

MobileIoT Toolkit: Connecting the EPC Network to Mobile Phones.

Dominique Guinard, Felix von Reischach
Florian Michahelles and Elgar Fleisch
Auto-ID Labs, ETH Zurich and University of St. Gallen
Sonneggstrasse 63
CH-8092 Zurich, Switzerland
+41 58 871 78 46
dguinard@ethz.ch

ABSTRACT

In this paper we discuss the MobileIoT Toolkit. This software framework offers a number of tools to ease the design and implementation of Java Mobile application prototypes interacting with the Internet of Things. In particular, we focus on mobile phones accessing a “standardized Internet of Things”, known as the EPC Network (Electronic Product Code). In this paper we introduce the EPC Network¹ and then describe the toolkit. Finally, we introduce two applications based on the toolkit.

1. INTRODUCTION

In general terms, Auto-ID techniques enable computers to identify the physical world. In particular the term refers to identifications techniques such as barcodes, RFID (Radio Frequency Identification), biometry, OCR (Optical Character Recognition) and the like.

The relatively new ability for mobile phones to read RFID tags as well as barcodes, makes them the ideal candidates for designing and prototyping all kind of applications involving ubiquitous services. Yet, in order to design such systems in a realistic way, a number of distributed components need to be installed and integrated. This makes the whole prototyping experience rather time-consuming and cumbersome.

A number of mobile service development toolkits do exist already. In [1] Rukzio et al. present a framework for supporting mobile interactions with real world objects. As the MobileIoT toolkit, this framework addresses the various technical steps in the interaction; from the reading of a tag (RFID, NFC, barcode, etc.) on a mobile phone to the invocation of a corresponding service on the server side. In [3] Adelman et al. a toolkit focusing on barcodes is discussed. In [2] Broll et al. propose a similar framework based on Web Services.

While our architecture leverages from these work the usage of the MobileIoT Toolkit (see Figure 6) the novelty of our approach is to offer connectivity to the EPC Network [4], a standardized Internet of Things. The adoption of standards at prototype level enables for the application to be more realistic and convincing. It also permits to build prototypes on top of one another. Finally it makes it easier to turn a research prototype into a real application.

In the MobileIoT Toolkit, we package and assemble the standards and their implementations in order to make them useable without too much overhead.

The EPC Network in a Nutshell

In the early years of the RFID technology the scope of applications was somewhat limited. The applications were rather local (access control, etc.) and did not require carefully thought frameworks to be based upon [5]. The advent of RFID in global business and in particular the vision of an Internet of Things [5], where common objects are part of a global network of information, changed the needs. From being local, Auto-ID applications became global. In 1999, the Auto-ID centers were to foster this challenge. After several years of research the EPC Network concept was born and handed over to EPCglobal to turn research into industry standards [4].

As of today, EPCglobal today represents about 12 major industries and over 50 industry segments ranging from healthcare and life sciences to transportation and logistics, footwear and apparel, and aerospace, automotive and high technology. As a consequence, the uniquely identifying numbering code: EPC (Electronic Product Code) and the standardized server-side components of EPCIS managing track & trace RFID events are the most adopted standards in business and industry (see [7] for further details).

When applied to mobile interactions with the Internet of Things these standards offer the luxury to think in the wide, i.e. about global use cases. Thus, mobile applications connected to the EPC Network take an interesting direction both in research and commercial terms.

In the world of prototyping for the EPC Network, the Accada project [5] prevails. This project proposes an open-source implementation of the EPC Network standards. While based on web services, Accada was not meant to prototype mobile applications. Thus, rather than re-inventing the wheel, the MobileIoT Toolkit integrates parts of the Accada project and provides a set of additional tools for mobile development.

2. MOBILEIOT TOOLKIT

This software toolkit is in essence an aggregation of various components as well as a design methodology. As shown on Figure 6, it is composed of two main parts: the Mobile Tools and the Server Tools, both implemented in Java. Additionally it integrates and abstracts a number of open source libraries.

