Building **R**adio frequency **ID**entification for the **G**lobal **E**nvironment

Supply Chain Integrity (D4.6.1)

Authors: Andrea Soppera (BT), Jeff Farr (BT), Oliver Kasten (SAP), Alexander Illic (ETH), Davide Zanetti (SAP), Mark Harrison (CAM)

December 2007

This work has been partly funded by the European Commission contract No: IST-2005-033546

About the BRIDGE Project:

BRIDGE (**B**uilding **R**adio frequency **ID**entification for the **G**lobal **E**nvironment) is a 13 million Euro RFID project running over 3 years and partly funded (€7,5 million) by the European Union. The objective of the BRIDGE project is to research, develop and implement tools to enable the deployment of EPCglobal applications in Europe. Thirty interdisciplinary partners from 12 countries (Europe and Asia) are working together on : Hardware development, Serial Look-up Service, Serial-Level Supply Chain Control, Security; Anti-counterfeiting, Drug Pedigree, Supply Chain Management, Manufacturing Process, Reusable Asset Management, Products in Service, Item Level Tagging for non-food items as well as Dissemination tools, Education material and Policy recommendations.

For more information on the BRIDGE project: www.bridge-project.eu

This document results from work being done in the framework of the BRIDGE project. It does not represent an official deliverable formally approved by the European Commission.

This document:

This report has two main goals. Firstly, to provide a simple, very broad-ranging model in which all the issues relating to supply-chain integrity can be more clearly understood. This goal is addressed in section 2 of the report. We will reflect on the meaning of integrity and categorize integrity conditions (or rather integrity violations) in four intuitive categories.
The second goal of this report is, to discuss the idea of using the EPC network as a tool for detecting violations to supply-chain integrity. Automated detection of integrity violations can trigger recovery mechanisms, thus helping to maintain the integrity of the supply chain. Along these lines we will present initial conceptual work, including a prototypical implementation of such a tool. In section 2 we will explore our approach of finding evidence of integrity violations through rules. We will substantiate the concept by providing several concrete examples. In Section 4 we will present our initial conception of a software implementation of such a integrity checking tool. Finally, in section 5 we will present an initial prototype in which we have put the general idea to a first test.
In the second half of BRIDGE's task 4.6, we intend to refine those ideas and we will extend the prototype into a configurable software tool, able to detect a wide range of supply-chain integrity violations.

Disclaimer:

Acknowledgements

We would especially like to thank Mark Harrison of Cambridge University for his contribution to the document and the entire AT4 wireless team for their contributions in discussions and their helpful comments. Cambridge University is not officially involved in WP4.

Contents

1 Introduction

This report has two main goals. Firstly, to provide a simple, very broad-ranging model in which all the issues relating to *supply-chain integrity* can be more clearly understood. This goal is addressed in section 2 of the report. We will reflect on the meaning of integrity and categorize integrity conditions (or rather integrity violations) in four intuitive categories.
The second goal of this report is, to discuss the idea of using the EPC network as a tool for detecting violations to supply-chain integrity. Automated detection of integrity violations can trigger recovery mechanisms, thus helping to maintain the integrity of the supply chain.
Along these lines we will present initial conceptual work, including a prototypical implementation of such a tool. In section 2 we will explore our approach of finding evidence of integrity violations through rules. We will substantiate the concept by providing several concrete examples. In Section 4 we will present our initial conception of a software implementation of such a integrity checking tool. Finally, in section 5 we will present an initial prototype in which we have put the general idea to a first test.
In the second half of BRIDGE's task 4.6, we intend to refine those ideas and we will extend the prototype into a configurable software tool, able to detect a wide range of supply-chain integrity violations.

2 Supply Chain Integrity

This section attempts to provide a fairly simple, very broad-ranging model in which all the issues relating to supply-chain integrity can be more clearly understood. The intention is to produce a single, consistent, clear, and simple description that can be used by the EPCglobal community and EPC practitioners alike, inside and outside of the BRIDGE consortium.

2.1 Definitions of Supply Chain Integrity

The term integrity (besides its meaning in ethics) typically refers to "the state of being whole, entire, or undiminished' [dictionary.com]. The concrete meaning of the word then largely depends on the context in which it is used. An appropriate definition of integrity for supply chains is the requirement that *the system performs its intended function in an unimpaired manner, free from deliberate or inadvertent manipulation* [see, for example, NIST-95].
To study this issue further, we first define and distinguish between the primary elements of the supply-chain system itself. The first element is the *physical supply chain*, which is centred on the physical objects being transported, and the associated manual processes (i.e., the actual products, containers, vessels; and also the different parties and handling processes that distribute those items through the supply chain).
The second element of the system is the *EPC network* . This, in its most general sense, is the RFID-based system designed to measure, record, analyse, and model the physical reality of the real-world supply chain.
We will define a supply chain integrity breach to have occurred when *either* of these systems deviate from their intended functions. The following two sub-sections focus on each of these components in turn.

2.2 Physical Integrity

Physical supply-chain integrity is the traditional (though limited) definition of supply-chain integrity, and refers to the properties and characteristics of physical entities (such as products, logistic units, organizations, and individuals) as well as of the processes in which those entities interact in the supply chain. A literature and Web search reveals a wide range of possible meanings of physical supply-chain integrity, which largely depend upon the industry sector in which the system operates. In fact, at this physical level there are relatively few processes that are completely generic to all industries, and so the detailed definitions of integrity vary considerably. For example:

- food and product safety (e.g., temperature control in the cold-chain)
- absence of tampered products (i.e., unauthorised product manipulations, e.g., re-labelling of Intel Pentium processors)
- absence of tampered containerised cargo (i.e., exchange, removal, and additions of cargo between/ from/ to a container) including cargo theft
- absence of store misplacements and wrong deliveries of products and shipments
- absence of counterfeit products, diversion, parallel trade, and smuggling
- bona fides of vendors (suppliers, distributors, or outsourced service providers such as packagers, forwarders, and advertising agents)
- absence of corruption and fraud
- regulatory compliance and correct handling of goods, (e.g., cold chain temperatures, quality control, customs)

Whilst BRIDGE is primary focused on the development and the application of the EPCglobal architecture, there are elements within the project (e.g. WP5 Anti-counterfeiting) in which the physical supply chain itself is relevant too. A key reason to include it within the scope of this Task is because, as it turns out, many of the integrity checking mechanisms we propose are unable to immediately distinguish whether the fault is in the physical supply chain itself, or in the EPC network. Hence our mechanisms are relevant to both types of problem.

