Pacific Asia Conference on Information Systems (PACIS)

PACIS 2009 Proceedings

Association for Information Systems *Year* 2009

RFID Context Data Management: The Missing Link to EPCIS-Based Supply Chain Monitoring

Christoph Tribowski* Christoph Goebel†

Oliver Günther‡

*Humboldt-Universität zu Berlin, christoph.tribowski@wiwi.hu-berlin.de

†Humboldt-Universität zu Berlin

‡Humboldt-Universität zu Berlin

RFID CONTEXT DATA MANAGEMENT: THE MISSING LINK TO EPCIS-BASED SUPPLY CHAIN MONITORING

Christoph Tribowski
Institute of Information Systems
Humboldt-Universität zu Berlin
Spandauer Str. 1, 10178 Berlin, Germany
christoph.tribowski@wiwi.hu-berlin.de

Christoph Goebel
Institute of Information Systems
Humboldt-Universität zu Berlin
Spandauer Str. 1, 10178 Berlin, Germany
christoph.goebel@wiwi.hu-berlin.de

Oliver Günther
Institute of Information Systems
Humboldt-Universität zu Berlin
Spandauer Str. 1, 10178 Berlin, Germany
guenther@wiwi.hu-berlin.de

Abstract

The potential of Radio Frequency Identification (RFID) for increasing supply chain efficiency has been stressed by practitioners and researchers alike. For the cross-company exchange of RFID-related data, the industry consortium EPCglobal has specified the EPC Information Services (EPCIS). According to EPCglobal, all RFID-related data recorded by an organization should be stored as EPCIS events in a dedicated database. The information that can be inferred from the EPCIS events stored in the EPCIS repositories will be sufficient to monitor the flow of goods. However, generating an EPCIS event does not only require the data that is provided by the RFID readers but also the corresponding context data (e.g. physical locations, related business process steps and related transactions). For this missing link, i.e. the association of read events and context data, we propose an architectural component called Event Capturing Application (ECA). In this paper we propose a data model for storing and exchanging EPCIS context data in an efficient and standardizable way. We also present an algorithm that can be used to assemble EPCIS events from read events and context data. The functionality of our prototypical ECA was evaluated using noFilis' RFID middleware CrossTalk and the Fosstrack EPCIS implementation.

Keywords: RFID, Context Data Management, Supply Chain Management, Prototype.

1 INTRODUCTION

The potential of Radio Frequency Identification (RFID) for increasing supply chain efficiency has been stressed repeatedly by practitioners and researchers alike (Niederman et al. 2007). The accuracy and detail of RFID data are expected to open up new and more efficient ways to steer the supply chain and reduce many of the inefficiencies plaguing today's businesses such as wrong deliveries, shrinkage and counterfeiting. Since production and distribution of physical goods are seldom in the hands of one organization and efficiency gains in supply chains oftentimes emanate from centralized supply chain control, it often makes sense to share selected RFID data among supply chain participants.

For the cross-company exchange of RFID reader events, the industry consortium EPCglobal has specified EPC Information Services (EPCglobal 2007). According to EPCglobal, all RFID-related data recorded by an organization should be stored in an EPCIS repository in the form of EPCIS events. EPCIS events are contextually enriched RFID reader events; they consist of reader and context data. The information that can be inferred from the EPCIS events stored in the EPCIS repositories along the supply chain enables item level visibility of the flow of goods.

Realizing the vision of EPCIS-based supply chain monitoring implies the existence of software components that connect the RFID reader infrastructure to the ERP systems of organizations in order to do two things: (1) make EPCIS events usable for higher level applications, and (2) generate EPCIS events from read events and the required context data in the first place. In our opinion, efficient ways to perform both of these tasks are critical for the success of the EPCIS. Moreover, they are similar in most companies and therefore standardizable. Only with a standardization of context enrichment is it possible to decouple the RFID reader infrastructure and middleware from the ERP layer. A clear separation of ERP systems and RFID-related systems including the standardization of the corresponding interfaces is a precondition for efficient competition on the emergent RFID market – it provides opportunities for smaller vendors who do not offer RFID middleware alongside ERP systems.

Numerous technical articles have been published on RFID and the so-called Internet of Things in recent years. Most of these publications either focus on the working of RFID reader protocols, i.e. the architectural layer below the EPCIS, or they investigate applications that access EPCIS data, i.e. the architectural layer on top (Beier et al. 2006, Dada and Magerkurth 2008, Goebel and Tribowski 2008, Kim and Kim 2007).