In general terms the toolkit provides tools for every step required in a typical mobile to IoT client-server application. These steps are listed on the top of Figure 6.

In order to better understand what the tools provide let us imagine a concrete application that enables a consumer to

¹ <http://www.epcglobalinc.org>

retrieve information about the places a product (say an MP3 player) traveled through before arriving at the store.

Mobile Tools

The Mobile tools contain a set of classes that can help the developer implementing the mobile side of the application. The first step (**Tag Access**) addresses the access to identifiers. The contribution here is to provide a unified way of fetching identifiers: the seeds of the Internet of Things!

Ideally, prototypes of mobile to IoT applications should not be restricted to use EPC tags or barcodes or NFC (Near Field Communication) tags. Rather they should be based on hybrid code that enables to switch between the Auto-ID readers in order to test them all and find the most suitable. However, implementing and using various readers is quite time-consuming as these do not implement a common API (Application Programming Interfaces).

We solve this using a simple object-oriented design as shown on Figure 1. The concrete implementation of Auto-ID readers (barcode, NFC, EPC Gen2, etc.) is hidden behind an abstract `IDReader` object that offers the same reading method for every reader (e.g. `readID()`). In the particular case of Figure 1, the `IDReader` interface hides the `UDPLink` and `UDPListener` classes which belong to the implementation of an RFID UHF reader on a mobile phone.

The current toolkit comes with three implemented readers: a 1D Barcode reader based on the BaToo toolkit [3], an NFC reader based on the JSR 257 (Java Specification Request), and more importantly in our case a reader for the E61i EPC UHF prototype from Nokia Research [9].

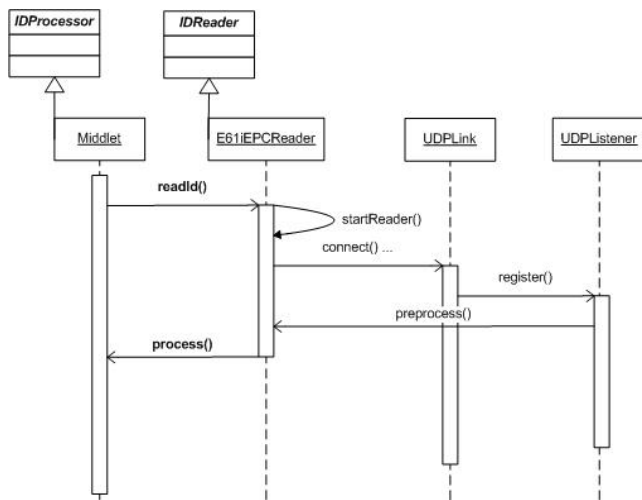


Figure 1: Interactions with the Server-side Software.

At this step in our concrete example, the mobile application read the unique identifier (EPC code) of the MP3 player: 880020EC420FE632.

After accessing the tag, one needs to be able to trigger the corresponding service (**Communication Medium and Service Interface** on Figure 6). We initially planned to achieve this invocation via direct Web Service calls, as Accada’s EPCIS offer such an interface, but faced a challenge: while standards for the invocation of web services within mobile Java applications do exist, (Java Specification Request 172), only few devices fully implement them yet. Thus, a common alternative to invoking services on mobile devices is to use “gateways” based on servlets.

In the MobileIoT Toolkit we provide a mechanism to issue light-weight, but still object-oriented, service consumption through these gateways. As shown on Figure 2, a component called `RESTLikeInvoker` takes care of converting the service calls into simple HTTP GET requests on a servlet.

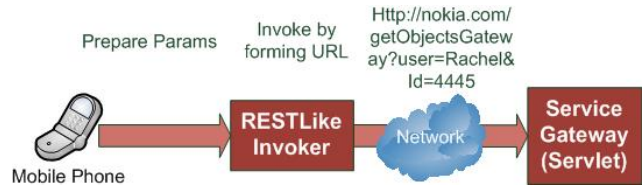


Figure 2: Invocation over the RESTLikeInvoker.

