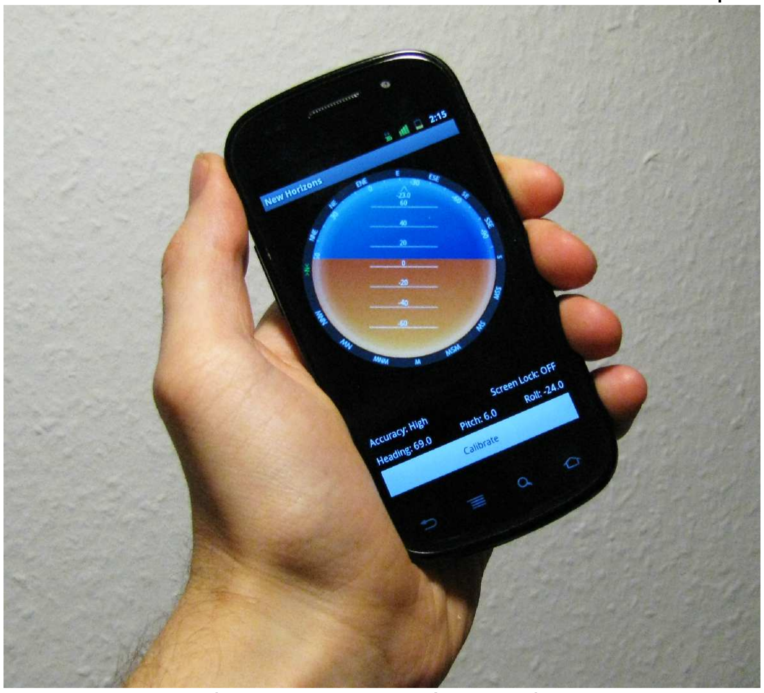


Figure 2: Smartphone measuring it's spatial orientation using gyroscope, accelerometer and compass.

  approximate the position using sensor data. A modern smartphone is typically equipped with acceleration and rotation sensors (gyroscopes). The data delivered by these sensors can be used to calculate an overall motion vector[11]. That however requires a very high level of sensor accuracy. Another possibility is to check the acceleration graph for characteristic patterns produced by e.g. walking (step counter). Other sensors are being implemented in phones, like barometers, so changes in height may be detected.

- **WLAN fingerprinting:** This real-time tracking method evaluates signals of WLAN networks that a device can pick up at a given time. The signal strengths of all available networks are matched against a data set of mapped signal strengths and the location is calculated[12]. This method requires the abundance of several distinguishable WLAN networks and calibration (mapping). This method is already being applied by several providers active in the mobile business, even though mainly for outdoor positioning[13].

- **GPS / GNSS extended indoors:** There are attempts to extend GPS coverage indoors. This is achieved by using more sensitive antennas and filtering algorithms, as well as the complementary use of other positioning methods[14]. This method requires highly-sensitive GPS antennas currently not implemented in smartphones.

11 http://www.geospatialworld.net/index.php?option=com_content&view=article&id=20580&Itemid=1925 [retrieved 2011-07-08]

12 http://www.mobile.ifi.uni-muenchen.de/studium_lehre/verg_semester/ws1011/msp/08_wlan_positioning.pdf [retrieved 2011-07-08]

13 http://www.indoorlbs.com/p/wifi-location-services-for-lbs_18.html [retrieved 2011-07-08]

14 http://forschung.unibw-muenchen.de/papers/4lut9jfxv9hd0ig1paaiqprktdwqko.pdf [retrieved 2011-07-08]

- **Pseudolite networks:** The term “pseudolite” is an abbreviation of “pseudo satellite” and refers to the usage of GNSS-like methods, but with fixed beacons sending the signals instead of satellites[15]. Just like GPS, this method calculates the distances of several signal sources using the respective transmitting time and subsequently deducts a position from those distances.

15 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.3605&rep=rep1&type=pdf [retrieved 2011-07-08]

2 Methods and General Considerations

2.1 Integration with the MAGUN Project

MAGUN is a research project investigating the feasibility and possible implementations of an indoor information infrastructure (iii)[16]. The goal is to develop open standards for location-dependent, context driven and internet-independent data delivery infrastructures. MAGUN is a collaboration between the Beuth University and several other partners[17]. The project has an overall duration of 18 months, ending in March, 2012. The author is employed as a working student within the project alongside his studies.

A key consideration is the use of established technologies (like WLAN, NFC) and existing protocols (FTP, HTTP) for building these infrastructures. A demonstrator for the usage of such an infrastructure is being developed: Information about a location is stored as an "electronic flyer" (eFlyer) on small local servers (modified WLAN access points) and may be downloaded, created, modified and uploaded by an application running on Android smartphones. The conceptual work consists of developing the technical infrastructure, data models for the content and protocols for data exchange.

Since the infrastructure model doesn't rely on a centralized (internet-based) architecture, an interface that makes client applications aware of the abundance of location- (or context-) specific data becomes necessary. That may either be done by recognizing the location itself, or by recognizing machine-readable objects (tags) that can be related to the specific data content.

The aforementioned requirement to link data and the fact that indoor localization and navigation may be beneficial for such an indoor information infrastructure marks the link between this thesis' field of work and the MAGUN project.

2.2 License

All software developed in the course of this thesis is licensed as open source under the Apache License, Version 2.0[18].

2.3 Technologies

This chapter provides an overview of the key technologies being used within this project.

16 http://magun.beuth-hochschule.de/#about [retrieved 2011-07-08]
17 http://www.magun-projekt.de/doku.php?id=public:partner [retrieved 2011-07-08]
18 http://www.apache.org/licenses/LICENSE-2.0.html [retrieved 2011-07-08]

2.3.1 Android OS

At the time of writing, Android is the smartphone operating system with the highest relative sales numbers[19].

There are a number of factors that make it suitable for this project: Android applications are typically developed in the Java language. Applications run within the Dalvik VM, a technology similar to the JVM. There is a high degree of compatibility with the JVM, which means that many libraries originally developed for the JVM can also be used for Android applications. That makes the platform popular with developers, which in turn results in a high availability of documentation and shared knowledge amongst developers. There is a free-of-charge developer tool set for the Android platform that may be integrated with the (equally free) Eclipse IDE.

Also, the high abundance of Android devices means that the technology being developed can be directly applied by a high number of users, which increases user feedback.

2.3.2 Near Field Communication (NFC)

Near field communication is a wireless technology based on RFID (Radio Frequency Identification), and is intended to transmit low amounts of data over short distances. Standards are being defined by the NFC Forum, a consortium of various companies involved in the development of the technology[20].

Data exchange using NFC typically involves an active and a passive part: The active reader/writer emits an electromagnetic field, which is picked up

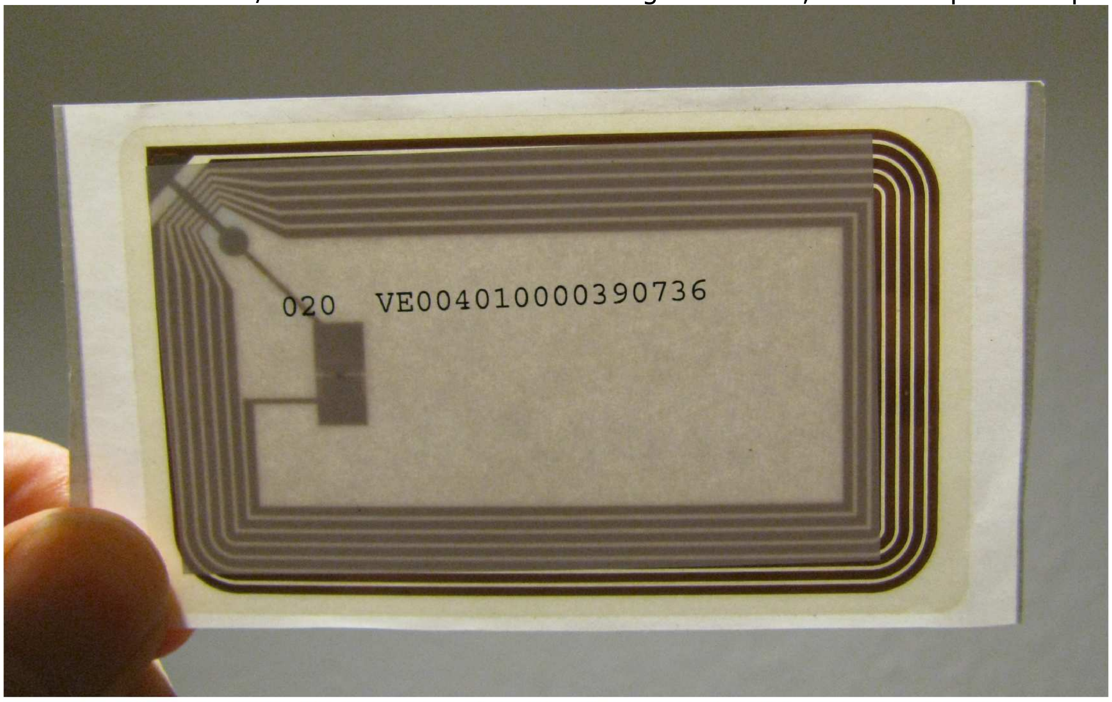


Figure 3: NFC tag (with it's UID printed on it). The antenna coil is connected to the microprocessor in the center of the transponder.

19 http://www.h-online.com/open/news/item/Smartphones-Android-overtakes-Symbian-Apple-loses-market-share-1181349.html [retrieved 2011-07-08]

20 http://www.nfc-forum.org/home/ [retrieved 2011-07-08]

by a coil antenna within the passive counterpart ("tag" or "transponder"). Thus, an electric current is induced, which powers a micro-processor on the passive transponder. The transponder modulates a (data) signal on the refracted electromagnetic field, which is in turn picked up by the reader[21].

There are reader devices that can emulate a passive tag. Also, communication between two active parts is possible. However, NFC always takes place as a one-to-one connection[22].

NFC uses a frequency of 13.56 MHz, which is legally available world-wide. There is a variety of RFID transponders that can be used for NFC[23], most notably those implementing the ISO 14443[24] standard.

The maximum distance over which data can be transmitted is intended to be no more than 5cm. However, some transponder technologies that are NFC-compliant are capable of transmitting over longer distances[25].

Transponder data capacities range from 48 bytes up to 9 kilobytes[26]. Transfer rates are per definition up to 424 kilobits per second[27]. Transponders can be either rewritable, read-only or write-once types.

Most NFC transponders feature a unique identifier (UID).

21 Finkenzeller (2003): RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification. – p.41ff.

22 http://www.radio-electronics.com/info/wireless/nfc/near-field-communications-tutorial.php [retrieved 2011-07-08]

23 http://www.radio-electronics.com/info/wireless/nfc/near-field-communications-tags-types.php [retrieved 2011-07-08]

24 http://www.openpcd.org/ISO14443 [retrieved 2011-07-08]

25 http://www.itwissen.info/definition/lexikon/ISO-15693-ISO-15693.html [retrieved 2011-07-08]

26 http://www.nxp.com/acrobat_download2/other/identification/173110_NFC_Forum_Type_Tags_WhitePaper.pdf, page 21 [retrieved 2011-07-08]

27 http://www.radio-electronics.com/info/wireless/nfc/near-field-communications-tutorial.php [retrieved 2011-07-08]

As of today, NFC is mostly used for payment systems, access control or

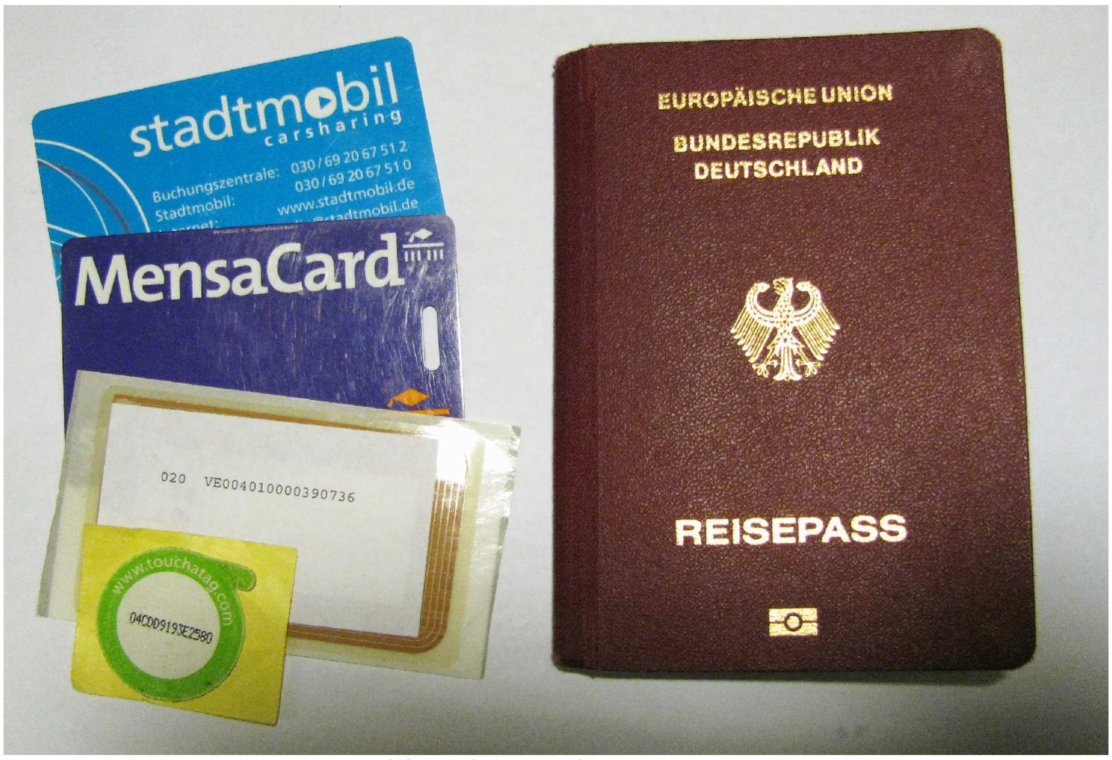


Figure 4: Various use cases and form factors of NFC tags.

personal identification.

NFC uses the NDEF data format[28]: Data is exchanged via NDEF messages (a transponder may contain one or more of them). A NDEF message consists of one or more NDEF records. A NDEF record contains the payload data itself as byte array, as well as data fields describing the data: The "type name format" (TNF) field contains an identifier value for the payload

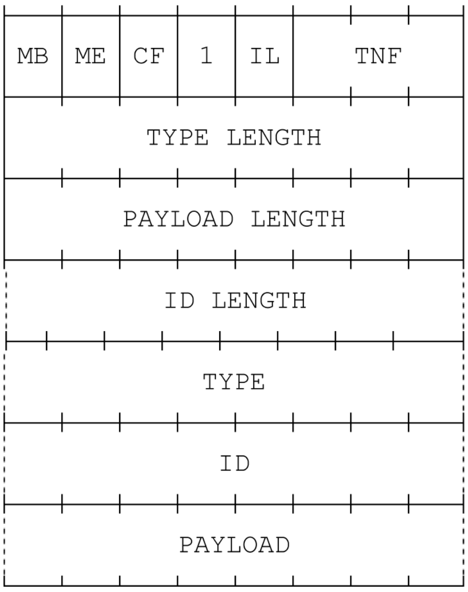


Figure 5: NDEF Record data fields (source: NFC Forum)

28 http://www.nfc-forum.org/specs/spec_license, "NFC Data Exchange Format (NDEF) Technical Specification" [retrieved 2011-07-08]

format description's syntax and the “type” field holds the payload format information itself. There is also an optional ID field and a field specifying the payload data size.

At the time of writing of this thesis, there are few smartphones offering NFC capability, with the Google/Samsung Nexus S being the only Android device supporting the technology.

Android implements NFC capability since it's version 2.3 (API level 9), with a major rework in version 2.3.3 (API level 10)[29].

As stated earlier, in this project, NFC transponders serve as machine-recognizable real-world objects, whose discovery by a device is used to retrieve linked data or a location.

2.3.3 QR Codes

Figure 6: QR Code.

QR codes (two-dimensional bar codes[30]) are used in this project alongside NFC, mainly due to the low abundance of NFC-capable Android handsets. This way, a broader audience can use the developed application (and thus, give feedback). Most smartphones are equipped with high resolution cameras and QR reader applications are freely available.

2.4 Development

This chapter provides an overview of the technologies being applied in the development process.

2.4.1 Software

The following tools are used to develop the application:

29 http://android-developers.blogspot.com/2011/02/android-233-platform-new-nfc.html [retrieved 2011-07-08]

30 http://code.google.com/p/zxing/wiki/BarcodeContents [retrieved 2011-07-08]

- **Eclipse IDE:** Development environment used for Java development.
- **Android SDK:** Toolset for Android development and debugging.
- **Altova XMLSpy:** XML toolset, used to develop XML data schema.
- **JOSM:** Tool for OpenStreetMap data digitalization
- **zxing online QR Generator:** QR code generator.

Android tools being used for development:

- **NXP TagWriter:** Standard application delivered with the Nexus S device, used for writing NFC tags.
- **NFC TagInfo:** NFC reading tool providing extensive information about the tag being read
- **NFC MIME:** NFC tool developed by the author for reading and writing various data formats (most notably MIME type defined data from files).

2.4.2 Hardware

The application is tested on the following devices:

- **Google / Samsung Nexus S:** NFC capable, running Android 2.3.3 and 2.3.4
- **HTC Legend:** not NFC capable, running Android 2.2
- **Google / HTC Nexus One:** not NFC capable, running Android 2.3.4

The following NFC tag types are used:

- NXP Mifare Ultralight (ISO 14443, 64B, write-only)
- NXP Mifare Classic 1k (ISO 14443, 1kB)
- NXP Mifare Classic 4k (ISO 14443, 4kB)
- NXP Mifare Desfire EV1 8k (ISO 14443-4, 8kB)
- NXP I-Code (ISO 15693, 112B)

3 Implementation

This chapter discusses the implementation of the demonstrator application.

Note: This document reflects the state of development at the time of writing, which corresponds to the application version 0.4. This version can be found in Appendix III: Demonstrator Tags and Project Data. However, development will continue after the writing of this thesis. Thus, a newer application/code version may be available by the time of reading. The latest stable version of the application is available in the Android Market[31].

3.1 Functionality Specification

This is a summary of the application's capabilities.

3.1.1 Map Display

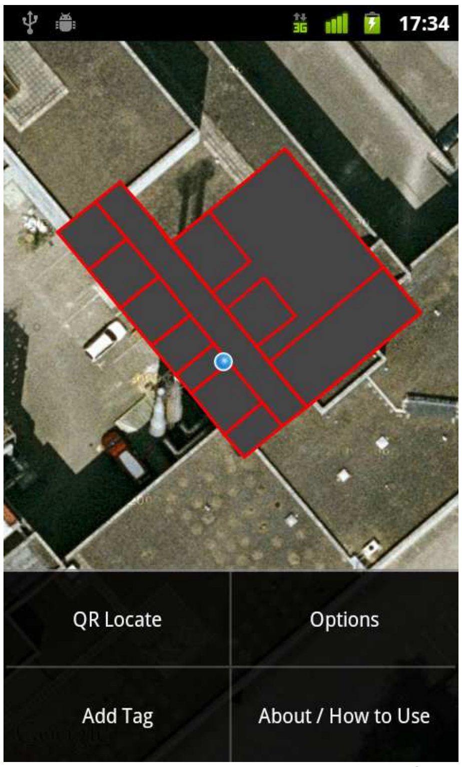


Figure 7: TagLocate main view: The map.

The application displays a map showing

- the current location
- simple indoor room geometries
- a satellite/aerial base map

31 https://market.android.com/details?id=de.bolz.android.taglocate [retrieved 2011-07-08]

This is the initial view of the application. If a new location is obtained by reading a NFC tag / QR code, the application draws a marker and centers the map at the new location. It is possible to pan the map and to freely zoom in and out.

There is an edit mode which shows an overlay symbol that points to the

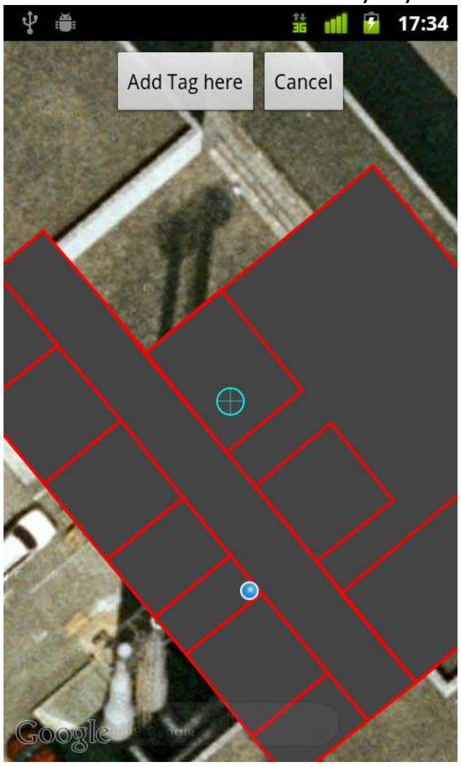


Figure 8: TagLocate in edit mode.

location a new tag is to be referenced. The user may pan the map to make the marker point to the desired location.

3.1.2 NFC Location Resolution

A location can be obtained by reading NFC tags. The location is deducted by using

- an explicit link written as payload on the tag, defining the location, or
- the UID, using a reference model which associates the UID with a location

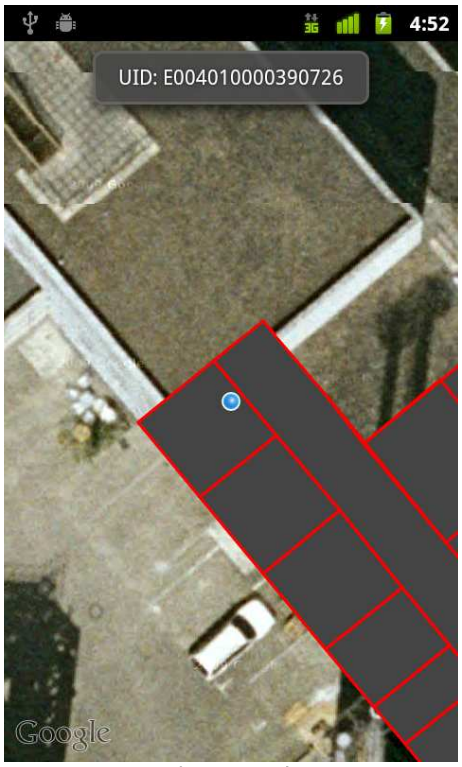


Figure 9: Notification about successful UID resolution.

NFC tags can be read by holding the device within reading range of a NFC tag while the application is running. The application displays a notification when a new location is detected.

3.1.3 QR Location Resolution

Alternatively, a QR code can be read. A location is deducted by

- evaluating the content for an explicit link defining the location, or
- treating the content as an ID, which is matched against a reference model

QR codes can be read by selecting the according menu item, which opens a separate view that allows aiming the device's camera at the QR code. When a QR code is read, the application switches back to the map view.

3.1.4 Location Provider

The Android OS provides a concept for handling different sources of location (e.g. GPS, GSM cell ID, WiFi fingerprint) by providing so-called location providers for each of these sources. Applications may implement and register listeners to these location providers in order to receive location updates[32].

Android also provides a way to create new location providers[33].

32 Ableson et al (2009): Unlocking Android – A Developer's Guide. – p. 275ff.
33 http://pedroassuncao.com/2009/11/android-location-provider-mock/ [retrieved 2011-07-08]

This application leverages these concepts by creating a location provider that is updated every time a location tag is detected. This way, the location information is not only accessible to the application itself for visualization purposes, but integrated with the operating system on a deeper level. Thus, other applications may use the location information that is generated.

3.1.5 Resolution Strategy Selection

It is possible to switch between localization strategies (see chapter 3.3: Referencing Concepts, p.23):

- ID only
- Link only
- ID first
- Link first

Switching is possible by selecting the strategy in the application's options menu.

3.1.6 NFC File Handling

It is possible to read files from NFC tags. The files may contain

- geometries, and / or
- reference information

The file reading capability may be toggled on/off in the options menu. If activated, the application reads geometry and reference files as soon as a NFC tag containing files (see chapter 3.8.11: NFC File Storage, p.33) is detected.

3.1.7 Data Storage

The application allows to store indoor room geometries and reference models locally on the device. The user may choose between available data sets.

3.1.8 Editing

The application includes an edit functionality that allows

- Writing a location link to an NFC tag
- Referencing a NFC tag's UID with a location
- Referencing a QR tag's content with a location

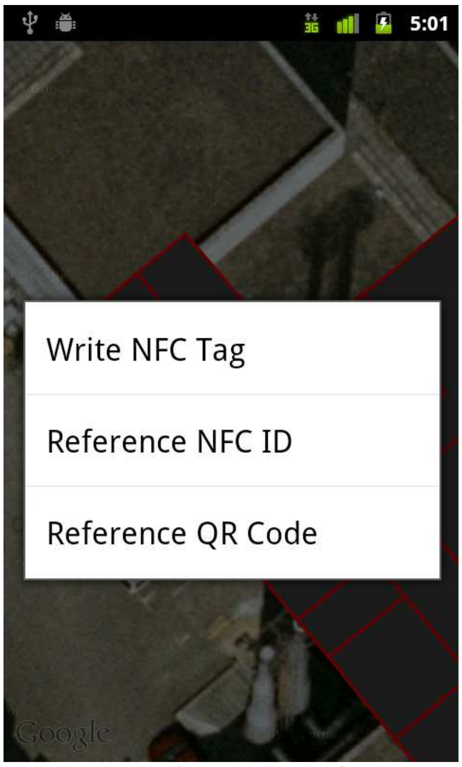


Figure 10: Referencing mode selection menu.

