# EPCIS Event-Based Traceability in Pharmaceutical Supply Chains via Automated Generation of Linked Pedigrees

Monika Solanki and Christopher Brewster

Aston Business School
Aston University, UK
{m.solanki,c.a.brewster}@aston.ac.uk

**Abstract.** In this paper we show how event processing over semantically annotated streams of events can be exploited, for implementing tracing and tracking of products in supply chains through the automated generation of linked pedigrees. In our abstraction, events are encoded as spatially and temporally oriented named graphs, while linked pedigrees as RDF datasets are their specific compositions. We propose an algorithm that operates over streams of RDF annotated EPCIS events to generate linked pedigrees. We exemplify our approach using the pharmaceuticals supply chain and show how counterfeit detection is an implicit part of our pedigree generation. Our evaluation results show that for fast moving supply chains, smaller window sizes on event streams provide significantly higher efficiency in the generation of pedigrees as well as enable early counterfeit detection.

## 1 Introduction

Recent advances in sensor technology has resulted in wide scale deployment of RFID enabled devices in supply chains. Timely processing of RFID data facilitates efficient analysis of product movement, shipment delays, inventory shrinkage and out-of-stock situation in end-to-end supply chain processes [1]. The scanning of RFID tags in production and storage facilities generates unprecedented volumes of events as data streams, when trading partners exchange and handle products from inception through to the end-of-life phase.

In this paper we present a methodology for the automated generation of event-based traceability/visibility information, referred to as "linked pedigrees". We propose a pedigree generation algorithm based on complex processing of real time streams of RFID data in supply chains. Our streams comprise of events annotated using RDF/OWL vocabularies. Annotating streams using standardised vocabularies ensures interoperability between supply chain systems and expands the scope to exploit ontology based reasoning over continuously evolving knowledge. We represent supply chain events as streams of RDF encoded linked data, while complex event patterns are declaratively specified through extended SPARQL queries. In contrast to existing approaches [6, 8, 9] where an element

P. Mika et al. (Eds.) ISWC 2014, Part I, LNCS 8796, pp. 82–97, 2014.

in a stream is a triple, our streams comprise of events where each event is represented as a named graph [5]. A linked pedigree is considered as a composition of named graphs, represented as an RDF dataset[1].

Our exemplifying scenario is an abstraction of the pharmaceutical supply chain. Counterfeiting has increasingly become one of the major problems prevalent in these chains. The WHO estimates that between five and eight percent of the worldwide trade in pharmaceuticals is counterfeit [11]. Many industry experts believe this to be a conservative estimate. Increased visibility of supply chain knowledge, enabled through exchange of event-based traceability data or pedigrees is anticipated to play a key role in addressing the problem.

In the fall of 2013, the U.S. House of Representatives and Senate passed the Drug Quality and Security Act (DQSA)[2]. The track-and-trace provisions, themselves known as *The Drug Supply Chain Security Act* (DSCSA)[3] within the DQSA outlines critical steps to build an electronic, interoperable system to identify and trace certain prescription drugs as they are distributed in the United States. In readiness for its implementation in the healthcare sector from 2015 onwards, the GS1 Healthcare US Secure Supply Chain Task Force has developed guidelines[4] to identify and serialise pharmaceutical products, in order to trace their movement through the U.S. pharmaceutical supply chains. The guidelines are based around the implementation of EPCIS[5] (Electronic Product Code Information Services) as a standard for event oriented, pedigree track and trace. In accordance to these guidelines, the algorithm proposed in this paper utilises EEM[6] (EPCIS Event Model) [14] - an OWL DL ontology for EPCIS, CBVVocab[7] an OWL DL ontology for the Core Business Vocabulary[8], as the specifications for encoding the event data streams and OntoPedigree[9] a content ontology design pattern for generating the linked pedigrees. To the best of our knowledge, stream processing of events annotated with semantics enriched metadata for the generation of traceability/visibility data has so far not been explored for EPCIS events within the Semantic Web or supply chain communities.

The paper is structured as follows: Section 2 presents our motivating scenario from the pharmaceuticals supply chain. Section 3 discusses background and related work. Section 4 presents the preliminaries needed for events and pedigrees that we use in Section 5 for generating our pedigree composition algorithm. Section 6 highlights our evaluation requirements, illustrates the execution environment and discusses evaluation results. Section 7 presents conclusions.

