RFID Event Data Processing — An Architecture for Storing and Searching

Matthieu-P. Schapranow, Jürgen Müller, Alexander Zeier, and Hasso Plattner

Hasso Plattner Institute, Enterprise Platform and Integration Concepts Chair, August–Bebel–Str. 88, 14482 Potsdam, Germany
{matthieu.schapranow|juergen.mueller|zeier|plattner}@hpi.uni-potsdam.de
https://epic.hpi.uni-potsdam.de

Abstract. The pharmaceutical industry suffers from increasing counterfeit rates. Federal law restrictions force manufacturers to guarantee authenticity of their products. RFID technology can be used as foundation for an integer and counterfeit-resistant pharmaceutical supply chain. In the given paper, we introduce an architecture optimized for storing and searching of RFID data. It is specifically designed for the needs of the pharmaceutical industry such as tracking and tracing of pharmaceutical products. We share our benchmark results measured with our implemented main-memory-based architecture prototype.

1 Introduction

The European pharmaceutical industry moved to public focus by operation MEDI-FAKE, an operation performed by custom authorities in all members of the European Union (EU) where 34 million fake drugs were detected in a period of only two months in 2008 [1]. This operation highlights that it becomes more and more important for the pharmaceutical industry to guarantee customers authentic products, i.e. effective medicines with proper medical effect. RFID is a possible technique to implement integer supply chains besides existing techniques such as one-dimensional barcodes. Reliable product tracking and tracing across all participants of the supply chain can be implemented by RFID [2]. However, RFID is not designed to be per se immunized against threats such as cloning, spoofing or eavesdropping of data [3].

The pharmaceutical industry has to find individual solutions addressing the authenticity of products while data integrity is ensured to shield the pharmaceutical supply chain against intrusion of product counterfeits [4]. Enhancing existing supply chains by introducing RFID tags comes with various prevailing advantages [5]. For instance, RFID tags can be read without establishing a direct line of sight, multiple tags can be read simultaneously, and they can cope with dirty environments in contrast to barcodes [6]. A dedicated service provider has to be introduced to perform plausibility checks. With the help of the virtual product history describing the product’s path through the supply chain the service provider automatically detects counterfeits. The virtual product history
is described by the set of all captured events for a certain product identified by its unique Electronic Product Code (EPC) stored on the product’s RFID tag. Thus, all participants in the pharmaceutical supply chain have access to the full virtual product history describing the product’s path within the supply chain. Implementing a dedicated service provider builds on the idea of frequent data exchange between participants of the supply chain. Implementation of an industry-specific solution on top of existing communication infrastructure such as the Internet is non-trivial. The communication medium suffers from the huge amount of event data required to be transferred for anti-counterfeiting [7].

We propose an architecture which addresses the following challenges:

– event data with approx. 100-200 bytes per event have to be stored in local event repositories,
– hundreds of thousands of events per hour will be captured decentralized at participants' sites,
– a huge amount of queries which need to be transferred to different EPC Information Services (EPCIS) repositories of each supply chain participants using the global communication infrastructure, and
– queries have to be processed in very fast time (sub-second time), because at intermediate locations in the supply chain the authenticity of hundreds of products – even of the same producer – are checked at the same time, e.g. when a truck arrives at a wholesaler.

The rest of the paper is structured as follows: Sect. 2 motivates the need for an architecture which supports integer RFID-aided pharmaceutical supply chains. Our work is placed in context of related work in Sect. 3. Sect. 4 shares details about architecture components, while Sect. 5 explains the interaction of all components of the architecture. Sect. 6 describes the benchmarking setup, presents gathered timing results and evaluates the measurements. The paper concludes in Sect. 7.

2 The Pharmaceutical Supply Chain

The United States Federal Food and Drug Administration (FDA) detected at least 21 counterfeit cases yearly since 2001; in 2004 there were more than twice that number with 58 confirmed opened cases [8].