When starting the edit function, the user first marks the location to be referenced on the map, then is prompted whether to write to NFC or to reference a new tag.

3.2 Protocols and URI Formats

TagLocate uses two URI formats for location references: The Geo-URI format[34] specified in RFC 5870[35] and the URI format defined for the iii protocol, that is still being developed in the framework of the MAGUN project.

3.2.1 Geo URI

The Geo URI allows to define locations in an URI format. The following values can be defined within a Geo URI according to it's specification:

- the scheme definition geo
- latitude
- longitude
- altitude (optional)
- projection information (optional, WGS84 assumed as default)
- position accuracy (optional)

The syntax is per definition:

34 http://geouri.org/ [retrieved 2011-07-08]
35 http://tools.ietf.org/rfc/rfc5870 [retrieved 2011-07-08]

```
geo:<lat>,<lon>[,<alt>][;u=<acc>][;crs=<crs>]
```

A simple example:

```
geo:51.96367589,7.61302172
```

The geo URI format is being used within the reference model, as well as well as for writing explicitly linked locations on NFC tags.

3.2.2 iii URI

The iii protocol is still under development within the MAGUN project at the time of writing. It is intended to encapsulate link information and carrier technology information within one super-protocol.

The motivation for developing such a protocol is, given the internet-independent nature of the iii concept, it cannot be assumed that the user already has access to the given carrier network in which the linked data is accessible. For example, a linked file may be accessible on a server that is connected to a specific local WiFi network. In this case, the iii URI would provide not only the protocol (e.g. FTP) and the path under which the file is accessible, but also the credentials of the WiFi network to connect to first.

TagLocate also recognizes iii URI's. Since the protocol is still under active development, the application uses a preliminary version. The syntax of this preliminary version is:

```
iii://<target-link>,<carrier-credentials>,<trigger>
```

An example:

```
iii://geo:51.96367589\,7.61302172,,
```

The comma within the Geo URI is escaped, since this is also a functional character of the iii URI syntax. At this stage, the application resolves linked location, so an iii link works with a Geo URI as target link and empty carrier credentials (since coordinates can be resolved without a carrier). The trigger element is also unused in this application (it has been extracted, see chapter 3.6: Reference Data Model, p.25 for more information).

At a later stage, the final specification of the iii protocol may be implemented in the application. This will facilitate integration with other indoor information applications developed in the framework of MAGUN.

3.3 Referencing Concepts

There are two fundamental ways of associating locations with objects, which are discussed in this chapter.

3.3.1 ID Reference Matching

Using this method, a unique feature of the object (NFC UID or unique QR code content) is read and matched against a reference model (see chapter 3.6: Reference Data Model, p.25).

The reference model maps unique features to links (Geo URI's and thus locations in the case of this application). If there is a match within the reference model, the application loads the referenced link/location.

Example: NFC tag is read. → UID is retrieved. → UID is looked up in the reference model. → There is a matching UID in the reference model, referencing a Geo URI. → Coordinates are extracted from the URI. → Location Provider and Map are updated.

3.3.2 Explicit Link

This method uses information directly stored on the object (data payload). If the data is compliant to the supported link formats, the link (Geo URI / iii URI) is loaded.

Example: NFC tag is read → NDEF payload is checked for Geo URI's and iii URI's → Tag contains a Geo URI → Coordinates are extracted from the URI → Location Provider and Map are updated.

3.3.3 Combinations

The application supports combinations of the two aforementioned methods given an order of priority: One method is executed first. If it has a successful result, the result is applied. If there is no result, the other method is executed.

3.4 NFC Tag Technologies

There are several types of NFC tags. The application reads tags implementing the ISO 14443 standard, as well as ISO 15693 type tags. At the time of writing, NFC doesn't include ISO 15693 tags. However, the standard is planned to be adopted by the NFC forum[36]. Since tests showed it is possible to write NDEF messages to those tags, the application supports them. The key difference between ISO 14443 tags and ISO 15693 tags is that ISO 15693 tags offer a greater transmitting distance[37].

3.5 NFC Data Models

This chapter describes how data are stored on NFC tags by the application.

3.5.1 Link Storage

URIs are stored using the WKT-URI format description and the URI string as payload, using UTF-8 encoding.

The following values of the NDEF record type definition fields are used:

36 http://www.rfidjournal.com/blog/entry/8182 [retrieved 2011-07-08]

37 http://www.itwissen.info/definition/lexikon/ISO-15693-ISO-15693.html [retrieved 2011-07-08]

- Type Name Format: `NFC Forum Well Known Type (0x01)`
- Type: `wkt:uri`

3.5.2 File Storage

Since NDEF records can only store basic byte arrays, the application uses ZIP archives to encapsulate files. This enables creating a quasi file system on an NFC tag. The ZIP archive's content is written as an NDEF record's payload.

The following values of the NDEF record type definition fields are used:

- Type Name Format: `media-type (0x02)`
- Type: `application/zip`

3.6 Reference Data Model

The reference model is stored in XML files using the extension `iir`. The schema can be found on p.49. It is basically a list of `Reference` elements that consist of `Trigger` and `Target` elements. The `Trigger` elements describe the circumstances under which a link is valid. In the case of this application, the fact that specific NFC tags or QR codes are read serves as trigger condition, hence the sub-element `Tag`. This model may be extended to use more complex conditions like context descriptions. The `Target` element stores the linked data that shall be retrieved if the trigger condition is met.

The `Tag` element contents further specify what kind of tag (QR or NFC) is being referenced in a URI manner.

Example:

```
<ReferenceList>
        <Reference>
                <Target>geo:52.545485,13.355754</Target>
                <Trigger>
                        <Tag>nfc:e00401000038a915</Tag>
                </Trigger>
        </Reference>
<ReferenceList>
```

3.7 Geometry Data Model

Due to the current lack of a suitable standard for indoor geometry data storage, a subset of OpenStreetMap XML is used for demonstration purposes. The data model will be replaced as soon as there is a standard (OpenFloorMap) for indoor geometries. The data model currently used only supports the basic geometry elements `node` and `way`[38].

38 http://wiki.openstreetmap.org/wiki/OSM_XML#OSM_XML_file_format [retrieved 2011-07-08]

3.8 Software Design

Note: This chapter provides an overview of the software design and about how the most important functionalities are implemented. However, the source code attached in Appendix I: TagLocate Source Code and Appendix II: NfcMimeLib Source Code contains additional comments and is in itself considered part of the documentation. Thus, the author encourages to refer to the source code for a deeper understanding of the application's work flow. The descriptions within this chapter provide references to the respective key parts of the code. There is also a Javadoc documentation on the CD delivered with this document.

Furthermore, this chapter assumes a basic understanding of Android and Java development fundamentals by the reader.

3.8.1 Basic Setup

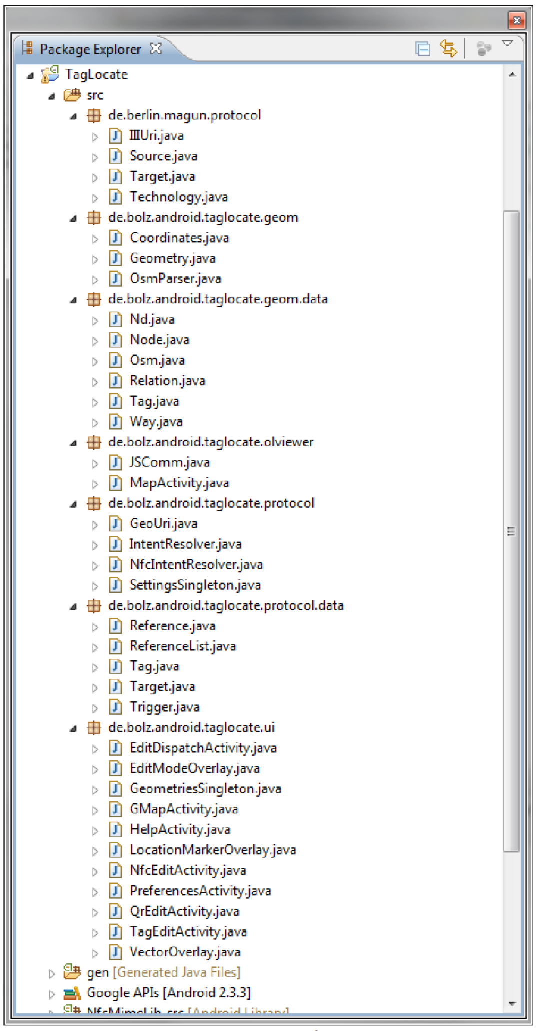
This is a fully formulated description of the fundamental application settings declared in the Android manifest file (see Android Manifest, p.44).

- Application version discussed here: 0.4
- Minimum required Android version: 2.2
- Android version required for NFC support: 2.3.3
- Android API level used for development: 10 (corresponds to Android version 2.3.3)
- Java namespace: `de.bolz.android.taglocate`
- Required Permissions[39]:
    - Access to location functionalities:
        - `ACCESS_COARSE_LOCATION`
        - `ACCESS_FINE_LOCATION`
        - `ACCESS_LOCATION_EXTRA_COMMANDS`
        - `ACCESS_MOCK_LOCATION`
    - Internet access:
        - `INTERNET`
    - File system access:
        - `WRITE_EXTERNAL_STORAGE`
    - Access to network information:
        - `ACCESS_NETWORK_STATE`
    - Access to NFC module:
        - `NFC`

39 http://developer.android.com/reference/android/Manifest.permission.html [retrieved 2011-07-08]

- Local folder used to store files:
  - `/mnt/sdcard/taglocate/`

3.8.2 Package Overview



```
Package Explorer
TagLocate
  src
    de.berlin.magun.protocol
      IIIUri.java
      Source.java
      Target.java
      Technology.java
    de.bolz.android.taglocate.geom
      Coordinates.java
      Geometry.java
      OsmParser.java
    de.bolz.android.taglocate.geom.data
      Nd.java
      Node.java
      Osm.java
      Relation.java
      Tag.java
      Way.java
    de.bolz.android.taglocate.olviewer
      JSComm.java
      MapActivity.java
    de.bolz.android.taglocate.protocol
      GeoUri.java
      IntentResolver.java
      NfcIntentResolver.java
      SettingsSingleton.java
    de.bolz.android.taglocate.protocol.data
      Reference.java
      ReferenceList.java
      Tag.java
      Target.java
      Trigger.java
    de.bolz.android.taglocate.ui
      EditDispatchActivity.java
      EditModeOverlay.java
      GeometriesSingleton.java
      GMapActivity.java
      HelpActivity.java
      LocationMarkerOverlay.java
      NfcEditActivity.java
      PreferencesActivity.java
      QrEditActivity.java
      TagEditActivity.java
      VectorOverlay.java
  gen [Generated Java Files]
  Google APIs [Android 2.3.3]
  NfcMimeLib_src [Android Library]
```

Figure 11: TagLocate package tree

This chapter provides an overview of the packages and their roles within TagLocate, in alphabetical order.

- **`de.bolz.taglocate.geom`:** This package contains classes that represent geometries and coordinates that are created by parsing OpenStreetMap XML files, as well as the according parsing functionality.
- **`de.bolz.taglocate.geom.data`:** This package contains classes representing elements of OSM XML files, to be used for (de-)serialization using the Simple XML framework[40].
- **`de.bolz.taglocate.olviewer`:** The classes of this package are deprecated. They result from an attempt to do the map rendering

40 http://simple.sourceforge.net/ [retrieved 2011-07-08]

using OpenLayers within a web view and the PhoneGap framework [41]. This was dismissed for various reasons (see chapter 3.8.12: Map View, p.33) The code is preserved for documentation purposes.

- **`de.bolz.taglocate.protocol`:** This package contains the business logic to extract location and/or files from tags as well as functionality to create reference models out of `iir` reference files.
- **`de.bolz.taglocate.protocol.data`:** Contains class representing elements of `iir` reference files, to be used for (de-)serialization using the Simple XML framework.
- **`de.bolz.taglocate.ui`:** Contains all user interface functionality (activities). Most notably the class `GmapActivity` (see p.88), which represents the main map view and also contains NFC and location handling business logic.

3.8.3 External Libraries, APIs and Packages

This chapter provides an overview on imported functionality.

- **Google APIs**
  - Author: Google Inc. and the Open Handset Alliance
  - Type: Android core APIs
  - Version: 2.3.3 (API level 10)
  - License:
    - Android API: custom license, free to use[42]
    - Google Location API: custom license, free to use under certain conditions[43]
  - Usage within this project: All Android-specific functionality.
- **Apache Commons Codec**
  - Author: Apache Software Foundation
  - Type: JAR library
  - Version: 1.5
  - License: Apache License, version 2.0
  - Usage within this project: Generating hexadecimal strings out of byte arrays (for NFC UID handling).
- **Apache Commons IO**
  - Author: Apache Software Foundation
  - Type: JAR library

41 http://www.phonegap.com/ [retrieved 2011-07-08]
42 http://developer.android.com/sdk/terms.html [retrieved 2011-07-08]
43 http://code.google.com/intl/de/android/maps-api-tos.pdf [retrieved 2011-07-08]

    - Version: 2.0.1
    - License: Apache License, version 2.0
    - Usage within this project: Generating byte arrays out of InputStreams (for NFC file handling).
- **Apache Commons Lang**
    - Author: Apache Software Foundation
    - Type: JAR library
    - Version: 2.6
    - License: Apache License, version 2.0
    - Usage within this project: Performing various operations on byte arrays.
- **Simple XML**
    - Author: Niall Gallagher
    - Type: JAR library
    - Version: 2.5.3
    - License: Apache License, version 2.0
    - Usage within this project: serialization and deserialization of XML files, specifically OSM geometry files and IIR reference files.
- **NfcMimeLib**
    - Author: Johannes Bolz, the MAGUN project
    - Type: Android Library Project[44]
    - Version: current developer version
    - License: Apache License, version 2.0
    - Usage within this project: NFC I/O operations.
- **Package de.berlin.magun.protocol**
    - Author: Johannes Bolz, the MAGUN project
    - Type: package imported into project workspace
    - Version: state of MAGUN release 0.7.0
    - License: Apache License, version 2.0
    - Usage within this project: Handling of iii URIs.

3.8.4 NFC Handling

TagLocate uses the Android library project NfcMimeLib (see Appendix II: NfcMimeLib Source Code, p.112ff) for read and write operations on NFC tags. The library is developed by the author in the context of the MAGUN

44 http://developer.android.com/guide/developing/projects/projects-eclipse.html#SettingUpLibraryProject [retrieved 2011-07-08]

project. The idea behind NfcMimeLib is to abstract NFC tag handling (which is mainly reading/writing raw byte arrays) to a higher level. Possible applications are:

- Write files' contents to NFC tags, using MIME type declarations
- Reading/writing encapsulated file/folder hierarchies (quasi file systems)
- Generating and writing checksums for data validation
- Reading/writing URI strings

The operations performed in TagLocate are reading / writing NDEF message content and reading the tag's UID. The classes `NdefMessageBuilder` and `RfidDAO` are used to perform these operations. For example, writing URIs to tags is done as follows (see NfcEditActivity.java, p.99 for details):

A `NdefMessageBuilder` is created, then an URI string is passed to the `addUriRecord` method:

```
NdefMessageBuilder builder = new NdefMessageBuilder();
(...)
builder.addUriRecord(uri.getString());
```

Then, an RfidDAO object is used to perform the write operation:

```
RfidDAO dao = new RfidDAO();
(...)
dao.writeMessage(tag, builder.buildMessage());
```

To get the NFC tag's UID, there is a method within `RfidDAO`:

```
String id = dao.getTagId(tag)
```

For information on file handling, see chapter 3.8.11: NFC File Storage, p.33).

The main Activity, `GmapActivity`, uses the NFC foreground dispatch[45] to read all NFC tags that match the specified intent filters. To resolve NFC intents, the `NfcIntentResolver` class is used (see NfcIntentResolver.java, p.76).

3.8.5 QR Code Handling

For reading QR codes, TagLocate calls an external application, ZXing Barcode Scanner[46], via an intent:

```
Intent intent = new Intent
      (getString(R.string.qr_scan_action));
intent.setPackage(getString(R.string.qr_reader_package));
intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
startActivityForResult(intent, 0);
```

45 http://developer.android.com/guide/topics/nfc/index.html#foreground-dispatch [retrieved 2011-07-08]

46 https://market.android.com/details?id=com.google.zxing.client.android [retrieved 2011-07-08]

The returned intent is then resolved using an IntentResolver (see next chapter).

3.8.6 Intent Resolution

For extracting the actual location data out of the intents generated by NFC/QR read events, the class de.bolz.android.taglocate.protocol.IntentResolver is used (e.g. within the method GmapActivity.onNewIntent):

```
IntentResolver i = new IntentResolver(
       getApplicationContext(), intent);
if (i.getLocation() != null) {
       updateLocation(i.getLocation().getLon(), I
       .getLocation().getLat());
}
```

IntentResolver basically checks the intent's data for supported URIs and file archives and provides functionality to extract these data.

3.8.7 URI Resolution

The package de.berlin.magun.protocol, imported from MAGUN's eFlyer project, is responsible for resolving iii URIs. The class IIIUri represents said URIs and also provides parsing functionality. The individual parts of the iii URI (in the state of development at the time of writing these are target, source and technology) also have class representations within the package. Since the iii URI syntax is still being developed, the classes representing it are bound to change.

Geo URI resolution follows the same principle: URI strings are parsed into objects. Geo URIs are represented by the class de.bolz.android.taglocate.protocol.GeoURI.

The resolutions itself is carried out by passing the URI string read from a tag to the constructor method of IIIUri/GeoURI:

```
iiiUri = new IIIUri(uri);
```

The parts of interest for further evaluation may then be accessed by calling their getter methods.

3.8.8 Reference Data Resolution

Reference data is stored using XML files with the extension iir. To perform matching, the currently used file is deserialized using the Simple XML framework, which returns an object representation of the (ReferenceList) root element (see IntentResolver.java, p.72):

```
ReferenceList list = null;
(...)
Serializer s = new Persister(new Format(new
CamelCaseStyle()));
list = s.read(ReferenceList.class, input);
(...)
```

Then, the application iterates through the list of references. During the iteration, each `Target` element's value is matched against the tag's ID:

```
List<Reference> links = list.getReferences();
String trigger;
(...)
trigger = (new Source(
       links.get(i).getTrigger().getTag().GetTagStr()))
       .getValue();
if (this.id.equalsIgnoreCase(trigger))
(...)
```

If there is a match, the location is updated.

When referencing new IDs, the ID is added to the data model, which is then again serialized (see TagEditActivity.java, p.106).

3.8.9 Geometry Data Resolution

The OSM geometry file is deserialized the same way as the reference file, as described in the previous chapter. Then `Geometry` objects are created that represent polygons out of the OSM `way` elements (see OsmParser.java, p.59):

```
List<Geometry> geometries = new ArrayList<Geometry>();
// Extract Way objects:
List<Way> ways = osm.getWaylist();
// Iterate through ways:
for (int i = 0; i < ways.size(); i++) {
       // Get Nd objects representing osm nd elements:
       List<Nd> nds = ways.get(i).getNdlist();
       List<Coordinates> coords = new
       ArrayList<Coordinates>();
              // Create Coordinates objects out of each node
              // element corresponding to the way's nd element:
              for (int j = 0; j < nds.size(); j++) {
                     double lat = nodes.get(String.valueOf(nds.get(j).
                                          getRef())).getLat();
                     double lon = nodes.get(String.valueOf(nds.get(j).
                                          getRef())).getLon();
                     coords.add(new Coordinates(lat, lon,
                            null));
              }
              geometries.add(new Geometry(coords,
                     ways.get(i).getId()));
       }
```

The Geometry objects are then used for map rendering (see chapter 3.8.12: Map View, p.33).

3.8.10 Location Provider Concept

The class `de.bolz.android.taglocate.ui.GMapActivity` is the main activity of the application. Thus, it contains methods to create and update the custom location provider. It also contains a nested implementation of the `LocationListener` interface[47] that is responsible for updating the map when the location is updated. The work flow is as follows:

First, the location provider is created within the method `createLocationProvider` with a unique identifier string:

```
locationManager.addTestProvider(TAG_LOCATION, (...)
```

If a new location could be deducted from a tag, the location provider is updated (method `updateLocation`):

```
Location newLocation = new Location(TAG_LOCATION);
      newLocation.setLatitude(lat);
      newLocation.setLongitude(lon);
      (...)
      locationManager.setTestProviderLocation(TAG_LOCATION,
            newLocation);
```

Finally, the `LocationListener` implementation's (`GmapActivity.TagLocationListener`) `onLocationChanged` method picks up the new location and updates the map.

3.8.11 NFC File Storage

If a zipped file system (see chapter 3.1.6: NFC File Handling, p.21) is detected on a NFC tag, the application writes the ZIP archive's content to the application folder on the SD card. A `ZipFileSystem` object is created out of the NDEF record. The `ZipFileSystem` class contains a `write` method that extracts and writes all contained files/folders to the local file system:

```
List<NdefMimeRecord> recordList =
      dao.getCachedMimeRecords(tag);
      (...)
      zfs = new ZipFileSystem(recordList.get(i));
      List<String> filenames = zfs.getFileNames();
      for (int j = 0; j < filenames.size(); j++) {
            String fname = filenames.get(j);
            // Write files to data directory:
            if (fname.endsWith(".iir") ||
            fname.endsWith(".osm")) {
                  zfs.write(SettingsSingleton.getInstance()
                        .getDatapath());
            }
```

3.8.12 Map View

The main map view is created by using a `MapActivity` (class `de.bolz.android.taglocate.ui.GMapActivity`). This is using the Maps API provided by Google. During the `onCreate` method, the map is initialized to display the standard Google satellite/aerial map and two overlays:

```
mapView.setSatellite(true);
mapOverlays = mapView.getOverlays();
mapOverlays.add(new VectorOverlay());
locOverlay = new LocationMarkerOverlay(
```

47 http://developer.android.com/reference/android/location/LocationListener.html [retrieved 2011-07-08]

```
            getApplicationContext());
```

VectorOverlay contains the room geometries specified in the OSM geometry file, and LocationMarkerOverlay displays the marker at the latest position read from the custom location provider. The EditModeOverlay is added in edit mode (that is when a new tag shall be written/referenced):

```
editOverlay = new EditModeOverlay(getApplicationContext());
mapOverlays.add(editOverlay);
```

In addition, the activity provides a menu that gives access to the QR reader mechanism, the options menu, the edit mode and a how-to page (which is a HTML view).

It was attempted to use a specific mobile version of OpenLayers in conjunction with PhoneGap[48], a multi-platform framework that enables communication between HTML/JavaScript code and the underlying mobile platform. However, due to performance considerations and poor JavaScript debugging support on Android, this idea was abandoned in favour of the Google Maps API. The classes have been preserved for documentation purposes in the package `de.bolz.android.taglocate.olviewer`.

3.9 Demonstrator Tag Installation

This thesis includes a set of tags to demonstrate the use cases. Due to the university's fire protection policy, the tags are not installed in the building, but attached to this document (see Appendix III: Demonstrator Tags and Project Data, p.127). The set comprises seven NFC tags and one printed QR code, each one of them referencing the location of a room in a part of the Bauwesen building[49] at Beuth University. The tags use different referencing mechanisms. Also, one of the NFC tags contains a geometry and a reference file that may be downloaded if the respective option is enabled in the application.

48 http://www.phonegap.com/ [retrieved 2011-07-08]

49 http://www.beuth-hochschule.de/fileadmin/oe/bau/dokumente/zeichnungen/bauwesen/Grundriss_Haus_Bauwesen_1.OG_A1.pdf [retrieved 2011-07-08]

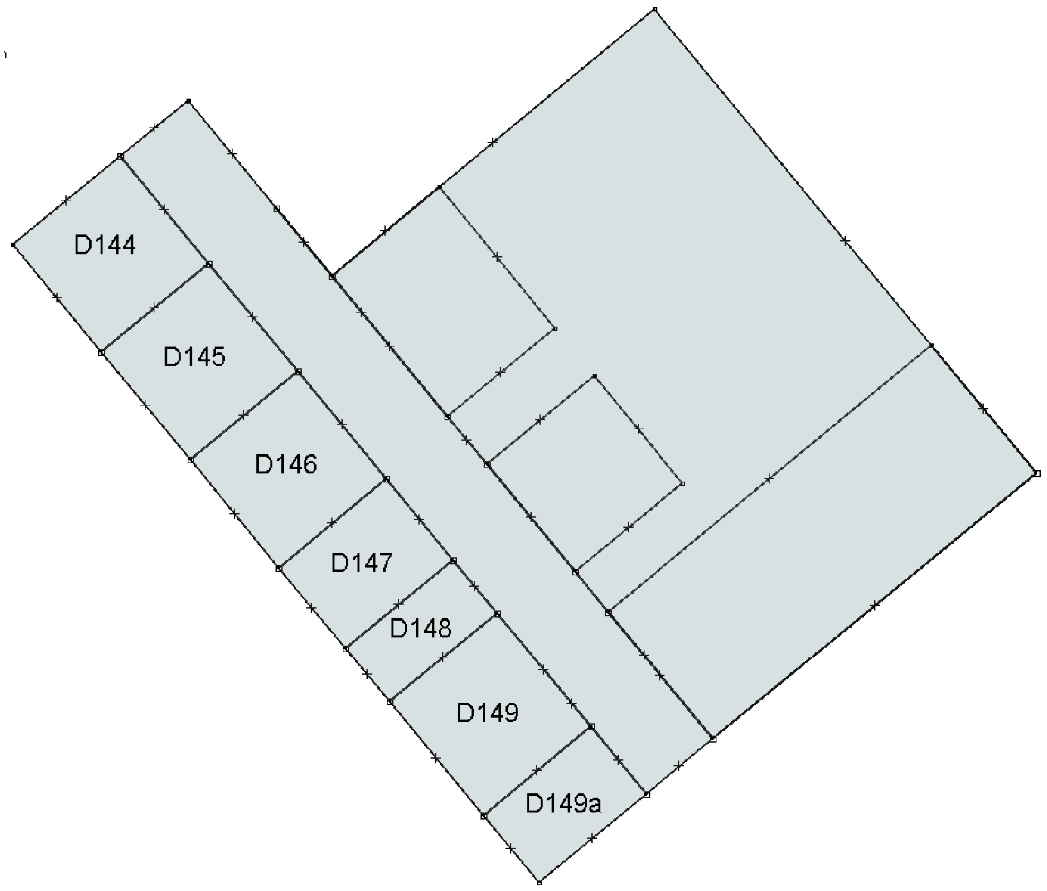


Figure 12: Demonstrator geometries with referenced room numbers.