2.3 Integrity of the EPC network

Besides physical-supply chain integrity, there is another interpretation of supply-chain integrity that is closer to the work of Workpackage 4 (Security of EPC Network). BRIDGE (as a whole) aims to support real-world supply-chain processes by building and employing software tools and IT infrastructure based on EPCglobal standards. Therefore, supply-chain integrity can also be understood as the integrity of the entire system of standards, tools, and infrastructure, including their actual implementation at the supply-chain partners (which, in this document, we will loosely refer to as the *EPC network*). This interpretation is closely related to the integrity notion used in the field of computer science.
As stated previously, the general (although largely implicit) purpose of the EPC network is that it acts as a measurement, recording and analysis system to accurately model real-world supply chains. The purpose and 'correct' operation of the network is somewhat defined by the EPCglobal network standards, but these documents only partially capture the above, more general purpose of the overall EPC network. In particular, standardised specifications of expected system operation will be supplemented by various other forms of written document and implicit expectations, and so it will *always* be the case that different parties will adopt somewhat differing definitions as to *the intended operation of the system*.
In the following sub-sections we classify the different authorities that have a role in writing (and so collectively defining) the purpose and intended operation of the EPC network. EPC-network integrity breaches can then be classified according to which sets of documentation the analyst wishes to use to define correct operation. The classifications of system specification we use are:

- Formally-defined global standard specifications (overwhelmingly, at present, the EPCglobal standards) ['global-level' specifications]
- Contractual commitments and SLAs (service-level agreements) signed between individual collaborating participants ['inter-organisational level' specifications]
- Internal system specifications as specified by an individual company for use within its own EPC network systems. ['intra-organisational level' specifications]

To clarify this classification, we might consider the analogy of the World Wide Web. 'Broken links' on the Web are a common occurrence, and significantly impede the successful use of the Web. However, they do not actually contravene W3C standards, and so they do not, by that definition, represent a breach of the system's integrity. On an intranet, however, a particular company might well have a policy that prohibits broken links within their network. Within the context of their organisation then, and that small part of 'the Web', any broken link there *would* constitute a breach of the system's integrity.
In the case of the EPC network, there are two important distinctions from the Web analogy. Firstly, although a prime purpose of the EPC network is to *accurately model* (some dedicated aspects of) the real world, there is almost nothing in the current formally-defined standards to capture this. The other two forms of system documentation (on the inter-organizational and intra-organizational levels), which may well capture this, are thus very important in this context. A second possible distinction is in the significance of such integrity breaches. Whilst the Web is intended for a very wide range of purposes, and an individual 'broken link' may prove inconvenient, the EPC network is specifically intended specifically for business purposes and any failures may therefore be of critical business importance. The EPC network may thus demand more thorough attention to its integrity than does, say, the Web.

The following subsections clarify the classification of system specifications outlined above.

2.3.1 EPC-network Standards Integrity [global-level]

As mentioned previously, some aspects of the intention of the EPC network are well captured by the EPCglobal standards. For example, those standards specify who can assign valid EPC numbers, which software components can inject data into an EPCIS, and the data fields that are mandatory in the event records. Non-compliance with any of the requirements specified in these standards constitutes a breach of (we will say) EPC network Standards Integrity.
In this category, we also include breaches in any of the other standards that are widely employed within the EPC network. For example, if Discovery Services are finally specified by the IETF (rather than EPCglobal) then that specification too is likely to become a *de facto* part of the EPC network. Similarly for the global and national standards defined by other industry consortia for different industry sectors.
Non-compliance with such public standards is likely to be widely regarded by the whole community as a breach of EPC network integrity. The possible causes and reasons for such a breach of integrity are considered further in Section 3, but it should be clear that they could at least be caused either accidentally (for example, through a software product that mis-implements a standard) or deliberately (through an party that consciously mis-configures their system). A more comprehensive set of examples of failure in this category are listed in Section 2.4.

2.3.2 EPC-network Contractual Integrity (Legal Compliance) [inter-organizational level]

This category captures the intentions of the EPC network as specified in SLAs and contracts agreed between co-operating supply-chain participants, and in legislation used to regulate particular industries. It is at this level that we expect to begin to see the goal of the EPC network as an accurate reflection of the real world described and specified. We expect these legal commitments to require that participating parties truthfully collect and share their information with particular other parties. In many supply chain environments, each partner may be obliged to record and share certain real-world information, and with a specified accuracy and robustness. Contraventions to such commitments constitute a second form of integrity violation to the integrity of the EPC network.
This form of integrity is not a general property of the entire EPC network but is instead highly dependent upon the business context and the agreements between individual business partners.

2.3.3 EPC-network Local Policies [intra-organizational level]

A final authority level at which the EPC network can be considered to be specified is within individual organisations. The majority of organisations will themselves have an inherent motivation to ensure that their systems accurately reflect the reality of their physical systems. It is likely that their RFID reader infrastructure, in particular, will have been carefully designed and specified so as to meet the demands for accuracy and dependability placed upon it. Hence local policies, developed internally by individual organisations, play some role in defining the expected operation of the EPC network.
The synthesis of new events from existing events and other information may be a particularly error-prone process. Synthesized events always bear the risk of being false, either because the inference algorithm proves to be flawed, or because the original information the synthesized event was based on was incorrect in the first place. For example, the reception of a tagged pallet (e.g., an SSCC) may be used to infer that a particular item has arrived on the pallet based on an advanced shipment notice (ASN), subsequently generating a receiving event for the item. But this inference is wrong, however, if the item had in fact been surreptitiously unloaded from the pallet or if the ASN proved to be

incorrect. For this reason we think it is likely that the storage of synthesised events in EPCISs should be minimised and their use reserved for application-level analytical track & trace services (see BRIDGE Work Package 3).
Again, some further examples of this form of integrity breach will be provided in the next subsection.