In contrast, in the years 1997 to 2000 the number of detected counterfeits did not exceed six per year [9]. This outlines two aspects. On the one hand, the number of counterfeits increased. On the other hand, counterfeit tests and detection methods are continuously improved and former undetected counterfeits are detected meanwhile. The European Commission reports an increase of 118 percent for pharmaceutical counterfeits detected at EU borders in 2008 compared to 2007. The pharmaceutical product category is the third largest product category in terms of quantities of intercepted articles besides the categories CDs/DVDs and cigarettes [10]. India is named as the top source of counterfeit
pharmaceutical products contributing more than 50 percent of all detected articles [11]. This example underlines that counterfeiters in countries with low law regulation benefit from pandemic diseases such as influenza H1N1 in 2009 when customers bought medicines without prescription via the Internet [12].

Counterfeits of goods are a risk for customers and for suppliers, because their efforts is neither tested nor validated and the customer may suffer from various kinds of medical complications. In 2004, it was estimated that more than 500 billion USD were traded in counterfeit, i.e. seven percent of the world trade in the same period [13]. It is argued, that this equals an increase of 150 billion USD compared to 2001 [14]. In contrast, in the same period the worldwide merchandise trade increased only by approx. 50 billion USD. The estimations about the monetary impact of counterfeits vary drastically. This underlines the fact that only a small number of counterfeits can be detected and the estimated number of unreported cases is hard to derive. Improvements in counterfeit detection and goods protection help to increase the amount of detected cases.

This health care scenario highlights the risks of counterfeits and the need for product protection. A high level of supply chain integrity is the basis for reliable product tracking to reduce the amount of counterfeit cases. The EU pharmaceutical supply chain consists of approx. 2,200 pharmaceutical producers, 50 thousand wholesalers, and more than 140 thousand retailers [15]. A model of an existing pharmaceutical supply chain is depicted in Fig. 1 showing the roles producer, reseller, and retailer. Every supply chain participant stores events in a local event repository describing handled products. Rather than exposing all event data, role-based data access, such as password or certificate-based authentication, has to be implemented to protect business secrets.

![Fig. 1. Product Flow in the Pharmaceutical Supply Chain](image)

3 Related Work

Fast storing and performing predefined search queries on RFID data is a current trend in research and various works address this topic. Acquiring real-world events by readers to map them to virtual product histories is task of enterprise RFID middleware systems [16]. They support either one of both categories of
RFID applications: history-oriented tracking and real-time monitoring. The latter needs to leverage sub-second response times to support supply chain tasks, such as incoming goods processing. Our Java prototype is optimized for real-time monitoring by leveraging very short response times for EPC-based queries.

Product flow and temporal event queries are expected to be major queries in RFID event analysis [17]. Existing data stream management systems build on enhancements of query languages such as SQL. Enhancements for performing temporal queries are available to reduce query complexity for temporal exploration of RFID data [18]. Special query templates exist for storing and retrieving flow and temporal data of a certain product [19]. They build on the separation of tag, path, and time data. All data is stored in separate database tables stored by a centralized event repository. But also building a decentralized application prototype was proved to work [20].

Our architecture approach follows the concept of storing event data in a decentralized way. Event stores can be uniquely identified by URLs which are looked up using the Object Name Services (ONS) [21]. The ONS builds on a tree-based data structure comparable to the one used by the Domain Name Service (DNS). The DNS architecture supports distribution of responsibilities across hierarchy levels with dedicated network information centers [22]. Although a centralized instance reduces costs for maintenance and can benefit from optimization, it introduces a single point of failure, e.g. when the system has to be placed offline during maintenance or when hardware failures occur. This needs dedicated redundant hardware components and data redundancy to overcome extreme situations which at least doubles infrastructure costs.

In our implementation we combine the ideas of an EPC Discovery Service (EPCDS) [23] and a DNS architecture. We implemented lookup mechanisms like the ONSs, because event data is stored decentralized to handle the bulky amount of events. Queries are forwarded to decentralized EPCIS event repositories to prevent load peaks that paralyze centralized architecture components. The huge amount of RFID data makes any kind of architecture fragile in terms of system load. Therefore, various researchers focus on data reduction. For instance, filtering duplicate and redundant RFID events can already be performed at enterprise middleware systems to reduce the total data load of the global architecture [24]. The cuboid architecture offers an implementation for partitioning RFID data [25]. This approach can be enhanced to implement static load balancing policies. Once a product reaches the end consumer associated events can be moved to slower storages. From our point of view establishing data retention management increases the total system performance [26].