At this step the application on the mobile device sent a request to the EPC Information Server in order to know where the MP3 player comes from.

Server Tools

Next (**Business Connectors** on Figure 6), the toolkit offers the Tag Data Translation component based on Mark Harrison’s implementation of the TDT². It basically offers to translate EPC codes from the form they are read in by a device into a form that is understood by the backend-services. Rather than being one standard the Electronic Product Code is an aggregation and an extension of various well-established standards. Furthermore, an EPC code can have various forms, e.g. binary, hexadecimal, URI, tag-encoding, etc. Since services on the EPCIS accept only one form (the URI or pure-identity in EPC jargon) there is a need for a conversion engine. The EPC polymorph object does this by accepting any form and any supported standard as an input and transforming it into any other form upon request.

In our concrete example we create the EPC object and invoke `getURIFORM()` on it to get the correct form for a service request on the EPCIS: `urn:epc:id:sgtin:0037000.030241.1041970`.

The next step is to invoke the service on the EPC Information Server. The connectors offer this last functionality. They make a subset of simple services available to be consumed by mobile devices using the toolkit. In our case we use the Accada EPCIS Connector.

When replying to a service invocation on the server side we experienced another challenge: we wanted to be able to transport the objects returned by the Accada EPCIS over the network and re-compose them back on the mobile device. This process is commonly called serialization and de-serialization; two unsupported mechanisms in Java Mobile.

Thus, on the server side and the mobile side a set of tools enable for objects to be transformed into strings. These are then sent as a reply to the service request, pretty much like a browser gets an HTML code back when asking for a webpage. This process is summarized on Figure 3.

In our case the EPCIS returns a set of records which are transformed to a single string by the `MobileFormatter` and sent over the network. On the mobile side they are then re-composed to objects corresponding to the places the MP3 players went through prior arriving at the store.

² <http://www.accada.org/tdt/index.html>

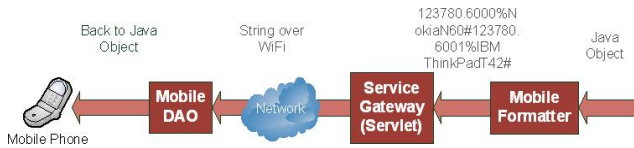


Figure 3: Transforming objects to strings and back to objects.

3. EVALUATION BY PROTOTYPING

Our approach has been to create a prototype of an application requiring interactions with the EPCNetwork first with the goal to extract a set of tools (namely classes) that could help the design of further applications.

Thus, the toolkit is based on the challenges we faced during the implementation of the first prototype: EPCFind [9]. Finally, the application was re-implemented using the toolkit for the sake of evaluation.

EPCFind

EPCFind is a community-based lost-and-found application improving the shortcomings of traditional lost and found approaches [9]. It demonstrates how our mobile companions can help us both tracking our belongings as well as easily and quickly reporting the recovery of objects.

In EPCFind all products are enhanced with an EPC tag. Every user is then given a mobile phone with an EPC reader. In our particular case we used a Nokia E61i extended with UHF EPC RFID reader as a functional cover. Since these mobile devices have a reading range of about 50 centimeters, they can silently (i.e. without explicit human interaction) register tagged objects in their vicinity. One can then use the application to locate where his laptop was last “seen” by the reader of the mobile phone of the owner. This implements the tracking of belongings.

The easy and quick reporting the recovery of items is implemented on a community-based model. Using the EPCFind application on his mobile phone one can report the finding of an object within a few seconds. The finder starts by scanning the object. The application then connects to a lost-and-found server and finds the object’s owner. Finally, the owner is automatically contacted through the EPCFind application on his mobile phone and can arrange a way of sending the lost item back with the finder.

Technically speaking EPCFind uses all the toolkit features exposed before. As shown on Figure 4, it is used to read the identifiers on the mobile phone and to invoke the services on the backend. Furthermore, the EPCFind server is using the EPC Information Server to store the RFID events and query for them.