Here is a list of the tags used for the demonstrator installation:

Tag ID:	Tag Technology:	Referencing room no.:	Referencing mechanism:
E004010000390726	NFC / ISO 15693	D144	UID
E00401000038DEBD	NFC / ISO 15693	D145	UID
E00401000038CA80	NFC / ISO 15693	D146 D147	UID Geo URI
E00401000038A94A	NFC / ISO 15693	D147	UID
E00401000038D051	NFC / ISO 15693	D148	Geo URI
E00401000038F402	NFC / ISO 15693	D149	iii URI
04806869BA2280	NFC / ISO 14443	D165, contains geometries and references	UID
-	QR Code	D149a	ID (unique content)

The ID references described in the table above are contained in the default reference file delivered with the application.

4 Results and Findings

This chapter summarizes findings resulting from the development, implementation and testing process.

4.1 Usability and Location Accuracy

Using object-based positioning, the error of the determined position equals the maximum transmitting range of the given transponder/reader system. – That is, assuming accurately georeferenced transponders. During tests, the range of NFC transponders turned out to be around 4cm for ISO 14443 tags and about 10cm for ISO 15693 tags. In theory, ISO 15693 tags are capable of transmitting ranges of around one meter[50].

On the one hand, the short range causes a high level of accuracy. On the other hand, it may be perceived as limitation to the usability. – Other than GPS or any other real-time positioning method, this method requires a physical action by the user (put device against tag). Another factor detrimental to usability is the fact that NFC communication always takes place as a one-to-one connection. So a NFC tag may not be read by multiple devices at once. This may be a disadvantage in crowded places.

The actual data transfer rates using the Nexus S device are around or less than a second for NFC transponders that are either empty (UID resolution) or only contain a location URI. For tags containing bigger amounts of data, like geometry files, reading speeds are around 1-2 kB per second, depending on the type of tag.

In comparison, QR codes offer the advantage of being readable by multiple devices at once. Also, they may be cheaper to install and the biggest share of today's smartphones are capable of reading them. The disadvantages are lower data capacities (in theory unlimited, but specifications exist for QR codes not bigger than 3kB of binary data[51]), the dependence on good lighting conditions and a free line of sight.

Another advantage of NFC over QR codes is the fact that the current application view must not be closed in order to show a view that allows aiming the camera at a QR code.

4.2 Reference Resolution

The major difference between the two ways of referencing discussed in previous chapters is the fact that ID referencing requires a separate means of storing the reference model. That makes a more complex data infrastructure necessary. On the other hand, having a centralized reference data store may make maintenance easier, for example if the target one particular tag is referencing needs to be changed.

50 http://www.itwissen.info/definition/lexikon/ISO-15693-ISO-15693.html [retrieved 2011-07-08]
51 http://www.denso-wave.com/qrcode/qrfeature-e.html [retrieved 2011-07-08]

Also, writing explicit links on NFC tags requires more expensive, writable transponders, that can hold the necessary data size, whereas ID resolution only requires a tag providing an ID. – That is the case for even the most simple NFC tags.

For using explicit references, write-once tags or tags providing access control may become necessary in order to prevent unwanted changes.

An implementation based on ID resolution may use already existing tag installations.

4.3 The Map

TagLocate at this point uses the Google Maps API for visualization and spatial operations. However, the API has limitations for the specific (indoor) use case:

First, the maximum possible zoom level may be perceived as too small for detailed indoor geometry display.

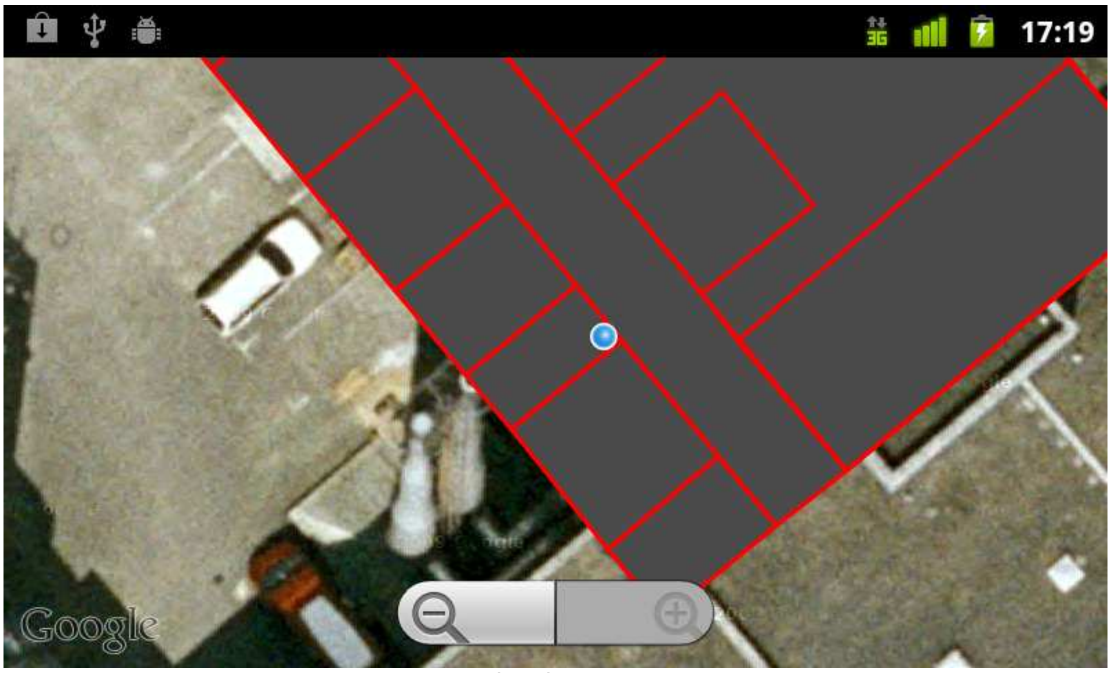


Figure 13: Map view at maximum zoom level.

Another limitation is the fact that the Google Maps API uses integer values to process coordinates: The decimal degree value is cut off after the sixth decimal place and converted to an integer[52]. When reprojecting pixel coordinates to geographic coordinates, a deviation by up to 30 microdegrees for the same pixel location can be observed. This can not be investigated any further since the source code of the Maps API is not public.

52 http://code.google.com/intl/de/android/add-ons/google-apis/reference/com/google/android/maps/GeoPoint.html [retrieved 2011-07-08]

5 Discussion

This chapter discusses how the application itself as well as object based referencing in general may be enhanced and refined in the future and concludes the thesis.

5.1 Future Work

Taking a look at how other technologies than GPS became commonly used, like WiFi or GSM cell ID based positioning, it is likely that several methods will become the norm for indoor use. – Especially, since there is no predominant technology to begin with, but several promising approaches. That means it will be vital to integrate several sources of location. The Android location provider concept, for example, offers a good way to deal with multiple location sources.

The reference model may be extended to enable context awareness in place of object awareness: In this case, a location or a data set would be resolved, if a multitude of machine-recognizable conditions are met. The reference data model was designed with that possible extension in mind (the `Trigger` element in this case would contain a set of conditions).

Furthermore, the application may be extended to recognize objects other than NFC tags and QR codes. For example, OCR software might be used to read room signs, or even more complex image recognition.

UID referencing could be leveraged by deploying a central repository of references. Reference resolution could take place by querying a web service accessing the repository. Using a VGI approach, new references could be added to the repository by users.

Another challenge for indoor mapping/navigation/information systems is handling the third dimension. This was a negligible factor for most outdoor scenarios, but knowing at least a floor number is vital for indoor applications. So current geometry data models need to be extended to include height information. In the framework of MAGUN such a data model ("OpenFloorMap") is being developed. The TagLocate application was developed with that extensibility in mind (for example, the class `Coordinates` already offers a field for height).

TagLocate currently uses mainly Geo URIs for referencing locations. Those Geo URIs include coordinates that represent the specific location. A consideration would be to extend those URIs so that they also can use geopaths, a syntax developed by MAGUN to describe locations in a hierarchical manner[53]. This way, not only points could be referenced, but also more complex geometries (like a polygon describing a room). Another possibility would be to extend the Geo URI syntax with a feature query functionality.

53 http://magun.beuth-hochschule.de/#about [retrieved 2011-07-08]

Concerning the application in particular, some changes and added functionalities appear reasonable: First, the location provider initialization should be extracted from the core application into a service. This would make sure that the location provider is immediately available once the device starts. The NFC localization mechanism may also be extracted from the main activity, so that localization doesn't require to the application to be running in the foreground.

Finally, due to the shortcomings of the Google Maps API described earlier (limited accuracy and zoom level), other visualization/spatial computation libraries should be evaluated.

On the hardware side, it would be beneficial for usability if NFC modules supported the entire transmitting range that some tag types (specifically ISO 15693) allow for. This would not necessarily sacrifice privacy, since the functionality could be restricted to reading mode.

5.2 Conclusion

It could be established that using NFC transponders and mobile devices for positioning and data referencing is a viable solution.

There may be drawbacks concerning usability, but on the other hand, the technology is easy to implement and maintain, and cheap.

As stated earlier, the best results in accuracy and usability will come from an integration of various localization methods. The method evaluated in this thesis may well integrate with other technologies currently being developed, like WiFi fingerprinting, barometric pressure measurement or inertial navigation.

6 Table of Figures

All figures except figure 5 created by the author.

7 Bibliography

Ableson et al. (2009): Unlocking Android: A Developer's Guide. – Greenwich.

Android Developers: API Reference (Javadoc). – http://developer.android.com/reference/. [Retrieved 2011-07-08]

Android Developers: Managing Projects from Eclipse with ADT. – http://developer.android.com/guide/developing/projects/projects-eclipse.html. [Retrieved 2011-07-08]

Android Developers: Near Field Communication. – http://developer.android.com/guide/topics/nfc/index.html. [Retrieved 2011-07-08]

Assunção (2009): Android location provider mock. – http://pedroassuncao.com/2009/11/android-location-provider-mock/. [Retrieved 2011-07-08]

Bach, Bächle (2001): Grundriss Haus Bauwesen 1.OG. – Beuth Hochschule für Technik. – http://www.beuth-hochschule.de/fileadmin/oe/bau/dokumente/zeichnungen/bauwesen/Grundriss_Haus_Bauwesen_1.OG_A1.pdf. [Retrieved 2011-07-08]

Beuth Hochschule für Technik (2011): MAGUN: A Open Source Project. – http://magun.beuth-hochschule.de/. [Retrieved 2011-07-08]

Bolz (2011): TagLocate. – Android Market. – https://market.android.com/details?id=de.bolz.android.taglocate. [Retrieved 2011-07-08]

Clark (2010-12-07): Google unveils first Android NFC phone – but Nexus S is limited to tag reading only for now. – Near Field Communications World. – http://www.nearfieldcommunicationsworld.com/2010/12/07/35385/google-unveils-first-android-nfc-phone-but-nexus-s-is-limited-to-tag-reading-only-for-now/. [Retrieved 2011-07-08]

Denso Wave Inc. (2010): QR Code Features. – http://www.denso-wave.com/qrcode/qrfeature-e.html. [Retrieved 2011-07-08]

Ducrohet (2011-02-09): Android 2.3.3 Platform, New NFC Capabilities. – Android Developers. – http://android-developers.blogspot.com/2011/02/android-233-platform-new-nfc.html. [Retrieved 2011-07-08]

Eisfeller, Zucker (2005): Indoor-GPS: Ist der Satelliten-empfang in Gebäuden möglich? – http://forschung.unibw-muenchen.de/papers/4lut9jfxv9hd0ig1paaiqprktdwqko.pdf. [Retrieved 2011-07-08]

Finkenzeller (2003): RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification. – Second edition, Munich.

Francica (2010-08-22): Indoor Positioning and Navigation: The Next LBS Frontier? – Directions Magazine. –http://www.directionsmag.com/articles/indoor-positioning-and-navigation-the-next-lbs-frontier/129925. [Retrieved 2011-07-08]

Gale (2010-11-23): A First Step Towards Indoor Navigation. Literally. – http://www.vicchi.org/2010/11/23/a-first-step-towards-indoor-navigation-literally/. [Retrieved 2011-07-08]

Google Code: APIs Add-On reference (Javadoc). – http://code.google.com/intl/de/android/add-ons/google-apis/reference/. [Retrieved 2011-07-08]

Google Inc. (2008): Android Maps APIs Terms of Service. – http://code.google.com/intl/de/android/maps-api-tos.pdf. [Retrieved 2011-07-08]

Google Inc. (2009): Terms and Conditions. – Android Developers. – http://developer.android.com/sdk/terms.html. [Retrieved 2011-07-08]

Heise Online (2010-07-23): Smartphone-Verkaufszahlen klettern um mehr als 40 Prozent. – http://www.heise.de/newsticker/meldung/Smartphone-Verkaufszahlen-klettern-um-mehr-als-40-Prozent-1044272.html. [Retrieved:2011-07-08]

Hönniger (2010-06-10): Indoor Navigation: Ein Überblick uns Stand der Forschung an der HCU. – http://www.geomatik-hamburg.de/forum-geomatik/2010/pdf/4-Indoor-Navigation_Hoenniger.pdf. [Retrieved 2011-07-08]

HTW Berlin (2010): Projekt Magun: MAGUN WIKI. – http://www.magun-projekt.de/. [Retrieved 2011-07-08]

IndoorLBS, LLC (2011): WiFi Location Services for LBS: Comparison. – http://www.indoorlbs.com/p/wifi-location-services-for-lbs_18.html. [Retrieved 2011-07-08]

ITWissen: ISO 15693. – http://www.itwissen.info/definition/lexikon/ISO-15693-ISO-15693.html. [Retrieved 2011-07-08]

James (2009-05-26): GPS. – TIME Magazine U.S. – http://www.time.com/time/magazine/article/0,9171,1901500,00.html. [Retrieved 2011-07-08]

Linnhoff-Popien (2010): WLAN Positioning. – http://www.mobile.ifi.uni-muenchen.de/studium_lehre/verg_semester/ws1011/msp/08_wlan_positioning.pdf. [Retrieved 2011-07-08]

Mayrhofer, Spanring (2010): A Uniform Resource Identifier for Geographic Locations ('geo' URI). – Internet Engineering Task Force (IETF). – http://tools.ietf.org/rfc/rfc5870. [Retrieved 2011-07-08]

Mayrhofer, Spanring: The "geo:" URI scheme. – http://geouri.org/. [Retrieved 2011-07-08]

MST250 WIKI (2009): The History of the Automotive Navigation System. – http://hst250.history.msu.edu/wiki-online/index.php/The_History_of_the_Automotive_Navigation_System. [Retrieved: 2011-07-08]

NFC Forum: http://www.nfc-forum.org/home/. [Retrieved 2011-07-08]

NFC Forum (2006-07-24): NFC Data Exchange Format (NDEF): Technical Specification. – http://www.nfc-forum.org/specs/spec_license: NFC Data Exchange Format (NDEF) Technical Specification. [Retrieved 2011-07-08]

NXP Semiconductors (2009): NFC Forum Type Tags: White Paper V1.0. – http://www.nxp.com/acrobat_download2/other/identification/173110_NFC_Forum_Type_Tags_WhitePaper.pdf. [Retrieved 2011-07-08]

O'Brien (2010-11-14): Smartphone Sales Taking Toll on G.P.S. Devices. – New York Times. –http://www.nytimes.com/2010/11/15/technology/15iht-navigate.html. [Retrieved 2010-07-08]

OpenPCD: ISO14443. – http://www.openpcd.org/ISO14443. [Retrieved 2011-07-08]

OpenStreetMap Foundation: OSM XML. – http://wiki.openstreetmap.org/wiki/OSM_XML. [Retrieved 2011-07-08]

Oszdenizci et al. (2011): Development of an Indoor Navigation System Using NFC Technology. – http://nfclab.isikun.edu.tr/papers/NFCInternal.pdf. [Retrieved 2011-07-08]

PhoneGap: http://www.phonegap.com/. [Retrieved 2011-07-08]

Radio-Electronics.com: NFC Near Field Communication. – http://www.radio-electronics.com/info/wireless/nfc/near-field-communications-tutorial.php. [Retrieved 2011-07-08]

Radio-Electronics.com: NFC Tags and Tag Types. – http://www.radio-electronics.com/info/wireless/nfc/near-field-communications-tags-types.php. [Retrieved 2011-07-08]

Rizos: Pseudolite Augmentation of GPS. – http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.3605&rep=rep1&type=pdf. [Retrieved 2011-07-08]

Roberti (2011): NFC Phones Can Read ISO 15693 Tags. – RFID Journal. – http://www.rfidjournal.com/blog/entry/8182. [Retrieved 2011-07-08]

Shaikh et al. (2005): Inertial navigation sensors for mobile mapping. – Geospatial World. – http://www.geospatialworld.net/index.php?option=com_content&view=article&id=20580&Itemid=1925. [Retrieved 2011-07-08]

Simple XML Serialization. – http://simple.sourceforge.net/. [Retrieved 2011-07-08]

The Apache Software Foundation (2004): Apache License, Version 2.0. – http://www.apache.org/licenses/LICENSE-2.0.html. [Retrieved 2011-07-08]

The H (2011-02-01): Smartphones: Android overtakes Symbian, Apple loses market share. – http://www.h-online.com/open/news/item/Smartphones-Android-overtakes-Symbian-Apple-loses-market-share-1181349.html. [Retrieved 2011-07-08]

ZXing (2010): Barcode Contents. – http://code.google.com/p/zxing/wiki/BarcodeContents. [Retrieved 2011-07-08]

ZXing (2011): Barcode Scanner. – Android Market. – https://market.android.com/details?id=com.google.zxing.client.android. [Retrieved 2011-07-08]

Appendix I: TagLocate Source Code

Android Manifest

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      android:versionCode="1"
      package="de.bolz.android.taglocate" android:versionName="0.4">

      <supports-screens
        android:largeScreens="true"
        android:normalScreens="true"
        android:smallScreens="true"
        android:resizeable="true"
        android:anyDensity="true"
        />
        <uses-sdk android:minSdkVersion="8"/>
        <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
        <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
        <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
        <uses-permission android:name="android.permission.INTERNET" />
        <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
        <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
        <uses-permission android:name="android.permission.NFC" />
        <uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />

    <application android:label="@string/app_name" android:icon="@drawable/taglocate"
android:debuggable="false">
    <uses-library android:name="com.google.android.maps" />
        <activity android:name="de.bolz.android.taglocate.ui.GMapActivity"
                  android:label="@string/app_name"
                  android:theme="@android:style/Theme.NoTitleBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
             <activity android:name="de.bolz.android.taglocate.ui.PreferencesActivity"
                          android:label="Options" />
             <activity android:name="de.bolz.android.taglocate.ui.HelpActivity"
android:label="@string/app_name"
                    android:theme="@android:style/Theme.NoTitleBar" />
      <activity android:name="de.bolz.android.taglocate.ui.EditDispatchActivity"
android:label="EditDispatchActivity"
                    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
             <activity android:name="de.bolz.android.taglocate.ui.NfcEditActivity"
android:label="NfcEditActivity"
                    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
             <activity android:name=".ui.QrEditActivity" android:label="QrEditActivity"
                    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
    </application>
</manifest>
```

Resource Files

/assets Folder

default.iir

```xml
<ReferenceList>
   <Reference>
      <Target>geo:52.545485,13.355754</Target>
      <Trigger>
         <Tag>nfc:e00401000038a915</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.54537,13.35586</Target>
      <Trigger>
         <Tag>nfc:e00401000038f91c</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.545417,13.356048</Target>
      <Trigger>
         <Tag>nfc:e00401000038f3fe</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.545301,13.355984</Target>
      <Trigger>
         <Tag>nfc:e004010000390725</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.545485,13.355761</Target>
      <Trigger>
         <Tag>qr:QR-ID1</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.545369,13.355862</Target>
      <Trigger>
         <Tag>qr:QR-ID2</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.545415,13.356049</Target>
      <Trigger>
         <Tag>qr:QR-ID3</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.545299,13.356001</Target>
      <Trigger>
         <Tag>qr:QR-ID4</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.545467,13.355739</Target>
      <Trigger>
         <Tag>nfc:e004010000390726</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.545431,13.355787</Target>
      <Trigger>
         <Tag>nfc:e00401000038debd</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.5454,13.355832</Target>
      <Trigger>
         <Tag>nfc:e00401000038ca80</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.545366,13.355877</Target>
      <Trigger>
         <Tag>nfc:e00401000038a94a</Tag>
      </Trigger>
   </Reference>
   <Reference>
      <Target>geo:52.545297,13.355971</Target>
      <Trigger>
         <Tag>qr:This is a unique QR code referencing room D149a.</Tag>
      </Trigger>
   </Reference>
   <Reference>
```

```
      <Target>geo:52.545593,13.355695</Target>
      <Trigger>
         <Tag>nfc:04806869ba2280</Tag>
      </Trigger>
   </Reference>
</ReferenceList>
```

default.osm