2.4 A General Model of Supply-Chain Integrity

As well as classifying the different types of supply-chain integrity failure, it is also useful (especially when designing measures to detect, prevent, or correct them) to consider their cause of failure. Integrity violations can be caused accidentally/inadvertently, or deliberately (due to selfish, malicious, or possibly altruistic reasons). At the physical supply-chain level, for example, items can be misplaced or mis-processed either consciously or inadvertently. At the EPC Network level, integrity could be violated deliberately in order to hamper the interpretation of trace data for anti-counterfeiting purposes or to cover up contractual failings. They might equally well arise due the careless mis-configuration of hardware and software components, such as the specification of access-control policies in information sharing services.
The following table summarises the key classifications and concepts described in this section in a comprehensive model of supply-chain integrity.

A Classification of System Failures

		Accidental	Intentional	
			Selfish	Malicious
EPC Network (Measurement, Recording and Analysis System)	Global Standards	•Prevention •Detection •Correction	•Prevention •Detection •Correction	•Prevention •Detection •Correction
	Contractural & SLAs			
	Local Policies			
Physical Supply Chain		•Prevention •Detection •Correction	•Prevention •Detection •Correction	•Prevention •Detection •Correction

To further clarify the model, and to better understand the sub-classifications defined in Section 2.3, we also provide a table showing a variety of example failures in each of the identified categories.

Examples of System Failures

		Accidental	Intentional: Selfish	Intentional: Malicious
EPC Network	Global Standards Contractural & SLAs Local Policies	•Software bugs and hardware faults due to the incorrect implementation of EPCglobal standards • Accidental re-use of the same EPC number for different items. •Catastrophic system component failures (e.g. a fault with an RFID tag, RFID Reader, the LAN, middleware or EPCIS). •System mis-configurations (e.g. time clocks wrong; or having the wrong bizStep or location associated with a Reader; premature deletion of records from an EPCIS; incorrectly-configured access control policies; mis-calibrated sensors within an RFID tag; timely publication of events to Discovery Services). •The accidental (though possibly negligent) introduction of synthesised events into an EPCIS based upon false real-world assumptions.	• unauthorized injection, modification, or deletion of events into EPCIS or DS. • malformed queries and replies, (e.g., missing mandatory fields and incompatible data formats) • using the same EPC number for different items at the same time (a mis-application of the EPC network) • use of invalid/unauthorised EPC numbers. •The deliberate injection of false events (or falsified attributes of an event) into an EPCIS, or Discovery Service. •The deliberate mis-configuration of system components (e.g. the application of incorrect time stamps; incorrect configuration of access control policies; deliberately late updating of Discovery Service records).
Physical Supply Chain		• Physical goods being accidentally mis-directed (including mis-placed and lost), or otherwise accidentally mis-processed. For example, in a cold chain a refrigeration unit may fail resulting in a batch of damaged or contaminated goods. • Absence of tampered products (e.g. mislabelled)	•Physical goods being deliberately mis-directed or mis-processed in some other way. i.e. stolen goods, or product sabotage. •The introduction of counterfeit goods into the supply chain. I.e. the deliberate insertion of unauthorised products into the supply chain.	

Finally, it is worth noting that our model of supply chain integrity clearly omits some other important, related systems, such as Electronic Data Exchange (EDI) [EDI] exchanges and the Global Data Synchronisation Network (GDSN) [GDSN]. It is possible that failures in those systems might have consequences on our domain of interest, and so, ultimately, they might need to be considered in a yet more comprehensive analysis of integrity.

2.5 Summary

This section has described a comprehensive model of supply-chain integrity, and has shown how the integrity of the EPC network (as defined by various sets of authority) can be compromised. We anticipate that the model will be particularly useful in scoping the types of fault a particular integrity mechanism is designed to manage, and in helping to ensure that (ultimately) adequate mechanisms are in place to handle the entire set of threats. Whilst there is clearly scope for some innovation in these mechanisms, it will also be apparent that we should easily be able to re-use many of the established integrity-supporting mechanisms already developed for other types of information system.
In the short term the model should be useful in helping us explain the scope of this task's research objectives, and in demonstrating the contribution of other BRIDGE work to the broad problem of supply-chain integrity. For example, it should now be clear that some of the work on the access-control framework (in the Network Confidentiality task of this work package) plays a strong role in preventing the injection of false information into the network by unauthorised parties, and so contributes to the maintenance of EPC-network integrity.

3 Detecting Integrity Violations

In the previous section we have presented our model of supply-chain integrity. In this section we first try to answer the question of how to handle integrity violations. We will argue that while preventing those integrity violations in the first place may be desirable, prevention is not always required and economically sensible. Another valid approach is to detect violations retrospectively and to restore the integrity through adequate recovery mechanisms. One approach for detecting violations to supply-chain integrity is to look for evidence in the EPC trace data. We will explore this approach by giving examples of what evidence can be extracted. The overall goal of this section is to understand how far we can get using this approach and to learn the general structure of rules in order to turn them into requirements for building an automated integrity tool.