Although the type of context data that should be associated with RFID reader events is well known since EPCglobal's definition of the EPCIS, there are only a few publications that address the management of context data. Nguyen et al. (2007) developed an extended Entity Relationship (ER) diagram for EPCIS data and described the characteristics of event and master data in detail. They reviewed other EPCIS data models, especially the ones used in industry solutions such as the Siemens and Sun RFID middleware. Although this work provides an in-depth insight into EPCIS event and master data, it implements the EPCIS specification without any contribution to EPCIS context data management. Kim et al. (2006) present a contextual event framework for RFID systems. Their goal is the transformation of RFID reader events into so-called contextual events. Indeed, their Contextual Event Assistant (CEA) is able to receive read events and also provides an API for high-level RFID applications that may query for the contextual events generated by the application; however, their application is supposed to retrieve reference data from the EPCIS, ONS and other pre-existent data sources in order to enrich read events with the necessary context data. This view contradicts the idea of EPCglobal to create one common format and access structure for contextualized events that can be used by higher level applications. For this reason, Kim et al.'s CEA does not comply with the proposed architecture of the EPCglobal network. A real-world application of the whole EPCglobal network is described by No (2008). For the architectural layer below the EPCIS, a middleware has been developed, which associates the observations with other information such as packaging containment and manufacturing meta-data; however, it is only stated that the association was done

prior to the recording. The functional and technical description on how this is done is not the focus of the paper.

The companies and research institutions that are involved in the definition of standards for the Internet of Things have done an excellent job in defining which context should be associated with RFID data and which format for the interchange of RFID data is most appropriate. However, to date they have not extensively addressed the issue of context data management including the retrieval and storage of all necessary context data outside EPCIS repositories. If EPCglobal's EPCIS specification is to be implemented, context data from sources such as ERP systems will have to be added at two architectural levels: the level above and below the EPCIS (see Figure 1).

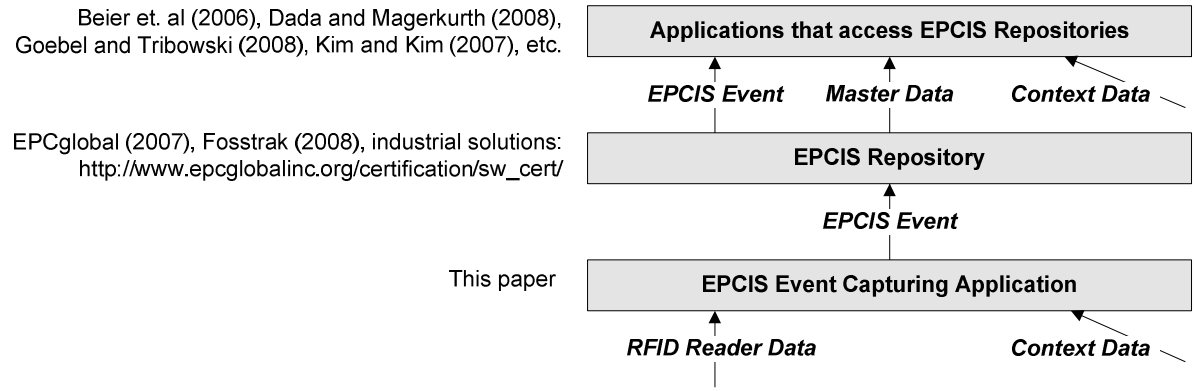

Figure 1. Scope of our contribution.

Above the EPCIS, higher level applications such as Supply Chain Event Management (SCEM) tools or cross-company pedigree applications have to be able to interpret EPCIS events based on pre-existent background knowledge about the supply chain. This context data may include information on admissible process step sequences, durations, and shipping quantities. EPCglobal has defined an additional query interface for retrieving the context data stored in an EPCIS separately using so-called master data queries (EPCglobal 2007, pp. 89-92 and 106-110). Although this can be seen as a first step towards realizing standardized access to context data, the data referred to as 'master data' in the EPCIS specification stems from the EPCIS and its scope does therefore not exceed the information inferable from the sum of EPCIS events and will not be sufficient to realize applications like SCEM.

Below the EPCIS, all the data received from the RFID readers has to be associated with the context data required for generating EPCIS events. In this paper, we will address possible ways to standardize context data management *below* the EPCIS level or *before* EPCIS events are interpreted by higher level applications. The rest of the paper is organized as follows: in Section 2 we analyze the typical functionality and requirements of an EPCIS event capturing application, propose a corresponding system architecture including an in-depth explanation of all components and interfaces, and finally provide a comprehensive use case. In Section 3 we outline the experiences gained from a prototypical implementation of the proposed architecture, before we conclude in Section 4.

2 EPCIS EVENT CAPTURING ARCHITECTURE

Realizing the vision of EPCIS-based supply chain monitoring implies the existence of a number of architectural components. These components should play together in order to guarantee a fast flow of the data stored on RFID transponders all the way to the decision maker who is steering the supply chain.