```
<?xml version='1.0' encoding='UTF-8'?>
<osm version='0.6' generator='JOSM'>
  <node id='-199' visible='true' lat='52.54543296230852' lon='13.355945208447102' />
  <node id='-198' visible='true' lat='52.54547609579596' lon='13.355886785426597' />
  <node id='-195' visible='true' lat='52.54540591720643' lon='13.355891217200888' />
  <node id='-192' visible='true' lat='52.54544905072045' lon='13.355832794180383' />
  <node id='-190' visible='true' lat='52.54538561874553' lon='13.356009266435544' />
  <node id='-189' visible='true' lat='52.545418525782075' lon='13.355964694948886' />
  <node id='-186' visible='true' lat='52.545391500777974' lon='13.355910743772313' />
  <node id='-183' visible='true' lat='52.545358593721176' lon='13.35595531525897' />
  <node id='-181' visible='true' lat='52.5454276622878' lon='13.356133922300142' />
  <node id='-179' action='modify' visible='true' lat='52.5455302721537'
lon='13.355994940329245' />
  <node id='-178' visible='true' lat='52.54538864265328' lon='13.356186773104811' />
  <node id='-176' visible='true' lat='52.545469513165095' lon='13.355805078519836' />
  <node id='-121' visible='true' lat='52.545346440664666' lon='13.355971776151286' />
  <node id='-119' visible='true' lat='52.54526353048552' lon='13.355937006759161' />
  <node id='-117' visible='true' lat='52.54528423555142' lon='13.355908962517839' />
  <node id='-116' visible='true' lat='52.545311237586006' lon='13.355962867707621' />
  <node id='-108' visible='true' lat='52.545318927041286' lon='13.355861974156058' />
  <node id='-106' visible='true' lat='52.545335296914274' lon='13.355839801745335' />
  <node id='-104' visible='true' lat='52.54535992401653' lon='13.355806445195672' />
  <node id='-102' visible='true' lat='52.54539285164741' lon='13.355761845840785' />
  <node id='-100' visible='true' lat='52.54545843702303' lon='13.35567301258284' />
  <node id='-99' visible='true' lat='52.54542572623822' lon='13.355717318293447' />
  <node id='-97' visible='true' lat='52.545452728185765' lon='13.355771223483252' />
  <node id='-95' visible='true' lat='52.545419853615165' lon='13.355815751030583' />
  <node id='-93' visible='true' lat='52.54538692600457' lon='13.35586035038545' />
  <node id='-91' visible='true' lat='52.54536229891745' lon='13.355893706935108' />
  <node id='-89' visible='true' lat='52.54534592905453' lon='13.355915879345838' />
  <node id='-87' visible='true' lat='52.54550232730057' lon='13.355760632779688' />
  <node id='-86' visible='true' lat='52.54530742095793' lon='13.356024626955955' />
  <node id='-85' visible='true' lat='52.545290532532825' lon='13.355990911948911' />
  <node id='-84' visible='true' lat='52.54548543895046' lon='13.355726917772643' />
  <way id='-200' visible='true'>
    <nd ref='-195' />
    <nd ref='-192' />
    <nd ref='-198' />
    <nd ref='-199' />
    <nd ref='-195' />
    <tag k='building' v='yes' />
  </way>
  <way id='-191' visible='true'>
    <nd ref='-183' />
    <nd ref='-186' />
    <nd ref='-189' />
    <nd ref='-190' />
    <nd ref='-183' />
    <tag k='building' v='yes' />
  </way>
  <way id='-182' visible='true'>
    <nd ref='-86' />
    <nd ref='-121' />
    <nd ref='-181' />
    <nd ref='-178' />
    <nd ref='-86' />
    <tag k='building' v='yes' />
  </way>
  <way id='-180' action='modify' visible='true'>
    <nd ref='-176' />
    <nd ref='-192' />
    <nd ref='-179' />
    <nd ref='-178' />
    <nd ref='-86' />
    <nd ref='-183' />
    <nd ref='-186' />
    <nd ref='-195' />
    <nd ref='-176' />
    <tag k='building' v='yes' />
  </way>
  <way id='-120' visible='true'>
    <nd ref='-117' />
    <nd ref='-116' />
    <nd ref='-85' />
    <nd ref='-119' />
    <nd ref='-117' />
    <tag k='building' v='yes' />
  </way>
  <way id='-118' visible='true'>
    <nd ref='-108' />
    <nd ref='-89' />
    <nd ref='-116' />
    <nd ref='-117' />
    <nd ref='-108' />
    <tag k='building' v='yes' />
  </way>
  <way id='-109' visible='true'>
    <nd ref='-106' />
    <nd ref='-91' />
    <nd ref='-89' />
    <nd ref='-108' />
    <nd ref='-106' />
    <tag k='building' v='yes' />
  </way>
  <way id='-107' visible='true'>
    <nd ref='-104' />
    <nd ref='-93' />
    <nd ref='-91' />
    <nd ref='-106' />
    <nd ref='-104' />
    <tag k='building' v='yes' />
  </way>
  <way id='-105' visible='true'>
    <nd ref='-95' />
    <nd ref='-93' />
    <nd ref='-104' />
    <nd ref='-102' />
    <nd ref='-95' />
    <tag k='building' v='yes' />
  </way>
  <way id='-103' visible='true'>
    <nd ref='-99' />
    <nd ref='-97' />
    <nd ref='-95' />
    <nd ref='-102' />
    <nd ref='-99' />
    <tag k='building' v='yes' />
  </way>
  <way id='-101' visible='true'>
    <nd ref='-84' />
    <nd ref='-97' />
    <nd ref='-99' />
    <nd ref='-100' />
    <nd ref='-84' />
    <tag k='building' v='yes' />
  </way>
  <way id='-88' action='modify' visible='true'>
    <nd ref='-84' />
    <nd ref='-87' />
    <nd ref='-176' />
    <nd ref='-195' />
    <nd ref='-186' />
    <nd ref='-183' />
    <nd ref='-121' />
    <nd ref='-86' />
    <nd ref='-85' />
    <nd ref='-89' />
    <nd ref='-91' />
    <nd ref='-93' />
    <nd ref='-95' />
    <nd ref='-97' />
    <nd ref='-84' />
    <tag k='building' v='yes' />
  </way>
</osm>
```

help.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
       <h2>TagLocate</h2>
       <p><b>v.0.4</b> <br>
              2011 <a href="mailto:johannes-bolz@gmx.net">Johannes Bolz</a>, <a
href="http://magun.beuth-hochschule.de">MAGUN project</a><br>
              This application is licensed under the <a
href="http://www.apache.org/licenses/LICENSE-2.0">Apache
              License, Version 2.0</a>.</p>

       <h3>What is this?</h3>
       <p>Tag Locate is a demonstrator application for object-based indoor positioning,
focusing mainly on NFC
       technology. However, it also supports QR codes.</p>

       <h3>How to locate</h3>
       <p><i><b>NFC</b></i>: Put the device against a NFC tag that contains location
information while the app is running. <br>
       <i><b>QR</b></i>: Press menu key, select "QR Locate". <a href="market://details?
id=com.google.zxing.client.android">'Barcode Scanner'</a>
       must be installed for this to work.</p>

       <h3>Tag data format</h3>
       <p>Tag Locate supports <a href="http://en.wikipedia.org/wiki/Geo_URI">Geo-URI's</a> as
well as
       <a href="http://magun.beuth-hochschule.de">iii-URI's</a> containing a Geo-URI target
element.<br>
       For QR-codes, use <a href="http://zxing.appspot.com/generator/">'Geo-QR code'</a>,
which is basically a Geo-URI.</p>

       <h3>ID resolution</h3>
       <p>In addition to explicit location URI's written on the tags, the unique ID of NFC
tags may also be used for getting a
       location. In this case, the application uses a reference file that links those ID's
with coordinates. QR codes may also be
       used as ID's, in which case their specific content will be no further evaluated. See
the default reference file
       (/sdcard/taglocate/default.iir) for more information.</p>

       <h3>Resolution strategies</h3>
       <p>There are multiple strategies to get location information out of tags:<br><br>
       <i><b>ID first:</b></i> The tag will only be evaluated for an explicit location link,
if it's ID cannot
       be found in the link file.<br>
       <i><b>Link first:</b></i> The ID will only be looked up if the tag doesn't contain an
explicit location link.<br>
       <i><b>ID only:</b></i> Only the ID will be evaluated.<br>
       <i><b>Link only:</b></i> Only the payload data will be evaluated.</p>

       <h3>Files</h3>
       <p>The /taglocate/ folder on the SD card contains geometry files and link files. You
can define your
       own files here. Note that the geometry files currently use a very simplistic <a
```

```
href="http://wiki.openstreetmap.org/wiki/XML">OSM format</a>, only
	supporting 'node' and 'way' elements.</p>

	<h3>File download from NFC</h3>
	<p>It is possible to store geometry and link files on NFC tags. Those files may be
downloaded and
	applied. To create a NFC tag containing files, put the desired files into a ZIP archive
and write
	the archive's byte array into a NDEF record using the MIME declaration
'application/zip'.</p>

	<h3>Reference new Tags</h3>
	<p>To reference new NFC tags or QR codes, select 'Add Tag' from the options menu. A
crosshairs symbol
	will appear in the center of the map. Move the view so that the symbol marks the
location you want to
	associate with the tag. Select 'Add Tag here'. Now you can choose between: <br><br>
	<b><i>'Write NFC Tag': </i></b>The location will be written to a NFC tag using a Geo-
URI.<br>
	<b><i>'Reference NFC ID': </i></b>The tag's ID will be associated with a location
within the local
	reference model (i.e. IIR file). If the ID has already been referenced, the reference
will be updated
	with the new location.<br>
	<b><i>'Reference QR Code': </i></b>The QR code will be referenced in the local
reference model just
	like a NFC ID. So make sure the QR's content is unique.<br>
	</p>
</html>
```

iir.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2011 sp1 (x64) (http://www.altova.com) by Johannes Bolz (Technische
Fachhochschule Berlin) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="ReferenceList">
		<xs:annotation>
			<xs:documentation>Root Element. Contains all
references.</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element name="Reference" minOccurs="0" maxOccurs="unbounded">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="Target" type="xs:anyURI"
maxOccurs="unbounded"/>
							<xs:element name="Trigger">
								<xs:complexType>
									<xs:choice>
										<xs:element name="Tag"
type="xs:string"/>
									</xs:choice>
								</xs:complexType>
							</xs:element>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

/res/layout Folder

help.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/mainlayout" android:orientation="vertical"
      android:layout_width="fill_parent" android:layout_height="fill_parent">

      <com.google.android.maps.MapView
            android:layout_width="fill_parent" android:layout_height="fill_parent"
            android:clickable="true" android:apiKey="###########################"
            android:id="@+id/mapview" />

      <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/btnLayout"
            android:visibility="invisible"
            android:orientation="horizontal"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_marginTop="10px"
            android:layout_centerHorizontal="true">
            <Button android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="@string/add_tag_btn"
                  android:id="@+id/addTagBtn" />
            <Button android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="@string/cancel_btn"
                  android:id="@+id/cancelAddTagBtn" />

      </LinearLayout>



</RelativeLayout>
```

options.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
</LinearLayout>
```

/res/menu Folder

map_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:title="@string/qr_locate" android:id="@+id/qr_menu_item"></item>
    <item android:title="@string/settings" android:id="@+id/settings_menu_item"></item>
    <item android:id="@+id/reference_tag_item" android:title="@string/reference_tag"></item>
    <item android:title="@string/howto" android:id="@+id/help_menu_item"></item>
</menu>
```

/res/values Folder

arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="strategy_values">
       <item>uid_f</item>
       <item>uri_f</item>
       <item>uid_o</item>
       <item>uri_o</item>
    </string-array>
    <string-array name="strategy_names">
       <item>ID first</item>
       <item>Link first</item>
       <item>ID only</item>
       <item>Link only</item>
    </string-array>

    <string-array name="edit_dialog_items">
        <item>Write NFC Tag</item>
        <item>Reference NFC ID</item>
        <item>Reference QR Code</item>
    </string-array>
</resources>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

       <!-- UI Strings: -->
             <!-- titles, static descriptions: -->
    <string name="app_name">TagLocate</string>
    <string name="qr_locate">QR Locate</string>
    <string name="settings">Options</string>
    <string name="howto">About / How to Use</string>
    <string name="title_gf_preference">Geometry File</string>
    <string name="files_preferences">Files</string>
    <string name="title_lf_preference">Reference File</string>
    <string name="processing_preferences">Processing</string>
    <string name="loadfiles_preference">Load Files from NFC</string>
    <string name="loadfiles_summary">Download and apply geometry and reference files, if
available.</string>
    <string name="title_strategy_preference">Resolution Strategy</string>
    <string name="message_bc_unavailable">Please install \'Barcode Scanner\' to read QR
codes.</string>
    <string name="title_ml_disabled">Mock locations disabled</string>
    <string name="message_ml_disabled">This application requires the \'Mock Locations\' system
setting
       to be enabled. Go to \'Application Preferences\' -> \'Development\'
       and check \'Allow Mock Locations\'.</string>
       <string name="message_nfc_not_supported">Not supported on this device.</string>
       <string name="add_tag_btn">Add Tag here</string>
       <string name="cancel_btn">Cancel</string>
       <string name="reference_tag">Add Tag</string>
       <string name="initial_nfc_dialog">Please scan NFC tag...</string>
       <string name="done">Done.</string>
       <string name="ok">OK</string>
       <string name="write_fail">Write failed!</string>
    <string name="format_msg">Formatted tag. Please scan again!</string>
    <string name="reference_done">Referenced tag: </string>
    <string name="reffile_found">Found Reference File!</string>
    <string name="geomfile_found">Found Geometry File!</string>

    <!-- Configuration Strings: -->
    <string name="default_geometry_file">default.osm</string>
    <string name="qr_reader_package">com.google.zxing.client.android</string>
    <string name="qr_scan_action">com.google.zxing.client.android.SCAN</string>
    <string name="help_url">file:///android_asset/help.html</string>
    <!-- Key Strings: -->
    <string name="strategy_preference">strategy preference key</string>
    <string name="link_file_preference">link file preference key</string>
    <string name="geometry_file_preference">geometry file preference key</string>
    <string name="hidden_prefs">hidden prefs key</string>
    <string name="lat_preference">lat preference key</string>
    <string name="lon_preference">lon preference key</string>
    <string name="zoom_preference">zoom preference key</string>
    <string name="edit_mode_preference">edit mode preference key</string>
    <string name="center_lon">view center lon key</string>
    <string name="center_lat">view center lat key</string>
    <string name="result_lat">result lat key</string>
    <string name="result_lon">result lon key</string>
    <string name="nfc_edit_mode">nfc edit mode key</string>


</resources>
```

/res/xml Folder

preferences.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">

    <PreferenceCategory
             android:title="@string/processing_preferences">
      <ListPreference
               android:key="@string/strategy_preference"
               android:title="@string/title_strategy_preference"
               android:dialogTitle="@string/title_strategy_preference"
               android:entryValues="@array/strategy_values"
               android:entries="@array/strategy_names"
               android:defaultValue="0" />
      <CheckBoxPreference
               android:key="@string/loadfiles_preference"
               android:title="@string/loadfiles_preference"
               android:summary="@string/loadfiles_summary"
               android:defaultValue="true" />
      </PreferenceCategory>
    <PreferenceCategory
            android:title="@string/files_preferences">
        <ListPreference
               android:key="@string/link_file_preference"
               android:title="@string/title_lf_preference"
               android:dialogTitle="@string/title_lf_preference"
               android:defaultValue="default.iir" />
        <ListPreference
               android:key="@string/geometry_file_preference"
               android:title="@string/title_gf_preference"
               android:dialogTitle="@string/title_gf_preference"
               android:defaultValue="@string/default_geometry_file" />
    </PreferenceCategory>
    <PreferenceCategory
             android:key="@string/hidden_prefs">
      <Preference
                   android:key="@string/lat_preference" />
             <Preference
                   android:key="@string/lon_preference" />
      <Preference
                   android:key="@string/zoom_preference" />

    </PreferenceCategory>
</PreferenceScreen>
```

Java Code

Package de.berlin.magun.protocol

IIIUri.java

```java
/*
 *    Copyright 2011 by the MAGUN project
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.berlin.magun.protocol;

import java.text.ParseException;

/**
 * Class representing a iii URI.
 * @author Johannes Bolz
 *
 */
public class IIIUri {
  private Source source;
  private Target target;
  private Technology technology;

  /**
   * @param uri the URI string.
   * @throws ParseException if the URI doesn't conform to the iii URI syntax specification.
   */
  public IIIUri(String uri) throws ParseException {
    // check if the URI is conform to the specification
    checkConformity(uri);

    // parse URI into objects, respect escaped characters
    String[] substrings = uri.replace("\\,", " ").split("," , -1);
    String targetname = substrings[0].replace("iii://", "");
    String sourcename = substrings[1];
    String technologyname = substrings[2];


    if (targetname.length() > 0) {
      this.target = new Target(targetname.replace(" ", ","));
    }
    if (sourcename.length() > 0) {
      this.source = new Source(sourcename.replace(" ", ","));
    }

    if (technologyname.length() > 0) {
      this.technology = new Technology(technologyname.replace(" ", ","));
    }


  }

  public void setSource(Source source) {
    this.source = source;
  }
  public Source getSource() {
    return source;
  }

  public void setTarget(Target target) {
    this.target = target;
  }

  public Target getTarget() {
    return target;
  }

  public void setTechnology(Technology technology) {
    this.technology = technology;
  }

  public Technology getTechnology() {
    return technology;
  }

  /**
   * Checks URI conformity to the definition
   * @param uri
   * @throws ParseException
   */
  private void checkConformity(String uri) throws ParseException {

    if (!uri.startsWith("iii://")) {
      throw new ParseException("Invalid syntax.", 0);
    }

    if (uri.contains(" ")) {
      throw new ParseException("URI must not contain white spaces.", 0);
    }

    uri  = uri.replace("iii://", "").replace("\\,", " ");

    //    "bla,bla","bla","bkl"bki"

    int commacount = 0;
    for (int i = 0; i < uri.length(); i++) {
      if (uri.substring(i, i + 1).equals(",")) {
        commacount++;
      }
    }
    if (commacount != 2) {
      throw new ParseException("Wrong number of separators or incorrectly escaped
characters.", 0);
    }
  }

}
```

Source.java

```
/*
 *    Copyright 2011 by the MAGUN project
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */
```

```java
package de.berlin.magun.protocol;

/**
 * Representation of the source (trigger) part of a iii URI.
 * @author Johannes Bolz
 *
 */
public class Source {
  private String content;
  private String scheme;
  private String value;

  /**
   * @param content the part of the iii URI describing the source (trigger)
   */
  public Source(String content) {
    this.content = content;
    this.scheme = content.substring(0, content.indexOf(":"));
    this.value = content.substring(content.indexOf(":") + 1);
    if (this.value.startsWith("//")) {
      this.value = this.value.substring(2);
    }
  }

  public String getUri() {
    return this.content;
  }

  public String getSchema() {
    return this.scheme;
  }

  public String getValue() {
    return this.value;
  }


}
```

Target.java

```java
/*
 *    Copyright 2011 by the MAGUN project
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.berlin.magun.protocol;

/**
 * Representation of the target part of a iii URI
 * @author Johannes Bolz
 *
 */
public class Target {
  public static final String FTP = "FTP";
  public static final String HTTP = "HTTP";

  private String schema;
  private String domain;
  private String path;
  private String url;

  /**
   * @param content the target part of a iii URI
   */
  public Target(String content) {
    this.url = content;
    this.schema = content.substring(0, content.indexOf(":"));

    if (content.contains("//")) {
      this.domain = content.substring(content.indexOf("//") + 2,
          content.indexOf("/", content.indexOf("//") + 2));
      this.path = content.substring(content.indexOf(this.domain) + this.domain.length());
    }
  }

  public String getUrl() {
    return this.url;
  }

  public String getSchema() {
    return this.schema;
  }

  public String getDomain() {
    return this.domain;
  }

  public String getPath() {
    return this.path;
  }
}
```

Technology.java

```java
/*
 *    Copyright 2011 by the MAGUN project
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.berlin.magun.protocol;

/**
 * Representation of the technology part of a iii URI
 * @author Johannes Bolz
 *
 */
public class Technology {

  public static final String WIFI = "WIFI";
  public static final String WPA = "WPA";
  public static final String WEP = "WEP";
  public static final String OPEN = "OPEN";

  private String technology;
  private String networkid;
  private String encryptiontype;
  private String password;

  /**
```

```
   * @param content the technology part of a iii URI
   */
  public Technology(String content) {
    // Syntax: WIFI:T:WPA;S:mynetwork;P:mypass;;

    if (content.toUpperCase().startsWith("WIFI")) {
      this.technology = WIFI;
      this.networkid = content.substring(content.indexOf("S:") + 2,
          content.indexOf(";", content.indexOf("S:") + 2));

      String encryption = content.substring(content.indexOf("T:") + 2,
          content.indexOf(";", content.indexOf("T:") + 2));

      if (encryption.equalsIgnoreCase("nopass")) {
        this.encryptiontype = OPEN;
      } else if (encryption.equalsIgnoreCase("WEP")) {
        this.encryptiontype = WEP;
        this.password = content.substring(content.indexOf("P:") + 2,
            content.indexOf(";", content.indexOf("P:") + 2));
      } else if (encryption.equalsIgnoreCase("WPA")) {
        this.encryptiontype = WPA;
        this.password = content.substring(content.indexOf("P:") + 2,
            content.indexOf(";", content.indexOf("P:") + 2));
      }

    }
  }

  public String getTechnology() {
    return this.technology;
  }

  public String getNetworkId() {
    return this.networkid;
  }

  public String getEncryptionType() {
    return this.encryptiontype;
  }

  public String getPassword() {
    return this.password;
  }

}
```

Package de.bolz.taglocate.geom

Coordinates.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.geom;
```

```java
/**
 * This class represents a geographic location, holding latitide, longitude and altitude
information.
 * @author Johannes Bolz
 */
public class Coordinates {
  private double lat;
  private double lon;
  private String alt;

  /**
   * @param lat latitude in decimal degrees
   * @param lon longitude in decimal degrees
   * @param alt altitude, can be in any appropriate format.
   */
  public Coordinates(double lat, double lon, String alt) {
    this.lat = lat;
    this.lon = lon;
    this.alt = alt;
  }

  public double getLat() {
    return lat;
  }

  public double getLon() {
    return lon;
  }

  public String getAlt() {
    return alt;
  }


}
```

Geometry.java

```java
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.geom;

import java.util.List;

/**
 * This class represents a polygon storing a set of {@link Coordinates} objects.
 * @author Johannes Bolz
 *
 */
public class Geometry {
  private List<Coordinates> points;
  private long id;

  /**
   * @param points List of {@link Coordinates} objects representing nodes of the geometry.
   * @param id unique id
```

```
    */
  public Geometry(List<Coordinates> points, long id) {
    this.id = id;
    this.points = points;
  }

  public List<Coordinates> getPoints() {
    return points;
  }

  public long getId() {
    return id;
  }

}
```

OsmParser.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.geom;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.apache.commons.io.IOUtils;
import org.apache.http.util.EncodingUtils;
import org.simpleframework.xml.Serializer;
import org.simpleframework.xml.core.Persister;

import android.content.res.Resources.NotFoundException;
import de.bolz.android.taglocate.geom.data.Nd;
import de.bolz.android.taglocate.geom.data.Node;
import de.bolz.android.taglocate.geom.data.Osm;
import de.bolz.android.taglocate.geom.data.Way;

/**
 * This class provides functionality to parse osm files into objects and to extract
 * {@link Geometry}s, {@link Way}s and {@link Node}s.
 * @author Johannes Bolz
 *
 */
public class OsmParser {
  private Osm osm;

  /**
   * @param f The osm file to be parsed.
   */
  public OsmParser(File f) {
```

```java
  parse(f);
}

/**
 * @return an {@link ArrayList} of {@link Geometry}s created from the way elements
 * within the osm file.
 */
public ArrayList<Geometry> getGeometries() {

  Map<String,Node> nodes = getNodeMap();

  List<Geometry> geometries = new ArrayList<Geometry>();
  // Extract Way objects:
  List<Way> ways = osm.getWaylist();

  // Iterate through ways:
  for (int i = 0; i < ways.size(); i++) {

    // Get Nd objects representing osm nd elements:
    List<Nd> nds = ways.get(i).getNdlist();
    List<Coordinates> coords = new ArrayList<Coordinates>();

    // Create Coordinates objects out of each node element corresponding to
    // the way's nd element:
    for (int j = 0; j < nds.size(); j++) {
      double lat = nodes.get(String.valueOf(nds.get(j).getRef())).getLat();
      double lon = nodes.get(String.valueOf(nds.get(j).getRef())).getLon();
      coords.add(new Coordinates(lat, lon, null));
    }
    geometries.add(new Geometry(coords, ways.get(i).getId()));
  }
  return (ArrayList<Geometry>) geometries;
}

/**
 * @return a HashMap with {@link Node} objects as values and their OSM IDs (String) as
 * keys.
 */
public HashMap<String, Node> getNodeMap() {
  Map<String,Node> nodemap = new HashMap<String, Node>();
  List<Node> nodelist = osm.getNodelist();
  for (int i = 0; i < nodelist.size(); i++) {
    nodemap.put(String.valueOf(nodelist.get(i).getId()),
        nodelist.get(i));
  }
  return (HashMap<String, Node>) nodemap;
}

/**
 * @return a HashMap eith {@link Way} objects as values and their OSM IDs (String) as
 * keys.
 */
public HashMap<String, Way> getWayMap() {
  Map<String, Way> waymap = new HashMap<String, Way>();
  List<Way> waylist = osm.getWaylist();
  for (int i = 0; i < waylist.size(); i++) {
    waymap.put(String.valueOf(waylist.get(i).getId()),
        waylist.get(i));
  }
  return (HashMap<String, Way>) waymap;
}

/**
 * Deserializes an OSM file into an {@link Osm} object.
 * @param f the OSM file
 */
private void parse(File f) {
      String input = "";
      try {
        // Make sure to pass the file's content to the Serializer with UTF-8 encoding:
    InputStream is = new FileInputStream(f);
    byte[] ba = IOUtils.toByteArray(is);
```

```
      input = EncodingUtils.getString(ba, "UTF-8");
    } catch (FileNotFoundException e1) {
      e1.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    }

    // Deserialize:
    Serializer serial = new Persister();
    try {
      osm = serial.read(Osm.class, input);
    } catch (NotFoundException e) {
      e.printStackTrace();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Package de.bolz.taglocate.geom.data

Nd.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.geom.data;

import org.simpleframework.xml.Attribute;

/**
 * Class representation of a 'nd' element within an OSM file
 * @author Johannes Bolz
 *
 */
public class Nd {

  @Attribute
  private long ref;

  public void setRef(long ref) {
    this.ref = ref;
  }

  public long getRef() {
    return ref;
  }
}
```

Node.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.geom.data;

import java.util.List;

import org.simpleframework.xml.Attribute;
import org.simpleframework.xml.ElementList;
import org.simpleframework.xml.Root;

/**
 * Class representation of a 'node' element within an OSM file.
 * @author Johannes Bolz
 *
 */
@Root(name="node")
public class Node {

  @Attribute
  private long id;

  @Attribute
  private double lat;

  @Attribute
  private double lon;

  @Attribute(required=false)
  private boolean visible;

  @Attribute(required=false)
  private String action;

  @ElementList(required=false, inline=true)
  private List<Tag> taglist;

  public long getId() {
    return id;
  }

  public void setId(long id) {
    this.id = id;
  }

  public double getLat() {
    return lat;
  }

  public void setLat(double lat) {
    this.lat = lat;
  }

  public double getLon() {
    return lon;
  }

  public void setLon(double lon) {
    this.lon = lon;
  }
  public boolean isVisible() {
    return visible;
  }

  public void setVisible(boolean visible) {
    this.visible = visible;
  }

  public void setTaglist(List<Tag> taglist) {
    this.taglist = taglist;
  }

  public List<Tag> getTaglist() {
    return taglist;
  }

  public String getAction() {
    return action;
  }

  public void setAction(String action) {
    this.action = action;
  }


}
```

Osm.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.geom.data;

import java.util.List;

import org.simpleframework.xml.Attribute;
import org.simpleframework.xml.ElementList;
import org.simpleframework.xml.Root;


/**
 * Class representation of the 'osm' root element within an OSM file.
 * @author Johannes Bolz
 *
 */
@Root(name="osm")
public class Osm {

  @Attribute(required=false, name="version")
  private String version;

  @Attribute(required=false, name="generator")
  private String generator;

  @ElementList(required=false, inline=true)
  private List<Node> nodelist;
  @ElementList(required=false, inline=true)
  private List<Way> waylist;

  @ElementList(required=false, inline=true)
  private List<Relation> relationlist;

  public Osm() {
    super();
  }

  public String getVersion() {
    return version;
  }

  public void setVersion(String version) {
    this.version = version;
  }

  public String getGenerator() {
    return generator;
  }

  public void setGenerator(String generator) {
    this.generator = generator;
  }

  public List<Node> getNodelist() {
    return nodelist;
  }

  public void setNodelist(List<Node> nodelist) {
    this.nodelist = nodelist;
  }

  public List<Way> getWaylist() {
    return waylist;
  }

  public void setWaylist(List<Way> waylist) {
    this.waylist = waylist;
  }

  public List<Relation> getRelationlist() {
    return relationlist;
  }

  public void setRelationlist(List<Relation> relationlist) {
    this.relationlist = relationlist;
  }

