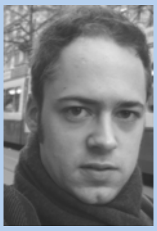
AUTO–ID LABS

Publishing and Discovering Information and Services for Tagged Products

Christof Roduner, Marc Langheinrich

Auto-ID Labs White Paper WP-SWNET-019



Christof Roduner
Institute for Pervasive Computing
ETH Zurich
Auto-ID Lab Switzerland

Marc Langheinrich
Institute for Pervasive Computing
ETH Zurich

Contact:

Institute for Pervasive Computing
ETH Zurich
Clausiusstr. 59
8092 Zurich
Switzerland

E-Mail: roduner@inf.ethz.ch
Internet: www.vs.inf.ethz.ch

Software & Network

Index

Abstract

Radio frequency identification (RFID), and more recently the development of Near Field Communication (NFC) technology, have popularized the idea of linking real-world products with online information and services. Apart from early prototypes, however, the benefits of such automated identification technologies have so far been mostly available to industry, rather than consumers. With the next generation of mobile phones capable of reading both traditional bar codes through their integrated cameras, as well as RFID tags using the NFC standard, end-users themselves could take full advantage of such ubiquitous identification labels, given novel information architectures that go beyond simple web pages or industrial enterprise resource planning (ERP) systems. This paper presents an open lookup infrastructure that allows commercial, public, and private entities to easily provide information and services associated with tagged items, thus facilitating the rapid development and deployment of applications based on everyday products.

1. Introduction

The idea of linking information and services to physical objects has been investigated in many research projects, either using printed markers [1–5], embedding RFID tags [6–8], or even by attaching small infrared beacons [9]. In its simplest form, product identification technology has been widely used as early as in the mid-1970s, when bar code labels began to speed up the checkout process in supermarkets. Today, bar codes have become truly ubiquitous, forming the backbone of many automated processes, such as in airline ticketing and baggage handling, in libraries and video rental shops, in hospitals, and – most of all – in industrial supply chain management.

During the past few years, radio frequency identification (RFID) labels have gradually begun to replace traditional bar code labels, as they offer two distinct advantages: Firstly, RFID labels do not require a line of sight between a reader and a tag, thereby allowing large numbers of tags to be read quickly. Secondly, traditional one-dimensional product bar codes (so-called EAN or UPC bar codes)[1] can only be used to identify products at class-level, while RFID tags allow the identification of individual items (as RFID offers more digits in a smaller area), thus allowing for more detailed product tracking capabilities.

However, so far the benefits of such identification technologies – be it bar codes or RFID tags – have mainly been limited to industry, i.e., manufacturers, distributors, wholesalers, and retailers, who were able to automate many of their logistical processes. This situation may change in the near future, as modern mobile phones are increasingly able to directly read

[1] EAN stands for *European Article Number*, UPC for *Universal Product Code*. They represent the official product identification codes for European and North American products, respectively.

identification tags found on consumer products: Mobile phones equipped with a *Near Field Communication* (NFC, see www.nfc-forum.org) module are already able to read specific types of RFID tags, while recent camera-equipped models can easily decode traditional EAN or UPC bar code symbols found on virtually all consumer goods [10].

While the above-cited projects certainly offer a wide variety of applications for tagged products, they nevertheless assume monolithic, centrally administrated services – such as calling up an online dictionary when putting a bound dictionary on the desk [8], opening a product’s web page [9], or launching an application-specific user interface [4]. However, the general availability of information tags and corresponding reader devices opens up the possibility for novel and innovative applications that cannot be planned for. Having to install and run separate applications and infrastructures for each of the potentially available services for a product (e.g., a price comparison service, allergy warnings, a calorie calculator, or warranty information) would soon overburden users, application developers, and system administrators.

Ideally, an open service infrastructure would allow any party, e.g., manufacturers, consumer interest groups, governmental agencies, or even enthusiastic end-users, to dynamically add services to a specific product or product group, which could then be presented to and selected by consumers right when they scan a product. This paper presents our open lookup infrastructure for tagged items, which allows

1. manufacturers, third-parties, and individuals to publish product-specific resources (i.e., information and services), and
2. consumers to dynamically find and use these resources.

We begin by describing a set of envisioned scenarios and an analysis of the corresponding requirements for our lookup infrastructure in Section 2. The overall architecture of our system is presented in Section 3, with implementation details given in Section 4. Section 5 concludes with a discussion of three prototype applications that we built on top of our infrastructure, demonstrating the value and feasibility of our approach.