3.1 Approaches to Handling Integrity Violations

When considering solutions to the integrity issues described above, we distinguish three distinct approaches to take:

- *prevention* of integrity violations (e.g., through the use of redundancy or through dedicated security mechanisms such as digital signatures)
- *detection* of integrity violations
- *correction* of integrity violations (i.e., mechanisms to recover integrity)

Some of the discussed integrity violations could potentially incur huge costs (on the participants and users of the system) and could have other negative effects (like companies not joining the network due to lack of trust). Therefore, prevention of integrity violations would seem to be the preferred approach. However, some of the integrity properties discussed above (both of the physical SC and EPCnetwork) are very hard or even impossible to achieve. That is, prevention of some integrity violations is not feasible or not economically sensible. In such cases, the detection of integrity breaches may be sufficient or even favourable, if the integrity can be restored by subsequently handling the incident. Controlling the physical supply chain is particularly hard. For example, ensuring that no counterfeits enter the supply chain would require very tight physical control and would likely severely hinder supply chain partners to adapt to quickly changing market situations. Though such a tight regime is desirable and necessary in some cases (e.g., pharmaceuticals), in many cases (e.g., luxury goods, toys, and hightech products) it may induce costs beyond the expected losses. On the other hand, the integrity of the supply chain can often be restored (and losses prevented) if the integrity violations are detected and appropriately handled, for example, detecting and removing the counterfeit before it reaches the customer. This is exactly what we intend to achieve and support by our integrity tool. As in the physical supply chain, maintaining the integrity of the EPCnetwork is not always possible. The EPCnetwork is a physically distributed system that has been deliberately built to be open, extensible, and loosely coupled. In general, it is a system with no centralized control (very much like the World Wide Web) though some parts may be more strictly controlled (such as the ONS and Discovery Service). As a consequence, it is often impossible to enforce (or even define) certain *global* system properties. For example, ensuring that each EPC number is only used once in the entire EPCnetwork would require a mechanism to authorize the writer of every tag and the EPC number to be written.

3.2 Example for Finding Evidence

In the following we will discuss in detail a specific violation to physical supply-chain integrity, namely the theft of goods in-transit, that is, during transportation from one supply-chain

partner to the next. In particular, we will preset approaches to detect evidence of this kind of integrity violation from EPC-network trace data and other information sources.

3.2.1 Evidence for Theft of Goods In-transit

A simple (but possibly over-simplistic) approach for detecting theft of goods in-transit is to count the number of products (more precisely, the number of distinct tags read) within a certain shipping unit both at the shipper and the receiver, and matching the numbers. A smaller count at the receiving end compared to the shipping end of the transportation link may indicate theft. However, such an approach could be easily deceived by replacing the stolen goods by plain RFID tags or cheaper tagged products with random EPCs. (Unauthorised goods replacement constitutes another integrity violation in itself.)
A more sophisticated approach would be to match the lists of (received versus shipped) identities. The rule could be formulated as: *All identities associated with a shipping unit at the sender must arrive with that shipping unit at the receiver.* This approach could also detect replaced products in a shipping unit. However, it would be still susceptible to replacing the original products with cloned tags carrying the products' identity.
We could also think of a situation where there is a distributor between the sender of goods and their receiver. If that distributor does not take part in the EPC network or does not share events with either side, the above approach would produce false alarms if the distributor (permissibly but unexpectedly) exchanges products between shipping units (for example, for optimization purposes). In this case one of the following two adapted approaches may be more suitable:

- *All items of a delivery must arrive before the next delivery*. This approach assumes a FIFO (first-in, first-out) ordering, that is, all items of a shipment are received before any item of a later shipment is received. In the face of priority shipments, this may not always be that case.
- *No item that belongs to a certain shipment does arrive later than* n *units of time after any other item of the same shipment.* This approach assumes that there is a deadline after which all items should have arrived. This deadline depends on several factors, such as the transportation means and the processes at the distributor. This approach may not work if the received products typically remain with the receiver for a time span that is shorter than the deadline. This may be the case if the receiver is another distributor (e.g. a distributor of a retailer versus the distributor of the manufacturer) performing cross-docking.

3.2.2 Different Causes may lead to the Same Evidence

An interesting observation is that a missing event could also have other causes (different from theft), which themselves could be considered integrity violations. In the process of finding evidence, events could be missing because

(1) of failed RFID reads (e.g., due to a defective reader or non-optimal read rates from collisions, shadowing, nulls etc.),

(2) a remote EPCIS is temporarily offline (and thus the events in question can not be retrieved remotely),

(3) the remote EPCIS is (accidentally or deliberately) configured to withhold the events in question.

Failed RFID reads (or a number of missing reads beyond a given threshold) and EPCIS downtime beyond a defined availability threshold (list items (1) and (2), respectively) could be considered integrity violations of the EPC-network information system, while withheld information (list item (3)) violates participation integrity. In an honest world without theft, the approaches described previously could be used verbatim to detect these violations of the

EPC-network integrity. This little example illustrates that EPC trace data cannot only be used to find evidence for physical supply-chain violations, but also for integrity violations of the EPC network.

3.2.3 False Alarms

In the previous section we pointed out that we must be careful not to interpret failed RFID reads as theft. Even in the case of an excellent RFID read rate of 99.999% (i.e., a miss rate of 0.001%) there is still one missing RFID read in 100,000 reads. If large volumes of items are processed, this could lead to a high number of false alarms.
In order to fix our previous example on theft in the face of failed RFID reads, we could include read events from multiple readers in the same company. More readers may be installed on the company site to observe other processes, such as shipping to the next partner in the supply chain.

3.3 Summary and Conclusions

In this section we have illustrated that preventing violations to supply-chain integrity may not always be possible. However, there are many situations where the integrity can be restored after the integrity violations have been detected. Therefore, detecting integrity violations is one approach to maintaining supply chain integrity.
In this section we have shown that it is indeed possible to detect integrity violations based on EPC traces (although there are situations where deciding which kind of violation exactly caused the evidence is ambiguous). Concretely we can formulate rules that evaluate this data in order to detect evidence of integrity violations. We have illustrated such rules using a concrete example, namely detecting theft during transportation of goods.
Our examples suggest that there is no single best solution for detecting evidence of integrity violations. Clearly the approach chosen should be carefully adapted to the situation or otherwise many false alarms can be expected, or, even worse, actual theft is not detected. As a consequence, a tool supporting the detection of evidence should be flexible enough to cover a wide range of situations and should allow for dynamic modifications to the rules.

4 Outlook: Rule-based Integrity Checking

In the previous section we have shown how to detect evidence of integrity violations using rules over EPC trace data (and possibly other data, such as advanced shipment notices). For the remainder of task 4.6, we plan to build a configurable software tool that automates much or all of this process. In this section we will present our initial conception of this tool.