}
```

Relation.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */
```

```
package de.bolz.android.taglocate.geom.data;

/**
 * Class representation of a 'relation' element within an OSM file
 * @author Johannes Bolz
 *
 */
public class Relation {
  // TODO: add corresponding objects for 'relation' element contents
}
```

Tag.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.geom.data;

import org.simpleframework.xml.Attribute;

/**
 * Class representation of a 'tag' element within an OSM file.
 * @author Johannes Bolz
 *
 */
public class Tag {

  @Attribute
  private String k;

  @Attribute
  private String v;

  public String getK() {
    return k;
  }

  public void setK(String k) {
    this.k = k;
  }

  public String getV() {
    return v;
  }

  public void setV(String v) {
    this.v = v;
  }

}
```

Way.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.geom.data;

import java.util.List;

import org.simpleframework.xml.Attribute;
import org.simpleframework.xml.ElementList;

/**
 * Class representation of a 'way' element within an OSM file.
 * @author Johannes Bolz
 *
 */
public class Way {

  @Attribute
  private long id;

  @Attribute(required=false)
  private boolean visible;

  @Attribute(required=false)
  private String action;

  @ElementList(inline=true)
  private List<Nd> ndlist;

  @ElementList(required=false, inline=true)
  private List<Tag> taglist;

  public long getId() {
    return id;
  }

  public void setId(long id) {
    this.id = id;
  }

  public boolean isVisible() {
    return visible;
  }

  public void setVisible(boolean visible) {
    this.visible = visible;
  }

  public String getAction() {
    return action;
  }

  public void setAction(String action) {
    this.action = action;
  }

  public List<Nd> getNdlist() {
    return ndlist;
  }
  public void setNdlist(List<Nd> ndlist) {
    this.ndlist = ndlist;
  }

  public List<Tag> getTaglist() {
    return taglist;
  }

  public void setTaglist(List<Tag> taglist) {
    this.taglist = taglist;
  }

}
```

Package de.bolz.taglocate.olviewer

JSComm.java

```
/*
 * This class was part of an attempt to use OpenLayers mobile together with PhoneGap as map
 * renderer. It is being preserved for documentation purposes.
 */

//package de.bolz.android.taglocate.olviewer;

//import java.util.ArrayList;
//import java.util.List;
//
//import com.phonegap.CallbackServer;
//
//import de.bolz.android.taglocate.geom.Coordinates;
//import de.bolz.android.taglocate.geom.Geometry;

//@Deprecated
//public class JSComm {
//  private CallbackServer callbackServer;
//
//  protected JSComm(CallbackServer cs) {
//    this.callbackServer = cs;
//  }
//
//  protected void drawMarker(String lon, String lat, String srs) {
//    callbackServer.sendJavascript((new StringBuilder("drawMarker(").append(
//        lon).
//        append(", ").
//        append(lat).
//        append(", \"").
//        append(srs).
//        append("\");")).toString());
//  }
//
//  protected void zoomTo(String lon, String lat, String srs, String zoomlevel) {
//    callbackServer.sendJavascript((new StringBuilder("zoomTo(").append(
//        lon).
//        append(", ").
//        append(lat).
//        append(", \"").
//        append(srs).
//        append("\", ").
//        append(zoomlevel).
//        append(");")).toString());
//  }
//
//  protected void clearMarkers() {
//    callbackServer.sendJavascript("clearMarkers();");
//  }
//
//  protected void drawGeometries() {
////  protected void drawGeometries(ArrayList<Geometry> geometries) {
```

```
////    for (int i = 0; i < geometries.size(); i++) {
////      StringBuilder sb = new StringBuilder();
////      sb.append("drawGeometry(\"");
////      List<Coordinates> coords = geometries.get(i).getPoints();
////
////      for (int j = 0; j < coords.size(); j++) {
////        Coordinates c = coords.get(j);
////        sb.append(c.getLon()).append(",")
////          .append(c.getLat()).append(";");
////      }
////      sb.append("\");");
////      String s = sb.toString();
////      callbackServer.sendJavascript(sb.toString());
////    }
//
callbackServer.sendJavascript("drawGeometry(\"13.41925811021193,52.48896316877748;13.419255615
1661,52.488937379946265;13.419165924755664,52.48894253890429;13.419168978011534,52.48896697372
756;13.41925811021193,52.48896316877748\");");
//
//
//  }


//}
```

MapActivity.java

```
/*
 * This class was part of an attempt to use OpenLayers mobile together with PhoneGap as map
 * renderer. It is being preserved for documentation purposes.
 */

//package de.bolz.android.taglocate.olviewer;

//import java.io.File;
//import java.io.FileInputStream;
//import java.io.FileNotFoundException;
//import java.io.IOException;
//import java.io.InputStream;
//import java.nio.charset.Charset;
//import java.util.List;
//
//import org.apache.commons.io.IOUtils;
//import org.simpleframework.xml.Serializer;
//import org.simpleframework.xml.core.Persister;
//
//import com.phonegap.DroidGap;
//
//import de.berlin.magun.nfcmime.core.NdefMimeRecord;
//import de.berlin.magun.nfcmime.core.RfidDAO;
//import de.bolz.android.taglocate.geom.data.Osm;
//import de.bolz.android.taglocate.geom.OsmParser;
//import android.app.PendingIntent;
//import android.content.Intent;
//import android.content.IntentFilter;
//import android.content.SharedPreferences;
//import android.content.res.Resources.NotFoundException;
//import android.nfc.NfcAdapter;
//import android.nfc.Tag;
//import android.nfc.tech.Ndef;
//import android.os.Bundle;
//import android.util.Log;

//@Deprecated
//public class MapActivity extends DroidGap {
//  private NfcAdapter adapter;
//  private PendingIntent pi;
//  private IntentFilter[] filters;
//  private String[][] techLists;
//  private String x, y;
//  private final String X_KEY = "xValue";
//  private final String Y_KEY = "yValue";
```

```
//
//
//    /** Called when the activity is first created. */
//    @Override
//    public void onCreate(Bundle savedInstanceState) {
//        super.onCreate(savedInstanceState);
//        SharedPreferences prefs = getPreferences(MODE_PRIVATE);
//        this.x = prefs.getString(X_KEY, "0");
//        this.y = prefs.getString(Y_KEY, "0");
//
//
//      OsmParser p = new OsmParser(new File("/mnt/sdcard/test.osm"));
////      comm.drawGeometries(p.getGeometries());
//
//        super.loadUrl("file:///android_asset/www/map.html");
//        JSComm comm = new JSComm(this.callbackServer);
//        this.callbackServer.sendJavascript("reloadPage();");
//      comm.clearMarkers();
//      comm.drawMarker(this.x, this.y, "EPSG:4326");
//      comm.zoomTo(this.x, this.y, "EPSG:4326", "16");
//
//      comm.drawGeometries();
//
//        adapter = NfcAdapter.getDefaultAdapter(this);
//        pi = PendingIntent.getActivity(this, 0,
//                new Intent(this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
//        IntentFilter filter = new IntentFilter(NfcAdapter.ACTION_TAG_DISCOVERED);
//        filters = new IntentFilter[] {filter,};
//        techLists = new String[1][1];
//        techLists[0][0] = Ndef.class.getName();
//
//        }
//
//    @Override
//    public void onResume() {
//        super.onResume();
//        adapter.enableForegroundDispatch(this, pi, filters, techLists);
//    }
//
//    @Override
//    protected void onNewIntent(Intent intent) {
//      super.onNewIntent(intent);
//
//      Tag tag = (Tag) intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
//      RfidDAO dao = new RfidDAO();
//      List<NdefMimeRecord> recordlist = dao.getCachedMimeRecords(tag);
//
//      if (recordlist != null) {
//        NdefMimeRecord coordsrec = recordlist.get(0);
//          if (coordsrec.getMimeString().equals("test/coordinates")) {
//            String coords = new String(coordsrec.getPayload(), Charset.forName("UTF-8"));
//            this.x = coords.substring(0, coords.indexOf(","));
//            this.y = coords.substring(coords.indexOf(",") + 1, coords.length());
//            JSComm comm = new JSComm(this.callbackServer);
//              comm.clearMarkers();
//              comm.drawMarker(this.x, this.y, "EPSG:4326");
//              comm.zoomTo(this.x, this.y, "EPSG:4326", "16");
//          }
//      }
//    }
//
//    @Override
//    public void onPause() {
//        super.onPause();
//
//        SharedPreferences prefs = getPreferences(MODE_PRIVATE);
//        SharedPreferences.Editor ed = prefs.edit();
//        ed.putString(X_KEY, this.x);
//        ed.putString(Y_KEY, this.y);
//        ed.commit();
//
//        adapter.disableForegroundDispatch(this);
//    }
//
////    @Override
////    public boolean onCreateOptionsMenu(Menu menu) {
////        MenuInflater inflater = getMenuInflater();
////        inflater.inflate(R.menu.map_menu, menu);
////        return true;
////    }
////
////    @Override
////    public boolean onOptionsItemSelected(MenuItem item) {
////        // Handle item selection
////        switch (item.getItemId()) {
////        case R.id.item1: {
////          Intent intent = new Intent("com.google.zxing.client.android.SCAN");
////            intent.setPackage("com.google.zxing.client.android");
////            intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
////            startActivityForResult(null, intent, 0);
////            return true;
////        }
////        default: {
////            return super.onOptionsItemSelected(item);
////        }
////        }
////    }
////
////    @Override
////    public void onActivityResult(int requestCode, int resultCode, Intent intent) {
////        if (requestCode == 0) {
////            if (resultCode == RESULT_OK) {
////                String contents = intent.getStringExtra("SCAN_RESULT");
////                String format = intent.getStringExtra("SCAN_RESULT_FORMAT");
////                // geo:52.520839,13.409413
////                if (contents != null && format != null) {
////                  if (contents.startsWith("geo:")) {
////                    this.y = contents.substring(4, contents.indexOf(","));
////                    this.x = contents.substring(contents.indexOf(",") + 1);
////                    JSComm comm = new JSComm(this.callbackServer);
////                      comm.clearMarkers();
////                      comm.drawMarker(this.x, this.y, "EPSG:4326");
////                      comm.zoomTo(this.x, this.y, "EPSG:4326", "16");
////                  }
////                }
////            } else if (resultCode == RESULT_CANCELED) {
////                // Handle cancel
////            }
////        }
////    }
//}
```

Package de.bolz.taglocate.protocol

GeoUri.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.protocol;

import de.bolz.android.taglocate.geom.Coordinates;

/**
 *
 * Class representing a Geo-URI string. (Syntax: geo:latitude,longitude[,altitude][;u=u]
[;crs=crs])
 * @author Johannes Bolz
 *
 */
public class GeoUri {

  private Coordinates coordinate;
  private String string;

  /**
   * @param uri URI string. Syntax: <i>geo:latitude,longitude[,altitude][;u=u][;crs=crs]</i>
   */
  public GeoUri(String uri) {
    this.string = uri;
    String data = new String(uri.substring(4));

    if(data.startsWith("osm")) {
      // TODO: OSM query handling: At a later point, query for certain OSM tags / attributes
      // will be implemented.
    } else {
      // parse URI String into objects
      String[] components = data.split(";")[0].split(",");
      String latStr = new String(components[0]);
      String lonStr = new String(components[1]);
      String height = null;
      if (components.length == 3) {
        height = components[2];
      }
      // build coordinate
      coordinate = new Coordinates(Double.parseDouble(latStr),
          Double.parseDouble(lonStr), height);
    }
  }

  /**
   * @param lat Latitude in decimal degrees
   * @param lon Longitude in decimal degrees
   */
  public GeoUri(double lat, double lon) {
    this.string = new StringBuilder("geo:").
    append(String.valueOf(lat)).
    append(",").
    append(String.valueOf(lon)).
    toString();
  }

  /**
   * @return the Coordinates specified in the Geo-URI.
   */
  public Coordinates getCoordinate() {
    return this.coordinate;
  }

  /**
   * @return the complete Geo-URI
   */
  public String getString() {
    return string;
  }
}
```

IntentResolver.java

```java
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 *
 */

package de.bolz.android.taglocate.protocol;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.text.ParseException;
import java.util.List;

import org.apache.commons.io.IOUtils;
import org.apache.http.util.EncodingUtils;
import org.simpleframework.xml.Serializer;
import org.simpleframework.xml.core.Persister;
import org.simpleframework.xml.stream.CamelCaseStyle;
import org.simpleframework.xml.stream.Format;

import de.berlin.magun.protocol.IIIUri;
import de.berlin.magun.protocol.Source;
import de.bolz.android.taglocate.geom.Coordinates;
import de.bolz.android.taglocate.protocol.data.Reference;
import de.bolz.android.taglocate.protocol.data.ReferenceList;
import android.content.Context;
import android.content.Intent;
import android.nfc.NfcAdapter;
import android.os.Build;
import android.view.Gravity;
import android.widget.Toast;

/**
 * This class provides functionality to resolve intents resulting from NFC or QR read events
and
 * to extract location information out of their contents.
 * @author Johannes Bolz
 */
public class IntentResolver {

  /*
   * SCENARIOS:
   * A) URI:
   *   1. Geo-URI
   *   2. III-URI
   * B) UID
   * C) DATA
   *   1. geometry file
   *   2. link file
   */


  // RESOLUTION STRATEGY KEYS:
  /**
```

```
   * The NFC tag's unique ID will be matched against the ID reference table
   * first. If the ID isn't associated with a location, the tag's content will
   * be checked for location information. In case of a QR tag, the tag's content
   * will first be looked up in the reference table (i.e. treated as an ID) and
   * then further evaluated.
   */
  public static final String UID_FIRST = "uid_f";

  /**
   * The NFC tag's unique ID will only be matched against the ID reference table.
   * A QR tag's content will be treated as an ID.
   */
  public static final String UID_ONLY = "uid_o";

  /**
   * The NFC tag will be checked for a location link first. If it doesn't contain
   * an explicit location link, it's unique ID will be matched against the reference
   * table.
   * A QR code will be first checked for a link and then be treated as an ID.
   */
  public static final String URI_FIRST = "uri_f";

  /**
   * The NFC tag or QR code will only be checked for a location link.
   */
  public static final String URI_ONLY = "uri_o";

  private String geoUriStr;
  private String iiiUriStr;
  private Context context;

  private Coordinates location;

  private String id;

  /**
   * @param context the {@link Context} of the calling class
   * @param intent the {@link Intent} to be examined. Must result either from a NFC or from
   * a QR read event.
   */
  public IntentResolver(Context context, Intent intent) {
    this.context = context;

    // Check for NFC intents only if device / Android version supports NFC:
    if (hasNfcSupport()) {
      NfcIntentResolver resolver = new NfcIntentResolver(context, intent);
      this.id = resolver.getId();
      this.iiiUriStr = resolver.getIiiUriStr();
      this.geoUriStr = resolver.getGeoUriStr();
    }

    // Resolve QR intent:
    if (intent.getAction().equals("com.google.zxing.client.android.SCAN")) {
      String result = intent.getStringExtra("SCAN_RESULT");
      this.id = result;

      // geo-URI
      if (result.startsWith("geo")) {
        geoUriStr = result;
      }

      // iii-URI
      if (result.startsWith("iii")) {
        iiiUriStr = result;
      }

    }

    // resolve
    resolveData(SettingsSingleton.getInstance().getStrategy());
  }
```

```
/**
 * Extracts location information from the intent according to the specified
 * link resolution strategy.
 * @param strategy the link resolution strategy.
 */
private void resolveData(String strategy) {

  if (strategy.equals(URI_ONLY)) {
    // Only resolve URI, if available:
    resolveUri();
    if (this.location != null) {
      showUriToast();
    }
  } else if (strategy.equals(UID_ONLY)) {
    // Only match ID:
    resolveUid();
    if (this.location != null) {
      showUidToast();
    }
  } else if (strategy.equals(URI_FIRST)) {
    // Try to resolve URI first. If no location can be extracted,
    // try ID matching:
    resolveUri();
    if (this.location != null) {
      showUriToast();
    } else {
      resolveUid();
      if (this.location != null) {
        showUidToast();
      }
    }
  } else if (strategy.equals(UID_FIRST)) {
    // Try to match ID first. If no location can be deducted, try
    // looking for an explicitly linked location:
    resolveUid();
    if (this.location != null) {
      showUidToast();
    } else {
      resolveUri();
      if (this.location != null) {
        showUriToast();
      }
    }
  }
}

/**
 * Determines whether a URI is a Geo-URI or a iii-URI and dispatches it
 * to the according URI resolution method.
 */
private void resolveUri() {
  if (this.geoUriStr != null) {
    resolveGeoUri(this.geoUriStr);
  }
  if (this.iiiUriStr != null) {
    resolveIIIUri(this.iiiUriStr);
  }
}

/**
 * Creates a {@link Coordinates} object from a Geo-URI
 * @param uri Geo-URI
 */
private void resolveGeoUri(String uri) {
  GeoUri geoUri = new GeoUri(uri);
  this.location = geoUri.getCoordinate();
}

/**
 * Creates a {@link Coordinates} object from a iii-URI, if it's target
 * element is a Geo-URI.
 * @param uri iii-URI
 */
private void resolveIIIUri(String uri) {
  IIIUri iiiUri = null;
  try {
    iiiUri = new IIIUri(uri);
  } catch (ParseException e) {
    e.printStackTrace();
  }
  if (iiiUri != null) {
    if (iiiUri.getTarget().getSchema().equals("geo")) {
      resolveGeoUri(iiiUri.getTarget().getUrl());
    }
  }
}

/**
 * Matches a NFC tag's ID or a Qr code's content against the reference model
 * stored in the IIR reference file.
 */
private void resolveUid() {

  // Deserialize link file:
  File referencefile = new File(SettingsSingleton.getInstance().getDatapath() +
      SettingsSingleton.getInstance().getDatafile());
  InputStream is;
  ReferenceList list = null;
  String input = "";
  try {

    // Make sure file's content is decoded using UTF-8:
    is = new FileInputStream(referencefile);
    byte[] ba = IOUtils.toByteArray(is);
    input = EncodingUtils.getString(ba, "UTF-8");

    // Create a serializer that adheres to the XML's camel case element style:
    Serializer s = new Persister(new Format(new CamelCaseStyle()));
    list = s.read(ReferenceList.class, input);
    if (list != null) {
      List<Reference> links = list.getReferences();
      String trigger;

      // Match ID against each item in the link list (=reference table):
      for (int i = 0; i < links.size(); i++) {
        trigger = (new Source(links.get(i).getTrigger().getTag().getTagStr()))
          .getValue();
        if (this.id.equalsIgnoreCase(trigger)) {
          resolveGeoUri(links.get(i).getTarget().getTargetStr());
        }

      }
    }
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  } catch (Exception e) {
    e.printStackTrace();
  }
}


/**
 * Checks if the device supports NFC and the Android version is 2.3.3 or higher.
 * @return true, if both conditions apply.
 */
private boolean hasNfcSupport() {
  return (Build.VERSION.SDK_INT >= 10 &&
      NfcAdapter.getDefaultAdapter(context) != null);
}

/**
 * Shows a Toast saying that a valid location link was found on the tag.
```

```java
   */
  private void showUriToast() {
    Toast t = Toast.makeText(context, "link", Toast.LENGTH_SHORT);
    t.setGravity(Gravity.TOP, 0, 0);
    t.show();
  }

  /**
   * Shows a toast saying that a location was found by matching an ID against
   * the reference table.
   */
  private void showUidToast() {
    Toast t = Toast.makeText(context, "UID: " + this.id.toUpperCase(), Toast.LENGTH_SHORT);
    t.setGravity(Gravity.TOP, 0, 0);
    t.show();
  }


  /**
   * @return the resolved location.
   */
  public Coordinates getLocation() {
    return this.location;
  }

}
```

NfcIntentResolver.java

```java
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.protocol;

import java.io.IOException;
import java.util.List;

import android.content.Context;
import android.content.Intent;
import android.nfc.FormatException;
import android.nfc.NfcAdapter;
import android.nfc.Tag;
import android.view.Gravity;
import android.widget.Toast;
import de.berlin.magun.nfcmime.core.NdefMimeRecord;
import de.berlin.magun.nfcmime.core.RfidDAO;
import de.berlin.magun.nfcmime.core.ZipFileSystem;
import de.bolz.android.taglocate.R;

/**
 * This class provides functionality to resolve intents resulting from a NFC read
 * process. It shall only be instantiated on devices supporting NFC and running Android
 * 2.3.3 or higher.
 * @author Johannes Bolz
 *
 */
public class NfcIntentResolver {
  private RfidDAO dao;
  private String id;
  private String scheme;
  private String geoUriStr;
  private String iiiUriStr;
  private ZipFileSystem zfs;
  private Context context;

  /**
   * @param context the {@link Context} of the calling class
   * @param intent the {@link Intent} to be examined. Must result either from a NFC
   * read event.
   */
  protected NfcIntentResolver(Context context, Intent intent) {
    this.context = context;
    parseNfcIntent(intent);
  }

  /**
   * Tries to get location information from the NFC tag and checks the tag for
   * contained files.
   * @param intent the {@link Intent} to be examined. Must result either from a NFC
   * read event.
   */
  private void parseNfcIntent(Intent intent) {

    if(intent.getAction().equals(NfcAdapter.ACTION_NDEF_DISCOVERED) ||
        intent.getAction().equals(NfcAdapter.ACTION_TECH_DISCOVERED)) {

      // All NFC intents:
      Tag tag = (Tag) intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
      dao = new RfidDAO();

      // Read tag ID:
      id = dao.getTagId(tag);

      // Extract payload data (files):
      if (SettingsSingleton.getInstance().isDownloadFromNfc()) {
        getFilesFromNfc(tag);
      }

      // Try to get links from NDEF intents:
      if (intent.getAction().equals(NfcAdapter.ACTION_NDEF_DISCOVERED)) {
        scheme = intent.getScheme();
        if(scheme != null) {

          // geo-URI
          if (scheme.equals("geo")) {
            geoUriStr = intent.getDataString();
          }

          // iii-URI
          if (scheme.equals("iii")) {
            iiiUriStr = intent.getDataString();
          }
        }

      }
    }
  }

  /**
   * Tries to extract geometry and link files from the NFC tag and to write them to the
   * local file system. <br>
   * To store files, create a NDEF record containing a ZIP-compressed dataset
   * containing the files, and write it to the tag specifying the MIME type 'application/zip'.
<br>
   * This function will also activate the loaded files in the application.
   * @param tag the {@link Tag} containing a zipped file dataset
   */
  private void getFilesFromNfc(Tag tag) {
    List<NdefMimeRecord> recordList = dao.getCachedMimeRecords(tag);
    if (recordList != null) {
      for (int i = 0; i < recordList.size(); i++) {
        if ("application/zip".equalsIgnoreCase(recordList.get(i).getMimeString())) {
          try {
            // Try to create a ZipFileSystem object out of the NDEF record:
            zfs = new ZipFileSystem(recordList.get(i));
            List<String> filenames = zfs.getFileNames();
            for (int j = 0; j < filenames.size(); j++) {
              String fname = filenames.get(j);
              // Write files to data directory:
              if (fname.endsWith(".iir") || fname.endsWith(".osm")) {
                zfs.write(SettingsSingleton.getInstance().getDatapath());
              }

              // Apply link file and show Toast:
              if (fname.endsWith("iir")) {
                SettingsSingleton.getInstance().setDatafile(fname);
                Toast t = Toast.makeText(context, context.getString(R.string.reffile_found),
Toast.LENGTH_SHORT);
                t.setGravity(Gravity.TOP, 0, 0);
                t.show();
              }

              // Apply geometry file and show Toast:
              if (fname.endsWith("osm")) {
                SettingsSingleton.getInstance().setGeometryfile(fname);
                Toast t = Toast.makeText(context, context.getString(R.string.geomfile_found),
Toast.LENGTH_SHORT);
                t.setGravity(Gravity.TOP, 0, 0);
                t.show();
              }
            }
          } catch (IOException e) {
            e.printStackTrace();
          } catch (FormatException e) {
            e.printStackTrace();
          }
        }
      }
    }

  }

  /**
   * @return the tag's ID
   */
  public String getId() {
    return id;
  }

  /**
   * @return the extracted URI's scheme
   */
  public String getScheme() {
    return scheme;
  }

  /**
   * @return the extracted Geo-URI as String
   */
  public String getGeoUriStr() {
    return geoUriStr;
  }

  /**
   * @return the extracted iii-URI as String
   */
  public String getIiiUriStr() {
    return iiiUriStr;
  }
}
```

SettingsSingleton.java

```java
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.protocol;

/**
 * This singleton class stores various global configuration values to make them concurrently
 * accessible from different classes during runtime and to avoid inconsistencies.
 * @author Johannes Bolz
 */
public final class SettingsSingleton {
  private static SettingsSingleton instance;

  private static String datapath;
  private static String datafile;
  private static String geometryfile;
  private static String strategy;
  private static boolean downloadFromNfc;

  /**
   * Keep constructor private.
   */
  private SettingsSingleton() {}

  /**
   * @return the instance of the singleton. If there is no instance yet, a new one will be
   * created.
   */
  public static synchronized SettingsSingleton getInstance() {
    if (instance == null) {
      instance = new SettingsSingleton();
    }
    return instance;
  }

  public String getDatapath() {
    return datapath;
  }

  public void setDatapath(String datapath) {
    SettingsSingleton.datapath = datapath;
  }

  public String getDatafile() {
    return datafile;
  }

  public void setDatafile(String datafile) {
    SettingsSingleton.datafile = datafile;
  }

  public String getGeometryfile() {
    return geometryfile;
  }
  public void setGeometryfile(String geometryfile) {
    SettingsSingleton.geometryfile = geometryfile;
  }

  public void setInstance(SettingsSingleton instance) {
    SettingsSingleton.instance = instance;
  }

  public String getStrategy() {
    return strategy;
  }

  public void setStrategy(String strategy) {
    SettingsSingleton.strategy = strategy;
  }

  public boolean isDownloadFromNfc() {
    return downloadFromNfc;
  }

  public void setDownloadFromNfc(boolean downloadFromNfc) {
    SettingsSingleton.downloadFromNfc = downloadFromNfc;
  }