2. Application Scenarios and Requirements

Augmenting physical products for end-user lookup is an attractive option, especially for manufacturers. A frozen food company could for example differentiate its frozen spinach by providing an instant “recipe of the week”-suggestion, which customers would be able to access simply by pointing their mobile phone at the product. At the same time, a consumer interest organization could provide background information about the product’s health benefits, e.g., praising it for the organic fertilizer that was used for it, or maybe warning consumers of genetically modified ingredients. Another example scenario for end-user lookup could be a faulty appliance, such as a printer or a coffee maker, which would provide a diagnostic code over an integrated NFC interface. By touching the appliance with a mobile phone, users could pick up this diagnostic information and receive instructions on how to resolve the problem. Alternatively, a list of nearby repair centers could be displayed. For missing consumables (e.g., printing paper, coffee capsules, or filter units), third party stores could offer users one-click reordering options. Last but not least, tagged products could also provide machine-readable instructions for other appliances, which would, e.g., allow a microwave oven to prepare a frozen meal, or a washing machine to warn users when the wrong temperature for a certain garment is selected.

Based on these scenarios, we can derive a number of high-level requirements for an infrastructure that should facilitate the discovery of resources (i.e., information and services) associated with a physical product.

Publication, search, and retrieval of resources

In general, many resources may be linked to a single tagged product at the same time. Our infrastructure must therefore provide mechanisms to store and find these resources. At the same time, an application might want to limit or focus the resources returned when looking up information and services associated with a given tagged product. Such search restrictions could be based on certain topics (e.g., “health aspects”) or certain types of resources to look up (e.g., “expiration date”). Furthermore, the concept of context [11] (e.g., the location or status of the appliance the user interacts with) should be supported as a search criterion.

Openness

As the example scenarios illustrate, a range of stakeholders could have a potential interest in associating resources with a product. We therefore want to allow both product manufacturers and third-parties (such as advocacy organizations, competitors, or individuals) to be able to publish resources for a particular product. Similarly, our system should also allow the sharing of publicly available resources. In line with Lessig’s “Creative Commons” approach [12], we expect the general availability of resources and a supporting infrastructure to prompt the development of numerous innovative applications, as is the case with current Web 2.0 mashups.

Extensibility

In our application scenarios above, resources can be of very different types. Our infrastructure should thus not try to define a limited set of foreseen resource types and their uses. Rather, it should provide extension points that allow third-parties, be it individuals or industries, to come up with their own resource types that can be shared using our system. Additionally, it must be possible to easily integrate existing information systems, such as enterprise resource planning (ERP) systems, to make their data accessible as resources.

Lightweight and secure architecture

Most of the discussed applications run on either mobile devices or embedded systems. Due to the resource constraints typically found on these platforms, we need to employ lightweight protocols in our lookup infrastructure. The open nature of our system mandates the use of security mechanisms. For example, in certain applications users must be able to determine the authenticity of a party providing a resource.

Resource discovery, i.e., finding resources by specifying a set of desired attributes in a distributed environment, has been an active field of research. While many of the protocols in this domain, such as Jini [13], UPnP [14], SLP [15], and Konark [16], focus on ad-hoc networks and do not scale beyond single communities, our application scenarios span larger networks (i.e., the Internet). While INS/Twine [17] seems to be well suited for large networks, it gives resource providers no control over where their resources are stored, which would be a problem for tagged product resolving, as it gives resource providers no control over service quality and cost sharing. UDDI [18] is a discovery service used in the domain of web services, so it should in principle scale well and offer finer control options. However, UDDI is not well suited for the lightweight lookup of simple and small bits of information. Very much related to our work is EPCglobal's EPCIS [19] – a standard for sharing product related information. However, its applicability is limited to the logistics domain. In summary, none of these existing solutions meet our requirements as discussed above.

3. Architecture

In this section, we discuss the architecture and four core concepts of our lookup service: resources and their descriptions, resource registries, a manufacturer resolver service, and search services (see Figure 1).

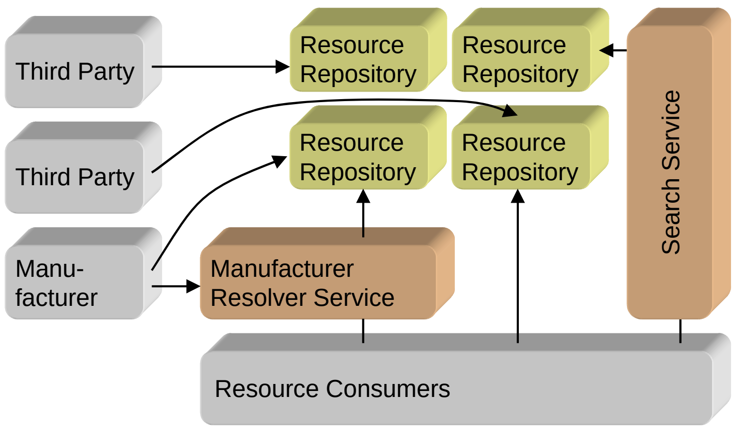


Fig. 1. *Open lookup infrastructure.* The center of our architecture are *resource repositories* containing *resource descriptions*. In order to find one or more repositories, given a particular product tag, mobile devices or stationary appliances either access a known repository (e.g., of their favorite consumer interest group), use the *manufacturer resolver service*, or query generic *search services*.

