Implementation of Fosstrak EPCIS RFID System

Thesis presented by
Fanny Ulldemolins Bes

Corresponing to
Engineering in Telecommunications

Czech Technical University
Prague, Czech Republic

2012
()

To my parents and sister.

Acknowledgments

First of all, I would like to thank my teacher and supervisor, Prof. Lukas Vojtech, for his support and patience in the final implementation of this project, for understanding all the setbacks I have come up against during the past year in trying to advance it. I would also like to thank everybody in the Telecommunication Engineering Department of the Czech Technical University of Prague, for having taken me in during my time Prague and helping me with everything I asked of them, and especially Prof. Louport, for his encouragement regarding my choice of topic for the project.

I wish to dedicate this thesis to my parents, and in particular to thank them for making it possible, for always supporting me and for giving me the courage to do everything I wanted to do, and for making a big effort to provide me with the means to study my degree of choice. So, every academic and personal achievement is dedicated to them: Baptiste Pedro and Herminia. And also to my sister Sheila, who every day transmits the spirit of self-improvement and the ambition to achieve whatever I want. She is my source of energy and my reference.

I would also like to take this opportunity to mention all the friends who have been close to me and have shared every happy and sad moment with me throughout these university years, full of stressful times and new experiences. And also my flatmates: Judit, Ivan, and Joan Jordi, who have been like family to me during the best years of our lives.

Finally, I would like to express my gratitude to everybody who has contributed in some way to producing this master's thesis. Those who gave me either technical or personal support while developing the thesis: Antonio Sejas, Gloria Monlleu, and especially Neus Amado and Adrià Martí, my second family, without whom this thesis would not have been possible - they taught me a new meaning of friend. Thank you so much.

Abstract

Nowadays, the importance of sharing information is increasing at a dramatic pace, as everything is connected in some way or other. This is also the case with logistics and supply chain management. The idea of the Internet of Things is growing, and intends to include information about every single item around the world. Such a huge project is attractive to many companies wishing to get involved and, as a result, several projects are emerging with the desire to contribute in some way.

The idea of this study was to analyze one of these projects aimed at facilitating work with RFID systems following EPCglobal standards. Fosstrak is open-source RFID prototyping platform which implements EPC Network specifications. It is composed of four different modules which are analyzed and compared with some other open-source software applications in order to ascertain the contributions and limitations that the tool could present for a RFID supply-chain management project.

Abstract

En els temps en els que vivim, la importàcia de compartir informació està creixent vertiginosament ja que tot està connectat d'alguna forma. Això també aplica per a l'àmbit de la logística i del control de magatzem, i està creixent la idea de l''Internet de coses", que intenta englobar informació de tots i cadascun del items de tot el món. Aquest gran projecte que es planteja es una oportunitat molt interesant per a moltes companyies, i estan apareixent molts profectes amb l'intent de contribuïr d'alguna forma a aquesta fita comú.

Aquest treball neix amb la idea d'analitzar un d'aquests projectes que intenten facilitar el treball amb sistemes RFID que volen seguir els estàndards que regula la organització d'EPCglobal. Fosstrak es una plataforma software de codi lliure que implementa les especificacions de l'arquitectura d'EPC Network. Es un projecte composat per quatre moduls diferents que s'analitzen un per un, comparant-los en algun cas amb altres softwares de codi lliure creats per amb el mateix fi, per comparar-los i extreure conclusions sobre quines son les aportacions i limitacions que pot presentar l'ús de l'eina de fosstrak per a un projecte RFID de control de magatzem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The Internet of Things is a term that has been around for several years. It was first introduced by the MIT Auto-ID Center, the precursor to the current EPCglobal organisation and at that time stood for the vision of a world where all physical objects are tagged with an RFID transponder with a globally unique ID - the EPC or electronic product code. RFID easily allows tracking the objects and the EPC servers as a link to data which can be queried over the Internet about each individual object.[21]

Today, RFID is the baseline of a common project and has the major responsibility for the sector's growth. It is starting to be used in different enterprises among of them: supply chain management and store's logistic control. RFID technology has improved and now it is possible to record additional information about the object or the object's environment inside of the object's tag. With all this information combined with easy software is possible the data processing on the item and local control procedures without human intervention. However, the complexity required in a sectors of, for example, warehouse management and supply chains makes necessary the use of a complex interface.

In the way to homogenize as much as possible utilization and adaptation of the whole huge network's customers has born an open-source software linked directly to the EPCglobal standards, called Fosstrak. It is a potential tool what is on the way to give the companies the complete possibility for getting inside of the EPCglobal Network.

The Internet of Things is a key part of the Future Internet. Many new opportunities can be foreseen for citizens as well as for businesses and other organizations, but also for the society as a whole.[21]

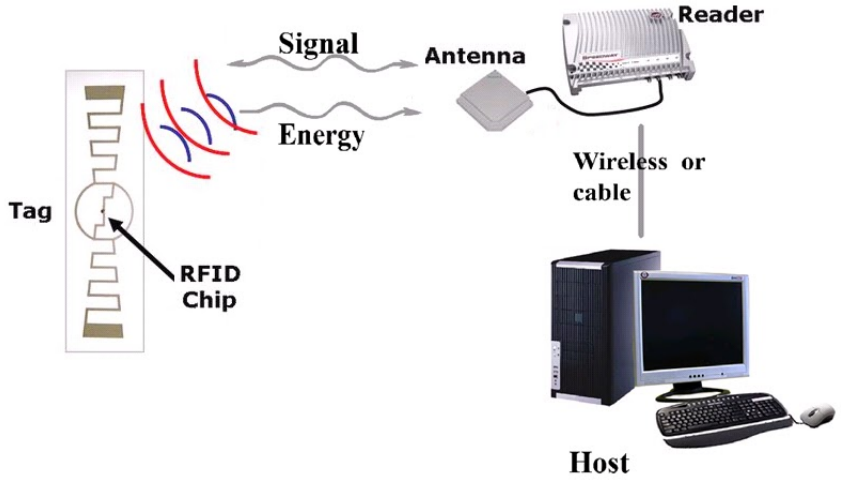


Figure 1.1: RFID System. Tag, reader and host computer for processing data. [1]

1.1.1 RFID

RFID (Radio Frequency Identification) is one of the most important developments of our time. One of its important application is the control of objects, with tagging, tracking and identification on the way to supersede traditional barcode system.

RFID system is implemented by a reader with an antenna (interrogator) and a transponder (RF tag) programmed with unique information. The purpose of an RFID system is to enable data to be transmitted by a portable device (tag), which is read by an RFID reader and processed according to the needs of a particular application (See figure 1.1).

RFID uses radio frequency waves and that allows not necessary direct visibility between the tag and the reader and also can detect more than one object simultaneously. Both of these reasons are the most important advantages that RFID gives to apply in our study.

The potential and current uses of RFID technology are many and very different, for example: security and access control, payment systems, retailing or animal and health control. In this thesis we will focus in the application with supply chains and warehouse management sectors, analyzing the current options and realizing the best way to work with it for future developments of new and adaptable technology.

1.1.2 EPCglobal

EPCglobal is leading the development of industry-driven standards for the Electronic Product Code (EPC) to support the use of Radio Frequency Identification (RFID) in today's fast-moving, information rich, trading networks. Their goal is increased visibility and efficiency throughout the supply chain and higher quality information flow between companies and their key trading partners.[5]

As a joint venture between GS1 (formerly know as EAN International) and GS1 US (formerly the Uniform Code Council, Inc.), EPCglobal leverages a nearly 30-year heritage

of successfully partnering with industry to promote and control the use of EPCglobal Network over all sectors and throughout the world.

EPCglobal Network

The EPCglobal Network is a framework that enables immediate, automatic identification and sharing of information on items in the supply chain. In that way, the EPCglobal Network will make organisations more effective by enabling true visibility of information about items in the supply chain. Using a combination of technologies and harnessing the power of current information systems, the EPCglobal Network will provide for immediate, automatic, and accurate identification and location of any item in the supply chain of any company, in any industry, anywhere in the world.[22]

Is also known as EPCglobal Architecture Framework and govern the rules between the EPCglobal standards and its five fundamental elements: EPC, RFID system, EPC middlewave, EPC Information Services and Discovery Services. The main goals of the EPCglobal Network are:

- to control and simplify the exchange of information and physical objects between authorized users.
- encourage innovation and adhesion to the wide-world network.

1.1.3 Supply chains and warehouse management VS RFID and EPCglobal Standards Integration

The supply chains and warehouse management is a key for the companies to be competitive nowadays. The main objective of supply chains and warehouse management is the most effective control and with the best price, that is the control of the flows, movements, inventories of finished products and all the information about any item, from a source place to the destiny.

RFID technology is being deployed widely and very efficiently into the supply chain management's world: to identify the product and in the warehouse to locate product easily without barriers and real-time, unlike a bar code, which must be visually located and scanned for on-demand operation. At figure 1.2is possible to see how it works in a warehouse and how makes easier to track and locate the objects tagged with RFID readers.

RFID is a technology that allows companies to: [23]

• Increase visibility and accuracy of the supply chain by providing real-time visibility to product flow.

• Reduce the time it takes to make important supply chain decisions by providing access to real-time information.