}
```

Package de.bolz.taglocate.protocol.data

Reference.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.protocol.data;

import org.simpleframework.xml.Element;

/**
 * Class representation of a 'Reference' element within a IIR link file.
 * @author Johannes Bolz
 */
public class Reference{

  @Element(required = true, name = "Trigger")
  private Trigger trigger;

  @Element(required = true, name = "Target")
  private Target target;

  public Trigger getTrigger() {
    return trigger;
  }

  public void setTrigger(Trigger trigger) {
    this.trigger = trigger;
  }

  public Target getTarget() {
    return target;
  }

  public void setTarget(Target target) {
    this.target = target;
  }
}
```

ReferenceList.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.protocol.data;

import java.util.List;

import org.simpleframework.xml.ElementList;
import org.simpleframework.xml.Root;

/**
 * Class representation of the root 'ReferenceList' element within a IIR link file.
 * @author Johannes Bolz
 */
@Root
public class ReferenceList {

  @ElementList(required = false, inline = true, name = "Reference")
  private List<Reference> references;

  public ReferenceList() {
    super();
  }

  public List<Reference> getReferences() {
    return references;
  }

  public void setReferences(List<Reference> references) {
    this.references = references;
  }
}
```

Tag.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
```

```java
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */


package de.bolz.android.taglocate.protocol.data;

import org.simpleframework.xml.Text;

/**
 * Class representation of a 'Tag' element within an IIR reference file.
 * @author Johannes Bolz
 *
 */
public class Tag {

  @Text
  private String tagStr;

  public String getTagStr() {
    return tagStr;
  }

  public void setTagStr(String tagStr) {
    this.tagStr = tagStr;
  }
}
```

Target.java

```java
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.protocol.data;

import org.simpleframework.xml.Text;

/**
 * Class representation of a 'Target' element within an IIR reference file.
 * @author Johannes Bolz
 *
 */
public class Target {

  @Text
  private String targetStr;

  public String getTargetStr() {
    return targetStr;
  }
  public void setTargetStr(String targetStr) {
    this.targetStr = targetStr;
  }

}
```

Trigger.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.protocol.data;

import org.simpleframework.xml.Element;

/**
 * Class representation of a 'Trigger' element within an IIR reference file.
 * @author Johannes Bolz
 *
 */
public class Trigger {

  @Element(required = false, name = "Tag")
  private Tag tag;

  public Tag getTag() {
    return tag;
  }

  public void setTag(Tag tag) {
    this.tag = tag;
  }


}
```

Package de.bolz.taglocate.ui

EditDispatchActivity.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.ui;

import de.bolz.android.taglocate.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.DialogInterface.OnCancelListener;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.nfc.NfcAdapter;
import android.os.Build;
import android.os.Bundle;
import android.view.Gravity;
import android.view.Window;
import android.widget.Toast;

/**
 * This class is responsible for calling the right edit activities when referencing a new NFC
Tag
 * or QR code.
 * @author Johannes Bolz
 *
 */
public class EditDispatchActivity extends Activity {

  // Impossible coordinate value:
  public static final int COORDINATENOTSET = -2147483648;

  private Intent intent;
  private AlertDialog dialog;
  private int latInt;
  private int lonInt;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.intent = getIntent();

        // Get integer coordinates:
        this.latInt = this.intent.getIntExtra(getString(R.string.center_lat),
COORDINATENOTSET);
        this.lonInt = this.intent.getIntExtra(getString(R.string.center_lon),
COORDINATENOTSET);

        requestWindowFeature(Window.FEATURE_NO_TITLE);
    this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    showDialog();

    }


    /**
     * Creates a chooser dialog and starts edit activities according to
     * the item selected from the list: 'Write NFC tag', 'Reference NFC tag'
     * or 'Reference QR code'.
     */
  private void showDialog() {

    AlertDialog.Builder builder = new AlertDialog.Builder(
        EditDispatchActivity.this);
    builder.setItems(R.array.edit_dialog_items,
        new DialogInterface.OnClickListener() {
          public void onClick(DialogInterface dialog, int which) {

            // Write NFC tag:
            if (which == 0) {
              if(hasNfcSupport()) {
```

```
              Intent i = new Intent(EditDispatchActivity.this,
                  NfcEditActivity.class);
              // Request write mode, pass coordinates:
              i.putExtra(getString(R.string.nfc_edit_mode), NfcEditActivity.WRITE);
              i.putExtra(getString(R.string.center_lon),
                  EditDispatchActivity.this.lonInt);
              i.putExtra(getString(R.string.center_lat),
                  EditDispatchActivity.this.latInt);
              startActivity(i);
              EditDispatchActivity.this.kill();
            } else {
              showNfcNotSupportedToast();
              EditDispatchActivity.this.showDialog();
            }
          }

          // Reference NFC ID:
          else if (which == 1) {
            if(hasNfcSupport()) {
              Intent i = new Intent(EditDispatchActivity.this,
                  NfcEditActivity.class);
              // Request read mode, pass coordinates:
              i.putExtra(getString(R.string.nfc_edit_mode), NfcEditActivity.READ);
              i.putExtra(getString(R.string.center_lon),
                  EditDispatchActivity.this.lonInt);
              i.putExtra(getString(R.string.center_lat),
                  EditDispatchActivity.this.latInt);
              startActivity(i);
              EditDispatchActivity.this.kill();
            } else {
              showNfcNotSupportedToast();
              EditDispatchActivity.this.showDialog();
            }
          }

          // Reference QR code:
          else if (which == 2) {
            Intent i = new Intent(EditDispatchActivity.this,
                QrEditActivity.class);
            // Pass coordinates:
            i.putExtra(getString(R.string.center_lon),
                EditDispatchActivity.this.lonInt);
            i.putExtra(getString(R.string.center_lat),
                EditDispatchActivity.this.latInt);
            startActivity(i);
            EditDispatchActivity.this.kill();
          }

        }
      });

  builder.setOnCancelListener(new OnCancelListener() {
    @Override
    public void onCancel(DialogInterface dialog) {
      EditDispatchActivity.this.kill();
    }
  });

  dialog = builder.create();
  dialog.show();
}



/**
 * Activity will be finished when pressing the back button.
 */
@Override
  public void onBackPressed() {
    this.finish();
    this.onDestroy();
  }
```

```java
  /**
   * Kills the activity.
   */
    private void kill() {
      this.onPause();
      this.finish();
      this.onDestroy();
    }

    // TODO: Remove duplication
    /**
   * Checks for NFC support on the device.
   * @return true if the device hardware supports NFC and is running Android
   * 2.3.3 or later.
   */
  private boolean hasNfcSupport() {
    return (Build.VERSION.SDK_INT >= 10 && NfcAdapter
        .getDefaultAdapter(getApplicationContext()) != null);
  }

  /**
   * Shows a Toast stating the device doesn't support NFC.
   */
  private void showNfcNotSupportedToast() {
    Toast t = Toast.makeText(getApplicationContext(),
getString(R.string.message_nfc_not_supported),
        Toast.LENGTH_SHORT);
    t.setGravity(Gravity.TOP, 0, 0);
    t.show();
  }

}
```

EditModeOverlay.java

```java
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.ui;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import de.bolz.android.taglocate.R;

/**
 * This class is responsible for drawing a crosshairs marker in the view center.
 * @author Johannes Bolz
 */
public class EditModeOverlay extends Overlay {

  private Point point;
  private Context context;

  /**
   * @param context the context of the calling class (activity)
   */
  public EditModeOverlay(Context context) {
    this.context = context;
  }

  /**
   * Standard Overlay method.
   */
  public void draw(Canvas canvas, MapView view, boolean shadow) {
    super.draw(canvas, view, shadow);

    point = new Point(view.getWidth() / 2, view.getHeight() /2);

    // Draw marker on the map's center:
    Bitmap locationMarker = BitmapFactory.decodeResource(
        context.getResources(), R.drawable.crosshairs);
    canvas.drawBitmap(locationMarker, point.x
        - (locationMarker.getWidth() / 2),
        point.y - (locationMarker.getHeight() / 2), null);
  }

}
```

GeometriesSingleton.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.ui;

import java.util.List;

import com.google.android.maps.GeoPoint;

/**
 * Singleton class that stores a {@link List} of {@link GeoPoint} arrays. Each array
 * should represent a polygon geometry. It is intended to provide a static
 * set of created geometries during runtime, so the {@link GeoPoint}s need to
 * be created only once.
 * @author Johannes Bolz
 */
public final class GeometriesSingleton {
  private static GeometriesSingleton instance;
  private static List<GeoPoint[]> geometries;

  /**
   * private constructor to avoid instantiation from outside the class
   */
  private GeometriesSingleton() {}

  /**
   * @return the instance of the singleton. If no instance exists, a new one will be
   * created.
```

```java
   */
  public static synchronized GeometriesSingleton getInstance() {
    if (instance == null) {
      instance = new GeometriesSingleton();
    }
    return instance;
  }

  /**
   * @return a {@link List} of {@link GeoPoint} arrays. Each array shall hold the
   * GeoPoints needed to create a polygon geometry.
   */
  public List<GeoPoint[]> getGeometries() {
    return geometries;
  }

  /**
   * Stores the geometries.
   * @param geometries a {@link List} of {@link GeoPoint} arrays. Each array shall hold the
   * GeoPoints needed to create a polygon geometry.
   */
  public void setGeometries(List<GeoPoint[]> geometries) {
    GeometriesSingleton.geometries = geometries;
  }



}
```

GmapActivity.java

```java
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 *
 *  ## CREDITS: Some parts of this class were developed upon code examples  ##
 *  ## and / or snippets from the following sources:                        ##
 *
 *  NFC foreground dispatch, Android API Demos:
 *
http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/nfc/Foreg
roundDispatch.html
 *
 *  Creating a custom location provider, discussion on the Stack Overflow forum:
 *  http://stackoverflow.com/questions/2531317/android-mock-location-on-device
 *
 *
 */

package de.bolz.android.taglocate.ui;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.List;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;

import de.bolz.android.taglocate.R;
import de.bolz.android.taglocate.protocol.IntentResolver;
import de.bolz.android.taglocate.protocol.SettingsSingleton;

import android.app.AlertDialog;
import android.app.PendingIntent;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.nfc.NfcAdapter;
import android.nfc.tech.Ndef;
import android.nfc.tech.NdefFormatable;
import android.nfc.tech.NfcA;
import android.nfc.tech.NfcV;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.preference.PreferenceManager;
import android.provider.Settings;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.Toast;

/**
 * This is the core activity of the application. It renders simple indoor room
 * geometries on top of an aerial (Google) map. It creates a custom {@link LocationProvider}
that contains
 * the location of the last read marker. That {@link LocationProvider} may also be used by
other apps by
 * calling the {@link LocationProvider} "taglocation". In addition, the current location
 * is displayed on the map. <br><br>
 * For reading NFC tags containing location information, the NFC Foreground dispatch (=all
intents matching the
 * specified intent-filters will be delivered to this activity) is enabled. It will
 * read NfcA (ISO 14443-3A), NfcV (ISO 15693) and all NDEF tags containing adequate data. All
NFC functionality will be disabled on
 * devices without NFC support or running an Android version prior to 2.3.3. <br><br>
 * The QR reader capability uses the ZXing Barcode Scanner app. It will not work without said
application
 * being installed. <br><br>
 * The application performs a check whether the 'Allow Mock Locations' setting is enabled, and
will not work
 * if it isn't.
 * @author Johannes Bolz
 *
 */
public class GMapActivity extends MapActivity {

  // custom LocationProvider ID:
  protected static final String TAG_LOCATION = "taglocation";

  // default configuration values:
  protected static final String DEFAULTLINKFILE = "default.iir";
  protected static final String DATAPATH = "/taglocate/";
  protected static final String DEFAULTGEOMETRYFILE = "default.osm";
  protected static final int DEFAULT_ZOOM = 20;
  protected static final double DEFAULT_LAT = 52.545490803744265;
  protected static final double DEFAULT_LON = 13.355747670676614;

  // private objects:
  private MapView mapView;
  private LinearLayout btnLayout;
  private List<Overlay> mapOverlays;
  private LocationMarkerOverlay locOverlay;
  private EditModeOverlay editOverlay;
  private LocationManager locationManager;
  private MapController mapController;
  private PendingIntent pi;
  private IntentFilter[] filters;
  private String[][] techLists;
  private LocationListener listener;
  private boolean editMode = false;
  private double lon = 0;
  private double lat = 0;
  private int zoom = 0;

  /**
   * Standard onCreate method.
   */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Check if the 'Allow Mock Location' system setting is enabled.
    // If not, display an info dialog and close the application.
    if (!mockLocationEnabled()) {
      showMLDialog();
    } else {

      // Get preferences from saved application state or apply default values:
      SharedPreferences prefs = PreferenceManager
          .getDefaultSharedPreferences(getBaseContext());

      this.lon = Double.parseDouble(prefs.getString(getString(R.string.lon_preference),
          String.valueOf(DEFAULT_LON)));
      this.lat = Double.parseDouble(prefs.getString(getString(R.string.lat_preference),
          String.valueOf(DEFAULT_LAT)));
      this.zoom = Integer.parseInt(prefs.getString(getString(R.string.zoom_preference),
          String.valueOf(DEFAULT_ZOOM)));

      this.editMode = prefs.getBoolean(getString(R.string.edit_mode_preference),
this.editMode);

      SettingsSingleton.getInstance().setDatafile(
          prefs.getString(getString(R.string.link_file_preference),
              DEFAULTLINKFILE));
      SettingsSingleton.getInstance().setDatapath(
          Environment.getExternalStorageDirectory().getName()
              + DATAPATH);
      SettingsSingleton.getInstance().setGeometryfile(
          prefs.getString(
              getString(R.string.geometry_file_preference),
              DEFAULTGEOMETRYFILE));
      SettingsSingleton.getInstance().setDownloadFromNfc(
          prefs.getBoolean(getString(R.string.loadfiles_preference),
              true));
      SettingsSingleton.getInstance().setStrategy(
          prefs.getString(getString(R.string.strategy_preference),
              IntentResolver.UID_FIRST));

      // Check if link and geometry files are available within the
      // application folder on the sdcard file system. If not, create default
      // ones:
      checkForFiles();

      // setup map view:
      setContentView(R.layout.main);
      mapView = (MapView) findViewById(R.id.mapview);
      mapView.setBuiltInZoomControls(true);
      mapView.setSatellite(true);
      mapOverlays = mapView.getOverlays();
      mapOverlays.add(new VectorOverlay());
      locOverlay = new LocationMarkerOverlay(getApplicationContext());
      locOverlay.setCoords(this.lon, this.lat);
      mapOverlays.add(locOverlay);
      mapController = mapView.getController();
      mapController.setZoom(Integer.valueOf(zoom));

      if (editMode) {
        startEditMode();
      } else {
        stopEditMode();
      }


      // Create a new LocationProvider for tag locations and an according listener
      // requesting location updates from it:
      locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
      listener = new TagLocationListener();
      createLocationProvider();
      locationManager.requestLocationUpdates(TAG_LOCATION, 0, 0, listener);

      // Update LocationProvider to last tag location:
      updateLocation(this.lon, this.lat);

      // Check for NFC support on the device. If NFC is supported,
      // create appropriate intent-filters:
      if (hasNfcSupport()) {
        setNfcFilters();
      }
    }
  }

  /**
   * Standard onPause method.
   */
  @Override
  public void onPause() {
    super.onPause();
    if (mockLocationEnabled()) {
      // Store persistent values into preferences:
      SharedPreferences prefs = PreferenceManager
          .getDefaultSharedPreferences(getBaseContext());
      SharedPreferences.Editor ed = prefs.edit();
      ed.putString(getString(R.string.lon_preference), String.valueOf(this.lon));
      ed.putString(getString(R.string.lat_preference), String.valueOf(this.lat));
      ed.putString(getString(R.string.link_file_preference),
          SettingsSingleton.getInstance().getDatafile());
      ed.putString(getString(R.string.geometry_file_preference),
          SettingsSingleton.getInstance().getGeometryfile());
      ed.putString(getString(R.string.strategy_preference), String
          .valueOf(SettingsSingleton.getInstance().getStrategy()));
      ed.putBoolean(getString(R.string.loadfiles_preference),
          SettingsSingleton.getInstance().isDownloadFromNfc());
      ed.putString(getString(R.string.zoom_preference),
String.valueOf(mapView.getZoomLevel()));
      ed.putBoolean(getString(R.string.edit_mode_preference), editMode);
      ed.commit();

      // Check NFC support, disable foreground intent dispatch:
      if (hasNfcSupport()) {
        NfcAdapter.getDefaultAdapter(this).disableForegroundDispatch(
            this);
      }
    }
  }

  /**
   * Standard onResume method.
   */
  @Override
  protected void onResume() {
    super.onResume();

    // Check for 'Mock Location' system setting:
    if (!mockLocationEnabled()) {
      showMLDialog();
    } else {
      // Check for NFC support, enable foreground intent dispatch:
      if (hasNfcSupport()) {
        NfcAdapter adapter = NfcAdapter.getDefaultAdapter(this);
        adapter.enableForegroundDispatch(this, pi, filters, techLists);
        adapter.enableForegroundDispatch(this, pi, filters, techLists);
      }
    }
  }

  /**
   * In this activity, this method handles NFC intents delivered by foreground dispatch.
   */
  @Override
  public void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    // Make sure the activity is receiving location updates by the LocationProvider:
    locationManager.requestLocationUpdates(TAG_LOCATION, 0, 0, listener);

    // Resolve intent and update location:
    IntentResolver i = new IntentResolver(getApplicationContext(), intent);
    if (i.getLocation() != null) {
      updateLocation(i.getLocation().getLon(), i.getLocation().getLat());
    }
  }

  /**
   * Required MapActivity method.
   */
  @Override
  protected boolean isRouteDisplayed() {
    return false;
  }

  /**
   * Creates the options menu view.
   */
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.map_menu, menu);
    return true;
  }

  /**
   * Handles touch events on options menu items.
   */
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    int id = item.getItemId();

    // QR Locate:
    if (id == R.id.qr_menu_item) {

      // Check availability of 'Barcode Scanner' (if it is installed, no
      // exception will be thrown:
      PackageManager pm = getPackageManager();
      try {
        pm.getPackageInfo(getString(R.string.qr_reader_package),
            PackageManager.GET_ACTIVITIES);

        // Start 'Barcode Scanner':
        Intent intent = new Intent(getString(R.string.qr_scan_action));
        intent.setPackage(getString(R.string.qr_reader_package));
        intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
        startActivityForResult(intent, 0);
        return true;

      // Handle unavailability of 'Barcode Scanner':
      } catch (PackageManager.NameNotFoundException e) {
        Toast.makeText(getApplicationContext(),
            getString(R.string.message_bc_unavailable),
            Toast.LENGTH_LONG).show();
      }

    // Settings:
    } else if (id == R.id.settings_menu_item) {
      startActivity(new Intent(this, PreferencesActivity.class));

    // About / How to Use:
    } else if (id == R.id.help_menu_item) {
      startActivity(new Intent(this, HelpActivity.class));

    // Create Tag:
    } else if (id == R.id.reference_tag_item) {
      startEditMode();
    }


    return super.onOptionsItemSelected(item);
  }

  /**
   * Handles results from Barcode Scanner in this activity.
   */
  @Override
  public void onActivityResult(int requestCode, int resultCode, Intent intent) {
    if (requestCode == 0) {
      if (resultCode == RESULT_OK) {

        // Make sure the activity is getting location updates:
        locationManager.requestLocationUpdates(TAG_LOCATION, 0, 0,
            listener);

        // Resolve resulting intent:
        IntentResolver i = new IntentResolver(getApplicationContext(),
            intent);
        if (i.getLocation() != null) {
          updateLocation(i.getLocation().getLon(), i.getLocation()
              .getLat());
        }
      } else if (resultCode == RESULT_CANCELED) {
        // Handle cancel = do nothing.
      }
    }
  }

  // TODO: Remove duplication
  /**
   * Checks for NFC support on the device.
   * @return true if the device hardware supports NFC and is running Android
   * 2.3.3 or later.
   */
  private boolean hasNfcSupport() {
    return (Build.VERSION.SDK_INT >= 10 && NfcAdapter
        .getDefaultAdapter(getApplicationContext()) != null);
  }

  // TODO: Remove duplication
  /**
   * Sets intent-filters for NFC foreground dispatch.
   */

  private void setNfcFilters() {
    // Foreground Dispatch:
    pi = PendingIntent.getActivity(this, 0, new Intent(this, getClass())
        .addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);

    // Create intent-filters for geo and iii scheme:
    IntentFilter geoFilter = new IntentFilter(
        NfcAdapter.ACTION_NDEF_DISCOVERED);
    geoFilter.addDataScheme("geo");
    IntentFilter iiiFilter = new IntentFilter(
        NfcAdapter.ACTION_NDEF_DISCOVERED);
    iiiFilter.addDataScheme("iii");

    // Create intent-filters for all NDEF compatible tags and
    // NfcA and NfcV RFID tags in order to read their UID's:
    IntentFilter ndefFilter = new IntentFilter(
        NfcAdapter.ACTION_NDEF_DISCOVERED);
    IntentFilter techFilter = new IntentFilter(
        NfcAdapter.ACTION_TECH_DISCOVERED);


    filters = new IntentFilter[] { geoFilter, iiiFilter, ndefFilter,
        techFilter };

    // Setup tech list filters for NfcA (ISO 14443) and NfcV (ISO 15693) tags
    techLists = new String[4][1];
    techLists[0][0] = NfcA.class.getName();
    techLists[1][0] = NfcV.class.getName();
    techLists[2][0] = Ndef.class.getName();
    techLists[3][0] = NdefFormatable.class.getName();
  }

  /**
   * Creates custom LocationProvider.
   */
  private void createLocationProvider() {

    // If LocationProvider already exists, remove it and renew it:
    if (locationManager.getProvider(TAG_LOCATION) != null) {
      locationManager.removeTestProvider(TAG_LOCATION);
    }
    locationManager.addTestProvider(TAG_LOCATION, "requiresNetwork" == "",
        "requiresSatellite" == "", "requiresCell" == "",
        "hasMonetaryCost" == "", "supportsAltitude" == "",
        "supportsSpeed" == "", "supportsBearing" == "",

        android.location.Criteria.POWER_LOW,
        android.location.Criteria.ACCURACY_FINE);
  }

  /**
   * Updates the LocationProvider with a new location:
   * @param lon geographic longitude
   * @param lat geographic latitude
   */
  private void updateLocation(double lon, double lat) {
    Location newLocation = new Location(TAG_LOCATION);

    newLocation.setLatitude(lat);
    newLocation.setLongitude(lon);

    locationManager.setTestProviderEnabled(TAG_LOCATION, true);

    locationManager.setTestProviderStatus(TAG_LOCATION,
        LocationProvider.AVAILABLE, null, System.currentTimeMillis());

    locationManager.setTestProviderLocation(TAG_LOCATION, newLocation);
  }

  /**
   * Creates a dialog saying the mock location system setting is disabled.
   */
  private void showMLDialog() {
    AlertDialog alert;
    AlertDialog.Builder builder = new AlertDialog.Builder(GMapActivity.this);
    builder.setTitle(getString(R.string.title_ml_disabled));
    builder.setMessage(getString(R.string.message_ml_disabled))
        .setCancelable(false)
        .setPositiveButton("OK", new DialogInterface.OnClickListener() {
          public void onClick(DialogInterface dialog, int id) {
            finish();
          }
        });
    alert = builder.create();
    alert.show();
  }

  /**
   * Checks if the 'Mock Location' system setting is enabled.
   * @return true if it is enabled.
   */
  private boolean mockLocationEnabled() {
    return "1".equals(Settings.Secure.getString(getContentResolver(),
        Settings.Secure.ALLOW_MOCK_LOCATION));
  }

  /**
   * Checks if there is a taglocate folder on the sdcard directory and whether
   * it contains the currently used geometry and link files. If not, the default
   * ones will be copied and applied.
   */
  private void checkForFiles() {
    if (!fileExists(SettingsSingleton.getInstance().getDatapath()
        + SettingsSingleton.getInstance().getDatafile())) {
      copyDefault(DEFAULTLINKFILE);
      SettingsSingleton.getInstance().setDatafile(DEFAULTLINKFILE);
    }
    if (!fileExists(SettingsSingleton.getInstance().getDatapath()
        + SettingsSingleton.getInstance().getGeometryfile())) {
      copyDefault(DEFAULTGEOMETRYFILE);
      SettingsSingleton.getInstance()
          .setGeometryfile(DEFAULTGEOMETRYFILE);
    }
  }

  /**
   * Checks if a specifies file exists.
   * @param uri the file's full path name
   * @return true if the file exists
   */
  private boolean fileExists(String uri) {
    File f = new File(uri);
    return f.exists();
  }

  /**
   * Copies default files to the taglocate sdcard folder.
   * @param file
   */
  private void copyDefault(String file) {
    File dir = new File(Environment.getExternalStorageDirectory().getName()
        + DATAPATH);
    if (!dir.exists()) {
      dir.mkdirs();
    }
    try {
      InputStream in = getAssets().open(file);
      OutputStream out = new FileOutputStream(new File(Environment
          .getExternalStorageDirectory().getName() + DATAPATH + file));

      byte[] buf = new byte[8192];
      int len;
      while ((len = in.read(buf)) > 0) {
        out.write(buf, 0, len);
      }
      in.close();
      out.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