3.1. Resources and Resource Descriptions

Resources are at the core of our system. They offer information on, or services for, a physical product. Typical examples for resources range from a simple website to complex web services. Resources can be provided by the original product manufacturer or any other party. Resource consumers can be product owners, business partners, or appliances. For every resource, a resource provider must create a *resource description* that specifies all the metadata that is needed to consume the information or service.

Figure 2 shows an example of a resource description. A resource description includes the following main elements:

- The *resource ID* element is a pseudo-random value that serves as a globally unique identifier (GUID) for the resource.
- The *tag ID* element denotes the identifiers of those tags on physical products that a resource is associated with. The tag ID can specify a product at an item- or class-

level. Different numbering schemes, such as EPC[2] and EAN/UPC, are supported. Note that a resource can carry several tag IDs and thus apply to several products. This can be helpful, e.g., when the same (or similar) product is sold under different identifiers.

- The *profile* element can be used to express that the resource adheres to the syntax and semantics that are defined in a certain profile. Typically, a profile will be defined by an industry (e.g., in a standardization group). The food industry could, for example, specify in a profile how the expiration date of a product is to be represented in a resource. Profiles are essential in cases where a resource is not interpreted by humans, but processed by an appliance.[3]
- The *url* element points to the actual resource (e.g., a website). Alternatively, the resource can be stored directly in the *data* element if it is relatively small (e.g., a product's expiration date), which avoids an additional roundtrip. The syntax and semantics of the data available via either the *url* or *data* element are defined by the resource's profile as indicated in the *profile* element.
- If specified, the *context* field defines in which situation the resource is relevant. In order to enable interoperability, we predefined the following context elements that can be used to restrict a resource's applicability: time (date, time, weekday), location (coordinates, city, country), and status (expressed as a simple string) of the appliance the user interacts with. Note that this list is easily extensible by resource providers. Exact values, value ranges, and regular expressions are supported for each context element.
- The *title* and *description* elements describe the resource in natural language.
- Finally, the resource provider can digitally sign the resource description using the optional *signature* element.

```
resource id:  f5f7305bf097af39c68b790d817d7889f788f222
     tag id:  urn:ean.ucc:7610200337481
    profile:  http://foodindustry.org/profiles/expiration-date/
        url:  (empty)
       data:  2007-05-31
    context:  (empty)
      title:  Expiration date
description:  Expiration date for OrganicMilk, 1 liter
  signature:  (empty)
```

Fig 2. Example resource description. **Descriptions can be expressed in various formats, e.g., XML or even binary, depending on the particular communication and storage needs of a product (example given in an abstract format).**

[2] EPC stands for *Electronic Product Code* and is the designated, global successor to both the EAN and UPC numbering scheme. It is administered by EPCglobal.
[3] Note that this element does not actually contain a syntactical or semantical description, but merely serves as an identifier for a format agreed upon by participants, similar to the *Content-Type* field in HTTP.

Figure 2 shows an example of a resource description, in this case describing the expiration date of a particular bottle of milk. Notice that the example is given in a generic format, which in practice can be instantiated in a number of formats, such as XML or even binary form, depending on the particular use case. Also, resource descriptions for food products might equally well be entire data sets (e.g., expiration date, allergy information, country of origin) instead of just a single data item (e.g., expiration date) as in the above example – this can be standardized as needed by the various standard bodies.

3.2. Resource Repository

The resource repository is responsible for storing resource descriptions and making them available to resource consumers. Resource repositories can be deployed by any party interested in offering resources, such as a manufacturer or an advocacy group. In this way, a single resource repository typically contains the descriptions of resources that are thematically related. Operators can flexibly configure access restrictions to their resource repositories. For example, a manufacturer will in most cases run a read-only repository, while a community operated product reviews repository might allow anyone to add or even update resource descriptions (very much like today's Wikis). The same applies to the querying side, where a consumer reviews publisher might limit access to its repository to paying customers only.

The three basic operations offered by a repository are *RegisterResource*, which is used to publish a resource description, *RemoveResource* to delete a published description, and *LookupResource*, which returns the descriptions of those resources matching the query conditions provided by the caller. A query can consist of up to four elements:

- The *tag ID* element must be provided to denote the product for which resources are looked up. A lookup can be performed at both class- and item-level.
- A *profile* element can be indicated to only fetch resources adhering to it.
- A *search term* element can be specified to restrict the resulting resources based on their textual description.
- Using the *context* element, the caller can specify an arbitrary number of context values. Each value must be marked as either a *hint* (favoring resources with a matching context element) or a *requirement* (excluding resources with no matching context element).