In the following sections, we concentrate on the technical aspects of EPCIS event capturing. In particular we focus on the definition of interfaces as well as efficient data storage and interchange schemes. The first step from raw RFID data to usable decision making information is the transformation of transponder signals to RFID reader events. Among other things multiple reads, i.e. signals from the same transponder that are received repeatedly, are removed in this step. The outcomes of this process according to the EPCglobal architectural specification are the so-called

Application Layer Events (ALEs). An Application Layer Event contains a list of EPCs transmitted by RFID transponders, the IDs of the readers that have received the transponder signals, and the time when the signals have been recorded. ALEs are usually generated by a read process that is started by an external trigger (e.g. a forklift passing a light barrier) and listens for incoming transponder signals until the reader is switched off by another trigger (e.g. a timer).

The generation of ALEs is contained in the typical functionality of RFID middleware. The solutions of a number of vendors include interfaces for the drivers of RFID hardware, facilities for steering and synchronizing large numbers of readers, and data tools to aggregate and export large volumes of data in different formats.

The process that takes ALEs as input and provides EPCIS events as output has so far neither been described in the literature nor standardized. Clearly, all vendors of RFID middleware who claim to offer EPCIS compatibility must have implemented this process in one way or another. We refer to the application in charge of this transformation as the EPCIS Event Capturing Application (ECA). As we mentioned in the introduction, the standardization of the ECA represents an important task left to be tackled since it allows for a transparent decoupling of the reader infrastructure management and the EPCIS.

2.1 Requirements

The implementation of the ECA has to fulfil three requirements: (1) it has to be fully compatible with EPCglobal's specification; (2) context data storage and exchange has to be done in an efficient and standardized way; and (3) the assembly of EPCIS events from the time of observation by RFID readers to the storage of EPCIS events in the corresponding repository has to be fast.

(1) Compatibility with the EPCIS specification implies in particular that the ECPIS ECA is able to handle all core event types. Four EPCIS event types have been defined by EPCglobal: the most general event is the object event which simply contains context information pertaining to one EPC. An aggregation event serves to describe the physical aggregation of several child EPCs to one parent EPC, e.g. the association of products packed in a box. A quantity event counts the number of objects that belong to a certain class of EPCs without saving their individual EPCs. Finally, the transaction event describes the association of one or more EPCs to a business transaction (e.g. purchase orders and invoices). In Section 2.3, we evaluate the compatibility requirement by presenting an example on the basis of the EPC Showcase (GS1 Germany 2008). It was jointly developed by GS1 Germany and Oracle and gives insights into the EPCIS concept by illustrating both the internal and the cross-company flow of data in a use case with a producer and distributor.

(2) The context data processed by the Event Capturing Application should be stored in an efficient manner and accessible via standardized interfaces. In Section 2.2, a corresponding database schema and data exchange formats are described in full detail.

(3) The timely access to EPCIS events is of essence for EPCIS-based supply chain control applications since the corresponding decisions may have to be made in real time. Therefore the matching of reader and context data has to be carried out very quickly; in view of the high data volumes that will be served by the RFID middleware, the access to all data items required for the assembly of EPCIS events has to be highly efficient.

2.2 Proposed System Architecture

Figure 2 provides an overview of the envisioned system infrastructure. The focus of the following details of our proposed ECA is on the specification of its major components including the Context Data Repository, the Rule Engine, and the required interfaces.