4.1 Mode of Operation

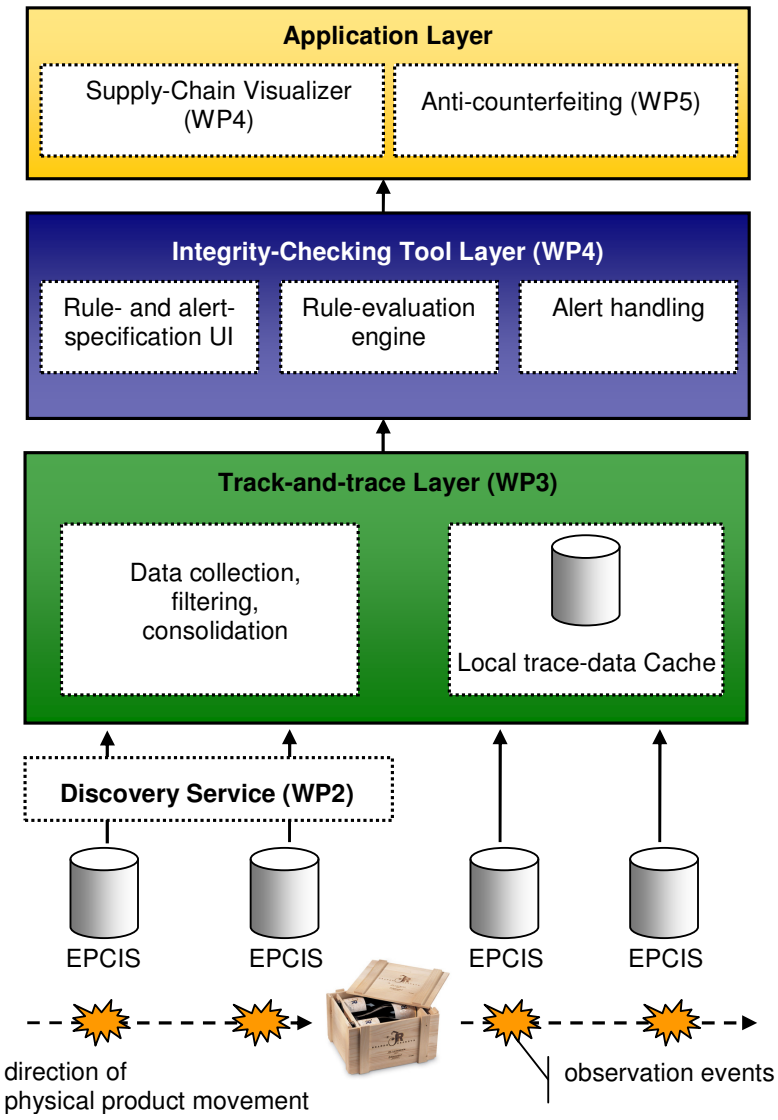
The core of the integrity checking tool is a rule engine that allows users to specify generic integrity rules, like those presented in the previous section 3.2 on page 11. Any rule or combination of rules can fire an alert. This alert can be configured to trigger recovery mechanisms in order to restore the supply-chain integrity. Alerts can also notify humans if further investigation of potential causes of integrity violations is required or if recovery requires human intervention ("hit team"). Alerting as well as recovery mechanisms are beyond the scope of this report and will not be discussed.

We expect that the integrity checking tool is not run as a central service in the EPC network, such as the ONS or Discovery Service. Instead, the tool can be run in parallel by multiple organizations or associations that are participating in the EPCglobal network. A central service would not provide the required flexibility to customize the system to individual needs and situations found in today's diverse supply-chain environments. Also, considering the concerns expressed by several companies regarding the sharing of their item-level EPC trace data [Eurich-07], it seems unlikely that a many companies would open up their EPCIS services to a central integrity service.

On the other hand, one key requirement is that the tool has access to the required EPC trace data, which may partly reside on remote EPCISes run by supply chain partners. We expect that the sharing of item level EPC trace data is governed by contracts between the individual parties, which is common practice today.

From a technical perspective, we expect that remote EPC trace data is retrieved, filtered, potentially condensed, and then cached locally. In order to retrieve remote EPC trace data, the integrity checking tool will utilize the infrastructure components developed in the BRIDGE workpackages WP2 and WP3, namely the Discovery Service and the enhanced Track&Trace framework, respectively. An initial system overview is depicted in the figure below.

If possible, remote data is retrieved in near real time in order to facilitate continuous rule evaluation. Since this may not always be possible (depending on the complexity of rules and the amount of traces involved), retrospective analysis will also be considered.



4.2 Rules and Rule Representation

In order to allow users to specify custom integrity rules, a representation language must be found that can express these rules. The rule language should be powerful enough to specify the most common integrity rules. However, at the same time it should be easy enough to be understood by end users configuring the tool. Languages serving a particular purpose in a specific domain are often called domain-specific languages. Unlike general-purpose programming languages, domain-specific languages are typically less complex.
We have not finished the process of evaluating which elements our rule-representation language must feature in order to be expressive enough. At the moment we are considering a (subset of) predicate logic (i.e., first-order logic). Our language mainly operates on EPCs, EPCIS events, sets of EPCIS events, as well as EPCIS event attributes and may contain:

- **Relations**, for example, (1) event e_1 occurred after event e_2, (2) EPC n_1 is of the same product class as n_2, (3) EPC n_1 is element of a set S.
- **Constants**, such as (1) true and false, (2) the empty set, etc.
- **Functions**, for example, (1) the set of child EPCs for a given aggregation event, (2) the ordered set of parent EPCs for a given EPC number and time period.

- **Variables**
- **Logic operators**, such as `not`, `and`, as well as `or`.
- **Quantifiers**: existential quantification (i.e., "there exists") and universal quantification (i.e., "for all")

We expect that the language will be based on a given fixed set of functions. Such a language would still allow the expression of a wide range of rules, without being fully programmable like general-purpose programming languages. We could, for example, express a rule such as the one presented in section 3.2.1 on page 12: "*All items of a delivery must arrive before the next delivery.*"

4.3 Programming Tools: Generic Rule Engines

In the open-source domain and through commercial offerings there exist a variety of generic rule engines and rule representations as programming libraries. Such libraries could significantly reduce the programming effort required to implement our integrity checking tool. We are planning to perform an evaluation about which of these libraries meet our requirements. Some of the tools we will consider are Drools (JBoss Rules, Drools Rule Language DRL), OOjDREW (RuleML), and Jess (CLIPS). We will also consider libraries for event-stream and complex-event processing, such as ESPER (EQL).