[1] http://www.w3.org/TR/rdf11-datasets/
[2] http://www.gpo.gov/fdsys/pkg/BILLS-113hr3204enr/pdf/BILLS-113hr3204enr.pdf
[3] http://www.fda.gov/Drugs/DrugSafety/DrugIntegrityandSupplyChainSecurity/DrugSupplyChainSecurityAct/
[4] www.gs1us.org/RxGuideline
[5] http://www.gs1.org/gsmp/kc/epcglobal/epcis
[6] http://purl.org/eem#
[7] http://purl.org/cbv#
[8] http://www.gs1.org/gsmp/kc/epcglobal/cbv
[9] http://purl.org/pedigree#

## 2 Motivating Scenario

We outline the scenario of a pharmaceutical supply chain, where trading partners exchange product track and trace data using linked pedigrees. Figure 1 illustrates the flow of data for four of the key partners in the chain. The *Manufacturer* commissions[10], i.e, assigns an EPC (Electronic Product Code) to the items, cases and pallets. The items are packed in cases, cases are loaded onto pallets and pallets are shipped. At the *Warehouse*, the pallets are received and the cases are unloaded. The cases are then shipped to the various *Distribution centers*. From the Distribution centers the cases are sent to retail *Dispenser* outlets, where they are received and unpacked. Finally, the items are stacked on shelves for dispensing, thereby reaching their end-of-life in the product lifecycle.

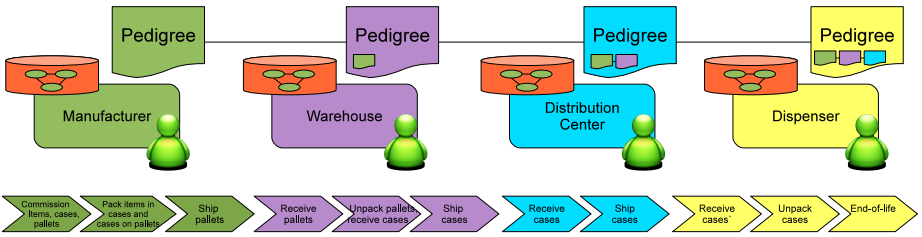


**Fig. 1.** Trading partners in a pharmaceutical supply chain and the flow of information

As the serialised items, cases and pallets move through the various phases of the supply chain at a trading partner's premises, EPCIS events are generated and recorded at several RFID reader nodes. Figure 2 illustrates the phases at a manufacturer's packaging setup and the event streams that can be possibly generated in these phases. For example, events are generated when `Case001` is tagged, i.e, commissioned and read by `Reader011`, when it is packed and read by `Reader013` and finally when the case is made a part of shipment `SSCC001` which is read by `Reader015`. When the pallets with the cases are shipped from the manufacturer's premises to the warehouse, pedigrees encapsulating the minimum set of EPCIS events are published at an IRI based on a predefined IRI scheme. At the warehouse, when the shipment is received, the IRI of the pedigree is dereferenced to retrieve the manufacturer's pedigree. When the warehouse ships the cases to the distribution center, it incorporates the IRI of the manufacturer's pedigree in its own pedigree definition. As the product moves, pedigrees are generated with receiving pedigrees being dereferenced and incorporated, till the product reaches its end-of-life stage.

Given this scenario, for a fast moving supply chain with high volumes (approx. 100,000 per day, cf. Section 6) of commissioned items, we evaluate the algorithm proposed in this paper against the time taken for pedigree generation and counterfeit detection. We experiment with varying sizes of event streams,

[10] Associating the serial number with the physical product.

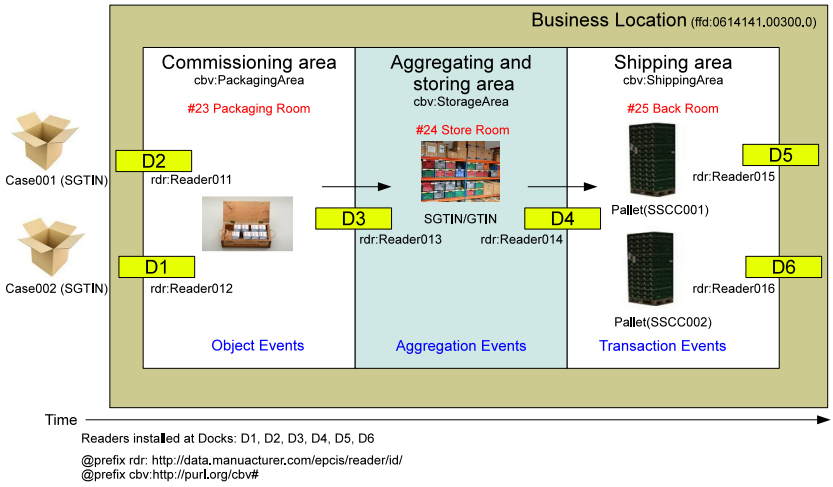


**Fig. 2.** EPCIS Read points and events at the Manufacturer's packaging setup

items, cases and pallets. In particular we would like to ascertain the trade offs between the time taken in generating a large number of pedigrees, each corresponding to a small number of commissioned items against generating a small number of pedigrees for large number of commissioned items, aggregated cases and pallets.