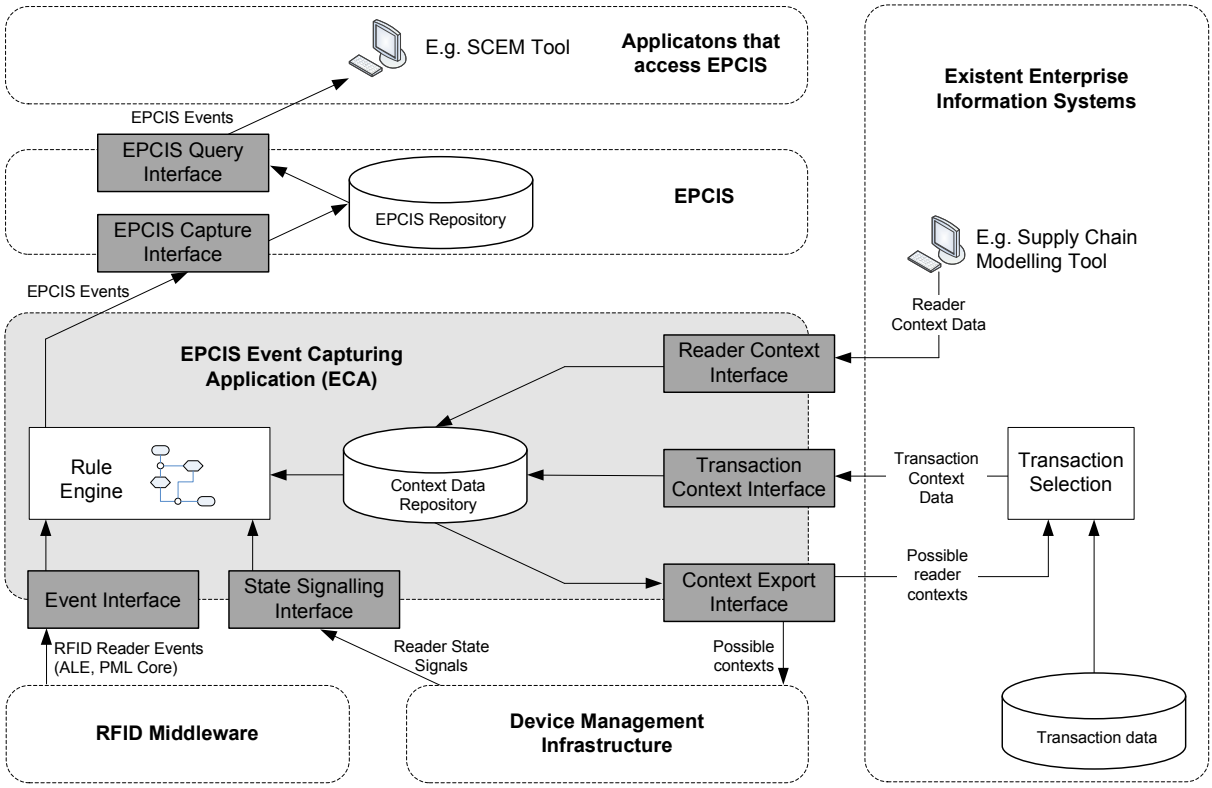

Figure 2. Proposed ECA embedment in system architecture.

In order to introduce the proposed system components in a structured manner, we will sketch the prototypical process of EPCIS event generation in the following. For the sake of clarity, the ECPIS event generation process can be separated into four separate sub-processes. In reality, these sub-processes usually execute in parallel.

The outcome of sub-process one is a cohesive model of the reader context. A straightforward way to generate the reader context is using a dedicated Graphical User Interface (GUI) that allows the supply chain manager to model a supply chain using predefined elements such as facility layouts and RFID readers and to interactively define the required context parameter values (bizStep, bizLocation, etc.). Initial work on such a GUI can be found in Cambridge University et al. (2007). For small, well arranged RFID infrastructures, adding this data manually using a text editor is also possible. The generated data is relatively stable. It only changes when new physical infrastructure is added or removed and the virtual representation needs to be updated consequently. We refer to the data generated in this phase as Reader Context Data. This data is sent to the Reader Context Interface in the XML format defined by the XML schema provided in Figure 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
   <xsd:element name="ReaderContextData">
      <xsd:complexType>
          <xsd:sequence>
             <xsd:element ref="Reader" maxOccurs="unbounded"/>
          </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="Reader">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="URN" type="xsd:anyURI"/>
            <xsd:element name="ReaderContext" maxOccurs="unbounded">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="validFrom" type="xsd:dateTime"/>
                  <xsd:element name="validTo" type="xsd:dateTime"/>
                  <xsd:element name="bizStep" type="xsd:anyURI"/>
                  <xsd:element name="bizLocation" type="xsd:anyURI"/>
```

```
21                    <xsd:element name="action" type="xsd:string" minOccurs="0"/>
22                    <xsd:element name="disposition" type="xsd:anyURI"/>
23                    <xsd:element name="eventType" type="xsd:string"/>
24                  </xsd:sequence>
25                </xsd:complexType>
26              </xsd:element>
27            </xsd:sequence>
28          </xsd:complexType>
29      </xsd:element>
30    </xsd:schema>
```

Figure 3. XML Schema for Reader Context Interface.

While the Reader Context Interface stores Reader Context Data in the Context Data Repository, a unique identifier for the context of a reader is assigned. For each RFID reader which is identified with a URN, more than one valid context can be given at a time. This is reflected by the possible repetition of the Reader element (Figure 3, row 6). The data elements describing the Reader Context (Figure 3, rows 19-23) are derived from the Core Event Types of the EPCIS specification and from the requirement to accurately preserve historical data. The latter is made possible by the elements *validFrom* and *validTo* (Figure 3, rows 17-18). Contexts that are no longer valid do not have to be overwritten this way, but can be kept in the database.

The second sub-process starts with the export of the possible reader contexts stored in the Context Data Repository to the existent ERP system of the company. This is done via the Context Export Interface. The corresponding XML schema is provided in Figure 4.

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
03     <xsd:element name="PossibleReaderContexts">
04        <xsd:complexType>
05            <xsd:sequence>
06               <xsd:element ref="Reader" maxOccurs="unbounded"/>
07            </xsd:sequence>
08        </xsd:complexType>
09     </xsd:element>
10     <xsd:element name="Reader">
11        <xsd:complexType>
12           <xsd:sequence>
13              <xsd:element name="URN" type="xsd:anyURI"/>
14              <xsd:element name="ReaderContext" maxOccurs="unbounded">
15                <xsd:complexType>
16                  <xsd:sequence>
17                    <xsd:element name="contextID" type="xsd:string"/>
18                    <xsd:element name="validFrom" type="xsd:dateTime"/>
19                    <xsd:element name="validTo" type="xsd:dateTime"/>
20                    <xsd:element name="bizStep" type="xsd:anyURI"/>
21                    <xsd:element name="bizLocation" type="xsd:anyURI"/>
22                    <xsd:element name="action" type="xsd:string" minOccurs="0"/>
23                    <xsd:element name="disposition" type="xsd:anyURI"/>
24                    <xsd:element name="eventType" type="xsd:string"/>
25                  </xsd:sequence>
26                </xsd:complexType>
27              </xsd:element>
28           </xsd:sequence>
29        </xsd:complexType>
30     </xsd:element>
31  </xsd:schema>
```