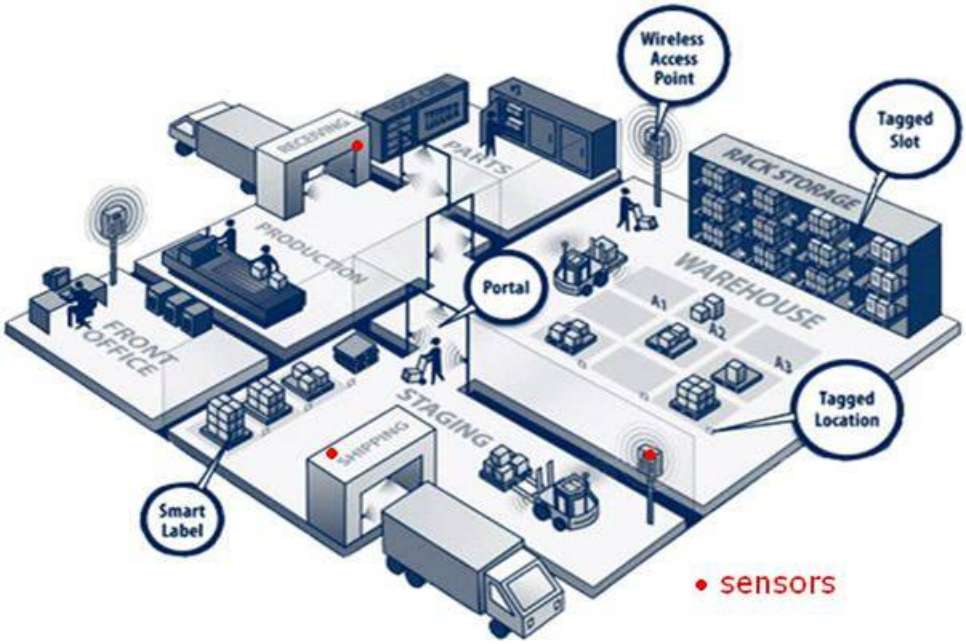


Figure 1.2: Warehouse management through RFID technology. Tracking and Locating Made Easier. [2]

• Respond quicker than ever before to customer demands through better information management.

• Improve error correction ratios by increasing levels of automation.

However, to operate all these facilities is very important the sharing of information to dispose and use it. Is in this point that EPCglobal appears and is absolutely essential, to provide a common rules to tag the items with Electronic Product Codes, to the RFID system's communication with EPC Information Services and to have a shared Database to look up and extract objects, definitely to work as a huge Network.

1.1.4 Power of Fosstrak tool

Many companies dealing with RFID technology have been fast to deploy any software product to apply in the supply chains and warehouse management using all their products and obtaining a complete package to offer another companies all the facilities and advantages already numerate from RFID systems. But it can be just a sharing dates between the companies with the same product, so, in an internal network, does not allow to share with some other product with some other companies. The first software what enables the interconnection between different companies, from different parts of the world, because follow the EPC global Standards in all the protocols inside of the communication between

reader and tag is Fosstrak:

"Fosstrak is an open source RFID software platform that implements the EPC Network specifications. It is intended to support application developers and integrators by providing core software components for track and trace applications."[11]

Fosstrak provides four separate modules: the Fosstrak EPCIS Repository, the Fosstrak Tag Data Translation (TDT) Library, the Fosstrak Filtering & Collection Middleware with ALE and LLRP Support and the Fosstrak LLRP Commander. They can be used together compliments each other or separately.

One of the main aims of this thesis is analyze deeply the complete tool, discovering how useful could be for a future project about a real case of supply chains management or which problems should be solved before.

1.2 Aim and structure

The main questions that this thesis attempts to answer are:

Is it possible to integrate and merge the Supply Chain Management from an organization using Fosstrak implementation to follow EPC global Standards? How to use it? Which advantages does it have and which disadvantages shall be solved? And finally if could be Motorola XR480 RFID reader compatible with this integration.

To deal with this task, this is the structure of the thesis work:

1. **Introduction:** the current chapter what presents the motivation what cause the study and the problem that the work attempts to answer.

2. **Background:** this chapter provides an introduction of RFID technology and EPC global word, trying to explain as much clear as possible all its Standards and its Architecture Framework, due to understand the connexion between both.

3. **Fosstrak tool:** this chapter tries to be kind of manual to help further Fosstrak users to use it correctly and to get the most of its possibilities as a novel user.

4. **Case of study:** this chapter presents the case of study, where the initiative of the study was coming from and how the thesis deal with it.

5. **Related work:** this chapter defines the developed work, the application of fosstrak tool to implement a RFID system with our own tools, interacting with EPCglobal framework. Presents the evolution on the work since the beginning trying with others ALE Servers a part of Fosstrak ALE Server, and presents individually each system component and the role that it takes.

6. **Evaluation:** this chapter analyzes deeply the problems which are being found during the related work and presents the results archived, as well as some comments to the EPCglobal solution and fosstrak.

7. **Conclusions:** this chapter introduces some final comments related the study made and encouraging the companies add to the EPC global Network movement.

Chapter 2

Background

2.1 RFID Technology

RFID (Radio Frequency IDentification) is a technology what uses radio waves to transfer data from an electronic tag, called RFID tag, what is attached to one object, through one reader with the purpose of identify and track the object. RFID technologies are grouped as so-called AUTO-ID (AUTOmatic IDentification).

A typical RFID system consists of 3 elements:

- One reader (usually known as interrogator or read/write device), which includes also an antenna and emits radio waves in ranges of anywhere to activate the RFID tags. Is the u to decode the tag information.
- One tag (also called Transponder), which encodes the data in a integrated circuit (silicon chip) and is attached to a radio antenna, that detects the reader's activation signal when it is through the reader's zone and send the data to the reader. Also can contain a battery.
- One controller (or host), which process the tag's data sent by the reader. Has to contain some control software or middleware.

Radio transmissions are used by the reader to send a query to the tag, and by the tag to return an answer, generally containing identifying information. The tag may not have its own radio transmitter, Instead, the tag uses the radio energy transmitted by the reader as its energy source (reflecting with a technique called backscatter). Finally, the reader sends the identifying information to a host computer for processing.

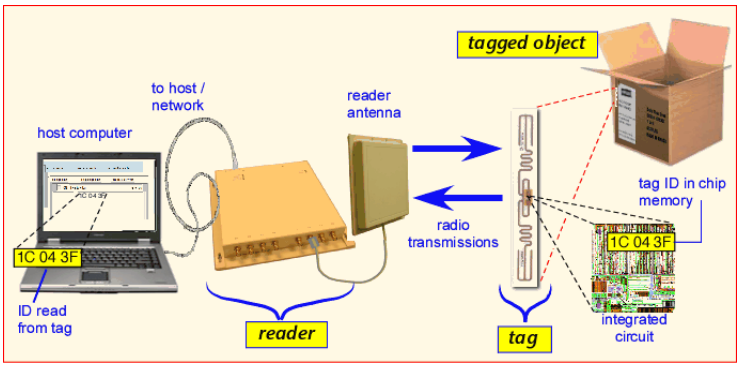


Figure 2.1: RFID system on supply chain application: an attached tag with integrate circuit, a reader with an antenna and a host. [3]

The technology has invented almost seventy years ago and its uses has been changed a lot. Since León Theremin invented an espionage tool for the Soviet Union in 1945 (considered the predecessor of RFID) as a result of many physics advances related to radio frequency, until the huge advance that the technology has develop in our days, RFID has past for a lot of different fields and applications, for example military applications during the WWII, transport or light and sounds transmissions.

Nowadays the use of the RFID technology is wide and diverse If you take care around you, for sure you will find many of RFID uses, but always with the same aim: identify as a unique one item . Some of the fields where it is used are: Manufacturing, Payment systems, Animal identification, hospital and Health-care, department of defense, passports, schools and universities, race timing, ski resort, etc. But in the sector where it is being implanted hardest because gives really good results in efficiency and speed is in the supply chain and warehouse management on the way to replace traditional barcode systems and achieve actuation without any human intervention.

2.1.1 RFID System

2.1.1.1 Reader or Interrogator

The reader or interrogator is the bridge to pas the information from the tag to the host or controller for manage it with the middleware software. It works sending and receiving radio frequency's waves to detect the tags and to be able to read and write from and to the tag.

Depending on the characteristic that we are taking on care there are several possible classifications of RFID readers. The most common and useful for the practice is:

1. **Stationary readers**: are devices which work with standardised interfaces (Ethernet)

that connect them to an operational host software data management system.

   (a) *Reader HP*: the classic RFID reader with more features. It is approved by all regulatory agencies by radio and communication protocols to be used for any application without any problems. They can be used in environments multireader because they accept up that 4 antennas.

   (b) *Reader LP*: with less features than reader HP and more economic.

   (c) *3D Reader*: which obtains a total visibility of the operating environment.

2. **Mobile readers**: compact solutions with an integrated antenna that enables a manual tags identification. Detected data can be stored in the RFID reader and then send on to its host computer system. Can be differentiate between as a Forklift reader or Handheld reader.

The RFID readers can be classified also by the complexity and capacities that reader has, as Passive readers or Active readers. The first ones are limited to only listening to transponders and the second ones are true interrogators which interrogate and listen to tags depending on the number of tags and communication protocols embedded in the system for data transmission between the tags and the readers.

There is another important classification of the RFID readers by the frequencies that they work, but this is a following topic.

At time to choose the reader for our application and system, we should consider some features and factors to obtain the reader that fits better for our needs. These are, for example, the sensibility (to get signals with low potency), the selectivity (to get the signals from the tags between a lot of others radio frequency signals), works under regulations and operational capacity between different manufacturers (EPCGlobal standards).[24]

LLRP reader LLRP readers are the devices ready to work following the rules of the communication protocol designed by EPCGlobal called LLRP. This allows that different readers belonging to a big network, can understand each other even if they come from different countries, companies, sectors...