## 3 Background and Related Work

An Electronic Product Code (EPC)[11] is a universal identifier that gives a unique, serialised identity to a physical object. EPCIS is a ratified EPCglobal[12] standard that provides a set of specifications for the syntactic capture and informal semantic interpretation of EPC based product information. As the EPC tagged object moves through the supply chain, RFID readers record and transmit the tagged data as "events". In this paper we are particularly concerned with three types of EPCIS events:

- *ObjectEvent* represents an event that occurred as a result of some action on one or more entities denoted by EPCs, e.g., *"This list of objects was observed entering warehouse #12 at 12:01AM, during Receiving"*.
- *AggregationEvent* represents an event that happened to one or more EPC-denoted entities that are physically aggregated (constrained to be in the same place at the same time, as when cases are aggregated to a pallet), e.g., *"This list of objects was just Palletized with this Pallet ID at Palletizer #27 at 12:32PM"*.

[11] `http://www.gs1.org/gsmp/kc/epcglobal/tds/tds_1_6-RatifiedStd-20110922.pdf`

[12] `http://www.gs1.org/epcglobal`

– *TransactionEvent* represents an event in which one or more entities denoted by EPCs become associated or disassociated with one or more identified business transactions, e.g., *"Order #123 was fulfilled with objects x, y and z"* .

A Pedigree is an (electronic) audit trail that records the chain of custody and ownership of a drug as it moves through the supply chain. Each stakeholder involved in the manufacture or distribution of the drug adds visibility-based data about the product at their end, to the pedigree. Recently the concept of "Event-based Pedigree"[13] have been proposed that utilises the EPCIS specification for capturing events in the supply chain and generating pedigrees based on a relevant subset of the captured events. In previous work [13] we introduced the concept of linked pedigrees, proposed a decentralised architecture and presented a communication protocol for the exchange of linked pedigrees among supply chain partners. In this paper we build upon that work and propose an automated pedigree generation algorithm using streams of EPCIS event data.

Several approaches [16, 17] have been proposed that utilise Complex Event Processing (CEP) for RFID events. In [16], an event composition language, declarative rules and a graph based event processing model are presented. In [17] the authors synthesise behavioural profiles of business processes for query optimisation, based on external process models that define potential sequences of events. Formal modelling of RFID events and roles for the pharmaceuticals supply chain has been proposed in [10], but the focus there is on addressing security threats and counterfeits rather than generating pedigrees for traceability. In contrast, the approach proposed in this paper addresses counterfeit detection implicitly while generating pedigrees.

RFID platforms built around EPCIS have also been made available by major IT vendors such as Oracle's Pedigree and Serialization Manager[14], Frequentz's IRIS Information Repository and Intelligence Server[15], Microsoft's BizTalk RFID[16] and SAP's Auto-ID Infrastructure[17]. In all the above frameworks, event descriptions are not interoperable, they cannot be shared and combined with background knowledge about the objects being tracked and traced. Further, none of these platforms provide any support for semantic descriptions of EPCIS events or generation of pedigrees as linked data nor do they provide any explicit mechanism for counterfeit detection. However our proposed approach could complement these implementations very well by providing a scalable data sharing model and framework for exchanging pedigrees using open standards.

[13] `http://www.gs1.org/docs/healthcare/Healthcare_Traceability_Pedigree_Background.pdf`

[14] `http://www.oracle.com/us/products/applications/life-sciences/pedigree-serialization/index.html`

[15] Originally IBM's InfoSphere Traceability Server, `http://frequentz.com/traceability-server/`

[16] `http://msdn.microsoft.com/en-us/biztalk/dd409102`

[17] `https://help.sap.com/aii`

Within the Semantic Web Community [7], Several frameworks for CEP and querying over RDF streams such as C-SPARQL [6], CQELS [8], EP-SPARQL [2] and INSTANS [9] have been proposed. Social media and smart cities have proved to be important use cases for the application of these frameworks, however there have been no applications in the business and supply chain sector. While most approaches assume streams to comprise of a sequence of time-stamped RDF triples, our streams are sequences of time-stamped RDF graphs. Some other approaches for streaming data sources such as [3,4] have also been proposed. A rule based approach to CEP of event streams that are semantically annotated is presented in [15].

## 4 Preliminaries

### 4.1 EPCIS Events

For the generation of pedigrees we are interested in the three types of EPCIS events outlined in Section 3. The set of predefined[18] EPCIS event types $E_{types} = \{O_e, A_e, T_e\}$, where $O_e$ is an Object event, $A_e$ is an Aggregation event and $T_e$ is a Transaction event. The set of predefined EPCIS business step types $B_{steps} = \{com, pck, shp\}$ represent the business steps of "commissioning", "packing" and "shipping" respectively as defined in the Core Business Vocabulary and correspondingly in our CBVVocab ontology.

An EPCIS event $E$ as defined in this paper, is a 6-tuple $\langle I_e, t_o, t_r, e_t, b_s, R_e \rangle$ where,

- $I_e \in I$ is the IRI for the event.
- $t_o$ is the time at which the event occurred.
- $t_r$ is the time at which the event was recorded, henceforth referred to as the *timestamp* of the event.
- $e_t \in E_{types}$ is the type of event.
- $b_s \in B_{steps}$ is the business step.
- $R_e$ is a non empty set of EPCs associated with the event.