Figure 4. XML Schema for Context Export Interface.

The ERP system is now responsible for matching the transaction data with the context data received from the Context Data Export Interface of the ECA. For instance, a rule could be applied that associates all customer orders with all reader contexts that contain ‘shipping’ as value of the bizStep variable and ‘distribution center: goods issue’ as value of the bizLocation variable. The generated Transaction Context Data is sent to the Transaction Context Data Interface of the ECA in XML format defined by the XML schema provided in Figure 5. The ReaderContext element (Figure 5, row 19-26) describes a business process in the supply chain by associating a number of Reader Contexts with one or several transactions each; this information only changes when business processes or RFID

hardware allocations change. The element TransactionEPCList allows for associating arbitrary physical objects to individual executions of a process (e.g. associating several physical products with a particular customer order).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
   <xsd:element name="TransactionContextData">
      <xsd:complexType>
         <xsd:choice>
            <xsd:sequence>
               <xsd:element ref="Reader" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:sequence>
               <xsd:element ref="TransactionEPCList" maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:choice>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="Reader">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="URN" type="xsd:anyURI"/>
            <xsd:element name="ReaderContext" maxOccurs="unbounded">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="contextID" type="xsd:string"/>
                  <xsd:element ref="bizTransaction" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="bizTransaction">
      <xsd:complexType>
         <xsd:simpleContent>
            <xsd:extension base="xsd:anyURI">
               <xsd:attribute name="type" type="xsd:string" use="required"/>
            </xsd:extension>
         </xsd:simpleContent>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="TransactionEPCList">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="value" type="xsd:anyURI"/>
            <xsd:element name="EPCList">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="URN" type="xsd:anyURI" maxOccurs="unbounded"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
         <xsd:attribute name="type" type="xsd:string" use="required"/>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Figure 5. XML Schema for Transaction Context Interface.

Upon the receipt of XML data the Transaction Context Interface stores the corresponding contexts in the Context Data Repository. The selection of the appropriate transactions is not included in the proposed ECA specification because it heavily depends on company specialities and thus cannot be standardized. The third sub-process of the EPCIS event generation process first exports the available context data to the Device Management Infrastructure via the Context Export Interface. The corresponding XML schema has already been provided in Figure 4. This data can be processed further and provided as input for the devices that interact with workers or other computerized systems. Based on the context data received via the Context Export Interface of the ECA, the Device Management Infrastructure is responsible for continuously reporting the current contextual state of the RFID readers via the State Signalling Interface. The corresponding XML schema is provided in Figure 6.

The selection of the appropriate reader context by the Device Management Infrastructure is not included in the proposed ECA specification. Similar to the selection of applicable transactions by the ERP system (see above) it heavily depends on company specialities and thus cannot be standardized.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
   <xsd:element name="ReaderStateSignals">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="contextID" type="xsd:string" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Figure 6. XML Schema for State Signalling Interface.

The fourth sub-process of the EPCIS event generation process is responsible for the import of reader data from the RFID middleware via the Event Interface. Because we are implementing the EPCglobal network with all its specifications, at a minimum we require the ALE format to be supported. Additionally, the event data interface should accept events delivered in the widespread Physical Markup Language (PML Core) format created by the Auto-ID Center. The corresponding XML schemas are provided in EPCglobal (2008) and Auto-ID Center (2003).

