Middleware Building Blocks for Architecting RFID Systems

Nikos Kefalakis[1], Nektarios Leontiadis[1], John Soldatos[1], and Didier Donsez[2]

[1] Athens Information Technology
19.5 Km Markopoulou Ave. 19002 Peania, Greece
{nkef,nele,jsol}@ait.edu.gr
[2] Université Joseph Fourier Grenoble
Avenue Centrale, Domaine Universitaire, 38041, Grenoble, France
Didier.Donsez@imag.fr

Abstract. RFID middleware is a cornerstone of non-trivial RFID deployments in complex heterogeneous environments. In this paper we present the principal middleware building blocks specified in the scope of the EPCglobal architecture. Alternative protocols and implementation frameworks for realizing these middleware blocks are also presented. At the same time we outline several middleware extensions to the EPCglobal architecture, towards meeting common requirements of automatic identification applications. Furthermore, we classify RFID applications into various categories based on their complexity, as well as based on their closed or open loop nature. Accordingly, we highlight the middleware blocks that are most important to each application category.

Keywords: RFID, Middleware, Architecture, EPCglobal, Business Event Generation.

1 Introduction

RFID middleware is gradually becoming the cornerstone of non-trivial RFID deployments in complex heterogeneous environments (e.g., logistics, supply chain management) comprising multiple readers, applications instances, legacy ICT systems, as well as sophisticated business processes and semantics. In such environments many distributed readers and antennas (e.g., in factories, warehouses, and distribution centers) capture RFID data, which must accordingly be conveyed to a variety of applications (e.g., enterprise resource planning (ERP) systems, warehouse management systems (WMS), corporate databases, process management systems).[1] Deployment and integration complexity are directly associated with the flexibility and versatility of the RFID middleware towards configuring and managing multiple heterogeneous devices, filtering and disseminating RFID data, translating low-level RFID data to high-level business semantics, as well as towards integrating RFID systems with legacy ICT systems and applications [5].

F. Granelli et al. (Eds.): MOBILIGHT 2009, LNICST 13, pp. 325–336, 2009.

The typical information flow within an RFID middleware system involves:

- Collecting RFID data from the physical readers, through reading the tagged items. At this level middleware implementations insulate higher layers from knowing what reader /models have been chosen. Moreover, they achieve virtualization of tags, which allows RFID applications to support different tag formats.
- Filtering the RFID sensor streams according to application needs, and accordingly emitting application level events. At this level middleware implementations insulate the higher layers from the physical design choices on how tags are sensed and accumulated, and how the time boundaries of events are triggered.
- Mapping the filtered readings to business semantics as required by the target applications and business processes. At this level middleware implementations insulate enterprise applications from understanding the details of how individual steps in a business process are carried out.

Software that implements any combination of these information flows can be conceived as an RFID middleware.

The above information flow is reflected in the EPCglobal architecture [2], where it is implemented based on the EPC-RP (Reader protocol), EPC-LLRP (Low-Level Reader Protocol), EPC-ALE (Application Level Events) and (EPC-IS) (Information Sharing) protocols and specifications [15]. Hence, the EPCglobal architecture specifies a middleware framework for a broad class of RFID applications. However it lacks some features that are extremely handy for many automatic identification applications. In this paper we present both the EPCglobal middleware layers, as well as additional middleware features, which are not completely covered by EPCglobal.

In this paper we introduce a middleware architecture (devised in the scope of the EC co-funded project ASPIRE [3]) which extends the EPCglobal architecture. To this end, we use the ASPIRE architecture to illustrate the various middleware layers and their possible implementations. Note that the proposed architectures have been devised in order to cover large scale open loop fully fledged RFID applications in the scope of inter-enterprise scenarios. Nevertheless, we are currently witnessing the proliferation of less complex closed loop applications, which can be implemented based on cut down versions of the proposed architectures. These applications require subsets of the presented middleware blocks as discussed in later sections.