A typical lookup request is shown in Figure 3. It shows a request as it could, for example, be sent to a printer manufacturer's resource repository in order to obtain troubleshooting instructions when the printer is in a malfunctioning state. The printer's status code is read by the mobile phone's NFC module and used as context information to narrow down the lookup.

A resource repository can also be configured to allow user feedback on resources. The incorporation of feedback allows community-based applications where the quality of content is controlled by users submitting confidence values for resources. At the moment, we only provide the *SendBinaryFeedback* operation, which can be called by users to express their approval or disapproval of a resource. The order in which resource descriptions are returned by the repository depends on these ratings. Finally, the resource repository can be configured to synthesize resource descriptions of a specific profile using custom-built *wrappers*. Wrappers can be used to integrate existing information systems, such as an ERP, into the lookup infrastructure.

Note that resource repositories are in principle no different from traditional Web servers. Therefore, the same well-established mechanisms for achieving security, reliability, and scalability can be used. For example, a repository could be replicated and made accessible through a load-balancer that routes traffic according to the individual repositories' availability and load.

```
     tag id:  urn:epc:id:sgtin:0652642.800031.400
    profile:  http://appliances.org/troubleshooting-hints/
search term:  (empty)
    context:  status=E683[hint]
```

Fig. 3. *Example lookup request sent to a resource repository.* Based on a particular printer status (as sent through the printer's NFC interface), a user could query directly for information on a particular printer in the context of a "status= E683" code.

3.3. Manufacturer Resolver Service and Search Service

In order to make use of resource descriptions, users must be able to locate the resource repositories containing them. This is the task of the *manufacturer resolver services* and *search service*. They connect a product EPC or EAN/UPC to a resource repository where this product's resource descriptions can be found.

The use cases in which the various deployed resource repositories are accessed by potential resource consumers can be divided into four groups. In the first group, only the product manufacturer's repository is of interest. An example for such a case is a washing machine that checks the handling instructions of every piece of clothing put into it. In the second group, there is a single repository that is used for every lookup. An example for this case is an application that allows a user to check prices offered by other dealers for a physical product at hand. In the third group, a lookup is performed in several repositories at the same time. An example for such a case is a browser application that lets users specify a number of repositories operated by interest groups (e.g., environmental, political, etc.) they care about. The browser would then, for example, display all reviews regarding a product that can be found in the repositories relevant to the user's interests. We envision repository directories similar to the Dmoz Open Directory Project (www.dmoz.org) from which users can pick the repositories they find interesting. In the fourth group, a user wants to search all repositories for resources associated with a given product. This case comes into play when no relevant resources can be found in the repositories the user has registered in his or her browser. In this case, the consumer would simply query his or her favorite search service for relevant repositories. It is clear from these considerations that the architecture needs to include both a *manufacturer resolver service* that links a tag ID to the manufacturer's resource repository and a *search service* to find resources across the boundaries of single repositories.

Why is there only a resolver service for manufacturers? Why not for distributors, vendors, or consumer interest groups? After all, the example scenarios in Section 2 above illustrated that a wide variety of parties might want to offer their descriptions to consumers, each for an equally valid reason. The question of who gets to supply information to a product, i.e., who gets to "define" its properties, is actually highly political. Our system uses a pragmatic approach, inspired both by technology and legal realities. Manufacturers already play a special role in the life of a product. They are responsible for its safety, they supply manuals, organize warranties and repairs, and often also handle its recycling. In many scenarios, manufacturers thus will be legally the main, if not the only, authoritative source for information. From a technical point of view, manufacturers are also much easier to localize, given their (industrial) ID. This is because the current EPC standard (and, to some extent, also the EAN/UPC standard) contain special mechanisms to quickly identify a product's manufacturer from an EPC or EAN/UPC code. Our manufacturer resolver service makes use of this mechanism (see Section 4 below for details), thus ensuring that users can always locate the repository of a product's manufacturer.

All other information and service providers are harder to identify and find. While one could conceive a central registry where all repositories would be registered, this would violate our openness and extensibility principles set forth in Section 2. Instead, we decided to complement our manufacturer resolver with an orthogonal, decentralized, search-based approach, building on existing web search technology. Just as today's web spiders and robots, specific resource search services would crawl repositories, create an index, and answer search queries. A query passed to a search service's *Search* operation consists of the same four elements (tag ID, profile, search term, context – see above) as a *LookupResource* request sent to a single resource repository.[4] Of course, users can also directly access repositories, e.g., of their favorite product review magazine, by manually entering its address, by receiving the address via Bluetooth or SMS, or by finding it in a directory of resource repositories.

3.4. Deployment and Use

How would these architectural parts be used to deploy and/or make use of individual product descriptions? This depends on the individual stakeholder.