2.1.1.2 Tag or Transponder

The transponder's function is store the information of the product which is labeled to and to make the communication with the reader possibility. In this aim, the tag contain a chip, which does not have a power supply by itself. Depending on the way that RFID chip is being fed, the RFID tags can be classified in 3 groups (figure 2.2):

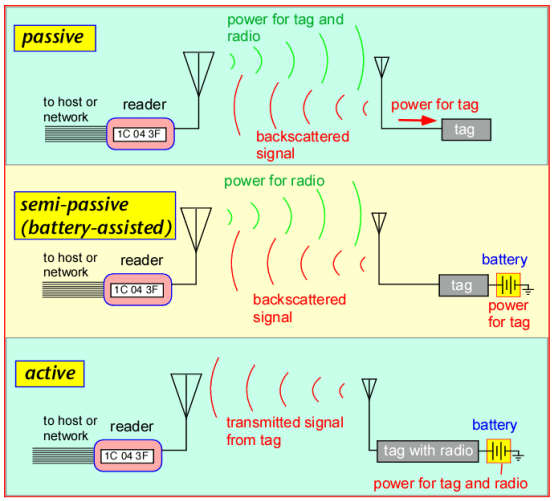


Figure 2.2: RFID tags types. *[3]*

1. **Active RFID tags**: have their internal battery to supply enough power to transmit the waves by themselves. It can achieve long distance of signal transmission but the main disadvantage is that it is more expensive.

2. **Passive RFID tags**: do not have an internal battery and it is supply by the power that the electromagnetic wave from reader has. The main disadvantage of passive tag is that its data transmission distance is shorter than an active tag. However, its price, dimension and easy-to-use determines that it is the main stream of RFID tags.

3. **Semi-Passive RFID tags** (or battery assisted tags): have an internal battery but they are just activated for supply the circuit when the tag has to be read but not to generate the radio frequency wave.

Another very important aspect to classify tags is if their internal memories can be written or, instead of this, can only be read. If the memory inside of the tag is a ROM memory (Read-Only-Memory) the information of the product what the tag contains was introduced by the manufacturer and it does not allow to be written more than once, while it can be read for many times. These tags are called **Read-only tag** (**RO**) and they are small in capacity because contain few data. Contrary, when the memory inside of the tag is a programmable memory like an EEPROM (Electrically Erasable Programmable Read Only Memory), the information that the tag contains can be read, erasure and written again every time that is needed. These tags are called **Read-Write tags (RW)** and the capacity is bigger than in the read-only tags because is normal that there is also other information about the identified item, such as production information, counterfeit-proof verification code etc.

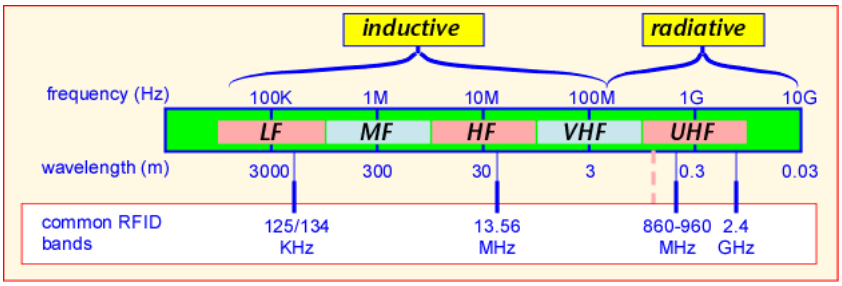


Figure 2.3: common frequency bands used for RFID.[3]

Also, depending on the frequency that the tags are working with, they can change considerably in their aspect and functionalities. But this is a following topic.

2.1.1.3 Host or Controller

The data acquired by the readers is then passed to a host computer, which may run specialist RFID software or middleware to filter the data and route it to the correct application, to be processed into useful information. It is usually a server running a database and some application software that enables control different readers together.

2.1.2 RFID Frequencies

As its name suggest, RFID (Radio Frequency Identification) is a technology which use radio waves to communicate between system components. The frequency of these waves depends on the application that the system is made for, but is always into a frequencies range (from 30Khz to 3GHz).

Some other common technologies are using radio frequencies as well, like Bluetooth or Wi-Fi, therefore is very important take care with the election of the frequency to do not generate interferences.

At figure2.3 we can see the four rather widely-separated frequency bands that are used by most of RFID applications.

Several issues are considered in choosing a frequency of operation. One of the most important is the wavelength, because affects directly to the antenna's size and the distance between the reader and the tag. When the antennas are very small compared to the wavelength, the effects of the currents flowing in the antenna cancel when viewed from a great distance, so there is no radiation. These are the ***inductive*** RFID systems and use antenna sizes from a few cm to a meter or so, and frequencies of 125/134 KHz (FL) or 13.56 MHz (HF). Contrarily, the ***radiative*** systems use antennas comparable in size to the wavelength. The very common 900 MHz range has wavelengths around 33 cm. Reader

Frequency Band	Description	Operating Range	Applications	Benefits	Drawbacks
125KHz to 134 KHz	Low Frequency	< .5M or 1.5ft.	• Access Control • Animal Tracking • Vehicle immobilizers • Product Authentication • POS applications	Works well around water and metal products.	Short read range and slower read rates
13.56 MHz	High Frequency	< 1M or 3ft.	• Smart Cards • Smart shelve tags for item level tracking • Library Books • Airline Baggage • Maintenance data logging	Low cost of tags	Higher read rate than LF
860 MHz to 930MHz	Ultrahigh Frequency (UHF)	3m or 9ft.	• Pallet tracking • Carton Tracking • Electronic toll collection • Parking lot access	EPC standard built around this frequency	Does not work well around items of high water or metal content
2.4GHz	Microwave	1m or3 ft.	• Airline Baggage • Electronic toll collection	Most expensive	Fastest read rates

Table 2.1: RFID frequencies and applications [19]

antennas vary in size from around 10 to >30 cm, and tags are typically 10-18 cm long. These systems use radiative coupling, and are not limited by reader antenna size but by signal propagation issues. [25]

The table shows the main applications, benefits and drawbacks to work with each band.

A second key issue in selection of frequency bands is the allocation of frequencies by regulatory authorities. RFID systems are typically operated in unlicensed bands. The very common frequency range for UHF RFID readers and tags is the 900MHz band, which is called the ISM (Industrial, Scientific and Medical) band. At figure 2.4there is a simplified summary of RFID allocations in the 900MHz region throughout the world.

Obviously, the system devices will depend of the frequency chosen. As we can see at the

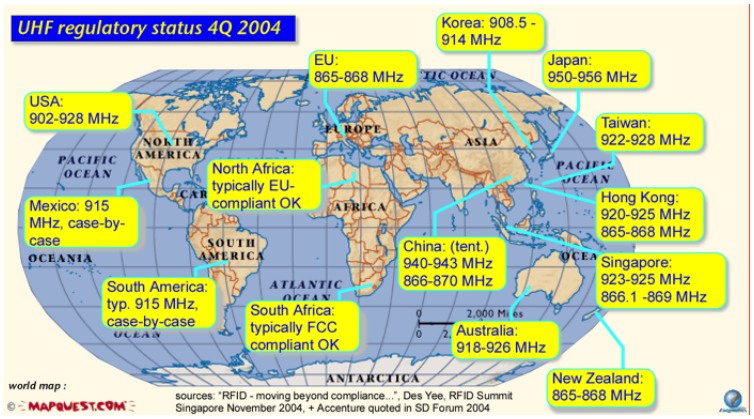


Figure 2.4: UHF regulatory status [3]

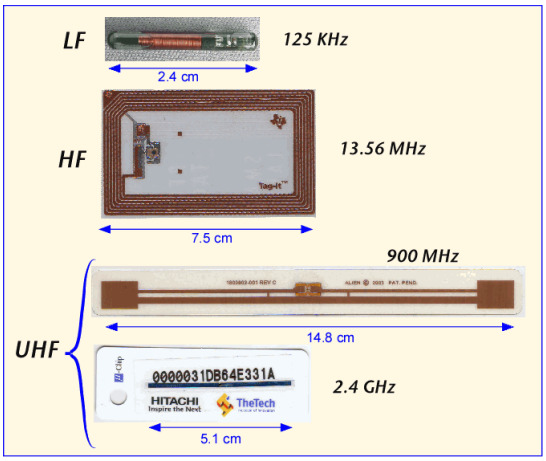


Figure 2.5: Different types of RFID tags depending on the frequency. [3]

figure 2.5, shape and size of the tags may change considerably depending on the frequency that is being used.

2.2 EPCglobal

EPCglobal Inc™ is the organisation entrusted by industry to establish and support the EPCglobal Network™ as the global standard for real-time. It was formed in October, 2003 as the successor of Auto-ID Center, the real developer of EPCglobal Network as an academic research project headquartered at the Massachusetts Institute of Technology (MIT) with labs at five leading research universities around the world.

The present obligation of EPCglobal is regulate and maintain active all the standards to manage the EPCglobal Network, what is getting bigger quickly. EPCglobal Network is

composed by five fundamental components:

- ***Electronic Product Code (EPC)***: is an identification scheme for universally identifying of physical objects via Radio Frequency Identification (RFID) tags and other means. The standardized EPC data consists of an EPC (or EPC Identifier) that uniquely identifies an individual object, as well as an optional Filter Value when judged to be necessary to enable effective and efficient reading of the EPC tags.

- ***The ID System:*** what is an EPC tag and EPC reader. The EPC tag is a RFID tag which contains EPC code inside of a microchip and the reader is a RFID reader getting the code from the tag through Radio Frequencies in UHF band and deliver the information to the EPC Middleware.

- ***EPC middleware***: manages real-time read events, provides alerts, and controls the basic read information for communication to EPC Information Services as well as a company's other existing information systems. EPCglobal is developing a software interface standard for services enabling data exchange between an EPC reader or network of readers and information systems.

- ***Discovery Services:*** A suite of services that enable users to find data related to a specific EPC and to request access to that data. Object Naming Service (ONS) is one component of Discovery Services.

- ***EPC Information Services (EPCIS):*** Enables users to exchange EPC-related data with trading partners through the EPCglobal Network.