The rest of this paper has the following structure: Section 2 discusses briefly the limitations of the EPCglobal architecture and introduces the ASPIRE architecture. This architecture is decomposed to the middleware building blocks dealing with readers and tags virtualization in Section 3, to filtering and collection blocks in Section 4 and Section 5 deals with the middleware blocks for addition of business context to RFID sensor streams. Section 6 classifies RFID applications into various categories and underlines the middleware building blocks that are relevant for each category. Finally, section 7 draws basic conclusions.

2 RFID Systems Architecture

The EPCglobal along with associated middleware implementations (see [4] for a comprehensive review) are subject to several limitations, some of which are inherent

to the EPC architecture. Specifically, the most prominent of these limitations relate to the following areas [4]:

- Configurable Business Events Generation: Current middleware implementations do not provide support for configurable and automated translation of filtered data (i.e. ECReports) to business events (i.e. EPCIS Events). RFID developers are therefore still required to allocate programming effort in mapping ALE outputs to information sharing constructs. We strongly believe the configurable interpretation of RFID readings in a specific business context should be an essential functionality of any RFID middleware suite.
- Support and integration for sensor data: In addition to identifying objects many applications (e.g., cold chain management) need to detect and consume physical measurements (e.g., temperature, humidity, weight, acceleration (for shock-tracking), lighting). Hence, middleware frameworks must to provide the means to integrate sensors and accordingly make their data accessible by the applications. EPCglobal covers mainly the coding of things identifiers. While ALE reports can include (as extensions) physical measurements acquired by RFID sensor tags or sensors attached to the environment (e.g., RFID interrogator, container) at reading time, current middleware frameworks do not provide support for the consumption of these metrics. This is they do not cater for aligning the coding of these measurements with main international units, quantities standards and specifications (such as ISO 31-0, JSR 275, Open Geospatial Consortium GML, Google KML). Middleware frameworks must therefore provide support for adapting and using sensor readings in accordance to these coding schemes.
- Integration of Actuators: Experience with automatic identification applications manifests that there is often a need to quickly interact with the physical world based on a wide range of actuating functions such as locks, LEDs or mechanical controllers. Hence, RFID middleware frameworks need to be enhanced with actuator control frameworks.
- Reader Connectors and Virtualization: EPC-RP and EPC-LLRP prescribe reader protocol standards aiming at achieving vendor independence. In the current reader landscape however, there are still many readers that do not fully support these protocols. As a result there is still a need to provide an adaptation layer for non EPC-RP or EPC-LLRP compliant readers, similar to the HAL (Hardware Abstraction Layer) implementation of the Accada project for EPC-RP [1],[6]. Most important, a middleware suite should include a uniform interface for communicating with upstream EPC layers (e.g., ALE).
- End-to-End Management: Non-trivial RFID solutions are supported by highly heterogeneous infrastructures comprising multiple tags, readers, sensors, as well as a host of middleware components and servers. Managing such an infrastructure end-to-end is certainly asset towards facilitating the deployment and operation of RFID solutions. The EPC architecture and related middleware products emphasize on single reader management (e.g., based on the Reader Management Protocol) and do not support complete end-to-end management of the RFID solutions.
- Programmability and (Visual) Integrated Development Environments: Integrated development environments (IDEs) and visual tools are a key prerequisite to boosting RFID implementation. Most OSS RFID platforms do not provide complete

integrated environments enabling visual development of RFID applications, which only few exceptions that are still in their infancy [7], [8]. In order for RFID deployment to go mainstream, complete IDEs enabling RFID consultants and business users to configure standards based solutions through minimal programming effort are urgently required.