[4] While this mechanism could in principle be also applied to the manufacturer's repository, thus eliminating the need for a special manufacturer resolver, we decided to make use of existing resolution mechanisms in order to guarantee users that at least the manufacturer information can be located.

A *manufacturer* would begin with setting up a public, read-only resource repository, e.g., using an add-in to a standard web server. It would then create resources for each of its products – either informational resources such as web pages or user manuals, or service resources, such as a recipe service or a diagnostics program – and prepare corresponding resource descriptions for each of these resources. These would be entered into its resource repository, which in turn would be registered with the manufacturer resolver service.[5]

A *third-party* wishing to provide information for a certain product (e.g., an advocacy organization or even a governmental agency) would start out similarly. After setting up a repository, creating a number of resources and publishing their descriptions in the repository, however, a third party would need to advertise this repository to potential users (as it cannot make use of the manufacturer resolver service). Instead, it would register its resource repository with a search service or repository directory (similar to Yahoo or the Dmoz Open Directory Project), and/or communicate its repository URL to end-users through traditional advertising, e.g., TV, SMS, and print media.

Without any special configuration, *end-users* can always contact the manufacturer's resource repository, which can be found via the manufacturer resolver service, in order to retrieve a list of "official" resources offered for a product. Similarly, they can use a search service to find resources available from third parties that have registered their repository with the search service. Alternatively, they can manually configure resource repositories that they find especially interesting, using the above mentioned out-of-band advertising mechanisms.

As with the World Wide Web, the cost of running our infrastructure is borne by those publishing resources. Parties interested in participating must either set up their own resource repository (i.e., a dedicated machine with a 24/7 Internet connection), or find someone to do so on their behalf (e.g., a hosting company). Just like the Web, our repository infrastructure can be built gradually and without central coordination.

Since anyone can publish arbitrary resources, data quality will become an issue. Until sophisticated search engines that can provide ranked results are available, we expect that word-of-mouth recommendation and independent editorial review (e.g., popular press) will lead to the emergence of a set of resource repositories that are known to provide quality content. Just as it has become standard with websites today, manufacturers and third parties will eventually run and advertise their repositories in both print and electronic media, treating them as yet another means for differentiating their products and services.

Given these characteristics, our approach differs from automatic service discovery as implemented in, e.g., UDDI [18] or E-Speak [20]. In the scenarios addressed by these technologies, selecting the right services is a matter of semantic description and automated matching. In the use cases presented above, however, selecting resources is simpler, as the search scope is limited to entries linked to a certain physical product at hand. We therefore believe that the adoption process for our infrastructure would be considerably faster.

[5] See Section 4 for details on how the manufacturer's resolver service is registered.

4. Implementation

Based on the concepts described above, we implemented a prototype of our resource lookup infrastructure. For each of its three building blocks, the implementation is reviewed in this section.

Resource Repository

Resource repositories are implemented using Java Servlets and a relational database for resource, feedback, and user management. Fulltext search capabilities are implemented using the Apache Lucene search engine. The implementation provides bindings to SOAP, XML-RPC, and REST [21]. Optionally, TLS can be used for increased security.

Manufacturer Resolver Service

Resolving the manufacturer's resource repository is implemented using the Object Naming Service (ONS) [22]. ONS is a global infrastructure that is used as part of EPCglobal's EPC Network to find the EPCIS[6] of a product's manufacturer. It resolves a product's identifier (its EPC number) to a URL pointing to the corresponding EPCIS by leveraging the existing Domain Name System (DNS) infrastructure. The basic principle of ONS is to reverse the elements of an EPC, discard the serial number element, and append ".sgtin.id.onsepc.com" to its string representation. Using standard DNS infrastructure, the resulting domain name (e.g., *000024.0614141.sgtin.id.onsepc.com*) is then queried for "NAPTR" records (a type of record as defined by the DNS specifications), which contain the URL to the manufacturer. We use a custom value (*EPC+ResRep*) in the *service* field of the NAPTR record in order to distinguish our URLs pointing to the manufacturer's resource repository from other data in the ONS (typically URLs pointing to an EPCIS).

Search Service

We believe that indexing of resource repositories is a task that could be best done by already existing web search services such as Google. In our prototype system, we developed a simple search service based on the Apache Lucene search engine. Our search service crawls all registered resource repositories, creates an index, and can be queried using the *Search* operation.

In addition to this, we extended our search service implementation beyond crawling resource repositories. The Internet is full of standard web pages containing information that pertain to physical products. Such information range from product reviews to user guides and blog entries. If we consider such standard web pages as potential resources linked to physical products, we can easily build a search service for these particular resources. Similarly to the Technorati blog search service, we use an empty anchor-tag (i.e., an *<a/>* HTML-element) to mark a web page as being a resource belonging to a certain physical product. A weblog author could for example link a posting to a physical product with EAN number 7610200337481 by including the element *<a href="http://tagged. example. org/tagid/ean/*

[6] EPCIS stands for *EPC Information Services* and is an integral part of the EPC Network. The EPCIS holds logistical information on a product in the EPC-enabled industrial supply chain.