2.2.1 EPCglobal Architecture Framework

The EPCglobal Architecture Framework is a collection of interrelated standards for hardware, software, and data interfaces ("EPCglobal Standards"), together with core services that are operated by EPCglobal and its delegates ("EPCglobal Core Services"), all in service of a common goal of enhancing the supply chain through the use of Electronic Product Codes (EPCs).

Into whole world of EPC Architecture Framework there are different levels of application with different standards applied and all of them are useful and necessary to share information containing EPCs. As it is possible to recognize at figure 2.6each level allows different exchanges of information, but always benefiting end users.[7]

- **EPC Physical Object Exchange:** End Users exchange physical objects that are identified with Electronic Product Codes (EPCs). The EPCglobal Architecture Framework defines EPC physical object exchange standards, designed to ensure that

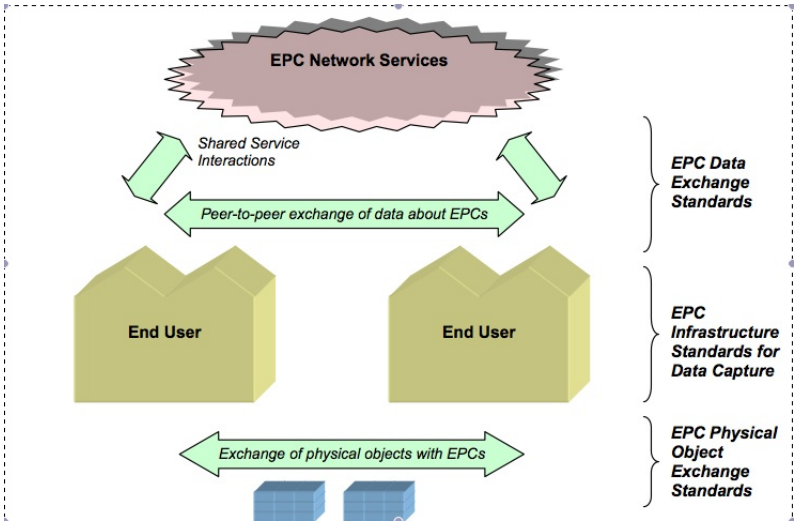


Figure 2.6: EPCglobal Architecture Framework overview. *[4]*

when one end user delivers a physical object to another end user, the latter will be able to determine the EPC of the physical object and interpret it properly.

- **EPC Data Exchange:** End Users benefit from the EPCglobal Architecture Framework by exchanging data with each other, increasing the visibility they have with respect to the movement of physical objects outside their four walls.
- **EPC Infrastructure for Data Capture**: The EPCglobal Architecture Framework defines interface standards for the major infrastructure components required to gather and record EPC data, thus allowing end users to build their internal systems using interoperable components.

Security is an important part of the EPCglobal Network. The EPCglobal Network uses industry recognizing best practices to protect its data. In addition, the EPCglobal Network functions like an extranet with a federated data model. Data lives both behind a Subscriber company's firewall and it is only referenced by the data registry system at EPCglobal.

2.2.2 EPCglobal Standards

As previously was indicated, the EPCglobal Architecture Framework and the global functionality of EPC Network is based on defined interrelated standards group that is doing possible the big exchange of information. The main goals of this standards are[7]:

- To facilitate the exchange of information and physical objects between trading partners. They must have prior agreement as to the structure and meaning of data to be

exchanged, and the mechanisms by which exchange will be carried out. EPCglobal standards are developed for global use.

- To foster the existence of a competitive marketplace for system components. EPCglobal standards define interfaces between system components that facilitate interoperability from components produced by different vendors (or in house).

- To encourage innovation. EPCglobal standards define interfaces, not implementations, what can be done by the way of implementers.

Inside of EPCglobal Architecture Framework, the standards are organized as figure 2.7shows. It is possible also to see at the figure, the three levels organization of the architecture, what makes easier to understand which standards apply in each data exchange phase.

All interfaces between architectural components are specified in open standards, developed by the EPCglobal Community through the EPCglobal Standards Development Process or an equivalent process within another standards organization to approve it due to ensure that all user requirements are heard and validated.

The most important standards for us, because they are involved of supply chain management and our topic of study, will be analyzed and defined in detail on the next sections.

2.2.3 EPC (Electronic Product Code)

The Electronic Product Code is an unique serial number designed to identify unequivocally one item. Is the proposal from EPCglobal for the future substitution of barcodes and it is considered as the evolution of Universal Product Code (UPC) in USA or the equivalent in Europe, the European Article Number (EAN).

Is the data of the EPC Global Network, and therefore the most important part of it. It assigns exclusively a number every physical object, track, load, location or other items involved on the chain. It is attached on a RFID tag, and through the RFID technology is a visible data during all the track process. The standard that control and unify the format of the code and equivalency between the different existing formats is **EPC Tag data Standard (TDS)**.

The structure of EPC has different versions but the most used schema is the 96 bits encoding. The information encrypted by these bits is defined at figure 2.8. [26]

- Header Code: The first component (4 bits) of the EPC that tells to the reader/interrogator how to parse the bits (decode the rest) of the EPC number.

- EPC Manager Number: The second component of the EPC, what describes which company or organization has authority over a group of products or things in the

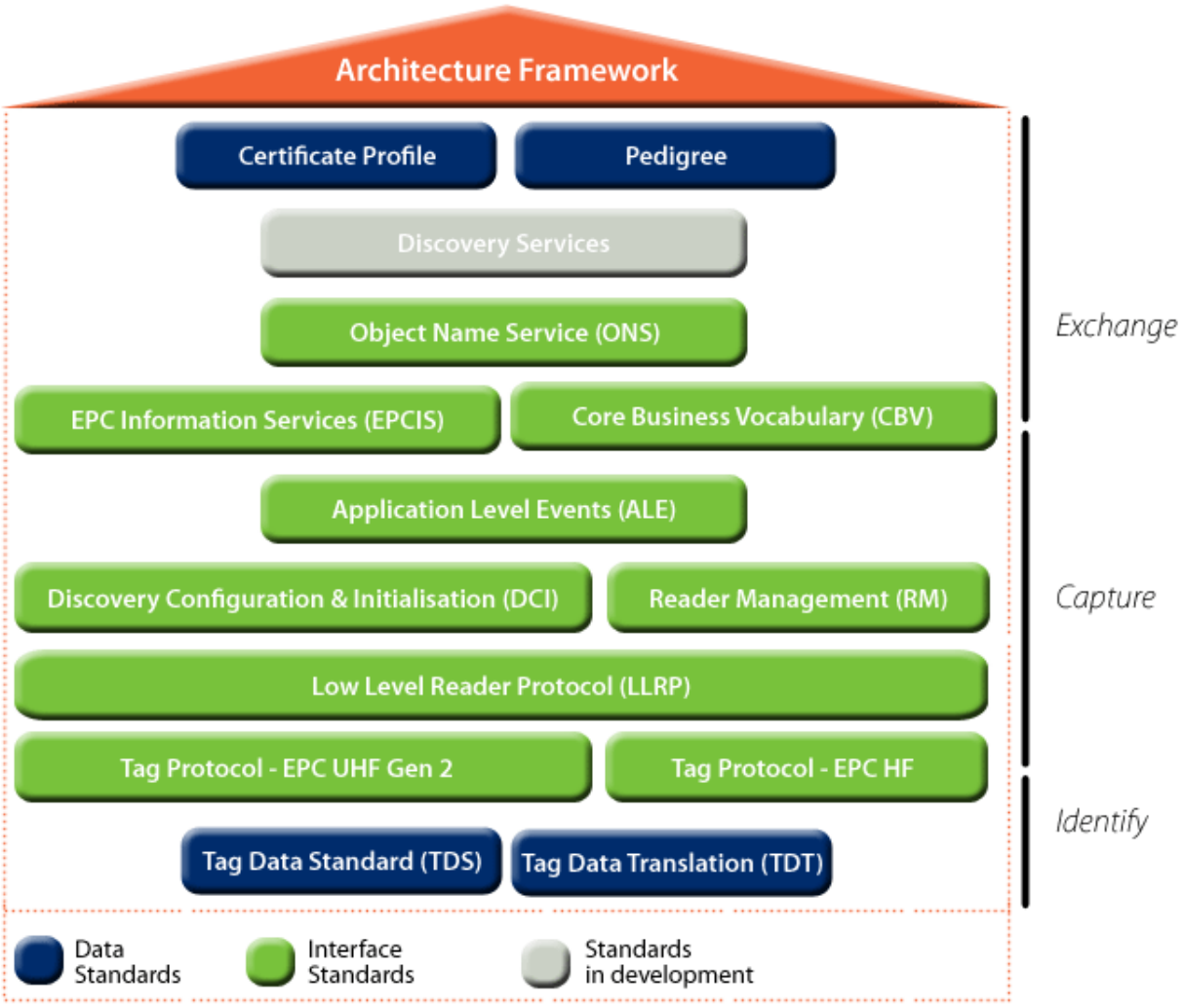


Figure 2.7: The EPCglobal Architecture Framework Standards [5]

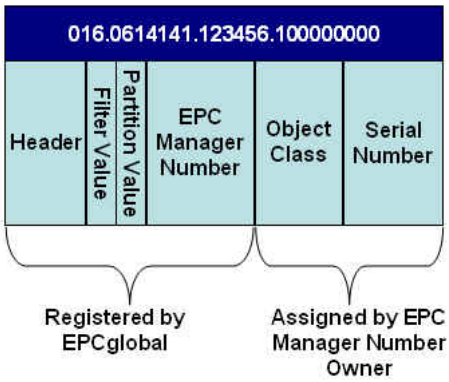


Figure 2.8: The 96 bits schema of EPC [6]

supply chain (much like the Company Prefix does in the GS1 system). Only End-Users have assigned EPC Manager Numbers and once them have been assigned are registered within the Object Naming Service (ONS). This will ensure that the product data is accessible via the EPCglobal Network. (28 bits, 268 millions of combinations)

- Object Class: The third component of the EPC that describes a category of things in the supply chain (much like an Item Reference or a SKU does in the GS1 System). (24 bits, 16 millions of combinations)
- Serial Number: The fourth component of the EPC that is critical to the reading and numeration of tags. How an individual company chooses to assign serial numbers (i.e. individual instances of the Object Class that precedes it) is up to them. (36 bits, 68 billions of combinations)

EPC URI Is the primary representation of an Electronic Product Code and is an Internet Uniform Resource Identifier (URI) called the Pure Identity EPC URI. It is the preferred way within an information system to denote a specific physical object and makes easier the exchange between software systems.