4 Architecture Components

In the following section we outline decisions for our architectural prototype while focusing on the aspect of throughput effectiveness. To satisfy incoming queries, all events associated with a certain EPC are acquired from decentralized event repositories. It involves the following steps:
– determination of involved EPCISs,
– query propagation to involved EPCISs,
– parallel and distributed processing of query at remote EPCISs,
– return of partial query results,
– aggregation of partial query results, and
– return of final query result to querying participant.

Queries are only forwarded to EPCISs that are involved in the supply chain path of a certain product. All EPCIS repositories offer interfaces to access their stored data in a unified way, regardless of internal format and vendor. Data unification is not part of the given architecture. Fig. 2 shows a query initiated by a retailer and its propagation to involved resellers and the derived producer of the product.

The outlined algorithm is chosen to reduce processing time by processing partial results in parallel comparable to the MapReduce algorithm [27]. By limiting the recipients for a query, the overall network load is kept minimal.

In terms of our architecture storing RFID data needs to be space-aware. On the one hand, a space-aware way of storing data increases the amount of data which can be stored on the same amount of storage, i.e. a more efficient way of using existing hardware resources is introduced. On the other hand, building architectures on top of existing infrastructure components such as the Internet involves fair use of network resources. In other words, the amount of transferred data has to be minimized to reduce the risk of flooding the existing communication network by RFID data and to prevent network congestion [28].

Motivated by these requirements we implemented an architecture prototype optimized for the needs of the global pharmaceutical supply chain. The following subsections introduce all components of our architecture, their functionality, and design decisions for their implementation.

Fig. 2. Requests for Product Tracing in the Pharmaceutical Supply

4.1 Global Name Server

The Global Name Server (GNS) is the central component of our architecture. Its implementation follows the concept of the DNS root servers [22]. The GNS
receives an incoming query containing the EPC of a certain product. Two parallel processing tasks follow. Firstly, the Global Aggregation Server is notified which creates a session to handle all incoming partial query results for its purposes. Secondly, the incoming query is forwarded to servers at lower hierarchy levels, i.e. Company Servers (CSs) which have access to a single EPCIS repository or multiple EPCIS repositories of a company or department.

Spanning a tree on top of the CSs hierarchy helps to identify the correct CS for a certain EPC in logarithmic time, i.e. \( O(n) \)-complexity, instead of performing a long sequential scan on a list of servers with \( O(n) \)-complexity. Our GNS implementation differs from a DNS root server, because the GNS also receives query results from servers at lower hierarchy levels to enable processing of the next query as described in Sect. 4. If one of the requested EPCIS repositories declines the information the requester is informed about the fact, that the returned data may be inconsistent, e.g. because of insufficient access rights. Access rights are implemented and managed by each EPCIS repository separately to guarantee data protection. EPCIS repositories are available as standardized product. They are used in a modular way in context of the given architecture because they are not in the focus of our architecture development.

### 4.2 Global Aggregation Server

The Global Aggregation Server (GAS) is responsible for collecting partial query results generated by the CSs. The GAS is queried by the GNS and receives partial query results by CSs. The GAS allocates memory for a temporary session to store the query, to save a list of known companies and to store partial query results received from the CS [23]. The GAS can be used to perform additional plausibility checks for anti-counterfeiting, but it is not an integrated part of the proposed architecture. We assume counterfeit detection to be implemented by a dedicated service provider.

### 4.3 Company Server

The CS has direct access to the EPCIS repositories of the company or department which contains event data. Based on the estimations for the dimensions of the pharmaceutical supply chain in the EU we decided to use the SAP TREX main memory database to store events. Instead of loading all attributes the columnar database layout makes it possible to load only affected attributes such as EPC and timestamp to answer queries of the following type:

"When did the product with the given EPC leave your company?"