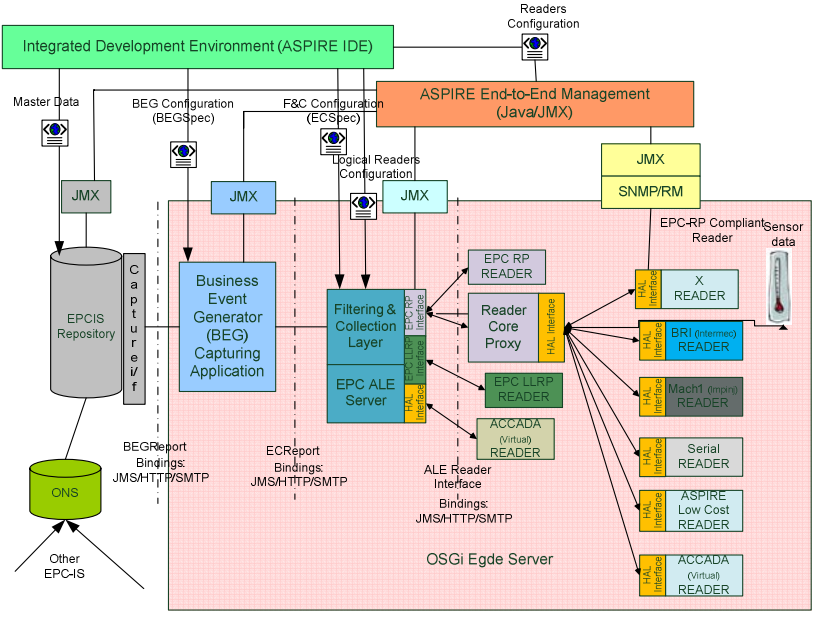

Fig. 1. ASPIRE Middleware Architecture

Driven by the above requirements and EPC limitations, the FP7 ASPIRE project has devised the middleware architecture depicted in Figure 1. It is based on the EPC architecture, but augments it with support for sensor data, end-to-end management, actuator control, as well as automated business context configuration functionality. These functionalities have been implemented in the scope of the AspireRfid [9] OSS project. Note that both the ASPIRE architecture and the AspireRfid project capitalize on lightweight container technologies, notably Open Services Gateway Interface (OSGi) (www.osgi.org) compliant for integrating and bundling the various middleware components comprising the architecture. In addition to being lightweight, an OSGi container constitutes a dynamic module system, which allows the deployment of various middleware blocks (described in later sections) as modules that can be flexibly (even at runtime) installed, started, stopped, activated, deactivated and updated. As a result, an OSGi based deployment facilitates the end-to-end management requirement, which is implemented based on JMX (Java Management Extensions) technology.

3 Readers and Tags Virtualization

3.1 Tags Virtualization

Tag virtualization capitalizes on a machine-readable version of the EPC Tag Data Standards specification [15]. This machine-readable version can be used for virtualizing underlying RFID tags through bridging and mapping different representations. Hence, a tag translation module is a (standalone or embedded) middleware component that enables the interpretation of machine readable version of the tag. The interpretation can be used in automated fashion. In addition the tag translation specification can be used for validating machine readable formats of the EPC tags. The TDT engine built in the scope of the AspireRfid [9] project supports the ISO15693, ISO14443, ISO15961, ISO15962, ISO15963, various GS1 formats (EAN/UPC, GS1 DataBar, GS1-128, ITF-14, GS1 DataMatrix, and Composite Component), as well as Bar Codes 1D and 2D: Note that barcode support is deemed particularly important given the vast number of legacy barcode applications, which need to be interoperable with emerging RFID applications.

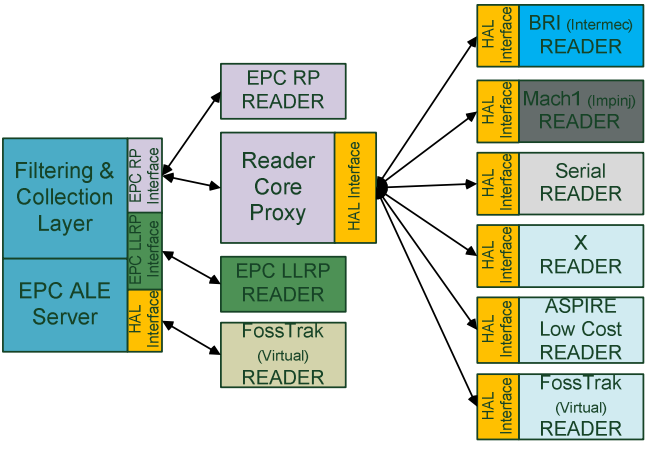

Fig. 2. Reader Virtualization Concept

3.2 Readers Virtualization

The role of the reader virtualization layer is to unify the way we interact with the miscellaneous hardware, by inserting a hardware abstraction layer and providing a fixed instruction set to the higher layers which require information from the hardware.