The construction of the EPC URI guarantees uniqueness across all categories of objects, provided that the URI is used in its entirety.

Example with sgtin scheme: *urn:epc:id:sgtin:0614141.112345.400*

2.2.4 EPCIS (Electronic Product Code Information Services)

EPC Information Services (EPCIS) is an EPCglobal standard designed to obtain a common data view for all the EPCglobal Network participants. The EPCIS specification is designed to be layered, extensible, and modular on the purpose of be easily used by any application in

any industry for the end users. Include therefore capabilities to improve efficiency, security and visibility in the global supply chain. It defines a data language understandable for all the network to raise the "what", "when", "where" and "why" of the data.

As it is very clear at figure 2.9 EPCIS sits at the highest level of the EPCglobal Architecture Framework and complements EPCglobal lower level (reader, tag and middleware) standards in the way between the EPCglobal subscriber or end user and the common EPC data base, through one capture interface and one query interface due to add and consult the global information.

Next sections will describe each of these modules in order to obtain a clear and general vision of how all the structure connects and how important is each module and each standard.

2.3 Interaction between RFID system and EPCglobal.

As we have seen at figure 2.9complete functionality of EPCglobal Network starts with a well functionality of RFID system. The EPCglobal Network will make organisations more effective by enabling true visibility of information about items in the supply chain. Having more accurate, immediate information about the location of items, the history of items, and the number of items in the supply chain will enable organisations to be more responsive to customers and consumer needs through more efficient, customer-driven operations. EPCglobal Architecture Framework contain standards that control and unify the acquisition of data trough RFID technology due to have it understood and compatible for next EPCIS Interfaces.

2.3.1 Communication Reader-Tag

An Interrogator transmits information to a Tag by modulating an RF signal in the 860 MHz – 960 MHz frequency range (Ultra High Frequency). The Tag receives both information and operating energy from this RF signal. Tags are passive, meaning that they receive all of their operating energy from the Interrogator's RF waveform.

An Interrogator receives information from a Tag by transmitting a continuous-wave (CW) RF signal to the Tag; the Tag responds by modulating the reflection coefficient of its antenna, thereby backscattering an information signal to the Interrogator. The system is ITF (Interrogator-talks-first), meaning that a Tag modulates its antenna reflection coefficient with an in- formation signal only after being directed to do so by an Interrogator. [27]

Interrogators and Tags are not required to talk simultaneously; rather, communications are half-duplex, meaning that Interrogators talk and Tags listen, or vice versa.

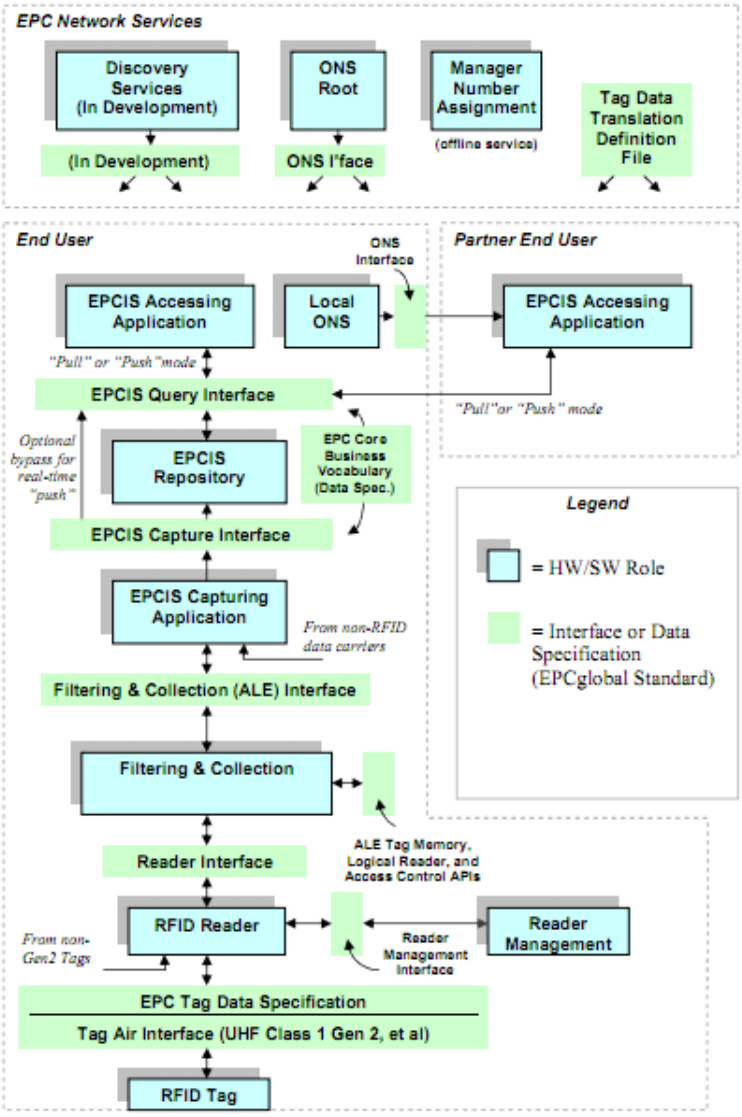


Figure 2.9: EPCglobal Architecture Framework [7]

This communication between the Interrogator or Reader and the Tag or Label is defined by the **UHF Class-1 Generation-2 air interface protocol EPCglobal standard**, commonly known as the "Gen 2" standard. As a summary, it covers following functions:

- physical interactions between reader and tag (radio waves).

- procedures and operations between RFID reader and tag to be able understand each other.

- collision systems used by the identification of the tag for the multi-Tag environments.

As we have seen at 2.2.3 the **Tag Data Specification (TDS)** is what describes the possibilities of EPC formats to get into the system.

Low Level Reader Protocol (LLRP) describes an interface between RFID Readers and Clients that provide means to command an RFID Reader to inventory tags (read the EPC codes carried on tags), read tags (read other data on the tags apart from the EPC code), write tags, and execute other protocol-dependent access commands (such as 'kill' and 'lock' from EPCglobal Class 1 Generation 2). However, the standard define how to retrieval Reader device capabilities and facilitate the addition of support for new air protocols. It has several advantages over the last Reader Protocol (not used nowadays) as that enables the communication between two LLRP endpoints using a binary protocol, more efficient and fast. In addition, LLRP is designed to be extensible in terms of supporting multiple air protocols,and that enables management and monitoring. [28] The responsibilities of the elements and interfaces below the Filtering & Collection role (see figure 2.9) are: tag data processing (Data path), Reader device management (Management path) and Reader control and coordination (Control path) which is taken by LLRP protocol.

2.3.2 ALE Server (Application Level Events Interface)

The ALE Server is an independent piece of software that speaks to RFID readers using their network wire protocols (as the middleware RFID system). Using ALE, an application makes a high-level description of what data it wants read from or written to tags, over what period of time, and with what filtering to select particular tags by different defined APIs (Application Programming Interface).

The role of the ALE interface within the EPCglobal Network Architecture is to provide independence between the infrastructure components that acquire the raw EPC data, the architectural component(s) that filter & count that data, and the applications that use the data. This allows changes in one without requiring changes in the other, offering significant benefits to both the technology provider and the end-user. The ALE interface described in the present specification achieves [29]this independence through five means:

- It provides a means for clients to specify, in a high-level, declarative way, what data

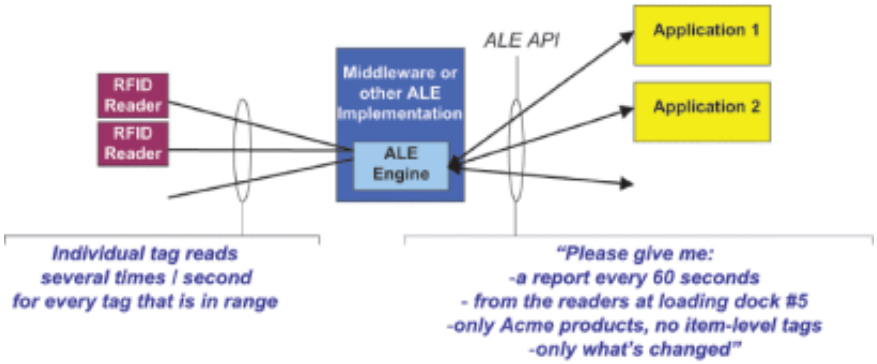


Figure 2.10: ALE role within EPCglobal Architecture Framework. [8]

they are interested in or what operations they want performed, without dictating an implementation.

- It provides a standardized format for reporting accumulated, filtered data and results from carrying out operations that is largely independent of where the data originated or how it was processed.
- It abstracts the channels through which data carriers are accessed into a higher-level notion of "logical reader," often synonymous with "location," hiding from clients the details of exactly what physical devices were used to interact with data relevant to a particular logical location.
- It abstracts the addressing of information stored on Tags and other data carriers into a higher-level notion of named, typed "fields," hiding from clients the details of how a particular data element is encoded into a bit-level representation and stored at a particular address within a data carrier's memory.
- It provides a security mechanism so that administrators may choose which operations a given application may perform, as a policy that is decoupled from application logic itself.