The results are forwarded via the communication network to the GAS. Each supply chain participant also knows details about successors a certain product was passed to. This information is combined with the data received from the EPCIS and returned to the GNS. The GNS restarts its algorithms and forwards the initial query now to the next participant in the supply chain.
Starting from the last participant traversing back to the producer the same steps are performed to improve processing time. This is possible, because the current owner of the product knows the predecessor. The last participant in the supply chain is defined by the querying party, e.g. when the product is checked during incoming goods processing. If the querying party is an intermediate and the product was already passed to the next participants the chain has to be reconstructed by the querying party only in one-way traversing to the end of the chain. By executing these two separate queries processing time can be divided by two if the querying party holds currently the product. If the communication network is segmented at a single site or partially unable, this approach can be used to reconstruct the chain around the unresponsive participant, because the query would stop from both ends at the same participant. If multiple participants are unresponsive or multiple paths are unavailable, the outage cannot be bridged and an incomplete result is indicated.

4.4 SAP TREX

SAP TREX is a columnar main memory database using compression techniques [29]. Queries are directly executed on compressed data in main memory. Our tests indicate that using a main memory database benefits in instant responses when querying a specific EPC.

We generated for two scenarios test data for a pharmaceutical supply chain. Each event consists of the following attributes: EPC, readpoint, time, action, disposition, bizlocation, bizstep. The EPC is stored as serialized global trade identification number and the locations (readpoint and bizlocation) are stored as serialized global location number as defined by the EPCIS Standard [30]. The remaining attributes are used by the service provider for anti-counterfeiting. The attribute parent is used to identify aggregated items during their containment relationship within a tagged parent object.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Events Size</th>
<th>Disk Size</th>
<th>RAM Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario I</td>
<td>≈1.1 Mio.</td>
<td>≈200 MB</td>
<td>≈20 MB</td>
</tr>
<tr>
<td>Scenario II</td>
<td>≈6.3 Mio.</td>
<td>≈1150 MB</td>
<td>≈120 MB</td>
</tr>
</tbody>
</table>

Table 1. Test Data Scenarios

In Table 1 the test data for our tests is characterized. Scenario I consists of about 1.1 million serialized events, using about 200 MB of disk capacity and approx. 20 MB of main memory once they are compressed and loaded into main memory by SAP TREX. Scenario II is about six times larger than scenario I: it consists of about 6.3 million serialized events, consuming 1150 MB on disk and approx. 120 MB in RAM.

Both scenarios show similar compression ratios of approx. 10:1, i.e. ten times more data can be stored in main memory by using compression techniques of
SAP TREX. The mean size of a single event is approx. 182 byte uncompressed and approx. 18 byte compressed. These observations can be used to estimate storage demands for the pharmaceutical supply chain.

4.5 Service Provider for Anti-Counterfeiting

The service provider performs specific tasks for anti-counterfeiting. For instances, it includes plausibility checks which are performed on the virtual product history of a certain product. It is necessary to access the set of events for a certain product. All participants of the supply chain have to agree on common access for this official party.

If a counterfeit is identified the service provider returns the value counterfeit and the product is removed immediately from the supply chain. If the virtual product history is valid, authentic will be returned. In case the outcome of the counterfeit detection cannot be derived automatically, e.g. if network partitioning or temporary failures occurred, unknown will be returned to indicate the need for manual processing.

\[
\text{service}_{\text{counterfeit}} : \text{epc}_i \mapsto \{\text{authentic}, \text{counterfeit}, \text{unknown}\}
\] (1)

We define a function \(\text{service}_{\text{counterfeit}}\) in Equation 1 performing checks with the help of a given \(\text{epc}_i\). It returns either one of the results authentic \(a\), counterfeit \(c\), or unknown \(u\).

5 Performance Aspects

The following section focuses on the interaction of architecture components as modeled in Fig. 3 and the data flow between involved supply chain participants.

Once a query for a certain product is created, it is sent to the GNS. The GNS registers the request and generates a temporary unique query ID which is used to identify related query results and to merge them to a final response. The query ID is length optimized, thus it reduces the amount of data transferred through the communication medium as the query identifier is short compared to the full query. With the help of query ID the GAS is contacted which creates a temporary session for query responses associated with a certain query ID. The query results received from the CS are assigned by the GAS to the corresponding session and therefore to the proper query.