Specifications exist that satisfy the need for a norm at this level; namely the EPCglobal Reader Protocol (RP) and the EPCglobal Lower Level Reader Protocol (LLRP). These protocols define the standard bindings through which an application can send messages in a standardized format, as described in relevant standards [15].

Towards achieving reader virtualization a Hardware Abstraction Layer (HAL), ensuring a graceful mapping of the standardized messages to the low-level vendor specific reader communication primitives is specified. The methods of communication between the HAL and the hardware itself will vary, depending on the hardware vendor and it may require a serial connection, an Ethernet connection, etc. The protocols of communication may also vary from a raw TCP connection, to SSL and HTTP. The same will apply for the command and message encodings, which may be text, XML or binary.

The layers above the HAL exchange messages that conform to a well-defined format – XML or text – using a set of standard network interfaces – Serial, TCP and HTTP. Any combination of the aforementioned is allowed. Figure 2 depicts the reader virtualization concept.

4 Filtering and Collection (F&C)

RFID technology when used in a large scale deployment generates an enormous number of object reads. Many of those reads represent non-actionable "noise." To balance the cost and performance of this with the need for clear accountability and interoperability of the various parts, the design of the ASPIRE Architecture (Figure 1) seeks to:

- Drive as much filtering and counting of reads as low in the architecture as possible.
- Minimize the amount of "business logic" embedded in the Tags.

The Filtering and Collection Middleware by applying EPC ALE (Application Level Events) [15] is intended to facilitate these objectives by providing a flexible interface to a standard set of accumulation, filtering, and counting operations that produce "reports" in response to client "requests." The client will be responsible for interpreting and acting on the meaning of the report. The client of the ALE interface may be a traditional "enterprise application," or it may be new software designed expressly to carry out an EPC-enabled business process but which operates at a higher level than the "middleware" that implements the ALE interface. Section 6 later in this paper elaborates on different deployment configurations depending on the application scale and nature.

The ASPIRE filtering & collection middleware represents a single interface to the potentially large number of readers that make up an RFID system deployment. This allows applications to subscribe to a specific already defined specification, which is then used along with the Logical Reader definition to configure the corresponding reader devices using the EPC global reader protocol (RP) or low level reader protocol (LLRP) (Figure 2).

Once the readers capture relevant tag data they notify the middleware which combines the data arriving from different readers in a report that is sent according to a predetermined schedule to the subscribed applications. Since the middleware receives data from multiple readers, it provides specific filtering functionality depending on the already defined specifications. Redundant read events from different readers observing the same location are not included to the dispatched report, which accomplishes the

reduction of filtering and delivers the level of required aggregation to the registered application(s) interpreting the captured RFID data.

The interfaces chosen to be used between the filtering & collection middleware and host applications is TCP/HTTP for the notification channel transferring XML reports and SOAP for the server operation programming (ECSpecifications definition/subscription, logical reader definition).

The primary data types associated with the ALE API (Application Programming Interface) are the ECSpec, which specifies how an event cycle is to be calculated, and the ECReports, which contains one or more reports generated from a single activation of an ECSpec. ECReports instances are both returned from the poll and immediate methods, and also sent to notification URIs when ECSpecs are subscribed to using the "subscribe" method of the specification.

An ECSpec describes an event cycle and one or more reports that are to be generated from it. It contains a list of logical Readers whose read cycles are to be included in the event cycle, a specification of how the boundaries of event cycles are to be determined, and a list of specifications each of which describes a report to be generated from this event cycle. There are two ways to cause event cycles to occur. A standing ECSpec may be posted using the define method. Subsequently, one or more clients may subscribe to that ECSpec using the subscribe method. The ECSpec will generate event cycles as long as there is at least one subscriber.