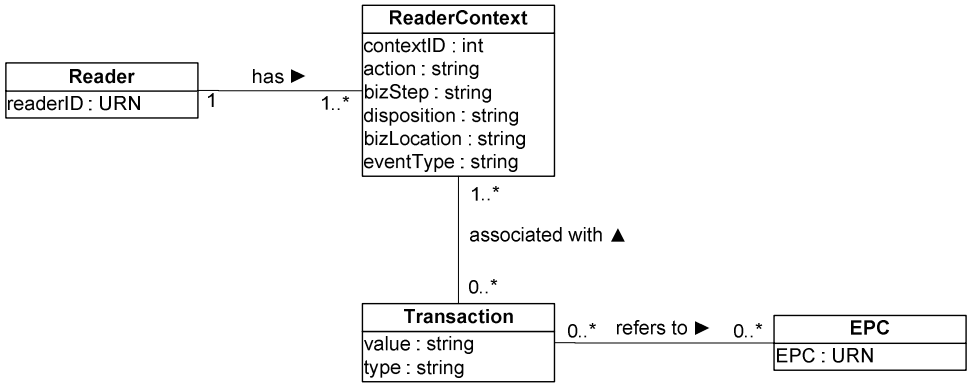

Figure 7. Data model for the Context Data Repository.

The data model of the Context Data Repository is provided in Figure 7 using the UML notation. We recommend the usage of a relational database to store this data. The data model reflects the scope and amount of context data stored in the Context Data Repository. For each reader, more than one context is allowed. There are two reasons for this: firstly, because one reader can be associated with several types of events, i.e. for one read event several EPCIS events (e.g. object and transaction) can be saved in the repository; secondly, because the outdated contexts have to be kept in the database as well. Each transaction can include a list of EPCs. One EPC in turn can be referred to by several transactions.

The component responsible for the assembly of read events and context data to EPCIS events and exporting them to the EPCIS repository is referred to as Rule Engine. The best manner to describe the working of the rule engine once a read event was imported by the ECA is to present the detailed algorithm. Because of programming-language independence, we use pseudo-code (see Figure 8).

As the pseudo-code of the EPCIS event assembly algorithm indicates, the biggest complexity is introduced by the association of the transactions. There are several reasons for this: an EPCIS event can contain more than one transaction; an EPC can be referred to by more than one transaction; and one RFID reader can capture more than one EPC at a certain time that are not necessarily referred to by to the same transaction. Therefore, the algorithm adds all transactions to an event as long as all EPCs in the current list of EPCs also belong to the first transaction; otherwise a new event is created and the remaining transactions are attached to the newly created event.

Upon reception of new Reader State Signal:

- Mark the corresponding Reader Context in the repository as active

Upon reception of new RFID Reader Event:

- For all active Reader Contexts that are related to the reader ID of the RFID Reader Event
  - Retrieve all context data from the context data repository
  - For all transaction IDs that are associated with the reader ID from the RFID Reader Event and refer to the EPCs from the RFID Reader Event
    - Prepare an EPCIS event of the type defined in the context data
    - Add all EPCs referred to by the current transaction in the EPCs from the RFID Reader Event
  - If the EPC lists from several EPCIS events are equal
    - Merge those transactions and delete all but one EPCIS event
  - If EPCs from the RFID Reader Event remain that have not been added to an EPCIS Event
    - Prepare an EPCIS event of the type defined in the context data
    - Add all EPCs from the RFID Reader Event that have not been added to an EPCIS Event
  - If type of EPCIS Event is Aggregation Event:
    - Identify the parent EPC and add all other EPCs from the RFID Reader Event to the list of child EPCs
    - Add all remaining data from the RFID Reader Event and the context data from the context data repository
  - Else if type of EPCIS Event is Quantity Event:
    - Identify the EPC class and add the number of all EPCs to it
    - Add all remaining data from the RFID Reader Event and the context data from the context data repository
  - Add all remaining data from the RFID Reader Event and the context data from the context data repository
  - Save the EPCIS events in the EPCIS repository using the EPCIS Capture Interface

Figure 8. EPCIS Event Assembly Algorithm of the Rule Engine.

An important requirement stated in Section 2.1 was that the matching of reader and context data has to be carried out very quickly and that the assembly of EPCIS events has to be highly efficient. Our algorithm requires the execution of two simple SQL queries in a relational database containing a limited amount of datasets. In comparison to the event data, the reader context data and the transaction context data is quite stable. In the best case, there is only one matching reader context to the read event and no transactions so that the EPCIS event can be assembled with a few instructions. In the average case, there is more than one matching reader context but still no transaction so that for each context one EPCIS event can be assembled with a few instructions. In the worst case, transactions have to be additionally added to the EPCIS event: if EPCs are read that belong to different transactions, the read event has to be split up into several EPCIS events. This procedure needs a few more instructions, but is still straightforward since the number of different transactions is naturally limited.

2.3 Exemplary Scenario

In this section we provide a comprehensive exemplary scenario to clarify the working of the ECA. The example is based on the EPC Showcase (GS1 Germany 2008). It refers to the receiving and shipping business step at a distribution centre. There is one RFID gate installed at the entrance of the distribution centre. Products that are received by or shipped from the distribution centre have to pass through this gate. Two light barriers installed at each side of the gate indicate the direction in which

the gate is crossed. When a shipment arrives, its completeness is checked against the corresponding delivery note; when a shipment leaves the distribution centre, the ERP system automatically creates a delivery note. The tagged products being shipped are associated with the transaction 'purchase order' at both business steps.