The EPC consists of an encoded company prefix, an encoded product reference, and a serial number [31]. The GNS contacts the producer of a certain product. Connection details for the producer can either be received from a local database or a cache. If no connection details for the producer are locally available these details are looked up via the ONS using the company prefix. The ONS is responsible for mapping a URL representation to the producer and vice versa.

The initial query and the query ID are forwarded to the CS. The CS performs two internal database lookups. The first lookup results in details about the
next participant in the supply chain the product was passed to. These details are uploaded by the current supply chain participant to enable the following next-hop algorithm. The second lookup performs the existence check for events associated with the given EPC.

We found out, that this query separation helps to improve response time, because the lookup for events associated with a certain EPC is time-consuming compared to the determination of the next participant in the supply chain. Meanwhile, the next CS identified by the result of the first query can be queried to start the next lookup. The algorithm starts over until no further successors for the product can be derived.

The GAS receives a notification which is used for anti-counterfeiting plausibility checks. The notification consists of these parts:

1. the query ID for mapping the query result to the corresponding query,
2. the company ID of the current CS, and
3. an indicator for the existence of EPC in the local event store of the company.

![Fig. 3. Architecture: Infrastructure Components and Data Flow](image)

5.1 Network

During the development, we switched the communication protocol from Transmission Control Protocol (TCP) to User Datagram Protocol (UDP). This decision was triggered by the fact that numerous messages are exchanged between GNS, CSs, and GAS. We figured out, that a significant amount of time was spent for TCP connection setup.

We identified the three-way handshake of TCP as a significant source of latency. We assume modern communication infrastructures to be reliable for data transmissions. If packets get lost during query retrieval we implemented
timeouts which trigger automatic retransmission. If responses from sites are received after the current timeout time, it is automatically adapted. Thus, next queries may wait longer for a response, e.g. when a participant uses a slow network or significant delay occurs because of congestion. If duplicated results are received, they are distinguished by the GAS using the **company ID** and **query ID**, i.e. duplicates are discarded. If expected responses are not received, the return value will be marked to be partially. This way, network segmentation and missing data can be identified without creating false-positives.

### 5.2 Sequence of Company Actions

Besides the selected communication protocol we figured out that the sequence of performed database queries has an impact on the total response time of our architecture. We decided to implement two different action settings as depicted in Fig. **4**. Both action settings differ at the CS. We focused on the CS because this component is involved multiple times during a trace for a single product. In other words, reduced response time will be multiplied by the number of involved participant for a certain trace. Our architecture builds on two database queries which are performed by the CS. In Fig. **1** action setting I shows, that both queries are performed simultaneously in protocol step five. Once both results are returned, the CS sends one part of the data to the GNS and the other part to the GAS.

For action setting II Fig. **4** outlines that the first query retrieves details about the participant who received the product after the current participant. In contrast to action setting I the GNS is now informed immediately in protocol step four. Afterwards, in protocol step five the second query is performed by the CS while the GNS can proceed traversing the trace. The result of the second query is sent asynchronously to the GAS in protocol step six.

![Fig. 4. Action Setting I (left), II (right)](image)
6 Timing Results

Using two communication protocols and two action settings, four implementation combinations exist for testing our prototype. These configurations are described and measurement results are presented in this section.

For the overall response time of the architecture, it is particularly important that the communication between GNS and CS is adequately fast. In other words, once the CS returns results, the next participant in the supply chain can be queried immediately.

6.1 TREX Database

SAP TREX was configured to run in a non-distributed environment. It was installed on a single IBM Blade HS 21XM Type 7995 running SuSE Linux Enterprise 10 SP 2 with the 64-bit kernel 2.6.16.60-0.21-smp. It was equipped with two CPUs Intel Xeon E5450 CPUs, each running at 3.0 GHz clock speed consisting of four logical cores with 32kB L1 cache, 12MB L2 cache. The system was equipped with eight 4GB DIMM DDR modules to form a total of 32GB available main memory. We measured the impact of the database size, i.e. the number of database entries, on the response time. Therefore, we used two different scenarios, one with 1.1 million and one with 6.3 million events, for benchmarking.