2.3.3 TDT (Tag Data Translation)

Tag Data Translation is designed to help to future-proof the EPC Network and in particular to reduce the pain / disruption in supporting additional EPC identifier schemes that may be introduced in the future, as the EPC Network is adopted by additional industry sectors and new applications and it is possible by providing a machine-readable framework (see figure 2.11) for validation and translation of EPC identifiers. The EPC Tag Data Standard (TDS) also describes in terms of human-readable encoding and decoding rules for each

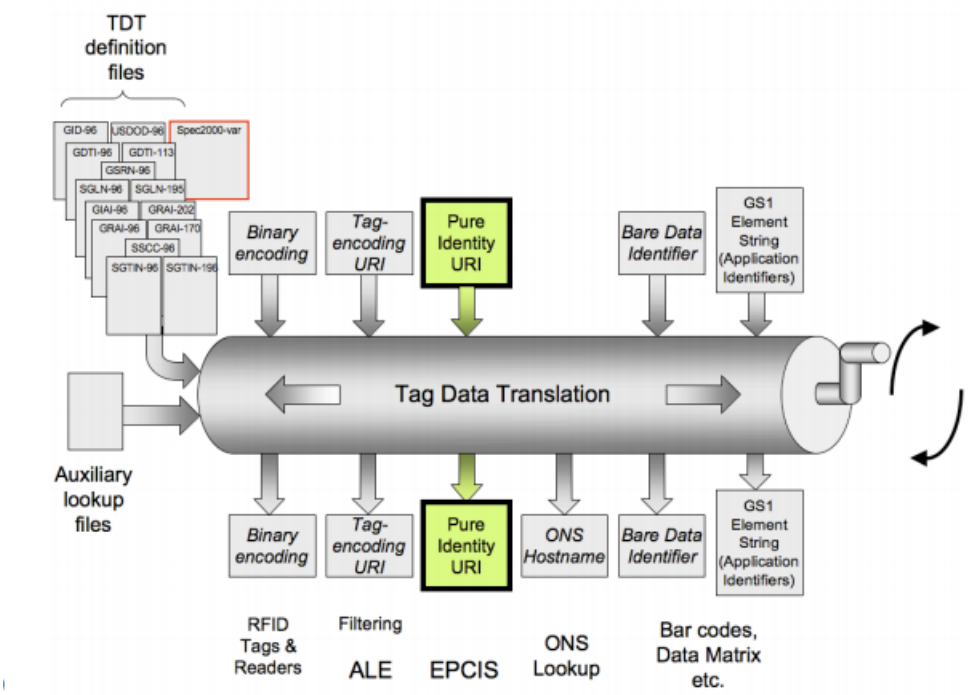


Figure 2.11: Tag Data Translation machine-readable framework [9]

coding scheme, how to translate between three representations of the electronic product code (EPC), namely the binary format and two formats of uniform resource identifiers (URI) (see 2.2.3), one for tag-encoding and another for pure identity.[30]

That is: The Tag Data Translation process translates one representation of EPC into another representation, within a particular coding scheme. For example, it could translate from the binary format for a GTIN on a 96-bit tag to a pure-identity URI representation of the same identifier, although it could not translate a SSCC into a SGTIN or vice versa.

The majority of the EPC Network components require the ability to consistently translate between binary data on tags and URI formats for information systems. However, it should be noted that levels of the stack above the Low Level Reader Protocol interface should normally be using the URI representation rather than the binary representation. This also enforces a need for a standard translation mechanism across the entire EPC network so that the translation process and resulting data is consistent and valid.

Tag Data Translation currently standard version must be directly linked and fully compatible with TDS currently standard version.

2.3.4 ONS (Object Name Service)

The Object Name Service (ONS) is an automated networking service similar to the Domain Name Service (DNS) that points computers to sites on the World Wide Web. When the RFID interrogator reads an RFID tag, the Electronic Product Code is passed to middleware, which, in turn, goes to an ONS local network or directly to Internet to find where product information is stored. ONS points middleware to server where a file about that product is stored. The middleware retrieves the file (after proper authentication), and the information about the product in the file can be forwarded to a company's inventory or supply chain applications.[31]

Note that the term DNS is used when the discussion is generally applicable to the DNS system and ONS is used when the discussion is specifically about querying the DNS for an EPC. ONS does not contain actual data about the EPC. It only contains the network address of services that contain the actual data. ONS is also authoritative in that the entity that has change control over the information about the EPC is the same entity that assigned the EPC to the item to begin with.

2.4 EPCIS Standard

EPCIS is the most complicate and complex standard because has as a goal to enable disparate applications to leverage Electronic Product Code (EPC) data via EPC-related data sharing, both within and across enterprises. Ultimately, this sharing is aimed at enabling participants in the EPCglobal Network to gain a shared view of the disposition of EPC objects within a whole business context.

The EPC Information Service approach will define a standard interface to enable this EPC-related data to be captured and queried using a defined set of service operations and associated data model.

As depicted in the figure 2.9, EPCIS sits at the highest level of the EPCglobal Architecture Framework, both above the level of raw EPC observations as well as above the level of filtered, consolidated observations. Differs from these lower layers elements in three key respects:

1. EPCIS deals explicitly with historical data (in addition to current data).

2. EPCIS often deals not just with raw EPC observations, but also in contexts that imbue those observations with meaning relative to the physical world and to specific steps in operational or analytical business processes. The lower layers of the stack are more purely observational in nature.

3. EPCIS operates within enterprise IT environments at a level that is much more diverse and multi-faceted than the lower level. This is due to the desire to share EPCIS data between enterprises which are likely to have different solutions deployed to perform similar tasks and due to EPCIS being at the highest level of the EPCglobal Network Architecture, and hence the natural point of entry into other enterprise systems, which vary widely from one enterprise to the next.

EPC-related data standards, all combined with appropriate security mechanisms that satisfy the needs of user companies. In many or most cases, this will involve the use of one or more persistent databases of EPC-related data, though elements of the Services approach could be used for direct application-to-application sharing without persistent databases.[10]

Technically, the specification is designed to be layered, extensible, and modular. In order to have clearly the interactions between these internal layers and how it implements the final behavior, see the figure 2.12, with the specification framework. The **Service Layer** defines service interfaces through which EPCIS clients interact, the **Data Definition Layer** specifies what data is exchanged through EPCIS, what its abstract structure is, and what it means, and finally, the **Abstract Data Model Layer** specifies the generic structure of EPCIS data and the general requirements for creating data definitions within the Data Definition Layer.

2.4.1 EPCIS Interfaces

The interfaces through which EPCIS data is delivered to enterprise-level roles, includes EPCIS Repositories, EPCIS Accessing Applications, and data exchange with partners. Events at these interfaces say, for example, "At location X, at time T, the following contained objects (cases) were verified as being aggregated to the following containing object (pallet)."[10]

In the next sections is described how EPCIS Interfaces works obtaining the EPC data from the lower layers and enables the interaction between end user and this data.

2.4.1.1 EPCIS Capturing Application and Capture Interface

Following the global Architecture Framework on figure 2.9, EPCIS Capturing Application supervises the operation of the lower-level architectural elements, and provides business context by coordinating with other sources of information. It understands the business process step or steps during which EPCIS data capture takes place and configures and routes events from the Filtering & Collection interface directly to an EPCIS-enabled Repository.

The EPCIS Capture Interface defines the delivery of EPCIS events from EPCIS Capturing Applications to other roles that consume the data in real time, including EPCIS

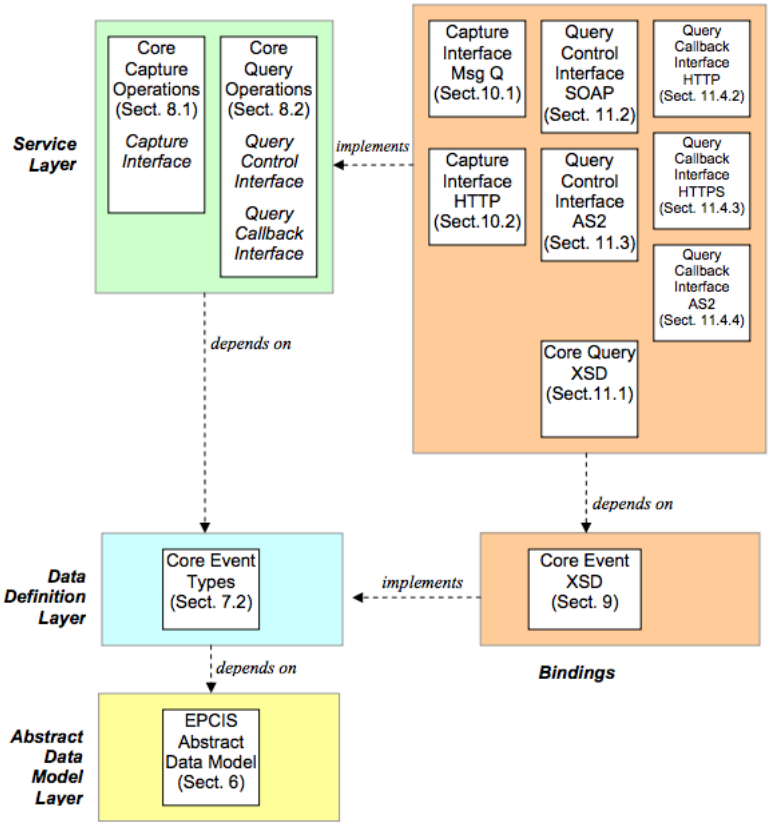


Figure 2.12: EPCIS Specification Framework [10]

Note: Bindings specify concrete realizations of the Data Definition Layer and the Service Layer.

Repositories, and real-time "push" to EPCIS Accessing Applications and trading partners.