Context Data for the Example	
ID of the RFID reader	urn:epcglobal:fmcg:loc:4023339.00005.0
Event Type	object event
Business Steps	urn:epcglobal:epcis:bizstep:received urn:epcglobal:epcis:bizstep:shipped
Transaction type (purchase order)	urn:epcglobal:fmcg:btt:po
Transaction value	http://gs1-germany.distribution/po/222222
EPCs referred to by transaction 222222	urn:epc:id:sgtin:4000001.099999.1 urn:epc:id:sgtin:4000001.099999.2
Event Data for the Example	
Read EPCs	urn:epc:id:sgtin:4000001.099999.1 urn:epc:id:sgtin:4000001.099999.2 urn:epc:id:sgtin:4000001.099999.3
Time of RFID Reader Event	2006-10-20T15:14:58+0200
ID of the RFID reader	urn:epcglobal:fmcg:loc:4023339.00005.0

Table 1. Example Data.

Before the ECA can process incoming event data in this scenario, the given contexts have to be imported via the Reader Context Interface. The Reader Context Data in particular contains the association of the business steps to the reader IDs, i.e. the association of the two business steps 'received' and 'shipped' to the reader ID of the entrance gate of the distribution centre.

In the given scenario, a transaction 'purchase order' has to be included into the EPCIS Events to be generated. The corresponding Transaction Context Data therefore has to be imported via the Transaction Context Interface. The selection of relevant Transaction Context Data can be performed based on the Reader Context Data that has previously been exported from the Context Data Repository via the Context Export Interface. Based on this information, the ERP system can infer which business steps have been defined and which RFID readers are currently used to monitor these business steps. As the data in Table 1 shows, the context selection can be done based on the business steps 'received' and 'shipped' and their association with the given reader ID. There exist two applicable Reader Contexts for the gate reader, one context for the 'receipt' and the 'shipped' business step respectively. The decision which context to associate with which RFID Reader Event is done based on the current Reader State updated by the Reader State Signals received via the State Signalling Interface. In the example scenario the state signals determining the applicable business step are sent to the ECA by the light barriers installed on both sides of the distribution centre's entrance. If the light barrier at the inside of the distribution centre is crossed, the rule engine marks the 'shipping' context; otherwise the 'receiving' context gets activated.

When the RFID reader at the gate captures the tagged objects, the RFID Middleware generates an RFID Reader Event containing the three EPCs listed in Table 1. The RFID Reader Event is pushed to the Event Interface of the ECA, which triggers the corresponding process of the Rule Engine. Following the algorithm provided in Section 2.3, the active context stored for the reader ID of the mobile RFID reader is fetched from the Context Data Repository. The context data listed in Table 1 requires the creation of an Object Event, which is created and filled with the data from the RFID Reader Event and the active context data, in particular the applicable business steps. The Rule Engine also checks all transactions that refer to the EPCs contained in the RFID Reader Event received from the RFID Middleware. In the example scenario, only two of the three EPCs contained in the RFID Reader Event are referred to by the purchase order. Therefore, one EPCIS event is created with the two EPCs associated to the transaction and a second one containing the remaining EPC without any associated transactions. After assembling the two EPCIS events, the Rule Engine saves them in the EPCIS repository using the EPCIS capture interface.

3 PROTOTYPICAL IMPLEMENTATION

To test the general functionality of the developed concept, we implemented the proposed ECA as an extension of the commercial RFID Middleware CrossTalk (developed by the software company noFilis) and tested it in our laboratory. CrossTalk is a device management software for the integration of smart devices into a distributed data network (noFilis 2007). According to our framework, CrossTalk has to be considered as RFID Middleware. It supports the integration of RFID hardware from major vendors and decouples the physical layer from the business layer. Furthermore, it allows for monitoring the state of all connected devices even in a physically distributed RFID infrastructure. CrossTalk can be used in very heterogeneous environments, spanning from an industry PC installation in a production environment to its use in multiple locations in the supply chain. It has been reported to be implemented in connection with SAP's Auto ID infrastructure (Bornhövd et al. 2004).

We deployed CrossTalk in a JBoss application server with a MySQL database and tested it first in our laboratory with two RFID readers: Feig LRU 2000 and Sirit Infinity 510. We created a CrossTalk service which performs the web service call to the Event Interface of the prototypical ECA. As EPCIS implementation we used Fosstrak (Fosstrak 2008). Fosstrak is a complete implementation of the EPCIS standard specification, certified by EPCglobal. We deployed both the ECA and Fosstrak in an Apache application server using the same MySQL database as CrossTalk.