5 Prototype: The Supply Chain Visualizer

In the previous sections, we have provided a conceptual perspective on detecting integrity breaches. In this section, we report first results of the prototype development during the first 18 months, which was carried out by ETH Zurich. A comprehensive description and evaluation of our prototype will follow in the next deliverable. Our prototype shows how to detect integrity violations across the supply chain through sophisticated analyses of EPC trace data. The prototype is targeted at a managerial audience and entitled "Supply Chain Visualizer" as its primary goal is to make supply chain integrity breaches visible to the supply chain manager. Currently, we have implemented an EPCIS connectivity layer (based on Accada), a track and trace event gathering layer, a trace data warehouse component, an extensible analysis component with several integrity checking rules, and also a first version of the map-based user interface.
In the following, we will first present the motivation and objectives of our prototype. Then we will present its architectural design, and details on the already implemented integrity rules of the first development phase. Finally, we conclude this section by summarizing the key points and giving an outlook about the second development phase of the prototype.

5.1 Motivation and Objectives

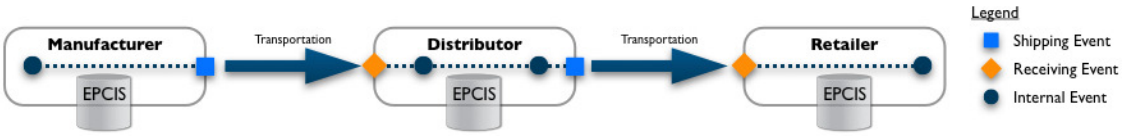
Today, supply chain managers typically have a very good view on the supply chain on an aggregate level. However, on the item level visibility is limited and actual processes remain hidden. The Supply Chain Visualizer is a tool that intends to raise the visibility of otherwise hidden processes on the item level, providing simple, aggregated output. The Supply Chain Visualizer displays "hot spots" to indicate weak links and integrity breaches in the supply chain. It provides bottom-up supply-chain consistency and performance analysis based on EPC-trace data.

5.2 Design

This Supply Chain Visualizer is based on Electronic Product Code Information Services (EPCIS) which is an industry-driven standard describing the use of RFID in business contexts. Within this standard, each object is identified by a unique Electronic Product Code (EPC). This code is a serialized number that could contain information on the product's type and origin. As the EPCs of individual products are read at each node in a supply chain, companies will be able to manage their supply chains on an item-level granularity, which is not yet possible today. The data can be used to describe, for example, the incoming number of items, current items in stock, and the outgoing number of items at a given location. Furthermore, analysis of the EPC data can reveal problems such as loss, damage, and theft of items in the supply chain.

5.2.1 Assumptions

As the Supply Chain Visualizer is a prototypical implementation and not a commercial product, we base our development on the following assumptions. We consider a supply chain where every product is equipped with an RFID tag. Every time a RFID tag is read, an event record is generated. As the Supply Chain Visualizer is focused on the inter-organizational supply chain aspects, we consider three different types of events. We use a so-called shipping/receiving supply chain model and differentiate between 1) shipping, 2) receiving and 3) internal events. An example for this model can be seen on the following picture.

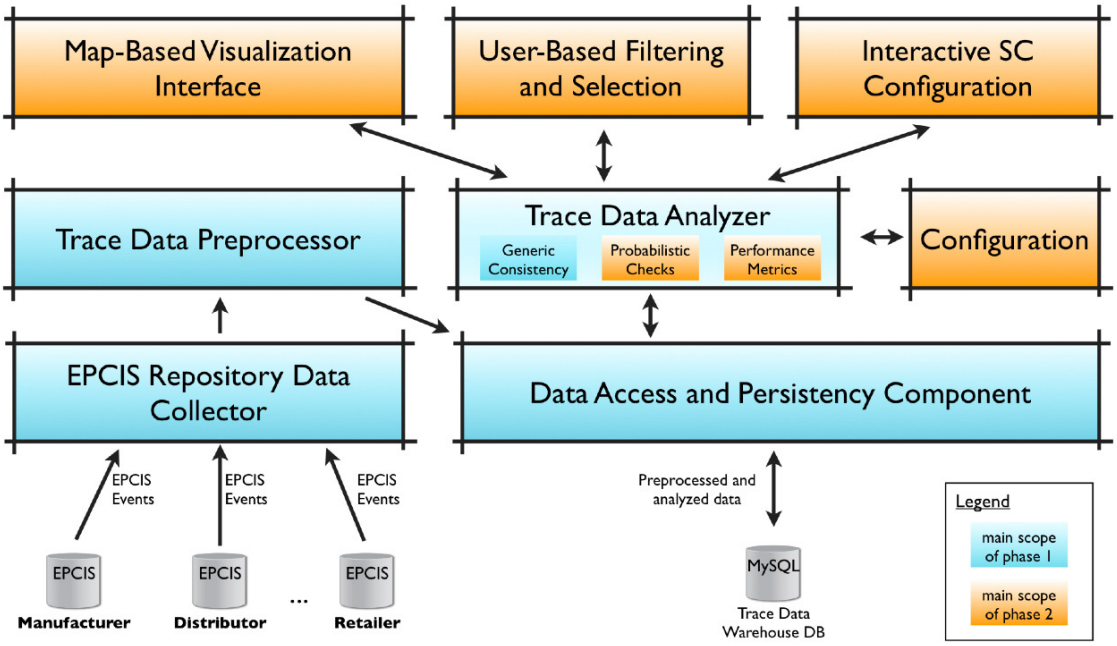


As input data, we expect events following the EPCIS event standard. However, as the EPCIS-event standard contains many optional fields and value types, we assume that all incoming events provide the following set of data fields: *epcList*, *eventTime*, *action*, *bizStep* and *bizLocation*. An input event could look like the following example.

```
<ObjectEvent>
      <eventTime>2007-11-01T08:24:38</eventTime>
      <epcList>
            <epc>1234.17.327115</epc>
            <epc>1234.9.652913</epc>
      </epcList>
      <action>OBSERVE</action>
      <bizStep>urn:autoidlabs:bizstep:receiving</bizStep>
      <bizLocation>
            <id>urn:autoidlabs:loc:coordinates:37.5667,127</id>
      </bizLocation>
</ObjectEvent>
```