Our timing results show that the query used to determine the next company in supply chain is performed in 11 ms in average. The result query is performed in 10 ms in average. These timing results are independent of the event scenario as shown in Fig. 5 for bars TCP 1 and UDP 1. This underlines the instant response time that is achieved by the SAP TREX main memory database in the given enterprise scenario. The available main memory in the used blade systems exceeds the amount of event data that is stored in the database. Therefore, the entire data sets of both scenarios fit completely into main memory and can be accessed instantly at main memory access time.

6.2 Network

For the network benchmark GNS and GAS were installed on workstations running Microsoft Windows 7. They were equipped with a single CPU, two cores running at 2.6 GHz clock speed, 4GB RAM, and Gigabit network interface cards. All systems were connected through an unmanaged Gigabit network switch.

The network throughput of our prototype is addressed by measuring the connection time, i.e. the time needed to transfer messages between peers. We focused on using existing standards rather than implementing highly optimized binary communication protocols such as Google Protocol Buffers [33]. The connection time for the communication protocols TCP and UDP are compared.

Our measurements showed that each data transfer using TCP needs six milliseconds in average whereas UDP is approx. six times faster with an average of one millisecond. With TCP one company needs 41 ms in average to complete query processing. Fig. 6 on the left shows the cumulated results which consist
of the connection time between GNS and CS (6 ms), processing the database queries for the next participant (11 ms) and the query for detecting the EPC in the local EPCIS (10 ms), evaluating the next company (2 ms), and two connection times: to the GNS (6 ms) and to the GAS (6 ms).

Switching our prototype to use UDP as communication protocol reduced the connection time to 1 ms in average. We were not confronted with unordered or lost packets, which might occur in more complex routed networks. The complete processing time when using UDP is reduced to an average of 26 ms as depicted in Fig. 6 on the right.

![Response Time Measurements](image)

**Fig. 5.** Comparison of Communication Method Configurations

### 6.3 Architecture Configurations

We enhanced our implemented architecture prototype by a configurable CS to switch between action settings. Action setting I uses two subsequent database queries. From the GNS’s point of view using action setting I showed to consume 35 ms in average with communication protocol TCP and 25 ms with communication protocol UDP as shown in Fig. 5 for TCP1 and UDP1. These response timings occur, because data transmission to the GAS is postponed until both results are available. By varying query processing in action setting II, it is possible to reduce processing time at the CS by 10 ms in average. Once the next
A company is determined, it is immediately sent to the GNS. The GNS receives its result set without the need to wait for the second database query. Using the communication protocol TCP in this case results in 25 ms processing time in average. The communication protocol UDP consumes 40 percent less processing time with 15 ms in average as depicted in Fig. 5 by UDP2.

Our timing results show, that the GNS is able to process one CS within 15 ms in average depending on the architecture configuration. By switching to action setting II using UDP, we were able to reduce processing time by approx. 60 percent compared to action setting I using TCP.

7 Conclusion

In the given paper, we shared our insights gathered by the implementation of a Java prototype specifically designed for storing and searching RFID event data. Our work is motivated by the need of counterfeit detection in the pharmaceutical industry, because fake drug cases increase worldwide.

Our proposed architecture consists of GNS, GAS and numerous decentralized CSs connected to EPCICs. For storing RFID event data we used the SAP TREX main memory database to leverage instant response times for real-time monitoring. To perform centralized anti-counterfeiting we query all participants involved in the product lifecycle for a certain product to return event details. A
linked list of companies is traversed in two directions to save time and to reconstruct single network segmentations. The amount of messages traveling through the communication network is minimized by only querying involved supply chain participants instead of performing broadcasts to all participants.

We outlined different architecture configurations, gave details about timing results and characterized the optimal setup for our prototype. Using action setting II with communication protocol UDP it is possible to contact each involved participant in the product’s lifecycle in 15 ms in average which is 60 percent faster than action setting I with communication protocol TCP. For our test data, we measured that an RFID event consumes in average 18 byte when stored by SAP TREX in a compressed way. For the European pharmaceutical supply chain, we assume one producer, four resellers, one reseller, and one customer. With approx. 15 billion prescriptions per year and nine read events – two per reseller and one per retailer – an annual amount of approx. 2.3 TB of compressed event data needs to be stored by all decentralized SAP TREX instances.

References

30. EPCGlobal: EPCIS Standard v. 1.0.1 (2007)