ECReports is the output from an event cycle. The essence of an ECReports instance is the list of ECReport instances, each corresponding to an ECSpec instance in the event cycle's ECSpec. In addition to the reports themselves, ECReports contains a number of "header" fields that provide useful information about the event cycle.

The ALE interface revolves around client requests and the corresponding reports that are produced. Requests can either be: (1) immediate, in which information is reported on a one-time basis at the time of the request; or (2) recurring, in which information is reported repeatedly whenever an event is detected or at a specified time interval. The results reported in response to a request can be directed back to the requesting client or to a "third party" specified by the requestor.

5 Business Context and Information Sharing

Adding business semantics to the low-level sensor streams is a key prerequisite to added-value deployments of RFID technology. For RFID to go mainstream companies must be offered tools and techniques for describing their RFID enabled business processes, without engaging in the low-level implementation details. To this end, a framework specification for describing business events must be provided. This framework should enable the description of business processes based on high-level semantics, which at the same time should be amendable by tools.

The EPC-IS framework [10] is standardized as an integral layer of the EPCglobal architecture. Its main function is to insulate enterprise applications from understanding the details of how individual steps in a business process are carried out at a detailed level. EPC-IS defines a data model for events associated with the lifetime of uniquely identified objects. As already outlined these events are industry and application agnostic. In this sense EPC-IS is a cross industry framework, which allows for

industry specific vocabularies and extensions. Furthermore, the framework is a supplement to (and not a replacement for) existing enterprise information systems. Specifically, EPC-IS events are used to push/pull events to/from other enterprise systems such as ERP (Enterprise Resource Planning Systems), WMS (Warehouse Management Systems) and corporate databases.

EPC-IS can operate in the scope of a layered service-oriented architecture, through persisting supply chain events in a repository and accordingly sharing these events with internal and external applications. The sharing is accomplished through interfaces for capture and query of event data. EPCIS data (i.e. events) are represented as records of activity happening in real world. These events provide context ("What, where, when, why") and proliferate as more business is transacted. EPCIS events are interpreted based on descriptive information (so called "master data"), which provides context for the events such as descriptions of locations, products and business transactions. Note that "mater data" grow at different timescales comparing to EPCIS events. In particular, master data grow slowly as companies grow, not as more business is conducted.

EPCIS events described within the specification can be classified as follows (Figure 3):

- Object Events, which correspond to observations of a collection of EPCs during a specific business step at a specified Location & Time.
- Aggregation Events, which reflect a physical association of a set of EPCs with a parent EPC along with a business step at a Location & Time.
- Quantity Events, which correspond to statements about an object Class (not individual objects), including a quantity, a Location & Time.
- Transaction Events, which records objects associated with a wider business transaction.

Having these events at hand, consultants, researchers and engineers can use them to describe RFID enabled business processes [11]. The starting point is the documentation of the business requirements, comprising the archetypical use cases. Accordingly, it is important to break each use case into a series of discrete business steps. Each one of these steps needs to be modeled as an EPCIS event, according to the above mentioned core EPCIS event types. In rare cases a new type could be defined i.e. when existing types are not sufficient to describing a business step. Note that it is important to define any necessary extension fields, as well as the full range of vocabularies that populate each field. Furthermore, fixed lists of identifiers with standardized meanings for concepts like business step and disposition must be provided, along with rules for population of user-created identifiers like read point and business location.

A novel characteristic of the ASPIRE architecture (Figure 1) in terms of business context handling is the introduction and implementation of the "Business Event Generator" (BEG) middleware module. This middleware module undertakes the automated and configurable mapping of reports (stemming from the F&C module) to EPCIS events. This automation will greatly simplify the development of capturing applications (according to the EPCglobal architecture).