  /**
   * Starts the edit mode. Adds the {@link EditModeOverlay} to the map and displays
   * edit buttons.
   */
  private void startEditMode() {
    this.editMode = true;
    editOverlay = new EditModeOverlay(getApplicationContext());
    mapOverlays.add(editOverlay);
    mapView.invalidate();
    btnLayout = (LinearLayout) findViewById(R.id.btnLayout);
    btnLayout.setVisibility(LinearLayout.VISIBLE);
    Button cancelBtn = (Button) findViewById(R.id.cancelAddTagBtn);
    cancelBtn.setOnClickListener(new OnClickListener() {
      @Override
      public void onClick(View v) {
        stopEditMode();
      }
    });
    Button addBtn = (Button) findViewById(R.id.addTagBtn);
    addBtn.setOnClickListener(new OnClickListener() {
      @Override
      public void onClick(View v) {
        // Create GeoPoint out of view center pixel coordinates:
        GeoPoint gp = mapView.getProjection().fromPixels((int)
Math.round(((double)mapView.getWidth()) / 2),
            (int) Math.round(((double)mapView.getHeight()) / 2));

        Intent i = new Intent(GMapActivity.this, EditDispatchActivity.class);
        i.putExtra(getString(R.string.center_lon), gp.getLongitudeE6());
        i.putExtra(getString(R.string.center_lat), gp.getLatitudeE6());
        startActivity(i);
        stopEditMode();
      }
    });
  }

  /**
   * Stops the edit mode. Removes edit buttons and the {@link EditModeOverlay}.
   */
  private void stopEditMode() {
    this.editMode = false;
    if (mapOverlays.contains(editOverlay)) {
      mapOverlays.remove(mapOverlays.indexOf(editOverlay));
    }
    mapView.invalidate();
    btnLayout = (LinearLayout) findViewById(R.id.btnLayout);
    btnLayout.setVisibility(LinearLayout.INVISIBLE);
  }


  /**
   * Implementation of a LocationListener that updated the map to changes in the current
   * location.
   * @author Johannes Bolz
   */
  private final class TagLocationListener implements LocationListener {
    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
    }

    @Override
    public void onProviderEnabled(String provider) {
    }

    @Override
    public void onProviderDisabled(String provider) {
    }

    @Override
    public void onLocationChanged(Location location) {
      // Get new location, set marker, center map on new location:
      GMapActivity.this.lon = location.getLongitude();
      GMapActivity.this.lat = location.getLatitude();
      GMapActivity.this.locOverlay.setCoords(location.getLongitude(),
          location.getLatitude());
      GMapActivity.this.mapView.invalidate();
      MapController mapController = GMapActivity.this.mapView
          .getController();
      mapController.animateTo(new GeoPoint((int) Math
          .round(GMapActivity.this.lat * 1000000), (int) Math
          .round(GMapActivity.this.lon * 1000000)));
    }
  }
}
```

HelpActivity.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */
package de.bolz.android.taglocate.ui;


import de.bolz.android.taglocate.R;
import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;

/**
 * This activity displays the application's HTML help / credits pages.
 * @author Johannes Bolz
 */
public class HelpActivity extends Activity{
  private WebView webView;

  /**
   * Standard onCreate method.
   */
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.help);

      // Populate WebVierw with HTML help page:
      webView = (WebView) findViewById(R.id.webview);
      webView.getSettings().setJavaScriptEnabled(true);
      webView.loadUrl(getString(R.string.help_url));
  }
}
```

LocationMarkerOverlay.java

```
/*
```

```java
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.ui;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.Projection;

import de.bolz.android.taglocate.R;

/**
 * This class is responsible for drawing a location marker on the Map View.
 * @author Johannes Bolz
 */
public class LocationMarkerOverlay extends Overlay {
  private Projection projection;

  private double lon, lat;
  private Point point;
  private GeoPoint geopoint;
  private Context context;

  /**
   * @param context the context of the calling class (activity)
   */
  public LocationMarkerOverlay(Context context) {
    this.context = context;
  }

  /**
   * Standard Overlay method. Called everytime the objects in this class are rendered.
   */
  public void draw(Canvas canvas, MapView view, boolean shadow) {
    super.draw(canvas, view, shadow);

    // Calculate pixel coordinates out of the location:
    projection = view.getProjection();
    geopoint = new GeoPoint((int) (Math.round(this.lat * 1000000)),
        (int) (Math.round(this.lon * 1000000)));
    point = new Point();
    projection.toPixels(geopoint, point);

    // Draw marker:
    Bitmap locationMarker = BitmapFactory.decodeResource(
        context.getResources(), R.drawable.marker_white_border);
    canvas.drawBitmap(locationMarker, point.x
        - (locationMarker.getWidth() / 2),
        point.y - (locationMarker.getHeight() / 2), null);
  }

  /**
```

```
   * Sets the marker's coordinates.
   * @param lon geographic longitude in decimal degrees
   * @param lat geographic latitude in decimal degrees
   */
  public void setCoords(double lon, double lat) {
    this.lon = lon;
    this.lat = lat;
  }

}
```

NfcEditActivity.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 *
 *  ## CREDITS: Some parts of this class were developed upon code examples  ##
 *  ## and / or snippets from the following sources:                        ##
 *
 *  NFC foreground dispatch, Android API Demos:
 *
http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/nfc/Foreg
roundDispatch.html
 *
 */

package de.bolz.android.taglocate.ui;

import java.io.IOException;
import de.berlin.magun.nfcmime.core.NdefMessageBuilder;
import de.berlin.magun.nfcmime.core.RfidDAO;
import de.bolz.android.taglocate.R;
import de.bolz.android.taglocate.protocol.GeoUri;
import android.app.AlertDialog;
import android.app.PendingIntent;
import android.content.DialogInterface;
import android.content.DialogInterface.OnCancelListener;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.ActivityInfo;
import android.nfc.NfcAdapter;
import android.nfc.Tag;
import android.nfc.tech.Ndef;
import android.nfc.tech.NdefFormatable;
import android.nfc.tech.NfcA;
import android.nfc.tech.NfcV;
import android.os.Bundle;
import android.view.Window;
import android.widget.Toast;

/**
 * Activity that handles referencing NFC tags. May either write an explicit link to an NFC tag
 * or reference the tag's unique ID in the reference model (iir file).
 * @author Johannes Bolz
 */
public class NfcEditActivity extends TagEditActivity{

  // NFC handling mode keys:
```

```java
public static final int WRITE = 0;
public static final int READ = 1;
private static final int NOTSET = -1;

private PendingIntent pi;
private IntentFilter[] filters;
private String[][] techLists;
private Tag tag;
private int mode;
private AlertDialog infoDialog;

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      Intent i = getIntent();

      this.mode = i.getIntExtra(getString(R.string.nfc_edit_mode), NOTSET);
      requestWindowFeature(Window.FEATURE_NO_TITLE);
  this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
  setNfcFilters();
  AlertDialog.Builder builder = new AlertDialog.Builder(this);
  builder.setMessage(getString(R.string.initial_nfc_dialog));

  // Finish activity when back button is pressed:
  builder.setOnCancelListener(new OnCancelListener() {
    @Override
    public void onCancel(DialogInterface dialog) {
      NfcEditActivity.this.finish();
    }
  });
  infoDialog = builder.create();
  infoDialog.show();
  }

  /**
   * Standard onPause method. Deactivates NFC foreground dispatch.
   */
  @Override
  public void onPause() {
    super.onPause();
    NfcAdapter.getDefaultAdapter(this).disableForegroundDispatch(this);
  }

  /**
   * Standard onResume method. Activates NFC foreground dispatch.
   */
  @Override
  public void onResume() {
    super.onResume();
    NfcAdapter adapter = NfcAdapter.getDefaultAdapter(this);
  adapter.enableForegroundDispatch(this, pi, filters, techLists);
  adapter.enableForegroundDispatch(this, pi, filters, techLists);
  }

  /**
 * In this activity, this method handles NFC intents delivered by foreground dispatch.
 */
@Override
public void onNewIntent(Intent intent) {
  super.onNewIntent(intent);
  this.tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
  // Write link on tag:
  if (mode == WRITE) {
    writeTag();

  // Reference NFC ID:
  } else if (mode == READ) {
    referenceTag();
  }
}
```

```
/**
 * Writes a Geo-URI to the NFC tag.
 */
  private void writeTag() {
  NdefMessageBuilder builder = new NdefMessageBuilder();

  // Build Geo-URI, recalculate coordinate int values to decimal degrees:
  GeoUri uri = new GeoUri(((double) this.latInt) / 1000000,
      ((double) this.lonInt) / 1000000);
  try {
    builder.addUriRecord(uri.getString());
    RfidDAO dao = new RfidDAO();

    // Format tag to NDEF, if not already done:
    if (NdefFormatable.get(tag) != null) {
      dao.formatNdef(tag);
      showToast(getString(R.string.format_msg), Toast.LENGTH_LONG);
    }

    // Write tag, if it is NDEF-compliant:
    if (Ndef.get(tag) != null
        && builder.getDataSize() <= Ndef.get(tag).getMaxSize()) {
      dao.writeMessage(tag, builder.buildMessage());
      showToast(getString(R.string.done), Toast.LENGTH_SHORT);
      infoDialog.dismiss();
      this.finish();

    // Otherwise state that there is a problem:
    } else {
      showToast(getString(R.string.write_fail), Toast.LENGTH_SHORT);
    }
  } catch (IOException e) {
    e.printStackTrace();
    infoDialog.dismiss();
    this.finish();
  } catch (Exception e) {
    // This exception will most likely happen, if tag is read-only:
    e.printStackTrace();
    showToast(getString(R.string.write_fail), Toast.LENGTH_SHORT);
  }

  }

  /**
   * References a NFC ID in the reference model.
   */
  private void referenceTag() {
    RfidDAO dao = new RfidDAO();
    String id = dao.getTagId(tag);
    referenceId(id, NFC);
    showToast(getString(R.string.reference_done) + " " + id, Toast.LENGTH_SHORT);
    infoDialog.dismiss();
    this.finish();
  }


  // TODO: Remove duplication
  /**
 * Sets intent-filters for NFC foreground dispatch.
 */
private void setNfcFilters() {
  // Foreground Dispatch:
  pi = PendingIntent.getActivity(this, 0, new Intent(this, getClass())
      .addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);

  // Create intent-filters for geo and iii scheme:
  IntentFilter geoFilter = new IntentFilter(
      NfcAdapter.ACTION_NDEF_DISCOVERED);
  geoFilter.addDataScheme("geo");
  IntentFilter iiiFilter = new IntentFilter(
      NfcAdapter.ACTION_NDEF_DISCOVERED);
  iiiFilter.addDataScheme("iii");
```

```
    // Create intent-filters for all NDEF compatible tags and
    // NfcA and NfcV RFID tags in order to read their UID's:
    IntentFilter ndefFilter = new IntentFilter(
        NfcAdapter.ACTION_NDEF_DISCOVERED);
    IntentFilter techFilter = new IntentFilter(
        NfcAdapter.ACTION_TECH_DISCOVERED);


    filters = new IntentFilter[] { geoFilter, iiiFilter, ndefFilter,
        techFilter };

    // Setup tech list filters for NfcA (ISO 14443) and NfcV (ISO 15693) tags
    techLists = new String[4][1];
    techLists[0][0] = NfcA.class.getName();
    techLists[1][0] = NfcV.class.getName();
    techLists[2][0] = Ndef.class.getName();
    techLists[3][0] = NdefFormatable.class.getName();
  }
}
```

PreferencesActivity.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 *
 *  ## CREDITS: Some parts of this class were developed upon code examples  ##
 *  ## and / or snippets from the following sources:                        ##
 *
 *  PreferenceActivity, Android Developers class description:
 *  http://developer.android.com/reference/android/preference/PreferenceActivity.html
 */

package de.bolz.android.taglocate.ui;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

import de.bolz.android.taglocate.R;
import de.bolz.android.taglocate.protocol.SettingsSingleton;
import android.nfc.NfcAdapter;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.preference.CheckBoxPreference;
import android.preference.ListPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.PreferenceCategory;
import android.preference.PreferenceManager;
import android.preference.PreferenceScreen;

/**
 * This activity provides a user interface to change application settings.
 *
 * @author Johannes Bolz
 */
```

```java
public class PreferencesActivity extends PreferenceActivity {
  private CharSequence[] osmCs;
  private CharSequence[] iirCs;
  private ListPreference geometryFilePreference, linkFilePreference,
      strategyPreference;
  private CheckBoxPreference loadFromNfc;

  /**
   * Standard onCreate method.
   */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    addPreferencesFromResource(R.xml.preferences);
    PreferenceManager.setDefaultValues(getApplicationContext(),
        R.xml.preferences, false);

    // Create checkbox for the 'Load Files from NFC' setting:
    loadFromNfc = (CheckBoxPreference) findPreference(getText(R.string.loadfiles_preference));
    if (!hasNfcSupport()) {
      loadFromNfc
          .setSummary(getString(R.string.message_nfc_not_supported));
      loadFromNfc.setChecked(false);
      loadFromNfc.setEnabled(false);
    }
    loadFromNfc
        .setOnPreferenceChangeListener(new Preference.OnPreferenceChangeListener() {
          @Override
          public boolean onPreferenceChange(Preference preference,
              Object newValue) {
            loadFromNfc.setChecked((Boolean) newValue);
            SettingsSingleton.getInstance().setDownloadFromNfc(
                (Boolean) newValue);
            return false;
          }
        });

    // Load file names into arrays:
    getFileArrays();

    // Create radio button chooser menu for geometry files:
    geometryFilePreference = (ListPreference)
findPreference(getText(R.string.geometry_file_preference));
    geometryFilePreference.setEntries(osmCs);
    geometryFilePreference.setEntryValues(osmCs);
    geometryFilePreference.setSummary(geometryFilePreference.getEntry());
    geometryFilePreference
        .setOnPreferenceChangeListener(new Preference.OnPreferenceChangeListener() {
          @Override
          public boolean onPreferenceChange(Preference preference,
              Object newValue) {
            geometryFilePreference.setValue((String) newValue);
            SettingsSingleton.getInstance().setGeometryfile(
                (String) newValue);
            geometryFilePreference
                .setSummary(geometryFilePreference.getEntry());
            return false;
          }
        });

    // Create radio button chooser menu for link files:
    linkFilePreference = (ListPreference)
findPreference(getText(R.string.link_file_preference));
    linkFilePreference.setEntries(iirCs);
    linkFilePreference.setEntryValues(iirCs);
    linkFilePreference.setSummary(linkFilePreference.getEntry());
    linkFilePreference
        .setOnPreferenceChangeListener(new Preference.OnPreferenceChangeListener() {
          @Override
          public boolean onPreferenceChange(Preference preference,
              Object newValue) {
            linkFilePreference.setValue((String) newValue);
            SettingsSingleton.getInstance().setDatafile(
                (String) newValue);
            linkFilePreference.setSummary(linkFilePreference
                .getEntry());
            return false;
          }
        });

    // Create radio button chooser menu for link resolution strategies:
    strategyPreference = (ListPreference)
findPreference(getText(R.string.strategy_preference));
    strategyPreference.setSummary(strategyPreference.getEntry());
    strategyPreference
        .setOnPreferenceChangeListener(new Preference.OnPreferenceChangeListener() {
          @Override
          public boolean onPreferenceChange(Preference preference,
              Object newValue) {
            strategyPreference.setValue((String) newValue);
            SettingsSingleton.getInstance().setStrategy(
                (String) newValue);
            strategyPreference.setSummary(strategyPreference
                .getEntry());
            return false;
          }
        });

    // remove preferences that shall not appear in the view:
    PreferenceCategory category = (PreferenceCategory)
findPreference(getString(R.string.hidden_prefs));
    PreferenceScreen screen = getPreferenceScreen();
    screen.removePreference(category);

  }

  /**
   * Builds an array with all geometry file names and one for all link file names within
   * the taglocate sdcard folder.
   */
  private void getFileArrays() {
    File dir = new File(Environment.getExternalStorageDirectory().getName()
        + GMapActivity.DATAPATH);
    String[] files = dir.list();
    List<CharSequence> osm = new ArrayList<CharSequence>();
    List<CharSequence> iir = new ArrayList<CharSequence>();
    for (int i = 0; i < files.length; i++) {
      if (files[i].endsWith(".osm")) {
        osm.add(files[i]);
      }
      if (files[i].endsWith("iir")) {
        iir.add(files[i]);
      }
    }

    osmCs = new CharSequence[osm.size()];
    for (int i = 0; i < osmCs.length; i++) {
      osmCs[i] = osm.get(i);
    }
    iirCs = new CharSequence[iir.size()];
    for (int i = 0; i < iirCs.length; i++) {
      iirCs[i] = iir.get(i);
    }
  }

  /**
   * Checks if there is NFC support on the device and the Android versio is at least 2.3.3.
   * @return true if both conditions apply.
   */
  private boolean hasNfcSupport() {
    return (Build.VERSION.SDK_INT >= 10 && NfcAdapter
        .getDefaultAdapter(getApplicationContext()) != null);
  }
}
```

QrEditActivity.java

```java
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.ui;

import de.bolz.android.taglocate.R;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.widget.Toast;

/**
 * Activity that handles referencing QR codes.
 * @author Johannes Bolz
 *
 */
public class QrEditActivity extends TagEditActivity {

  /**
   * Standard onCreate method. Calls the 'Barcode Scanner' app
   * to get QR contents.
   */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Check availability of 'Barcode Scanner' (if it is installed, no
    // exception will be thrown:
    PackageManager pm = getPackageManager();
    try {
      pm.getPackageInfo(getString(R.string.qr_reader_package),
          PackageManager.GET_ACTIVITIES);

      // Start 'Barcode Scanner':
      Intent intent = new Intent(getString(R.string.qr_scan_action));
      intent.setPackage(getString(R.string.qr_reader_package));
      intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
      startActivityForResult(intent, 0);

    // Handle unavailability of 'Barcode Scanner':
    } catch (PackageManager.NameNotFoundException e) {
      Toast.makeText(getApplicationContext(),
          getString(R.string.message_bc_unavailable),
          Toast.LENGTH_LONG).show();
    }
  }

  /**
   * Handles results from Barcode Scanner in this activity, i.e. reference
   * a scanned QR code.
   */
  @Override
  public void onActivityResult(int requestCode, int resultCode, Intent intent) {
    if (requestCode == 0) {
      if (resultCode == RESULT_OK) {
        String id = intent.getStringExtra("SCAN_RESULT");
        referenceId(id, QR);
        showToast(getString(R.string.reference_done) + " " + id, Toast.LENGTH_SHORT);
        this.finish();
      } else if (resultCode == RESULT_CANCELED) {
        this.finish();
      }
    }
  }

}
```

TagEditActivity.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.bolz.android.taglocate.ui;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

import org.apache.commons.io.IOUtils;
import org.apache.http.util.EncodingUtils;
import org.simpleframework.xml.Serializer;
import org.simpleframework.xml.core.Persister;
import org.simpleframework.xml.stream.CamelCaseStyle;
import org.simpleframework.xml.stream.Format;

import de.berlin.magun.protocol.Source;
import de.bolz.android.taglocate.R;
import de.bolz.android.taglocate.protocol.GeoUri;
import de.bolz.android.taglocate.protocol.SettingsSingleton;
import de.bolz.android.taglocate.protocol.data.Reference;
import de.bolz.android.taglocate.protocol.data.ReferenceList;
import de.bolz.android.taglocate.protocol.data.Tag;
import de.bolz.android.taglocate.protocol.data.Target;
import de.bolz.android.taglocate.protocol.data.Trigger;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.widget.Toast;

/**
 * Abstract superclass to store common functionality for tag referencing activities.
 * @author Johannes Bolz
 *
 */
public abstract class TagEditActivity extends Activity{

  public static final String NFC = "nfc";
  public static final String QR = "qr";
```

```java
protected int latInt;
protected int lonInt;

/**
 * Standard onCreate method. Gets the latitude and longitude int's from the
 * Intent.
 */
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  Intent i = getIntent();
  this.latInt = i.getIntExtra(getString(R.string.center_lat),
          EditDispatchActivity.COORDINATENOTSET);
      this.lonInt = i.getIntExtra(getString(R.string.center_lon),
          EditDispatchActivity.COORDINATENOTSET);
}

/**
 * Standard onPause method.
 */
@Override
public void onPause() {
  super.onPause();
}

/**
 * Standard onResume method.
 */
@Override
public void onResume() {
  super.onResume();
}

/**
 * Standard onNewIntent method.
 */
@Override
public void onNewIntent(Intent intent) {
  super.onNewIntent(intent);
}

/**
 * Adds an ID together with a location to the reference model.
 * @param id the ID itself
 * @param tagtype the type of tag, either 'qr' or 'nfc'.
 */
protected void referenceId(String id, String tagtype) {
  try {
    // Make sure file's content is decoded using UTF-8:
    InputStream is = new FileInputStream(new File(SettingsSingleton
        .getInstance().getDatapath()
        + SettingsSingleton.getInstance().getDatafile()));
    byte[] ba = IOUtils.toByteArray(is);
    String input = EncodingUtils.getString(ba, "UTF-8");

    // Create a serializer that adheres to the XML's camel case element style:
    Serializer s = new Persister(new Format(new CamelCaseStyle()));
    ReferenceList linkList = s.read(ReferenceList.class, input);

    // Check if ID is already referenced:
    boolean isInDataSet = false;
    List<Reference> links = linkList.getReferences();
    for (int i = 0; i < links.size(); i++) {
      Reference link = links.get(i);
      if ((new Source(link.getTrigger().getTag().getTagStr())).getValue()
          .equalsIgnoreCase(id)) {
        String geoUri = new GeoUri(((double) this.latInt) / 1000000,
            ((double) this.lonInt) / 1000000).getString();
        link.getTarget().setTargetStr(geoUri);
        isInDataSet = true;
        break;
        }
      }

      // Create new Reference element in reference model, if necessary:
      if (!isInDataSet) {
        Reference l = new Reference();
        l.setTarget(new Target());
        l.getTarget().setTargetStr(new GeoUri(((double) this.latInt) / 1000000,
            ((double) this.lonInt) / 1000000).getString());
        l.setTrigger(new Trigger());
        l.getTrigger().setTag(new Tag());
        l.getTrigger().getTag().setTagStr(tagtype + ":" + id);
        links.add(l);
      }

      // Overwrite link file with updated reference:
      File result = new File(SettingsSingleton
          .getInstance().getDatapath()
          + SettingsSingleton.getInstance().getDatafile());
      s.write(linkList, result);

    } catch (FileNotFoundException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }

  /**
   * Shows a Toast on top of the view.
   * @param msg the message to display.
   * @param length the Toast duration
   */
  protected void showToast(String msg, int length) {
    Toast t = Toast.makeText(this, msg,
        length);
    t.setGravity(Gravity.TOP, 0, 0);
    t.show();
  }
}
```

VectorOverlay.java

```
/*
 *    Copyright 2011 by Johannes Bolz and the MAGUN project
 *    johannes-bolz (at) gmx.net
 *    http://magun.beuth-hochschule.de
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 *
 *  ## CREDITS: Some parts of this class were developed upon code examples  ##
 *  ## and / or snippets from the following sources:                        ##
 *
 *  Drawing polygon overlays, discussion on the Stack Overflow forum:
 *  http://stackoverflow.com/questions/2176397/drawing-a-line-path-on-google-maps
 */

package de.bolz.android.taglocate.ui;

import java.io.File;
```

```java
import java.util.ArrayList;
import java.util.List;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.Point;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.Projection;

import de.bolz.android.taglocate.geom.Coordinates;
import de.bolz.android.taglocate.geom.Geometry;
import de.bolz.android.taglocate.geom.OsmParser;
import de.bolz.android.taglocate.protocol.SettingsSingleton;

/**
 * This overlay class draws polygon geometries on the map. For better performance, it will
 * cache calculated GeoPoints using a Singleton ({@link GeometriesSingleton}).
 * @author Johannes Bolz
 *
 */
class VectorOverlay extends Overlay {
  private Paint paint, fillpaint;
  private Path path;
  private Projection projection;
  private List<Geometry> geometries;
  private Canvas canvas;
  private String geometryfile;

  public VectorOverlay() {
    setGeometries();

    // Polygon line style:
    paint = new Paint();
    paint.setDither(true);
    paint.setColor(Color.RED);
    paint.setStyle(Paint.Style.STROKE);
    paint.setStrokeJoin(Paint.Join.ROUND);
    paint.setStrokeCap(Paint.Cap.ROUND);
    paint.setStrokeWidth(3);
    paint.setAntiAlias(true);

    // Polygon fill style:
    fillpaint = new Paint();
    fillpaint.setColor(Color.DKGRAY);
    fillpaint.setStyle(Paint.Style.FILL);
  }

  /**
   * Standard Overlay method. Will be called each time the overlay is rendered.
   */
  public void draw(Canvas canvas, MapView view, boolean shadow){
        super.draw(canvas, view, shadow);

        // Check if GeoPoints have already been calculated out of the current geometry file.
        // If not, get the geometried from the file and calculate the GeoPoints:
        if (!(SettingsSingleton.getInstance().getDatapath() +
            SettingsSingleton.getInstance().getGeometryfile()).equals(geometryfile)) {
          setGeometries();
          calculateGeoPoints();
        }

        // Make sure the GeometriesSingleton is populated with GeoPoints:
        if (GeometriesSingleton.getInstance().getGeometries() == null) {
          calculateGeoPoints();
        }