7610200337481"/> into the HTML source code.[7] As most search engines support a *link* operator to find all web pages linking to a given URL, it is possible to leverage these systems to easily find pages marked with such an *<a/>* element. Our original intention was to implement the search service around one of the large Internet search engines. However, as this turned out not to work reliably, we again used Apache Lucene as the underlying search technology. When the search service receives a *Search* request, it internally queries the Lucene search engine, converts the search results into resource descriptions with the *profile* element set to “webpage” and the *url* element set to the respective web page’s address, and returns these resource descriptions to the caller.

Depending on the client’s request, matching resource descriptions found in resource repositories and synthesized from web pages are returned either separately or aggregated. Our search service implementation provides bindings to both XML-RPC and REST.

5. Prototype Applications

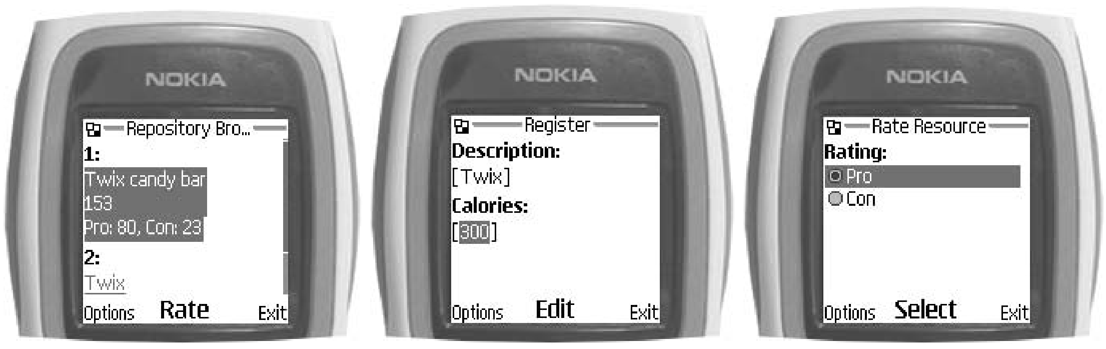
To illustrate the value that our lookup infrastructure offers to the development of applications around tagged products, and to validate our architectural design choices, we built three demonstrator applications. All prototypes were implemented as Java MIDlets on a Nokia 3220 mobile phone. The MIDlets use the REST binding to connect to both the resource repositories and the search service. XML parsing of service responses is implemented using kXML, a lightweight parser for J2ME with minimal memory footprint. Our demonstrators rely on the Nokia 3220’s integrated NFC reader, even though conventional EAN/UPC barcode symbols could be equally used as tagging technology.

5.1. Calorie Tracker

The first demonstrator allows users to track their daily calorie intake (see Figure 4). Calorie information on products is fetched from a user-extendable resource repository. The application demonstrates the possibility of a community-built and -maintained resource repository, by creating new resources and adding feedback to them directly on a mobile phone. To ensure basic quality control, we borrow a concept from community websites and let users approve or disapprove resources created by other users. For every resource, the number of positive and negative votes is recorded and taken into account when resources are ranked in response to a query. If there are no entries for a product, or if a user does not agree with any of the returned values, a new resource can be created. When a user touches a product with the NFC phone, a *LookupResource* request with the acquired tag ID is performed on the “calories repository”. The result contains a list of resource descriptions, consisting of the textual description, the calorie number, and feedback, as partly shown in

[7] Notice how this link does *not* enclose any text, which is how traditional hyperlinks work. Instead, this singular anchor indicates that this entire page applies to the referenced resource.

Figure 4(a). While browsing through the results, the user has the possibility to rate a result. Figure 4(b) shows the form for entering a rating for a resource. If none of the suggestions are correct, the user can add a new resource as shown in Figure 4(c).



Repository Bro...
1:
Twix candy bar
153
Pro: 80, Con: 23
2:
Twix

Register
Description:
[Twix]
Calories:
[300]

Rate Resource
Rating:
○ Pro
○ Con

Fig. 4. *“Calorie Tracker” application.* An example for a community-built and -maintained product repository.