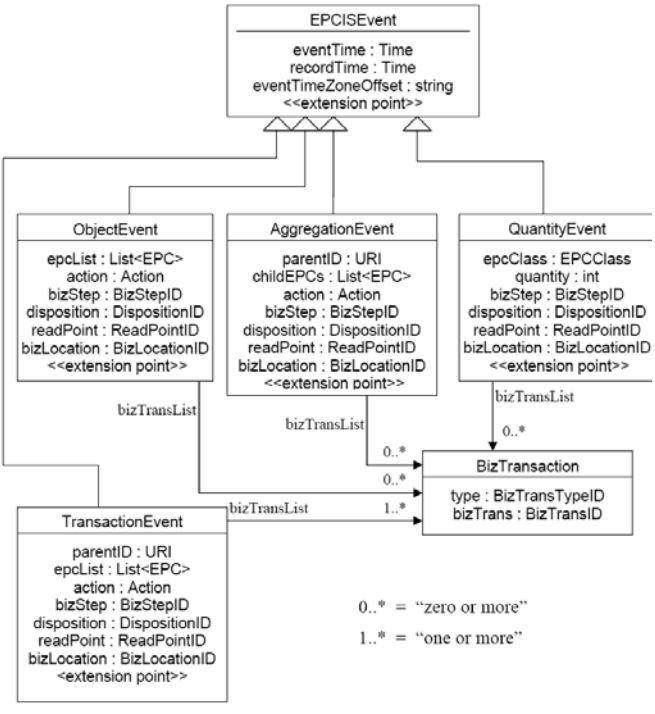

Fig. 3. Core EPCIS event types and the object-oriented relationship between them

6 Application Classification

The proliferating RFID applications, pilots and deployment vary in functionality and scale. The RFID vision was initially articulated through the specification of large scale "open loop" systems and deployments. Prominent examples of such systems are those developed and trialed by Wall-Mart and the U.S Department of Defense (DoD). A main characteristic of these systems is that they span different locations across multiple companies and/or organizations. Note that these trials manifested several problems, both technical (e.g., information sharing, interoperability and scalability at a large scale) and business ones (e.g., business model related issues). Large scale deployments can provide a crash test for protocols like EPCIS and ONS (Object Naming Service) [15].

Following these early complex and visionary deployments, the RFID community has gradually starting to dispel the hype (and its associated complexity). Hence, during the last couple of years we are witnessing a proliferating number of smaller scale solutions covering a wide range of asset tracking and inventory management scenarios, as well as other ROI (return-on-investment) generating case studies. These case studies focus on very specific business problems, which an AIS (automatic identification system) can solve even a single enterprise. A main characteristic of the smaller scale deployment is also the fact that tagging occurs at the case and pallet level

tagging rather than item level. Furthermore, tracking, traceability and identification occur within a warehouse or a single supply chain. Note that these smaller scale solutions are also a reality for RFID vendors, which are gradually refocusing their strategies towards smaller-scale opportunities. Nevertheless, the vision still exists, since small applications (i.e. closed loop islands) could one day become integrated into larger scale open loop systems.

We believe that middleware developers and RFID consultants should prioritize middleware modules development based on the scale of the target application. Open loop solutions must pay emphasis on implementing the full range of middleware layers described in this paper. On the other hand smaller scale closed loop systems must prioritize the F&C, reading and tag virtualization building blocks. Moreover, for some very simple systems our experience shows that custom filters over a HAL for the target reader(s) could provide a rapid and acceptable solution. Table1 presents our view regarding the middleware building blocks that are required to implement each of the above application categories. This is based on our experience with RFID implementations and demonstrations across diverse deployments of varying scale (e.g., [9], [12], [13]).

Table 1. RFID application Classification and Middleware Building Blocks

Application Type/Middleware Block	HAL	EPC-RP, EPC-LLRP	F&C	Business Context
Simple	Yes	Recommended	Recommended	No
Simple Closed Loop	Yes	Yes	Recommended	No
Complex Closed Loop	Yes	Yes	Yes	Recommended
Open Loop	Yes	Yes	Yes	Yes