2.4.1.2 EPCIS Accessing Application and Query Interface

The EPCIS Accessing Application Responsible for carrying out overall enterprise business processes, such as warehouse management, shipping and receiving, historical throughput analysis, and so forth, aided by EPC-related data.

The EPCIS Query Control Interface defines a means for EPCIS Accessing Applications and trading partners to obtain EPCIS data subsequent to capture, typically by interacting with an EPCIS Repository. The EPCIS Query Control Interface provides two modes of interaction. In "on-demand" or "synchronous" mode, a client makes a request through the **EPCIS Query Control Interface** and receives a response immediately. In "standing request" or "asynchronous" mode, a client establishes a subscription for a periodic query. Each time the periodic query is executed, the results are delivered asynchronously (or "pushed") to a recipient via the **EPCIS Query Callback Interface**. The EPCIS Query Callback Interface may also be used to deliver information immediately upon capture; this corresponds to the "optional bypass for real-time push" arrow in the figure 2.9.

2.4.2 EPCIS Repository

The EPCIS-enabled Repository Records EPCIS-level events generated by one or more EPCIS Capturing Applications, and makes them available for later query by EPCIS Accessing Applications. It is as a big database interacting simultaneously with Capture Interface and Query Interfaces, and must be extremely demanding with the data structure. In that purpose EPCglobal has an standard schema to implement the EPCIS events.

While the EPCIS Capture Interface and EPCIS Query Interfaces provide no means for an application to explicitly request the deletion of an event, EPCIS Repositories MAY implement data retention policies that cause old EPCIS events to become inaccessible after some period of time.

2.4.3 EPCIS Events

The Core Event Types data definition module specifies the Event Types that represent EPCIS data capture events. These events are typically generated by an EPCIS Capturing Application and provided to EPCIS infrastructure using the data capture operations. These events are also returned in response to query operations that retrieve events according to query criteria.

This module defines five event types, one very generic event and four subclasses that can represent events arising from supply chain activity across a wide variety of industries.

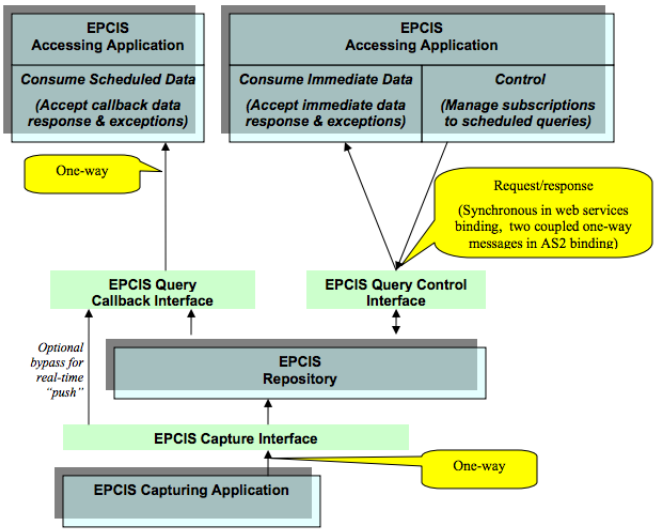


Figure 2.13: Service Layer[10]

EPCISEvent It is a common base type for all EPCIS events. Contain the basic information with the fields *eventTime* (The date and time at which the EPCIS Capturing Applications asserts the event occurred), *recordTime* (The date and time at which this event was recorded by an EPCIS Repository.) and *eventTimeZoneOffset* (The time zone offset in effect at the time and place the event occurred, expressed as an offset from UTC).

ObjectEvent An ObjectEvent captures information about an event pertaining to one or more physical objects identified by EPCs. it can be used for any event a Capturing Application wants to assert about EPCs, including for example capturing the fact that an expected observation failed to occur.

It includes the *action* field what describes the event's relationship to the lifecycle of the EPC(s) named in the event. Possible options are:

- ADD: the EPCs in *epcList* (another ObjectEvent field) were commissioned (issued for the first time) and may appear in subsequent events.
- OBSERVE: The event represents a simple observation of the EPC(s) named in the event, not their commissioning or decommissioning.
- DELETE: the EPCs in *epcList* were decommissioned (retired from future use) and should not be observed again.

AggregationEvent The event type AggregationEvent describes events that apply to objects that have been physically aggregated to one another. In such an event, there is

a set of "contained" objects that have been aggregated within a "containing" entity that's meant to identify the physical aggregation itself.

Aggregation means an association where there is a strong physical relationship between the containing and the contained objects such that they will all occupy the same location at the same time, until such time as they are disaggregated. An example of an aggregation is where cases are loaded onto a pallet and carried as a unit.

It includes two mainly important fields:

- *parentID*: is the URI of the parent of the association (often a "containing" entity). It is mandatory when the action is ADD or DELETE and the URI may be an EPC or another identifier drawn from a suitable private vocabulary.
- *childEPCs*: An unordered list of the EPCs of the contained objects.

The *action* field of an AggregationEvent describes the event's relationship to the lifecycle of the aggregation:

- ADD: The EPCs named in the child list have been aggregated to the parent during this event.
- OBSERVE: The event represents neither adding nor removing children from the aggregation, only observation, and may be incomplete, what means that there may be children that are part of the aggregation but not observed during this event and therefore not included in the childEPCs field of this AggregationEvent.
- DELETE: The EPCs named in the child list have been disaggregated from the parent during this event. If the field is empty all the child may be disaggregated.

QuantityEvent A QuantityEvent captures an event that takes place with respect to a specified quantity of an object class. This Event Type may be used, for example, to report inventory levels of a product. Therefore, the most important field within the event is *quantity* field what describes the quantity of object within the class described by the event.

TansactionEvent The event type TransactionEvent describes the association or disassociation of physical objects to one or more business transactions. While other event types have an optional bizTransactionList field that may be used to provide context for an event, the TransactionEvent is used to declare in an unequivocal way that certain EPCs have been associated or disassociated with one or more business transactions as part of the event.

As AggregationEvent it contains parentID and childEPCs fields and with the same purpose. At the same time action field for TransactionEvent describes the event's relationship to the lifecycle of the transaction:

- ADD: The EPCs named in the event have been associated to the business transaction(s) during this event. This includes situations where the transaction(s) is created for the first time, as well as when new EPCs are added to an existing transaction(s).
- OBSERVE: The EPCs named in the event have been confirmed as continuing to be associated to the business transaction(s) during this event.
- DELETE: The EPCs named in the event have been disassociated from the business transaction(s) during this event. This includes situations where a subset of EPCs are disassociated from the business transaction(s), as well as when the entire business transaction(s) has ended. As a convenience, the list of EPCs may be omitted from the TransactionEvent, which means that all EPCs have been disassociated. Note that if action is DELETE, *epcList* is empty and *parentID* is omitted, all EPCs have been disassociated from the business transactions enumerates in *bizTransactionList*.

  Observation: In the case where AggrgationEvent action is ADD and a non-empty bizTransactionList is specified, the semantic effect is equivalent to having an AggregationEvent with no bizTransactionList together with a TransactionEvent having the bizTransactionList and all same field values as the AggregationEvent. Note, however, that a AggregationEvent with a non-empty bizTransactionList does not cause a TransactionEvent to be returned from a query.

The EPCIS Events define with their fields:

- WHAT (product): with epcList, childEPC or parentID.
- WHERE (location): with readPoint (location where an event took place) or BusinessLocationID (where the item is immediately after the event occurs).
- WHEN (time): eventTime (states when the event took place) or recordTime (indicates when the event was received through the EPCIS Capture Interface). These two fields can be different due to processing delay and clock synchronization.
- WHY (business step and status): BusinessStepID (what business operation was taking place at the time of the event, i. e. receiving, shipping,...) or DispositionID (status of the product immediately after the event occurs, i. e. in progress, destroyed,...).

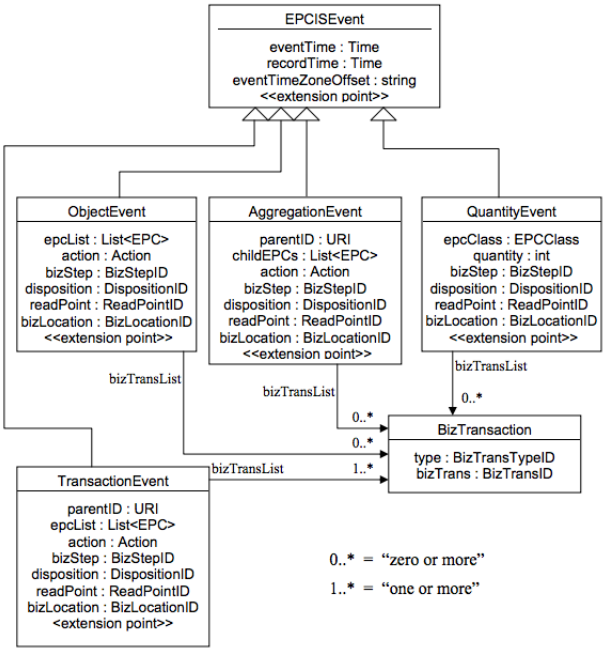


Figure 2.14: EPCIS Events Types[10]

Event Data and Master Data EPCIS deals in two kinds of data: event data and master data. Event data arises in the course of carrying out business processes, and is captured through the EPCIS Capture Interface and made available for query through the EPCIS Query Interfaces. Master data is additional data that provides the necessary context for interpreting the event data. It is available for query through the EPCIS Query Control Interface, but the means by which master data enters the system is not specified in the current specification.[10] The relationship between both kinds of data is shown at figure 2.15.

The modeling of real-world business information as event data and master data is the responsibility of the Data Definition Layer inside of the EPCIS framework and the structure of the data the Abstract Data Model Layer.

As a practical example of Event Data: "At 1:23pm on 15 March 2004, EPC X was observed at Location L.", and for Master Data: "Location L refers to the distribution center located at Elm Street, Anytown, US." Is easy to note that event data grows in quantity as more business is transacted and master data generally does not.