An EPCIS event named graph, $E_g$, is a pair $(I_n \in I, G_e \in G)$, where $I_n$ is the name (as well as the IRI) of the event graph and $G_e$ is the RDF event graph. Additionally, we define functions $\text{eventGraph}(E_g)$ and $\text{eventIRI}(\text{eventGraph}(E_g))$ that return the event graph and the event IRI respectively for the EPCIS event represented by $E_g$. Further, we define a function, $\text{eventOccurrenceTime}(\text{eventGraph}(E_g))$ that returns the time of occurrence $t_o$ of the event represented in $G_e$.

An EPCIS stream $(G_s)$ is an ordered sequence of RDF triples $\langle (I_n, \text{eventRecordedAt}, t_r) : [t_o] \rangle$ published at an IRI $I_s \in I$, and ordered by $t_o$. The set of triples in $G_s$ are valid at timestamp $t_r$.

We use the notations and definitions defined above in the pedigree generation algorithm proposed in Section 5.

[18] Also referred to as an enum.

### 4.2 Provenance Based Linked Pedigrees

In [13] we proposed a content ontology design pattern,"OntoPedigree"[19], for the modelling of traceability knowledge as linked pedigrees. As provenance of information in supply chains is of critical importance, in this paper we extend the pedigree definition to include provenance.

As pat of the core supply chain knowledge, a linked pedigree includes IRIs for products, transaction and consignment. For provenance, we exploit the PROV-O[20] ontology. In particular, we define provenance assertions for the service and the organisation that created the pedigree as well as for the events and other information artifacts used in the creation of the pedigree. It is worth noting that our event ontology EEM has also been mapped to PROV-O[21]. This implies that we can trace the provenance associated with any event encapsulated within a pedigree. This capability has proved immensely useful in associating authorities with counterfeits (cf. Section 6) when they are detected.

Figure 3 illustrates the graphical representation of OntoPedigree augmented with provenance information. In particular we link pedigrees to the creating authority through the `prov:wasAttributedTo` property. Pedigrees are created by every partner in the supply chain. Apart from the pedigree initiated and created by the first partner, all other linked pedigrees include IRIs to the pedigree datasets for the stakeholders in the immediate upstream or downstream of the supply chain. Pedigrees received from external partners are related via the `ped:hasReceivedPedigree` property which is defined as a subproperty of `prov:wasDerivedFrom` and the time of pedigree generation is captured via the `prov:generatedAt` property.

## 5 Incremental Linked Pedigree Generation Algorithms

### 5.1 Extracting Events from EPCIS Streams

Central to the generation of linked pedigrees from streaming EPCIS events is the notion of "windows" that allow the extraction of an event graph from the streams for further processing. We extract events from EPCIS event streams using windows in two steps: In the first step, the window is selected based on the time interval. In the second step, the filtering of the event IRIs for inclusion in the pedigree is carried out based on the business steps that generated the event graphs.

The following SPARQL queries corresponding to the window selection criteria identified above are defined.The prefix `eem` corresponds to the EEM ontology.

**Window Selection: Time Interval ($Q_t$)**

In this step all event IRIs within a time interval of X hrs (tumbling windows) are selected from the event stream serialised in the TRIG[22] format. As TRIG is

[19] `http://purl.org/pedigree#`
[20] `http://www.w3.org/TR/prov-o/`
[21] `http://fispace.aston.ac.uk/ontologies/eem_prov#`
[22] `http://www.w3.org/TR/trig/`

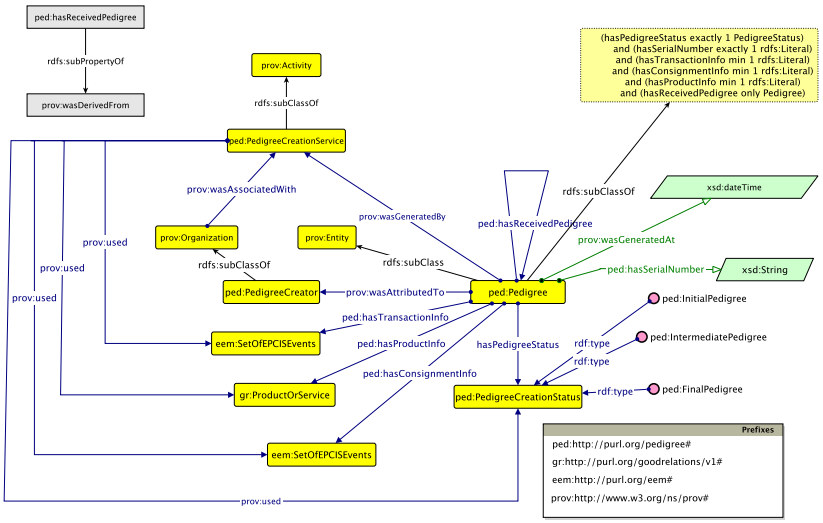


**Fig. 3.** Graphical Representation of Provenance based OntoPedigree

currently not supported by any of the stream processing frameworks, we perform this step as part of our implementation using customised functions.

```
SELECT DISTINCT ?g ?eventGraphIRI ?time WHERE{
 GRAPH ?g {
  ?eventGraphIRI eem:eventRecordedAt ?time ;
  BIND(now() AS ?t1)
  FILTER(fun:extractHours(?t1, ?time) <= X) }}
```