7 Conclusion

In this paper we have presented the middleware components and layers, which are commonly implemented in RFID applications. The EPCglobal architecture, as well as its extensions in the scope of the ASPIRE architecture provide a general middleware framework that can address the needs of many RFID applications. We argue however that the full range of middleware layers and building blocks are necessary only in the scope of large scale open loop middleware implementations. Simpler applications can leverage cut down versions of these architectures, towards economizing on performance overhead as well as implementation complexity and cost. Specifically, trivial applications can be implemented via customized filtering mechanisms on top of HAL layers or event the EPC-LLRP and EPC-RP protocols. Also, a wide range of closed loop intra-enterprise scenarios could be implemented without a need for sophisticated information sharing layer. Overall, we think that lightweight low-overhead implementations are essential for the smooth transition to fully fledged RFID deployments. This could reinforce a ‘start small, think big’ approach towards the Internet of Things (IoT)

vision. The open source AspireRfid project of the OW2 (www.ow2.org) community provides distinct implementations of the various building blocks, in order to enable researchers and developers to gradually leverage the various middleware functionalities, as required by their target deployments.

Acknowledgments. Part of this work has been carried out in the scope of the ASPIRE project (FP7-215417). The authors acknowledge help and contributions from all partners of the project.

References

1. Floerkemeier, C., Roduner, C., Lampe, M.: RFID Application Development with the Accada Middleware Platform. IEEE Systems Journal 1(2), 82–94 (2007)
2. Architecture Review Committee, The EPCglobal Architecture Framework, EPCglobal (July 2005), http://www.epcglobalinc.org.
3. The ASPIRE FP7 Project, http://www.fp7-aspire.eu
4. Cezon, M., Vaudaux-Ruth, G., Laurens, L., Soldatos, J., et al.: Review of State-of-the-Art Middleware. ASPIRE Project Public Deliverable D2.1 (June 2008)
5. Sarma, S.: Integrating RFID. ACM Queue 2(7), 50–57 (2004)
6. The Accada RFID Middleware Project, http://www.accada.org
7. The Rifidi project, An open source IDE for RFID, http://www.rifidi.org
8. Sun's JCAPS (Java Composite Application Platform) for RFID, http://java.sun.com
9. AspireRfid project, http://wiki.aspire.objectweb.org/xwiki/bin/view/Main/WebHomewiki
10. EPC Information Services (EPCIS) Version 1.0, Specification, EPCglobal (April 2007), http://www.epcglobalinc.org
11. BEAWebLogic RFID Enterprise Server™, Understanding the Event, Master Data, and Data Exchange Services, Version 2.0, Revised: October 12 (2006)
12. Zarokostas, N., Dimitropoulos, P., Soldatos, J.: RFID Middleware Design for enhancing traceability in the Supply Chain Management. In: The Proc. of the 18th IEEE Personal Indoor and Mobile Radio Communications, Athens, Greece, September 3-7 (2007)
13. Rudametkin, W., Touseau, L., et al.: NFCMuseum: an Open-Source Middleware for Augmenting Museum Exhibits. In: IEEE International Conference on Pervasive Services (ICPS 2008), Sorrento, Italy, July 6-10 (2008) (Public demonstration)
14. Lampe, M., Floerkemeier, C.: High-Level System Support for Automatic-Identification Applications. In: Maass, W., Schoder, D., Stahl, F., Fischbach, K. (eds.) Proceedings of Workshop on Design of Smart Products, Furtwangen, Germany, March 2007, pp. 55–64 (2007)
15. EPCglobal standards, http://www.epcglobalinc.org/standards

Abbreviations

ALE – Application Level Events
BEG – Business Events Generator
EC –European Committee

ECReport – Event Cycle Report
ECSpec – Event Cycle Specification
EPC – Electronic Product Code
EPC-ALE - Electronic Product Code Application Level Events
EPC-IS - Electronic Product Code Information Service
EPC-LLRP - Electronic Product Code Low Level reader protocol
EPC-RP - Electronic Product Code Reader Protocol
ERP – Enterprise resource Planning
HAL – Hardware Abstraction Layer
ICT – Information and Communications Technologies
JMX – Java Management Extensions
ONS – Object Naming Service
OSGi – Open Service gateway Initiative
OSS – Open Source Software
RFID – Radio Frequency Identification
WMS – Warehouse management System