5.2. Shopping Assistant

A second example application provides users with background information on products. Upon scanning a tagged product, the “shopping assistant” contacts three resource repositories: First, the manufacturer to obtain allergy information according to the “allergy” profile that we assume has been defined by the food industry. Second, a repository implementing a wrapper to the product’s price information at Amazon.com. Third, a repository offering information on environmental issues of a given product. Based on the resources obtained from these repositories, the assistant informs the user if the product conflicts with his or her allergy profile, if it is available from Amazon and for what price, and if there are any environmental issues with it. Table 1 shows queries for an example product sent to the three repositories, while Figure 5 illustrates two received responses. All results are aggregated and displayed as shown in Figures 6(a) and 6(b). The Amazon book price resources are automatically created by a custom wrapper that leverages the Amazon Web Services to fetch the current price of books.

Table 1. *Resource repository queries.* Three examples for a shopping assistant (see Section 5.2), trying to find information pertaining to an identified product.

Repository	*LookupResource* elements
manufacturer	`tagid=urn:ean.ucc:9783540240037, profile=allergy`
price information	`tagid=urn:ean.ucc:9783540240037, profile=price`
env. information	`tagid=urn:ean.ucc:9783540240037, profile=review`

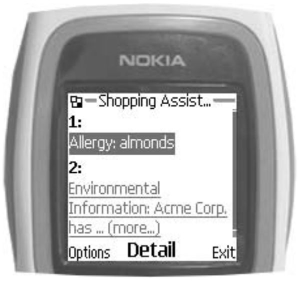
```
<resDescriptions repository="http://repos.allergy.org/">
  <item resId="b5fe3a5bf077af32c68b790d817d7339f724f209">
    <profile>allergy</profile><title>Allergy Information</title>
    <data><almonds/></data>
  </item>
</resDescriptions>

<resDescriptions repository="http://repos.envprot.org/">
  <item resId="73cd125bf097af69c64b790d817d7899f788ffa7">
    <profile>review</profile><title>Environmental Information</title>
    <data>Acme Crop. has repeatedly distributed its toxic waste...</data>
  </item>
</resDescriptions>
```

Fig. 5. *Responses from resource repositories.* These (abbreviated) replies illustrate potential replies to the queries shown in Table 1.

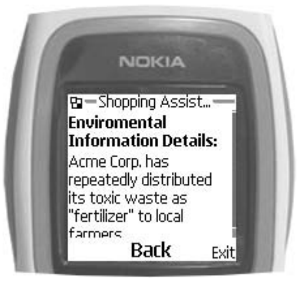
5.3. Appliance Support

Our last application uses context in the form of a status code obtained from a malfunctioning appliance, such as a printer, to find information that can help solve the problem. We use the search service to locate web pages, blog entries, or other sources of information that are marked as relevant to the product at hand in the status encountered. Figure 6(c) shows an overview of the results found for a printer in a certain status. By selecting “Goto”, the user can launch the device’s web browser and open the web page (Figure 6(d)). The two special links that mark the pages that were found as relevant for a product with EAN tag ID “6420256000052” and context “status=3762” were *<a href="http: //tagged. example. org/ tagid/ean/6420256000052"/>* and *<a href="http: //tagged. example. org/context/status/ 3762"/>*, respectively.



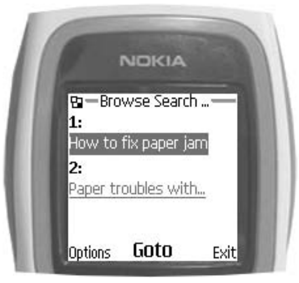
1:
Allergy: almonds
2:
Environmental Information: Acme Corp. has ... (more...)

(a) Assistant



Enviromental Information Details:
Acme Corp. has repeatedly distributed its toxic waste as "fertilizer" to local farmers

(b) Details



1:
How to fix paper jam
2:
Paper troubles with...

(c) Printer Help

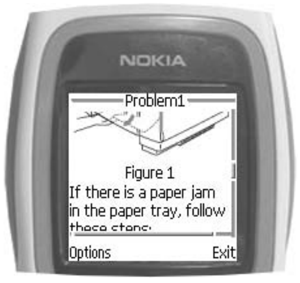


Figure 1
If there is a paper jam in the paper tray, follow these steps:

(d) Help Details

Fig. 6. “Shopping Assistant” and “Appliance Support” applications

6. Conclusion

The idea of linking information and services with physical objects is a powerful concept, especially when we are able to augment millions of everyday products with such resources. Realizing the vision of every product being augmentable raises the question of how interested parties can flexibly associate information and services with a product. We address this issue by presenting the concept and architecture of an *open lookup infrastructure* for resource descriptions that fulfils the requirements derived from a range of example application scenarios. We validated the infrastructure by implementing its key components prototypically. We also implemented three demonstrator applications to illustrate how it facilitates the development of novel applications involving digitally augmented, tagged products. In a corresponding user study [23], our demonstrators received very positive reviews from our test subjects. In summary, our open lookup infrastructure offers four key benefits to the various stakeholders involved. Firstly, it allows users to find out what information and services are available for a physical product. Secondly, it gives resource providers access to potential consumers. Thirdly, it enables manufacturers to increase the value of their products by adding information and services to them. And finally, it provides application developers with concepts and services that facilitate the implementation of novel applications.