### Window Selection: Business Step ($Q_{bs}$)

Event based consignment information to be included in a pedigree consists of events corresponding to (a) the commissioning of the items, cases and pallets, (b) aggregation of the items in the cases (c) loading of cases on the pallets and shipping. The following SPARQL query, extracts the events corresponding to these business step from each event graph.

```
SELECT DISTINCT ?objEvt ?aggEvt ?shpEvt WHERE
{
 ?objEvt a eem:ObjectEvent;
    eem:hasBusinessStepType ?x;
    eem:associatedWithEPCList ?y.
    ?y  <http://purl.org/co#element> ?epc1.
    {
        ?aggEvt a eem:AggregationEvent;
        eem:hasAggregationURI ?au;
        eem:hasBusinessStepType ?x1;
```

```
        eem:associatedWithEPCList ?y1.
        ?y1  <http://purl.org/co#element> ?epc1.
        FILTER( contains(str(?x1), "packing"))
        {
         ?shpEvt a eem:ObjectEvent;
         eem:hasBusinessStepType ?x2;
         eem:associatedWithEPCList ?z1.
         ?z1  <http://purl.org/co#element> ?au.
         FILTER( contains(str(?x2), "shipping"))
        }
    }
   FILTER (contains(str(?x), "commissioning"))
}
```

### Counterfeit EPC Checking

The basis of our counterfeit detection mechanism mandates that all EPCs that are part of an Aggregation event have been actually commissioned and asserted as part of an Object event. This implies that if the business step is "packing" for an Aggregation event, we further check if the EPCs included in the event have indeed been commissioned as part of an Object event with business step "commissioning".

We experimented with various forms of aggregates and joins in our SPARQL queries to efficiently retrieve and compare the EPCs in the commissioning and aggregation events at the query level itself, however this proved to be highly inefficient and time-intensive. Simple queries for individually retrieving the EPCs and running the counterfeit checks within our implementation gave us a much better performance for counterfeit detection. We reproduce only one of the queries here due to space constraints.

```
SELECT ?epc1  WHERE{
    ?event1IRI a eem:ObjectEvent;
    eem:hasBusinessStepType ?x1;
    eem:associatedWithEPCList ?y1.
    ?y1  <http://purl.org/co#element> ?epc1.
    FILTER( (contains(str(?x1), "commissioning")))}
```

## 5.2 Pedigree Generation Algorithm: Commissioning-Packing-Shipping

In accordance to the scenario presented in Section 2, Algorithm 1 generates the pedigrees. It instantiates the pedigree graph, applies various checks as per the SPARQL queries defined above, retrieves and integrates external datasets, before finally publishing the linked pedigrees. The steps in the algorithm have been illustrated in a self explanatory way within the pseudo code itself for brevity.

# 6 Evaluation

## 6.1 Evaluation Requirements

Our evaluation for the pharmaceutical scenario outlined in Section 2 focuses on two critical timing requirements for pedigree generation in the pharmaceutical supply chain:

- **The time taken to detect counterfeit products in varying volumes of shipments:** This is important as counterfeits have to be detected either before or along with the pedigree generation. We consider the case where counterfeits may be introduced as additional items or as replacement of existing items, when items are being packed into cases. The items have EPCs assigned and tagged to them, although they have not been commissioned at the manufacturing unit.
- **The time taken for pedigree generation:** This time is crucial as pedigree generation for a specific shipment must be initiated as soon as a shipping event for the shipment is recorded. The pedigree must be published imminently when the shipment is dispatched. Given this requirement, we evaluate the time taken for the execution of the various queries in the algorithm for varying number of commissioning, aggregation and shipping events, as well as the overall time taken to generate the pedigrees.

## 6.2 EPCIS Event Volumes

In order to estimate the volume and velocity of events generated in pharmaceutical event streams, we referred to grey literature and interviewed people closely involved in the pharmaceutical sector and EPCIS experts. We referred to a survey [12] that studied the cost benefit analysis of introducing EPCIS event based pedigrees in the pharmaceutical supply chain. As per the survey, the average volume (number) of pallets, cases and items per month being shipped out of a typical manufacturing unit is 290, 5800 and 580,000 respectively. Interviews with experts corroborated the facts, however they also stated that for some large scale units, the number of items shipped could be as high as 100,000 per day.