5.2.2 High-level Architecture

The following picture shows the high-level architecture of the Supply Chain Visualizer prototype. Blue boxes indicate components that are already completed. Implementation details in the following parts refer to these components. Orange boxes refer to components that are not completed yet and will be described in the future work section.



5.3 Trace Data Analyzer

The core of the Supply Chain Visualizer is the Trace Data Analyzer. It consists of a generic consistency rule-engine, probabilistic checks, and performance metrics. In the following, we focus solely on the generic consistency rule-engine, as the other two components are yet to be implemented. The generic consistency rule-engine conducts a data-driven integrity analysis based on the activated rules and the content of the data-warehousing component.

The rule-engine offers several methods to speed up the development of new rules. To add a new rule, only one interface method must be implemented. Several shortcut functions do exist for common data filtering or selection tasks.
In the following, we will explain some of the already implemented rules. Based on the data requirements specified above, the rules are intended to be universally applicable to different types of supply chains. Thus, they are parameterized and can be adapted to specific supply chain needs by simply changing the configuration file of the Supply Chain Visualizer.
Let $ooe\,(epc)$ be the set of events sorted in ascending order by time for a particular object with electronic product code epc.

5.3.1 Movement Consistency

5.3.1.1 Description

The flow of an object through the supply chain must adhere to the laws of movements which are bound to physical constraints. Before an object exists it must be created. At the end the life cycle of the object, it gets terminated. During its life, the object flows through a series of shipping and receiving operations, whereas in between of a receiving and a shipping operation there might be some internal operations. Therefore the object can not move faster than the transportation mechanism allows it to.

5.3.1.2 Implementation

The object event in $ooe\,(epc)$ with the lowest event time must have an action field set to 'ADD'. There must not be an object event for an object following an object event which has an action field set to 'DELETE'. All object events which occur after the object event with the add action and before the potential object event with the delete action at the end, must have an action field set to 'OBSERVE'.
In $ooe\,(epc)$ the first object event must have an internal business step. For every object event which is not the last object event, the following must be true. An object event with an internal business step must be followed by an object event with an internal or a shipping business step. An object event with a shipping business step must be followed by an object event with a receiving business step. An object event with a receiving business step must be followed by an object event with a internal object event.
Let $(oe0,\ oe1)$ be two subsequent object events in $ooe\,(epc)$ where $oe0$ has a shipping business step and $oe1$ has a receiving business step and $oe0$ being the object event with the lower event time. For these object events the following must be true.

$$calculatedLag\ (oe0,\ oe1) := \frac{distance\,(oe0,\ oe1)}{speed\,(distance\,(oe0,\ oe1))}$$

$$calculatedLag\ (oe0,\ oe1) - \varepsilon \leq lag\,(oe0,\ oe1) \leq calculatedLag\ (oe0,\ oe1) + \varepsilon$$

Thereby $speed$ is a function which returns the speed of an appropriate transportation mechanism for the distance provided as the argument.

5.3.2 Stopover Time Consistency

5.3.2.1 Description

The amount of time a certain object spends within a business location must be upper bounded by a maximal time span.

5.3.2.2 Implementation

Every sub-sequence of $ooe\,(epc)$ in a sequence having the same business location defines a stopover. The duration of this stopover is given as the lag between the event times of the first object event and the last event within the sub-sequence and must not exceed a certain threshold.

5.3.3 Inventory Consistency

5.3.3.1 Description

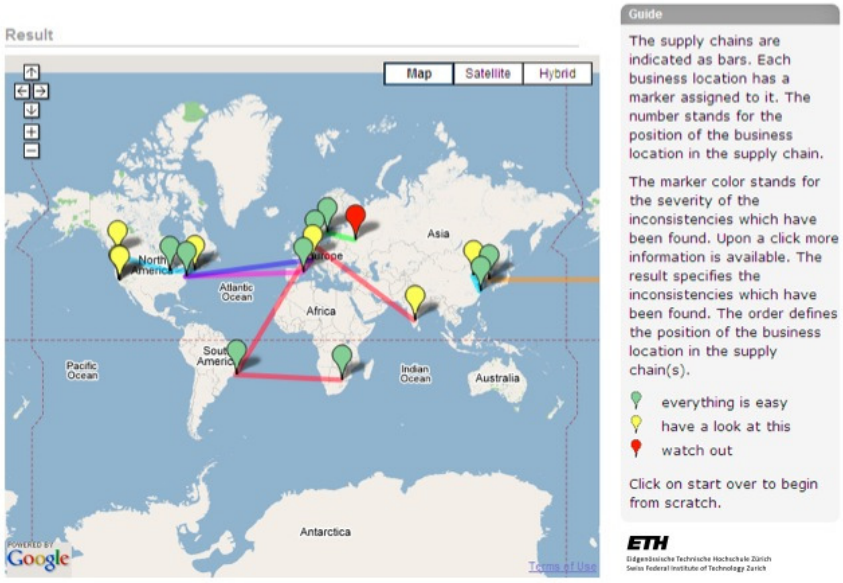
The inventory indicated by the EPC data must correspond to the expected inventory. A mismatch can either be due to inventory shrinkage or inventory increase.

5.3.3.2 Implementation

The objects currently present in a facility of an organization are given as all of the objects which have an object event with a receiving business step and a business location which correlates to the facility, but without a subsequent object event with a greater or equal event time and a shipping business step. These objects need to be synchronized with the inventory data of the facility's ERP system.