Figure 5: A screenshot of APriori

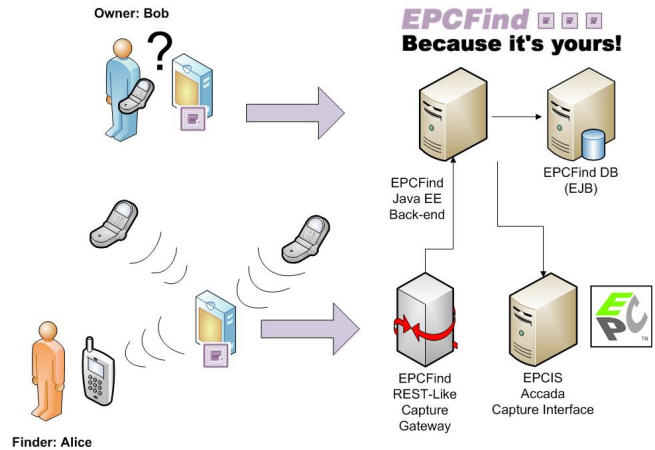


Figure 4: Interactions with the Server-side Software.

While no formal evaluation has been undertaken yet, the EPCFind application was successfully demonstrated during the HotMobile Workshop 2008 [10].

APriori

In the Internet of today, many people use so called product recommendation systems before taking their buying decision. These systems allow consumers to share the experiences they have made with certain products in a community-like fashion.

With APriori, consumers utilize their Auto-ID enabled mobile phone (in our prototype: NFC) to retrieve and write recommendations. On the sales floor, they identify products by scanning associated tags, and based on the identification, they access recommendations made by other users. Having bought and used particular products, a similar procedure allows them to actively rate and recommend these products.

System-wise, APriori consists of a mobile application and a server application, where all recommendations are stored. When developing the APriori prototype, we used the MobilIoT Toolkit to help us realizing the communication between the recommendation server and the numerous mobile application clients.

Our experience in the development of the APriori prototype is that the MobileIoT Toolkit facilitates the development of distributed applications where mobile clients are implemented using Java ME. Using the toolkit, we could work with Java objects on the client side as well as the server side, making the communication through servlets (using strings) transparent.

Future Work

While we already have experienced these applications were easier to design using the toolkit, it was also identified that further work is needed to make it easier to handle. Indeed, in its current state the toolkit rather hints a methodology for programming mobile applications interacting with the Internet of Things and provides a set of tools to achieve that. We believe the toolkit could be further extended in order to completely encapsulate the methodology. That is: one could build a mobile application only by using and extending classes of the toolkit.

Furthermore we want to extend the toolkit to encapsulate more standards such as the Object Naming Service (ONS) used to retrieve authoritative sources of information given an EPC

number, as well as the JSR 257 (Java Specification Request) which encapsulates Auto-ID in a standard fashion.

4. ACKNOWLEDGMENTS

This project was supported by Nokia Research. Our special thanks to Sassan Iraj and his team in Helsinki for their work on the E61i UHF EPC RFID prototype. We would also like to thank the whole Accada team as well as Robert Adelman for his barcode reading module (BaToo).