        // Draw:
```

```
      projection = view.getProjection();
      this.canvas = canvas;
      drawPolygons(fillpaint);
      drawPolygons(paint);
  }

/**
 * Draws polygons from the GeoPoints stored in GeometrySingleton.
 * @param paint the {@link Paint} style information that shall be applied when
 * rendering
 */
private void drawPolygons(Paint paint) {

  // Get GeoPoints from Singleton:
  List<GeoPoint[]> geomList = GeometriesSingleton.getInstance().getGeometries();

  // Declare only one Point object that will be overriden during each iteration:
  Point point = new Point();

  // Each array within the list represents one polygon. Each polygon will be drawn
  // separately using a Path object:
  for (int i = 0; i < geomList.size(); i++) {
    path = new Path();
    GeoPoint[] points = geomList.get(i);

    // Reproject coordinates to pixel coordinates:
    projection.toPixels(points[0], point);

    // Set start point of the path:
    path.moveTo(point.x, point.y);

    // Build the path:
    for (int j = 1; j < points.length; j++) {
      projection.toPixels(points[j], point);
      path.lineTo(point.x, point.y);
    }

    // Close the path, making it a polygon:
    path.close();

    // Draw the path:
    canvas.drawPath(path, paint);
  }
}

/**
 * Calculates a {@link List} of {@link GeoPoint}s from the {@link Geometry}s
 * that was extracted from the geometry (osm) file. The list will be stored in
 * {@link GeometriesSingleton} so that they are accessible without recalculating
 * during each rendering process.
 */
private void calculateGeoPoints() {
  List<GeoPoint[]> geometryList = new ArrayList<GeoPoint[]>();

  // Each Geometry object represents a polygon:
  for (int i = 0; i < geometries.size(); i++) {
    GeoPoint[] geopoints = new GeoPoint[geometries.get(i).getPoints().size()];
    List<Coordinates> coords = geometries.get(i).getPoints();
    GeoPoint geopoint;

    // Create GeoPoints for each point within the polygon:
    for (int j = 0; j < coords.size(); j++) {
      geopoint = new GeoPoint((int) (Math.round(coords.get(j).getLat() * 1000000)),
          (int) (Math.round(coords.get(j).getLon() * 1000000)));
      geopoints[j] = geopoint;
    }
    geometryList.add(geopoints);
  }

  // Store into singleton:
  GeometriesSingleton.getInstance().setGeometries(geometryList);
}
```

```
  /**
   * Creates {@link Geometry} objects from the geometry (osm) file.
   */
  private void setGeometries() {
    this.geometryfile = SettingsSingleton.getInstance().getDatapath() +
    SettingsSingleton.getInstance().getGeometryfile();
    OsmParser p = new OsmParser(new File(geometryfile));
        geometries = p.getGeometries();
  }

}
```

Appendix II: NfcMimeLib Source Code

Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="de.berlin.magun.nfcmime"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
    </application>
</manifest>
```

Java Code

Package de.berlin.magun.nfcmime.core

CrcGenerator.java

```
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

import java.util.zip.CRC32;
import java.util.zip.Checksum;

import org.apache.commons.lang.ArrayUtils;
import android.nfc.NdefRecord;

/**
 * This class provides methods to generate and verify checksums for NdefRecord arrays.
 * @author Johannes Bolz
 *
 */
public class CrcGenerator {

  /**
   * Generates a CRC32 checksum for an array of NdefRecords.
   * @param records
   * @return CRC32 checksum represented as long.
   */
  public long getChecksum(NdefRecord[] records) {
    byte[] overallBytes = getByteArray(records);
    Checksum cs = new CRC32();
    cs.update(overallBytes, 0, overallBytes.length);
    return cs.getValue();
  }

  /**
   * Checks if the CRC32 checksum of a NdefRecord[] array matches a reference
   * checksum
   * @param records NdefRecord[] to check against checksum
   * @param checksum long representation of a CRC32 reference checksum
   * @return true if the CRC32 checksum of records matches checksum.
   */
  public boolean checkHash(NdefRecord[] records, long checksum) {
    String chksumStr = String.valueOf(checksum);
    String toValidate = String.valueOf(getChecksum(records));
    return (toValidate.equals(chksumStr));
  }

  /**
   * Creates a single byte array out of a NdefRecord[] array.
   * @param records
   * @return byte[]
   */
  private byte[] getByteArray(NdefRecord[] records) {
    byte[] overallBytes = new byte[0];
    for (int i = 0; i < records.length; i++) {
      overallBytes = ArrayUtils.addAll(overallBytes, records[i].toByteArray());
    }
    return overallBytes;
  }

}
```

FormatThread.java

```
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

import android.nfc.NdefMessage;
import android.nfc.NdefRecord;

/**
 * Thread that formats a NDEF compatible NFC tag.
 * @author Johannes Bolz
 */
public class FormatThread extends Thread{

  public void run() {
        try {
            int count = 0;

            // connect I/O
            NdefSingleton.getInstance().getNdefFormatable().connect();
            // wait 5 seconds for connection, if lost
            while (!NdefSingleton.getInstance().getNdefFormatable().isConnected()) {
                if (count > 500) {
                    throw new Exception("Unable to connect to tag");
                }
                count++;
                sleep(10);
            }

            // create an empty NDEF message as initial dataset
            NdefRecord[] mockRecords = new NdefRecord[1];
            mockRecords[0] = new NdefRecord(NdefRecord.TNF_EMPTY, new byte[0], new byte[0],
new byte[0]);
             NdefSingleton.getInstance().getNdefFormatable().format(new
NdefMessage(mockRecords));
            NdefSingleton.getInstance().getNdef().close();

        } catch (Throwable t) {
            // TODO Export exception handling
        }
```

```
    }
}
```

IOTask.java

```
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

import java.io.IOException;

import android.nfc.FormatException;

/**
 * Interface that is required by {@link IOThread} methods.
 * @author Johannes Bolz
 *
 */
public interface IOTask {
  /**
   * Specifies the operations to be performed in an {@link IOThread}
   * @throws IOException
   * @throws FormatException
   */
  public void doTask() throws IOException, FormatException;
}
```

IOThread.java

```
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

/**
 * Provides a thread for I/O operations on an NDEF formatted RFID transponder.
 * @author Johannes Bolz
 *
 */
public class IOThread extends Thread{

  /**
   * Will do nothing. Use run(IOTask task).
   */
  @Deprecated
  public void run() {}

  /**
   * Runs the thread.
   * @param task IOTask that specifies the operation to be performed on the transponder.
```

```
    */
  public void run(IOTask task) {
        try {
            int count = 0;

            // connect I/O
            NdefSingleton.getInstance().getNdef().connect();
            // wait 5 seconds for connection, if lost
            while (!NdefSingleton.getInstance().getNdef().isConnected()) {
                if (count > 500) {
                    throw new Exception("Unable to connect to tag");
                }
                count++;
                sleep(10);
            }

             task.doTask();
            NdefSingleton.getInstance().getNdef().close();

        } catch (Throwable t) {
            // TODO Export exception handling
        }
    }
}
```

NdefMessageBuilder.java

```
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

import java.io.IOException;
import java.io.InputStream;
import java.nio.charset.Charset;
import java.util.ArrayList;

import org.apache.commons.io.IOUtils;
import org.apache.commons.lang.ArrayUtils;

import android.nfc.NdefMessage;
import android.nfc.NdefRecord;

/**
 * Provides methods to create NdefMessages.
 * @author Johannes Bolz
 *
 */
public class NdefMessageBuilder {
  private ArrayList<NdefRecord> recordlist;
  private boolean hasChecksum = false;
  private NdefMessage message;

  public NdefMessageBuilder() {
    this.recordlist = new ArrayList<NdefRecord>();
  }

  /**
    * Stores an NDEF record from an InputStream. Will throw an IOException if the last NDEF
record is
    * a checksum added with addChecksum().
    * @param is InputStream that holds the payload data of the NDEF record.
```

```
   * @param mimetype MIME type definition, e.g. "text/plain"
   * @throws IOException
   */
  public void addMimeRecord(InputStream is, String mimetype) throws IOException {
    if (!this.hasChecksum) {
      recordlist.add(new NdefRecord(NdefRecord.TNF_MIME_MEDIA,
mimetype.getBytes(Charset.forName("US-ASCII")),
          new byte[0], IOUtils.toByteArray(is)));
    } else {
      throw new IOException("Cannot add record - last record is a checksum.");
    }
  }

  /**
   * Stores an NDEF record containing a URI, using the URI format definition.
   * @param uri the URI to be written
   * @throws IOException
   */
  public void addUriRecord(String uri) throws IOException {
    if (!this.hasChecksum) {
      recordlist.add(new NdefRecord(NdefRecord.TNF_WELL_KNOWN, NdefRecord.RTD_URI,
          new byte[0], ArrayUtils.addAll(new byte[] {(byte) 0x00},
uri.getBytes(Charset.forName("UTF-8")))));
    } else {
      throw new IOException("Cannot add record - last record is a checksum.");
    }
  }

  /**
   * Builds an NdefMessage from all stored NDEF records.
   * @return NdefMessage
   */
  public NdefMessage buildMessage() {
    NdefRecord[] records = new NdefRecord[recordlist.size()];
    for (int i = 0; i < recordlist.size(); i++) {
      records[i] = recordlist.get(i);
    }
    this.message = new NdefMessage(records);
    return this.message;
  }

  /**
   * Creates a CRC32 checksum out of all previously added NDEF records and adds it as last
record entry.
   * After that, it will not be possible to add additional records. The record payload will be
a String
   * consisting of the prefix 'crc32:' and a String representation of the checksum long, e.g.
   * 'crc32:2868450884'. The record will be MIME formatted as 'text/plain'.
   */
  public void addChecksum() {
    CrcGenerator generator = new CrcGenerator();
    NdefRecord[] records = new NdefRecord[recordlist.size()];
    for (int i = 0; i < recordlist.size(); i++) {
      records[i] = recordlist.get(i);
    }
    recordlist.add(new NdefRecord(NdefRecord.TNF_MIME_MEDIA,
"text/plain".getBytes(Charset.forName("US-ASCII")),
        new byte[0], ("crc32:" +
String.valueOf(generator.getChecksum(records))).getBytes(Charset.forName("US-ASCII"))));
  }

  /**
   * @return the overall NdefMessage size.
   */
  public int getDataSize() {
    return buildMessage().toByteArray().length;
  }
}
```

NdefMessageSingleton.java

```
/*
```

```java
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

import java.util.HashMap;
import java.util.Map;

import android.nfc.NdefMessage;

/**
 * This singleton class stores a map of NdefMessages, identified by a unique Long
 * (e.g. timestamp).
 * @author Johannes Bolz
 *
 */
public final class NdefMessageSingleton {

  private static NdefMessageSingleton instance;
  private static Map<Long, NdefMessage> messageMap;

  /**
   * empty, private constructor to avoid multiple instantiation
   */
  private NdefMessageSingleton() {}

  /**
   * @return the instance of the singleton object. Creates a new one if none exists.
   */
  protected static synchronized NdefMessageSingleton getInstance() {
    if (instance == null) {
      instance = new NdefMessageSingleton();
    }
    if (messageMap == null) {
      messageMap = new HashMap<Long, NdefMessage>();
    }
    return instance;
  }

  /**
   * Add a NDEF message to the map
   * @param key long to identify the stored NdefMessage
   * @param message NdefMessage to store
   */
  protected synchronized void addMessage(Long key, NdefMessage message) {
    messageMap.put(key, message);
  }

  /**
   * @param key the unique identifier that was assigned to specific {@link NdefMessage}
   * @return NdefMessage
   */
  protected synchronized NdefMessage getMessage(Long key) {
    return messageMap.get(key);
  }

  /**
   * Clears the map of all NdefMessage entries.
   */
  protected synchronized void clearMap() {
    messageMap.clear();
  }
}
```

NdefMimeRecord.java

```java
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

import org.apache.http.util.EncodingUtils;

/**
 * Wrapper class that holds the payload and MIME type byte arrays that may form an NdefRecord.
 * @author Johannes Bolz
 */
public class NdefMimeRecord {
  private byte[] payload;
  private byte[] mime;

  /**
   * Constructor.
   * @param payload byte[]
   * @param mime byte[]
   */
  public NdefMimeRecord(byte[] payload, byte[] mime) {
    this.payload = payload;
    this.mime = mime;
  }

  /**
   * @return payload byte array
   */
  public byte[] getPayload() {
    return payload;
  }

  /**
   * @return MIME definition as byte array
   */
  public byte[] getMime() {
    return mime;
  }

  /**
   * @return MIME definition as String.
   */
  public String getMimeString() {
//    return new String(this.mime, Charset.forName("US-ASCII"));
    return EncodingUtils.getString(this.mime, "UTF-8");
  }

}
```

NdefSingleton.java

```java
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

import android.nfc.tech.Ndef;
import android.nfc.tech.NdefFormatable;

/**
 * Singleton class that holds a Ndef and a NdefFormatable object
 * @author Johannes Bolz
 *
 */
public final class NdefSingleton {

  private static NdefSingleton instance;
  private static Ndef ndef;
  private static NdefFormatable ndefFormatable;

  /**
   * private, empty constructor to avoid multiple instances
   */
  private NdefSingleton() {}

  /**
   * @return the instance of NdefSingleton.
   */
  protected static synchronized NdefSingleton getInstance() {
    if (instance == null) {
      instance = new NdefSingleton();
    }
    return instance;
  }

  /**
   * Sets the Ndef object within the singleton
   * @param ndef Ndef
   */
  protected synchronized void setNdef(Ndef ndef) {
    NdefSingleton.ndef = ndef;
  }

  /**
   * @return the Ndef object within the singleton
   */
  protected synchronized Ndef getNdef() {
    return NdefSingleton.ndef;
  }

  /**
   * Sets the NdefFormatable object within the singleton
   * @param ndefFormatable NdefFormatable
   */
  protected synchronized void setNdefFormatable(NdefFormatable ndefFormatable) {
    NdefSingleton.ndefFormatable = ndefFormatable;
  }

  /**
   * @return the NdefFormatable object within the singleton
   */
  protected synchronized NdefFormatable getNdefFormatable() {
    return NdefSingleton.ndefFormatable;
  }
}
```

ReadTask.java

```java
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

import java.io.IOException;
import android.nfc.FormatException;

/**
 * IOTask implementation to read NdefMessages
 * @author Johannes Bolz
 *
 */
public class ReadTask implements IOTask{

  private long id;

  public ReadTask(long id) {
    this.id = id;
  }

  /**
   * Reads the NdefMessage from the Ndef object stored in NdefSingleton.
   */
  @Override
  public void doTask() throws IOException, FormatException {
    NdefMessageSingleton.getInstance().addMessage(this.id,
        NdefSingleton.getInstance().getNdef().getNdefMessage());
  }

}
```

RfidDAO.java

```
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;

import android.nfc.FormatException;
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
import android.nfc.Tag;
import android.nfc.tech.Ndef;
import android.nfc.tech.NdefFormatable;
import android.nfc.tech.NfcV;

import org.apache.commons.codec.binary.Hex;
import org.apache.commons.lang.ArrayUtils;
import org.apache.http.util.EncodingUtils;
```

```
/**
 * Data access object class that contains various read/write functions for RFID
 * transponders
 * @author Johannes Bolz
 */
public class RfidDAO {

  private NdefRecord[] records;

  /**
   * @param tag Tag
   * @return a hexadecimal String representation of a Tag's ID or UID
   */
  public String getTagId(Tag tag) {

    // get the ID byte array
    byte[] id = tag.getId();

    if (NfcV.get(tag) != null) {
      ArrayUtils.reverse(id);
    }

    return new String(Hex.encodeHex(id));
  }

  /**
   * @param tag Tag
   * @return true, if the Tag is NDEF formatted
   */
  public boolean isNdef(Tag tag) {
    return (Ndef.get(tag) != null);
  }

  /**
   * @param tag Tag
   * @return true, if the Tag is NDEF formatable.
   */
  public boolean isNdefFormatable(Tag tag) {
    return (NdefFormatable.get(tag) != null);
  }

  /**
   * @param tag Tag
   * @return an ArrayList of NdefMimeRecord objects out of the Tag. Performs a read operation
   * on the NFC adapter.
   * @throws IOException
   * @throws FormatException
   */
  public ArrayList<NdefMimeRecord> getMimeRecords(Tag tag) {
    readRecords(tag);
    return extractMimeRecords(this.records);
  }

  /**
   * Reads URI records from NDEF records
   * @param tag
   * @return an ArrayList of URIs, represented as Strings.
   */
  public ArrayList<String> getUriRecords(Tag tag) {
    readRecords(tag);
    if (this.records != null) {
      ArrayList<String> uriStrings = new ArrayList<String>();
      for (int i = 0; i < this.records.length; i++) {
        if(this.records[i].getTnf() == NdefRecord.TNF_WELL_KNOWN &&
            Arrays.equals(this.records[i].getType(), NdefRecord.RTD_URI)) {
          uriStrings.add(EncodingUtils.getString(ArrayUtils.remove(records[i].getPayload(),
0), "UTF-8"));
        }
      }
      return uriStrings;
    } else {
      return null;
    }
  }


  /**
   * Checks all NDEF records against the CRC value contained in the last record.
   * @return true, if the checksum is correct.
   */
  public boolean verifyMimeRecords() {
    CrcGenerator generator = new CrcGenerator();
    NdefRecord checksumRecord = this.records[this.records.length -1];
    NdefRecord[] payloadRecords = (NdefRecord[]) ArrayUtils.remove(this.records, 
this.records.length -1);
    if (checksumRecord.getTnf() == NdefRecord.TNF_MIME_MEDIA &&
      EncodingUtils.getString(checksumRecord.getPayload(), "UTF-8").startsWith("crc32:")) {
      String checksumStr = EncodingUtils.getString(checksumRecord.getPayload(), "UTF--8");
      long checksum = Long.parseLong(checksumStr.substring(checksumStr.indexOf("crc32:") + 
6));
      return generator.checkHash(payloadRecords, checksum);
    } else {
      return false;
    }
  }

  /**
   * @param tag Tag
   * @return an ArrayList of cached NdefMimeRecords, i.e. doesn't perform a read operation
   * on the NFC adapter. Will return null if the cache doesn't hold a Ndef object.
   */
  public ArrayList<NdefMimeRecord> getCachedMimeRecords(Tag tag) {
    NdefSingleton.getInstance().setNdef(Ndef.get(tag));
    if (NdefSingleton.getInstance().getNdef() != null) {
      if (NdefSingleton.getInstance().getNdef().getCachedNdefMessage() != null) {
        this.records = 
NdefSingleton.getInstance().getNdef().getCachedNdefMessage().getRecords();
        return extractMimeRecords(this.records);
      }
    }
    return null;
  }


  /**
   * @param records Ndefrecord[] array
   * @return an ArrayList of the NdefRecords within records. Will only put MIME formatted
   * NdefRecords in the ArrayList. Returns null if the array is null.
   */
  private ArrayList<NdefMimeRecord> extractMimeRecords(NdefRecord[] records) {
    if (records == null) {
      return null; // TODO Give more specific feedback?
    }
    ArrayList<NdefMimeRecord> recordlist = new ArrayList<NdefMimeRecord>();
    for (int i = 0; i < records.length; i++) {
      // check if NdefRecord is MIME formatted
      if (records[i].getTnf() == NdefRecord.TNF_MIME_MEDIA) {
        recordlist.add(new NdefMimeRecord(records[i].getPayload(), records[i].getType()));
      }
    }
    return recordlist;
  }

  /**
   * Stores a NDEF into the {@link NdefMessageSingleton}.
   * @param tag
   */
  private void readRecords(Tag tag) {
    NdefSingleton.getInstance().setNdef(Ndef.get(tag));
    // Create timestamp for NDEF message.
    long timestamp = (new Date()).getTime();
    // Create ReadTask, which will the message into a Map in NdefMessageSingleton, identified 
by timestamp.
```

```
    ReadTask rt = new ReadTask(timestamp);
    IOThread readThread = new IOThread();
    readThread.run(rt);

    // Retrieve NDEF message out of NdefMessageSingleton by it's timestamp ID, extract
records.
    NdefMessage m = NdefMessageSingleton.getInstance().getMessage(timestamp);
    if (m != null) {
      this.records = NdefMessageSingleton.getInstance().getMessage(timestamp).getRecords();
    } else {
      this.records = null;
    }
  }

  /**
   * Writes an NDEF message on a NFC-compliant RFID transponder.
   * @param tag Tag
   * @param message the NdefMessage to write
   * @throws Exception
   */
  public void writeMessage(Tag tag, NdefMessage message) throws Exception {

    // use NdefSingleton for I/O operations (which is also accessed by the separate Thread)
    NdefSingleton.getInstance().setNdef(Ndef.get(tag));

    // make sure the tag can be written to
      if (!NdefSingleton.getInstance().getNdef().isWritable()) {
        throw new Exception("Tag is read-only!"); // TODO Create custom Exception for that
purpose
      }


      // make sure the NDEF message is neither null, nor larger than the available tag memory
      if (message != null) {
        if (message.toByteArray().length > NdefSingleton.getInstance().getNdef().getMaxSize())
{
              throw new Exception("NDEF message too long!"); // TODO Create custom Exception
for that purpose
              }
        } else {
              throw new NullPointerException();
            }

      // create and run a IOThread that writes the message
      WriteTask wt = new WriteTask(message);
      IOThread writeThread = new IOThread();
      writeThread.run(wt);

  }

  /**
   * Formats an NDEF formatable RFID transponder, so NDEF messages can be written to it.
   * @param tag Tag
   */
  public void formatNdef(Tag tag) {
    // check if Tag is NDEF formattable. If it is, NDEF-format it
    if (NdefFormatable.get(tag) != null) {
      // use NdefSingleton for I/O operations
      NdefSingleton.getInstance().setNdefFormatable(NdefFormatable.get(tag));
      FormatThread ft = new FormatThread();
      ft.run();
    }
  }



}
```

WriteTask.java

```
/*
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package de.berlin.magun.nfcmime.core;

import java.io.IOException;

import android.nfc.FormatException;
import android.nfc.NdefMessage;

/**
 * IOTask implementation to write NdefMessages
 * @author Johannes Bolz
 *
 */
public class WriteTask implements IOTask{

  private NdefMessage message;

  public WriteTask(NdefMessage message) {
    this.message = message;
  }

  /**
   * writes the NdefMessage to the Ndef stored in NdefSingleton
   */
  @Override
  public void doTask() throws IOException, FormatException {
    NdefSingleton.getInstance().getNdef().writeNdefMessage(message);
  }
}
```

ZipFileSystem.java

```
/*
 *    Copyright 2011 by the MAGUN project
 *    http://magun.beuth-hochschule.de
 *    johannes-bolz (at) gmx.net
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software
 *  distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

package de.berlin.magun.nfcmime.core;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.zip.ZipEntry;
import java.util.zip.ZipInputStream;

import android.nfc.FormatException;
```

```java
/**
 * This class provides funtioniality to extract files from a {@link NdefMimeRecord} record
containing a ZIP
 * archive.
 * @author Johannes Bolz
 *
 */
public class ZipFileSystem {
  private NdefMimeRecord record;
  private List<String> files;
  private long size = 0;

  /**
   * @param record the {@link NdefMimeRecord} containing a ZIP archive
   * @throws FormatException if the MIME format isn't 'application/zip'
   */
  public ZipFileSystem(NdefMimeRecord record) throws FormatException {
    if (!"application/zip".equalsIgnoreCase(record.getMimeString())) {
      throw new FormatException("Bad MIME format!");
    }
    this.record = record;
  }

  /**
   * Writes the contained files to the local file system.
   * @param root The path to store the files
   * @throws IOException
   */
  public void write(String root) throws IOException {

    ZipInputStream zInStream = new ZipInputStream(
        new ByteArrayInputStream(record.getPayload()));

    // Extract ZIP entries:
    try {
      ZipEntry entry = null;
          byte[] entryBuffer = new byte[8192];
          int len = 0;
          while ((entry = zInStream.getNextEntry()) != null) {
            String separator = System.getProperty("file.separator");

            // In case there is a folder path specified with the file, create the
            // path before writing the file:
            if (entry.getName().contains(separator)) {
              String pathname = new String(entry.getName().substring(0,
                  entry.getName().lastIndexOf(separator)));
              File path = new File(root, pathname);
              path.mkdir();
            }

            // Write file:
            if (!entry.isDirectory()) {
                File entryFile = new File(root, entry.getName());
                FileOutputStream fos = new FileOutputStream(entryFile);
                while ((len = zInStream.read(entryBuffer)) > 0) {
                    fos.write(entryBuffer, 0, len);
                }
                fos.flush();
                fos.close();
            }
        }
    } finally {
      zInStream.close();
    }
  }

  /**
   * @return The names of the files contained in the archive.
   * @throws IOException
   */
  public ArrayList<String> getFileNames() throws IOException {
    if (files == null) {
      resolveMetadata();
    }
    return (ArrayList<String>) files;
  }

  /**
   * @return the overall uncompressed size of the archive.
   * @throws IOException
   */
  public long getSize() throws IOException {
    if (size == 0) {
      resolveMetadata();
    }
    return size;
  }

  /**
   * Extracts the file names and the data size of the archive.
   * @throws IOException
   */
  private void resolveMetadata() throws IOException {
    files = new ArrayList<String>();

    ZipInputStream zInStream = new ZipInputStream(
        new ByteArrayInputStream(record.getPayload()));
    try {
      size = 0;
      ZipEntry e = null;
      while ((e = zInStream.getNextEntry()) != null) {
        size += e.getSize();
        files.add(e.getName());
      }
    } finally {
      zInStream.close();
    }

  }

}
```

Appendix III: Demonstrator Tags and Project Data