Assuming an average rate of production as 6 days per week and 10 hours per day, we ran a simulation that replicated the volume and velocity of event generation. We generated the commissioning events based on the number of items ranging from 24,000 to 102,000 per day or approximately 40 to 170 per minute[23]. As the number of items packed per case and the number of cases loaded per pallet could vary across manufacturing units, we generated aggregation and shipping events, considering aggregated items ranging from 100 to 500 (increments of 100) per case and number of cases per pallet ranging from 20 to 100 (increments

[23] Since the number of pallets and cases commissioned is significantly lower than that of the items, as a close approximation we assume that commissioning of the items subsumes the commissioning of the cases and pallets and therefore do not consider these events separately.

**Algorithm 1.** Pedigree generation: Commissioning-Packing-Shipping

```
// Input is the event graph stream and output is the linked pedigree
Data: I_s
Result: G_p
// Set up the pedigree graph
Instantiate the pedigree provenance graph, G_pp
Insert triple (I_p, prov:wasGeneratedAt, t_p) in G_pp
Instantiate the pedigree default graph, G_pd
Insert triple (I_p, hasStatus, ped:Initial) in G_pd
Insert triple (I_p, hasSerialNumber, n_p) in G_pd
while I_s has events do
    // extracts the event graphs based on the window length for time
       interval.
    Execute Q_t on the incoming stream I_s to get the event IRI result set R_e
    for E_g ∈ R_e do
        Execute counterfeit EPC checking queries on R_e to get counterfeit EPC
        result set, R_c
        if (R_c is non-empty) then
            Send notification for counterfeit EPC
        else
            Retrieve event graph, eventGraph(E_g)
            Transform E_g to N-Triples representation
            Execute Q_bs on E_g using a RDF stream processor
            Extract event URIs for commissioning, aggregation and shipping
            events in R_com, R_agg, R_shp respectively
            for I_a rdf:type A_e ∈ R_agg do
                Insert triple (I_p, eem:hasConsignmentInfo, I_a) in G_pd
            end for
            for I_e rdf:type O_e ∈ R_com do
                Retrieve product master IRI, I_m for EPCs in I_e
                Insert triple (I_p, eem:hasProductInfo, I_m) in G_pd
                Insert triple (I_p, eem:hasConsignmentInfo, I_e) in G_pd
            end for
            for I_s rdf:type T_e ∈ R_shp do
                Insert triple (I_p, eem:hasTransactionInfo, I_e) in G_pd
            end for
        end if
    end for
    Merge graphs G_pp and G_pd
    Publish G_p at IRI I_p
end while
```

of 20). We experimented with tumbling window sizes of 3, 5, 7 and 10 hours respectively. For the window size of 10 hours and rate of 120 and 170 items per minute in the stream, the number of commissioning, aggregation and shipping events are highlighted in Table 1 giving an indication of the overall volume of events we considered. Based on the rate of counterfeits as highlighted in Section 1, we introduce 8% of the total items as counterfeits in order to estimate the time taken for detection. The event dataset dumps used for the various runs of the algorithm as part of our simulation have been made available[24].

**Table 1.** Number of commissioning, aggregation and shipping events for a window size of 10 hours and item commissioning rate of 120 and 170 per minute

| | | | 100-500 per case | 20-100 per pallet |
|---|---|---|---|---|
| Window size (hrs) | Items/min. event stream velocity | Commissioned events | Aggregation events (increments of 100) | Shipping events for each of the aggregates (increments of 20) |
| 10 | 120 | 72000 | 720/360/240/180/144 | 36/18/12/9/7 |
| | | | | 18/9/6/5/4 |
| | | | | 12/6/4/3/3 |
| | | | | 18/9/6/5/4 |
| | | | | 7/4/3/2/2 |
| | 170 | 102000 | 1020/510/340/255/204 | 51/26/17/13/11 |
| | | | | 26/13/9/7/5 |
| | | | | 17/9/7/5/4 |
| | | | | 13/7/5/4/3 |
| | | | | 10/5/4/3/2 |

### 6.3 Pedigree Generation Framework

Figure 4 illustrates the workflow and execution environment of our EPCIS stream processing framework. We have developed a library, LinkedEPCIS[25] for encoding EPCIS events as linked data. RFID tag data, read by readers is converted into a stream of linked EPCIS event named graphs using the library, which is deployed on the edge server or as part of a custom app. EPCIS event streams are accessed by the Linked Pedigree server that deploys various components for facilitating the generation of linked pedigrees from event IRIs.

As our event streams are named graphs, we natively generated the streams in TRIG. However currently none of the stream processing engine support the TRIG format. We therefore implemented an event extractor component that extracts the event graphs from the stream based on predefined window sizes, computed using the rate of event generation identified above.

[24] `http://fispace.aston.ac.uk/pharma/eventDatasets`
[25] `https://github.com/nimonika/LinkedEPCIS`

As part of our machinery, we incorporate an enhanced version of an existing semantic stream processing engine, INSTANS [9] for continuously executing queries over our event streams. As INSTANS accepts event streams in NTriples, we convert the extracted event stream into the N-Triples serialisation before piping it with INSTANS, where the queries are executed. The results are event graph IRIs, that are passed to the pedigree generator which compiles the pedigrees. The pedigree generator integrates the pedigrees with any external datasets such as location based information, product master data or any other information the trading partner may consider useful. The pedigrees are persistently stored for history based analysis and are also published at IRIs that can be accessed by invoking the REST services published on the Linked Pedigree server.