5. REFERENCES

- [1] E. Rukzio, S. Wetzstein, A. Schmidt: "A Framework for Mobile Interactions with the Physical World.", Wireless Personal Multimedia Communication (WPMC'05), Aalborg, Denmark, 2005.
- [2] G. Broll, S. Siorpaes, E. Rukzio, M. Paolucci, J. Hamard, M. Wagner, A. Schmidt: „Supporting Mobile Service Usage through Physical Mobile Interaction” Pervasive 07 USA, 2007.
- [3] R. Adelman, M. Langheinrich, C. Flörkemeier: "Toolkit for Bar Code Recognition and Resolving on Camera Phones – Jump Starting the Internet of Things" MEIS'06 at Informatik '06, Dresden, Germany, 2006.
- [4] F. Thiesse, F. Michahelles: "An overview of EPC technology", *Sensor Review*, Vol. 26, No. 2. (April 2006), pp. 101-105.
- [5] C. Floerkemeier, C. Roduner, M. Lampe: "RFID Application Development with the Accada Middleware Platform", IEEE Systems Journal, Special Issue on RFID Technology, 2007.
- [6] E. Fleisch, F. Mattern: "Das Internet der Dinge: Ubiquitous Computing und RFID in der Praxis", Springer-Verlag 2005.
- [7] EPCglobal Inc.; "The EPCglobal Architecture Framework", Technical Report, 2007. <http://www.epcglobalinc.org/standards/architecture>
- [8] T. Wiechert, F. Thiesse, F. Michahelles, P. Schmitt, E. Fleisch: "Connecting Mobile Phones to the Internet of Things: A Discussion of Compatibility Issues between EPC and NFC.", AMCIS '07, Keystone, Colorado, USA, 2007.
- [9] D. Guinard, O. Baecker, F. Michahelles.: "Supporting a Mobile Lost and Found Community", MobileHCI '08, Amsterdam, Netherlands, 2008.
- [10] D. Guinard, F. Michahelles, S. Iraj, H. Hirvola, E. Fleisch: "EPCFind: Implementing a Mobile Lost and Found Infrastructure", Demonstration at HotMobile '08, Napa Valley CA, USA, 2008.
- [11] F. von Reischach, F. Michahelles: „APriori: A Ubiquitous Product Rating System“, PERMID'08 at Pervasive 08, Sydney, Australia, 2008.

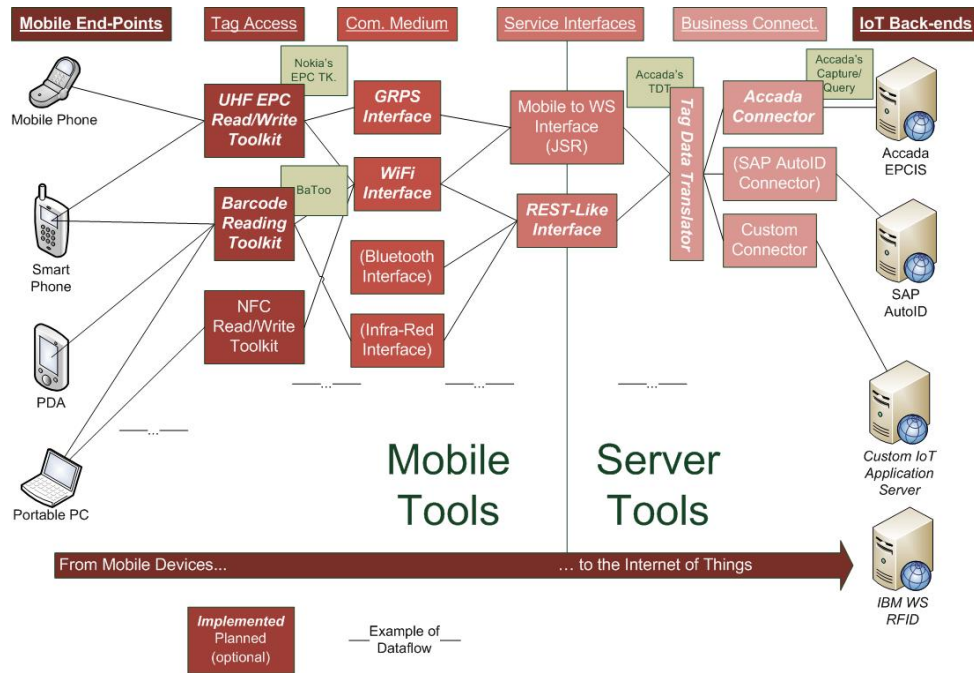


Figure 6: The MobileIoT Toolkit.