5.4 User experience of the prototype

The Supply Chain Visualizer is a web-based application. A standard computer with a web-browser and Internet connection is sufficient. No specific client software is needed. The screenshot shows how a supply chain manager can pinpoint problematic locations at a glance. Product flows are highly aggregated to single lines in order to provide a clear view on the whole supply chain. With an easy to understand three-color scheme, a supply chain manager can now see where the problems are. Green pins indicate that the integrity analysis did not find any problems at these places. Yellow means that there are some problems detected which deserve attention and red indicates that the number of problems exceed a critical threshold. By clicking on the pins, more details about the type of integrity violation is revealed.



5.5 Technical Details

The Supply Chain Visualizer is a web-based application. Our prototype runs on a Debian Linux 2.6.18-4. The CPU is an Intel P4 2.53GHz. The Supply Chain Visualizer prototype is based solely on open source software. As servlet engine we use Apache Tomcat 6.0.13 with the MySQL Connector 5.0.5. The supply chain data and the analyzer's metadata are stored in a MySQL database based on the MySQL Server 5.0.38.
The programming language is Java. Specifically, we use the Java J2EE SDK V1.5.0. Implemented servlets follow the Servlet Spec V2.5 and Java Server Pages V2.1 specifications. To improve the extensibility and code quality, the Spring MVC Framework is used. The Spring Framework separates application, presentation and storage logic in a

convenient way. As a persistency interface with object-relational mapping, we use Hibernate 3 with a Spring Hibernate template. To visualize the output, the Google Maps API is used together with the DWR Ajax interface.
The prototype was tested successfully with EPCIS systems based on the Accada EPCIS V0.2.2 software. All software was programmed in an Eclipse 3 development environment. Build and source code management is based on Maven 2 and SVN V1.4.2.

5.6 Conclusions and Future work

We have created a first version of a powerful tool that can assist supply-chain managers to visually locate integrity violations in their Supply Chain. In a next step, we want to put our tool into action and collaborate with Work Package WP9 and other business work packages of BRIDGE to visualize their RFID trace data. We expect that this novel approach of Supply Chain Visualization should not be limited to the attention of the academic community, but should also be of great interest to industry. In addition, we want to further improve the Supply Chain Visualizer in a second development phase. Hereby we focus on the following fields:

- **Performance Metrics**: For every node in the supply chain, performance metrics can be inferred from the EPC data available. The metrics can consist of measurements of the node's performance in terms of average number of items handled per day, average number of items lost per day, etc. Further analysis will be made in order to decide which performance measures are most relevant. The goal is to maximize the scope of application of the tool by showing simple, yet powerful performance metrics.
- **Probabilistic-reasoning Approaches**: Probabilistic-reasoning approaches can be used to perform further analysis on the data in the EPCIS repository. An example of an analysis based on this approach is detection of cloned EPCs. This issue is a problem in many industries, where copies are tagged with a fake EPC and at some point introduced in the supply chain. Cloned EPCs severely threaten the integrity of supply chains. Rules could be developed to detect such scenarios, as two instances of the same EPC are not supposed to exist in the supply chain at the same time. We aim to collaborate in this issue also with work package WP5 (Anti-Counterfeiting).
- **Handling Aggregation**: Each time a group of products are packed for transportation, the pallet or container is typically tagged with a new EPC. Rules can be made to check if the number of the items during the packing corresponds to the number of items when the box is opened again. This way, also business integrity problems in the supply chain can be detected.
- **Visualization**: As the amount of data in real world EPCIS repositories are typically very large, additional algorithms for selecting relevant data will be implemented before the data are presented in the Visualizer. An important issue to address is the selection of data to be displayed. The selection can be based on criteria such as requiring that items of the same type and with the same time stamp are only displayed once.
- **Generalization of Rules**: Currently, the Supply Chain Visualizer can handle a selected few rules of general applicability, which are hard-coded into the system and cannot be changed dynamically. In order to support the dynamic creation and modification of rules (to adapt to new and non-standard situations), we intend to implement flexible rule representations and a rule engine as described in Section 3 of this document.

Finally the result of the analysis will be presented in a way that is easy to understand and easy to communicate. Emphasis will be put on creating an interface where few interactions are needed and where the results are self-explanatory.

6 Bibliography

[NIST-95] National Institute of Standards and Technology, An Introduction to Computer Security: The NIST Handbook, Special Publication 800-12, Oct 1995. Available at http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf

[NRC-92] National Research Council, *Computers at Risk*, Washington, DC: National Academy Press, 1991.

[NRC-a] Department of Defense, *Glossary of Computer Security Terms*, Pub. NCSC-TG-004, National Computer Security Center, Ft. Meade, MD 20755 (Oct. 1988). Also known as the "Teal Green Book." Available at http://packetstormsecurity.org/docs/rainbow-books/NCSC-TG-004.txt

[Fearne-00] Andrew Fearne and David Hughes, *Success factors in the fresh produce supply chain - Insights from the UK*, British Food Journal, Vol. 102 No. 10, 2000, pp. 760-772. Available at http://www.wye.imperial.ac.uk/CFCR/pdfdoc/fresh-produce.pdf

[Bridge-412] EU FP6 IP Project “BRIDGE: Building RFID Solutions for the Global Environment”. Deliverable D-4.1.2 *A Threat Model Analysis of EPC-based Information Sharing Networks, 2007*

[GDSN] GS1 GDSN (Global Data Synchronisation Network). Available at http://www.gs1.org/productssolutions/gdsn/, accessed December 21, 2007

[EDI] *Electronic Data Interchange*, Wikipedia, The Free Encyclopedia, 18 December 2007, 13:25 UTC, Available at: http://en.wikipedia.org/w/index.php?title=Electronic_Data_Interchange&oldid=178713359, accessed December 21, 2007

[Eurich-07] M. Eurich and N. Oertel, *Interorganisationales Teilen von Ereignisdaten auf Stückebene: Gegenwärtige Barrieren und Möglichkeiten zu deren Überwindung*," SAP Research. To be published.