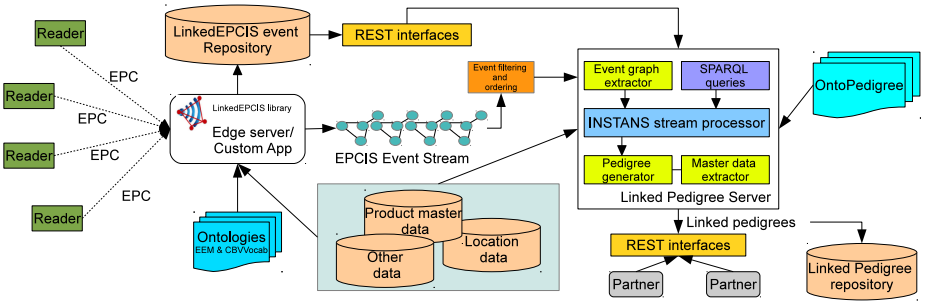


**Fig. 4.** Generating linked pedigrees from EPCIS event streams

### 6.4 Evaluation Results

We carried out an extensive and exhaustive evaluation of the pedigree generation algorithm. For the four window sizes and varying number of commissioning, aggregation and shipping events, we ran a total of 400 iterations of the algorithm. The evaluations were made on Mac OSX 10.9.2, 1.7GHz Intel core i5, 4GB 1333 MHz DDR3. Figures 5 and 6 illustrate some of the key findings of our experiments. We observed that the time taken for the generation of pedigrees was most influenced by and increased with the number of commissioning events. Surprisingly, varying the number of items per case (100-500 with increments of 100) or the number of cases per pallet (20-100 with increments of 20) for the same number of commissioning events had little influence. The time taken for the detection of counterfeits did increase with the number of commissioning events, however the increase was not as significant as that observed with the pedigree generation time. The formulation of the SPARQL queries did have an influence on the time taken to detect counterfeits as noted in Section 1. For high numbers of commissioned items, we consistently ran out of memory when we tried to run a combined query for retrieving the EPCs from the commissioning and aggregation events. However splitting the query and checking for counterfeits within the implementation resolved the problem. Further, as the EEM ontology

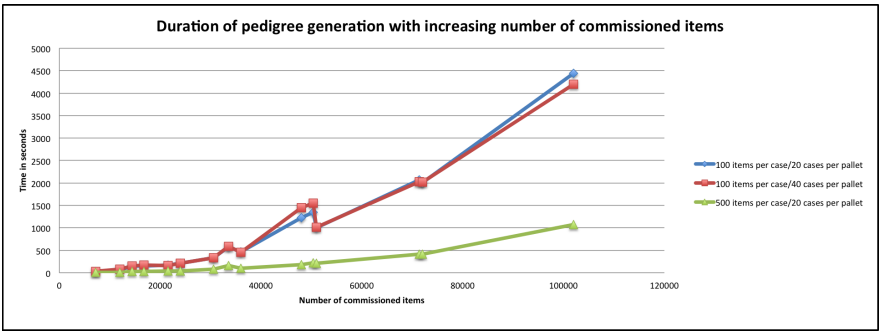


**Fig. 5.** Pedigree generation duration for increasing number of commissioning events

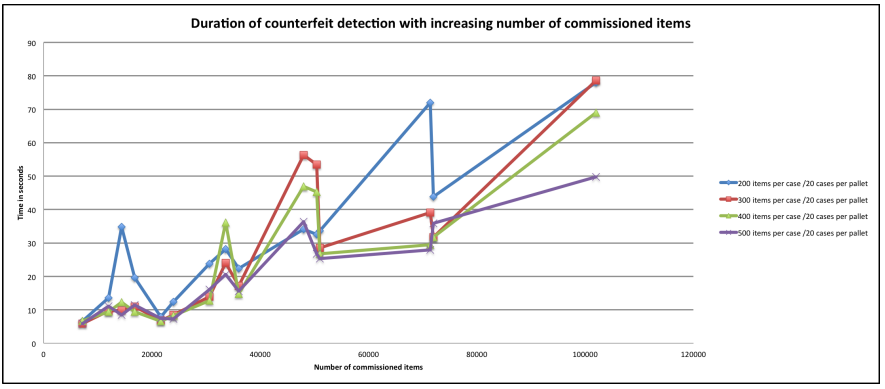


**Fig. 6.** Counterfeit detection duration for increasing number of commissioning events

and the OntoPedigree pattern heavily exploit the PROV-O vocabulary, when a counterfeit was detected, we were able to trace it back to the agent responsible for the generation of the pedigree using the `prov:wasAttributedTo` relationship. This feature of Semantic Web/Linked data technologies for counterfeit detection in pedigree generation, immediately gives us a significant advantage compared to the various commercial efforts reviewed in Section 3. Update queries took comparatively little time as compared to querying, so we do not report those results here.