A series of tests demonstrated the general working of the developed concept and prototype. Context data similar to the EPC Showcase was imported via the Reader and Transaction Context Interfaces of our ECA implementation. RFID tags conforming to ISO/IEC 18000-6 were read by the RFID readers, transformed to the ALE format by the created CrossTalk service and imported to the ECA via the Event Interface. Manual queries from the EPCIS repository showed that the ECA created all EPCIS events correctly. To date we have not conducted any performance tests measuring the feasible data throughput that can be handled by our prototype for more than two RFID readers that capture events simultaneously. Further performance tests of the EPCIS event generation process on the basis of a simulative approach are an important future area of research.

4 CONCLUSIONS

The industry consortium EPCglobal and associated research institutions have developed a stack of standards for the Internet of Things. Besides the hardware related specifications, which have already been adopted by the ISO, there exist three architectural layers designed to enable the EPCIS-based cross-company usage of RFID-related data: the EPCIS itself as well as the event capturing and applications. While the applications that access the EPCIS repositories are the focus of numerous research projects, the event capturing applications required to generate EPCIS events in the first place have so far not been in the focus of attention although their development and standardization is both challenging and important.

This paper fills this gap by presenting an in-depth analysis of the requirements for event capturing applications. Our results show that the connection of the different data sources required to generate EPCIS events is very demanding. The required data ranges from rather static process descriptions to highly dynamic state signals from the physical infrastructure.

We have proposed the specification of a standardizable ECA including the required interfaces, database and event assembly algorithm. To demonstrate the correctness of our specifications, we have developed a prototype that implements them and tested it in our RFID laboratory. The architectural layers interacting with our ECA prototype were represented by solutions accepted in the market (CrossTalk and Fosstrak) in order to assure its usability.

References

Auto-ID Center (2003), “PML Core Specification”, Version 1.0.

Beier, S., Grandison, T., Kailing, K. and Rantzau, R. (2006), “Discovery Services – Enabling RFID Traceability in EPCglobal Networks”. In Proceedings of the 13th International Conference on Management of Data, Delhi, India.

Bornhövd, C., Lin, T., Haller, S. and Schaper, J. (2004), “Integrating Automatic Data Acquisition with Business Processes Experiences with SAP's Auto-ID Infrastructure”. In Proceedings of the 30th International Conference on Very Large Data Bases (VLDB04), Toronto, Canada, pp. 1182-1188.

Cambridge University, BT Research and SAP Research (2007), “Serial-Level Inventory Tracking Model”, Public Deliverables of the BRIDGE project, http://www.bridge-project.eu.

Dada, A. and Magerkurth, C. (2008), “Anti-Counterfeiting Based on Supply Chain Proximity”. In Proceedings of the 4th European Workshop on RFID Systems and Technologies. Freiburg, Germany.

EPCglobal (2008), “The Application Level Events (ALE) Specification”, Version 1.1 Part I: Core Specification.

EPCglobal (2007), “EPC Information Services (EPCIS) Specification”, Version 1.0.1.

Fosstrak (2008), “Free and Open Source Framework for Track and Trace”. http://www.fosstrak.org/, last access on October 22nd, 2008.

Goebel, C. and Tribowski, C. (2008), “EPCIS-based Decision Support for Assembly Networks”. In Proceedings of the 16th European Conference on Information Systems, Galway, Ireland.

GS1 Germany (2008), “EPC Showcase”. http://www.gs1-germany.de/internet/content/produkte/epcglobal/epc_showcase/index_ger.html, last access on October 22nd, 2008.

Kim, J. and Kim, H. (2007), “E-Pedigree Discovery System and its Verification Service for Consumer’s mobile RFID Device”. In Proceedings of the IEEE International Symposium on Consumer Electronics, (ISCE 2007), Dallas, Texas, USA, pp. 1-4.

Kim, Y., Moon, M. and Yeom, K. (2006), “A Framework for Rapid Development of RFID Applications”. In Proceedings of the 2nd Ubiquitous Web Systems and Intelligence Workshop (UWSI 2006), Glasgow, UK.

Niederman, F., Mathieu, R. G., Morley, R. and Kwon, I.-W. (2007), “Examining RFID Applications in Supply Chain Management”. Communications of the ACM, 50 (7), pp. 92-101.

Nguyen, T., Lee, Y.-K., Huq, R., Jeong, B.-S. and Lee, S. (2007), “A Data Model for EPC Information Services”. In Proceedings of the 18th Data Engineering Workshop, Hiroshima, Japan.

No, J. P. T (2008), “Development of a National Electronic Product Code Network for the Tracking of Fast Moving Consumer Goods”, *Int. Journal of Enterprise Network Management*, 2:1, pp. 25-46.

noFilis (2007), “CrossTalk. White Paper 2.0”. http://nofilis.com/documents/CrossTalk_WhitePaper_2.0_EN.pdf, last access on October 22nd, 2008.