References

[1] Ishii, H., Ullmer, B.: Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In: CHI ’97: Proc. of the SIGCHI conference on Human factors in computing systems, ACM Press (1997) 234–241

[2] Ljungstrand, P., Redström, J., Holmquist, L.E.: WebStickers: Using Physical Tokens to Access, Manage and Share Bookmarks to the Web. In: DARE ’00: Proc. of DARE 2000 on Designing augmented reality environments, ACM Press (2000) 23–31

[3] Rekimoto, J., Nagao, K.: The World through the Computer: Computer Augmented Interaction with Real World Environments. In: UIST ’95: Proc. of the 8th annual ACM symposium on User interface and software technology, ACM Press (1995) 29–36

[4] Rohs, M., Bohn, J.: Entry Points into a Smart Campus Environment – Overview of the ETHOC System. In: IWSAWC ’03: Proc. of the 23rd International Conference on Distributed Computing Systems, IEEE Computer Society (2003) 260–266

[5] Smith, M.A., Davenport, D., Hwa, H., Turner, T.: Object AURAs: A Mobile Retail and Product Annotation System. In: EC ’04: Proc. of the 5th ACM conference on Electronic commerce, ACM Press (2004) 240–241

[6] Lampe, M., Metzger, C., Fleisch, E., Zweifel, O.: Digitally Augmented Collectibles. Adjunct Proc. of 8th Annual ACM Symposium on User Interface Software and Technology (UIST), Seattle (2005)

[7] Römer, K., Schoch, T., Mattern, F., Dübendorfer, T.: Smart Identification Frameworks for Ubiquitous Computing Applications. Wireless Networks 10(6) (2004) 689–700

[8] Want, R., Fishkin, K.P., Gujar, A., Harrison, B.L.: Bridging Physical and Virtual Worlds with Electronic Tags. In: CHI ’99: Proc. of the SIGCHI conference on Human Factors in Computing Systems, Pittsburgh, PA, USA (1999) 370–377

[9] Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., Spasojevic, M.: People, Places, Things: Web Presence for the Real World. Mob. Netw. Appl. 7(5) (2002) 365–376

[10] Adelmann, R., Langheinrich, M., Floerkemeier, C.: Toolkit for Bar Code Recognition and Resolving on Camera Phones – Jump Starting the Internet of Things. In: Informatik 2006 workshop on Mobile and Embedded Interactive Systems (MEIS’06). (2006)

[11] Dey, A.K.: Understanding and Using Context. Personal Ubiquitous Comput. 5(1) (2001) 4–7

[12] Lessig, L.: The Future of Ideas: The Fate of the Commons in a Connected World. Random House Inc., New York, NY, USA (2001)

[13] Sun Microsystems: Jini Architectural Overview (1999) *www.sun. com/software/ jini/whitepapers/architecture.pdf*.

[14] UPnP Forum: UPnP Device Architecture (2000) *www.upnp.org*.

[15] Guttman, E.: Service Location Protocol: Automatic Discovery of IP Network Services. IEEE Internet Computing 3(4) (1999) 71–80

[16] Helal, S., Desai, N., Verma, V., Lee, C.: Konark - A Service Discovery and Delivery Protocol for Ad-Hoc Networks. In: IEEE Wireless Communications and Networking Conference (WCNC 2003). Volume 3 (2003) 2107–2113

[17] Balazinska, M., Balakrishnan, H., Karger, D.: INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery. In: Proc. of the First International Conference on Pervasive Computing. Volume 2414 of Lecture Notes in Computer Science, Springer-Verlag (2002) 195–210

[18] UDDI: UDDI Technical White Paper (2000) *www.uddi.org/pubs/Iru_UDDI_ Technical_White_Paper.pdf*.

[19] EPCglobal: EPCglobal Architecture Framework Version 1.0 (2005)

[20] Kim, W., Graupner, S., Sahai, A., Lenkov, D., Chudasama, C., Whedbee, S., Luo, Y., Desai, B., Mullings, H., Wong, P.: Web E-Speak: Facilitating Web-Based E-Services. IEEE MultiMedia 9(1) (2002) 43–55

[21] Fielding, R.T., Taylor, R.N.: Principled Design of the Modern Web Architecture. ACM Trans. Inter. Tech. 2(2) (2002) 115–150

[22] EPCglobal: Object Naming Service (ONS) Specification Version 1.0 (2005)

[23] Roduner, C., Langheinrich, M., Floerkemeier, C., Schwarzentrub, B.: Operating Appliances with Mobile Phones – Strengths and Limits of a Universal Interaction Device. In: Proc. of Pervasive 2007. LNCS, Springer (2007)