The results of our experiments provide noteworthy insights into improving the performance of the supply chain and optimising the process of pedigree generation in real time. In Section 2, we set out to establish the trade off between generating a large number of pedigrees, each corresponding to a small number of commissioned items against a small number of pedigrees for large number of commissioned items, aggregated cases and pallets. Based on our observation we can conclude that using smaller window sizes of 3 - 5 hrs for generating pedigrees, yields less number of commissioning events which can not only significantly reduce the running time of the algorithm, but it can also enable quicker detection of

counterfeits and consequently make a significant impact on the overall efficiency of tracking and tracing within the supply chain. Another important conclusion is that variations in aggregation and shipping loads do not significantly impact the pedigree generation time for small window sizes.

## 7 Conclusions

Data visibility in supply chains has received considerable attention in recent years. In the healthcare sector, visibility of datasets that encapsulate track and trace information is especially important in addressing the problems of drug counterfeiting. In this paper we have shown how Semantic Web standards, ontologies and linked data can be utilised to represent and process real time streams of supply chain knowledge, thereby significantly contributing to the vision. We have presented an algorithm that illustrates how linked pedigrees can be automatically harnessed from streaming EPCIS event datasets. Our algorithm, besides generating the pedigrees, also checks an important constraint of EPC mismatch, which can play a major role in identifying counterfeit drugs illegally introduced in the supply chain. Provenance, which is a critical aspect of supply chain knowledge is an integral part of our framework. We have performed an exhaustive evaluation of the algorithm using various combinations of commissioning, aggregation and shipping events. Our results provide very useful insights in improving the overall efficiency of the supply chain.

Much work still needs to be done. We are extending our algorithms to automatically assert aggregation and containment relationships using stream reasoning techniques, and persistently update the knowledge base with the newly derived relationships.

**Acknowledgements.** The research described in this paper has been partially supported by the EU FP7 FI PPP projects, SmartAgriFood (`http://smartagrifood.eu`) and FISpace `http://www.fispace.eu/`. The authors would like to thank Mikko Rinne from the INSTANS team for the extensive support with their stream processing engine.

## References

1. Alexander Ilic, F.M., Andersen, T.: EPCIS-based Supply Chain Visualization Tool. Auto-ID Labs White Paper WP-BIZAPP-045 (2009)
2. Anicic, D., et al.: EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011. ACM (2011)
3. Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL - Extending SPARQL to Process Data Streams. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 448–462. Springer, Heidelberg (2008)

4. Calbimonte, J.-P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 96–111. Springer, Heidelberg (2010)
5. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. In: Proceedings of the 14th International Conference on World Wide Web, WWW 2005. ACM (2005)
6. Francesco, D., et al.: C-SPARQL: a Continuous Query Language for RDF Data Streams. Int. J. Semantic Computing (2010)
7. Della Valle, E., et al.: It's a Streaming World! Reasoning upon Rapidly Changing Information. IEEE Intelligent Systems (2009)
8. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)
9. Rinne, M., Abdullah, H., Törmä, S., Nuutila, E.: Processing Heterogeneous RDF Events with Standing SPARQL Update Rules. In: Meersman, R., et al. (eds.) OTM 2012, Part II. LNCS, vol. 7566, pp. 797–806. Springer, Heidelberg (2012)
10. Schapranow, M.-P., et al.: A Formal Model for Enabling RFID in Pharmaceutical Supply Chains. In: Hawaii International Conference on System Sciences. IEEE Computer Society (2011)
11. Schuster, E.W., Koh, R.: Auto-ID Labs, Massachusetts Institute of Technology Cambridge, MA
12. Sinha, A.: Systems engineering perspective of e-pedigree system. Master's thesis, Systems Design and Management (SDM). MIT (2009)
13. Solanki, M., Brewster, C.: Consuming Linked data in Supply Chains: Enabling data visibility via Linked Pedigrees. In: Fourth International Workshop on Consuming Linked Data (COLD 2013) at ISWC. CEUR-WS.org proceedings, vol. 1034 (2013)
14. Solanki, M., Brewster, C.: Representing Supply Chain Events on the Web of Data. In: Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE) at ISWC. CEUR-WS.org proceedings (2013)
15. Teymourian, K., Rohde, M., Paschke, A.: Fusion of background knowledge and streams of events. In: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, DEBS 2012. ACM (2012)
16. Wang, F., Liu, S., Liu, P.: Complex RFID event processing. The VLDB Journal (2009)
17. Weidlich, M., Ziekow, H., Mendling, J.: Optimising Complex Event Queries over Business Processes Using Behavioural Profiles. In: Muehlen, M.z., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 743–754. Springer, Heidelberg (2011)