At this point we may note that current specification[10] of EPCIS standard is still missing to clarify several important points, among them:

- the means by which master data enters the system through capturing application.

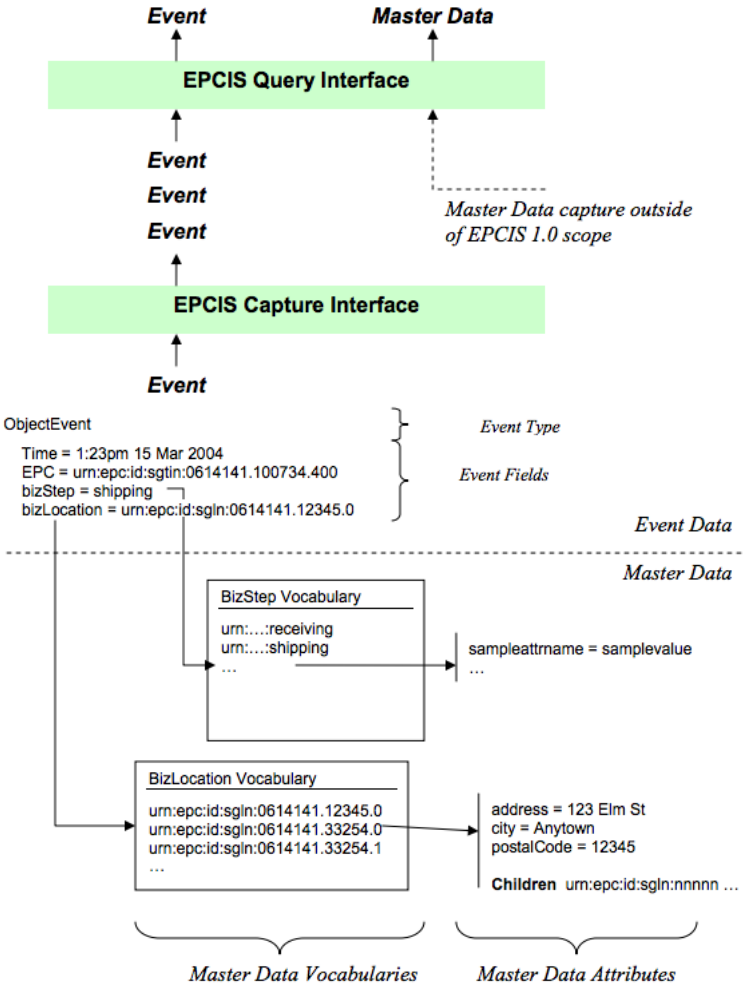


Figure 2.15: Relationship between Event data and Master data.[10]
NOTE: Event Data= a set of events.

- the means by which EPCIS Capture Interface and EPCIS Query Interfaces provide an application to explicitly request the deletion of an event.
- define any standard vocabulary elements and master data attribute names for the exchange of information instead of only the definition. This point could lead to problems understanding some event fields with name definition (i.e. bizstep) between different partners (calling differently the step).

Chapter 3

Fosstrak tool

This chapter is a kind of guide line for basic users of RFID technology that wants to introduce into EPCglobal environment or students who start working with this topic and are begging to work with fosstrak. With this purpose, after a short introduction of the software and platform and the possibilities that it provides, each module implemented by fosstrak will be explained first as a theoretical way and after with a practical point of view, to make easier the execution by the reader/user.

Note that the following sections further detail the development of each Fosstrak projects by the structure of the last version of Fosstrak (June 2009).

3.1 What is Fosstrak and how it interacts with EPCglobal?

Fosstrak (Free and Open Source Software for Track and Trace) is an open source RFID prototyping platform that implements the EPC Network specifications. It is intended to foster the rapid prototyping of RFID applications. It was initiated by Christian Floerkemeier, Matthias Lampe and Christof Roduner of the Distributed Systems Group at ETH Zurich led by Friedemann Mattern and the Auto-ID Lab at ETH Zurich/University of St. Gallen led by Elgar Fleisch. [11]

The Fosstrak project has an active developer community with significant RFID middleware experience and insights into the RFID/EPC standardisation process. There is a strong focus on implementing the specifications developed by EPCglobal and providing useful tools for these implementations.

It must be said that Fosstrak is not a reference implementation of the EPCglobal Network, because the current Architecture Framework Standards are going beyong of what is implemented by fosstrak modules, but is an excellent open-source software implementation to work with EPCs and RFID technology and EPCIS Repository has been EPCglobal certified.

Standard	Implemented by Fosstrak	Fosstrak project
Tag Data Standard	No	-
Tag Data Translation (TDT)	Yes (v1.0)	Fosstrak Tag Data Translation (TDT)
Tag Protocol EPC UHF Gen 2	No	-
Tag Protocol EPC HF	No	-
Low Level Reader Protocol (LLRP)	Yes	Fosstrak LLRP Commander
Discovery Configuration & Initialization (DCI)	No	-
Reader Management (RM)	Yes	Fosstrak LLRP Commander
Application Level Events (ALE)	Yes (v1.1)	Fosstrak ALE Middleware
EPC Information Services (EPCIS)	Yes (v1.0.1)	Fosstrak EPCIS
Core Business Vocabulary (CBV)	No	-
Object Name Service (ONS)	No	-
Discovery Services	No (standard in development)	-
Certificate Profile	No	-
Pedigree	No	-

Table 3.1: The current implementation of EPCglobal Standards by Fosstrak.[11]

The fosstrak platform consists of four separate modules also considered projects:

- Fosstrak EPCIS Repository.
- Fosstrak LLRP Commander.
- Fosstrak ALE Middleware (with LLRP Commander Support).
- Fosstrak Tag Data Translation (TDT) Library.

All Fosstrak components are licensed under the GNU Lesser General Public License v2.1 except Fosstrak LLRP Commander that is licensed under the GNU General Public License v3.

The EPCglobal Standards implemented currently by Fosstrak modules are showed in the table 3.1.

Per each module, Fosstrak has a section devoted to developer community who likes to interact with source and contribute to the Fosstrak project. Therefore,it gives to the developers some accurate guidelines to understand perfectly the background of each project and be able to implement part of the code. For the purpose of build whole project, Fosstrak uses maven environment.

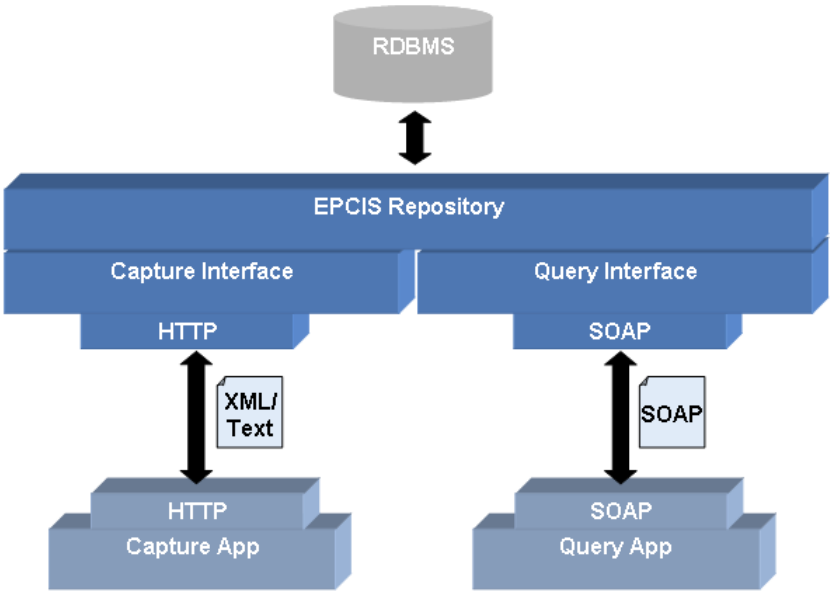


Figure 3.1: Fosstrak EPCIS Project overview. [11]

3.2 EPCIS Repository

3.2.1 What is EPCIS Repository?

Fosstrak EPCIS is a complete implementation of the EPCIS standard specification (Version 1.0.1 of September 21, 2007). It successfully passed conformance testing and is EPCglobal-certified.

The Fosstrak EPCIS Project comprises three separate modules:

- an EPCIS Repository implementation.
- an interactive EPCIS Capture Application.
- an interactive EPCIS Query Application.

At figure 3.1is shown an overview of Fosstrak EPCIS project implementation. The main idea of the functionality is that there is a Capture Interface implemented by HTTP that allows to generate and capture an EPCIS event and sends it to the Repository, what keep it into the database. Ones the event is registered inside the database the user has the data at ones disposal by the Query Interface, through SOAP environment.

EPCIS Repository Implementation It allows one to run your own instance of EPCIS repository and store the EPC-related data captured by Capture Application and to retrieve data from the repository by Query Application. It is implemented throught a database

that must respect a structure of data events. This structure of data is provided by Fosstrak as well and describes the certificated way to store data and masterdata to be shared and understandble for each user who wants to work with EPCIS platform.

The database backend officially supported by Fosstrak is MySQL, and we, working following fosstrak lines, will use this one as well, but is possible to implement the EPCIS repository using other databases as PostgreSQL, for example.

EPCIS Capture Application For the capture procedure, fosstrak provides a java implementation called EPCIS Capture Interface Client, through what is possible to define each even type specifying wished fields for create a real EPCIS event. The application sents the event to the specified Capture interface url, and translates it to a http format due to be understood for the interface and to be integrated inside the RFID repository database.

Note that, although there are efforts put for capturing EPCISMasterDataDocument in CaptureClient from version 0.5, it is still missing subject for fosstrak implementation because the means by which master data enters the system is not specified in the EPCIS 1.0 specification. Nevertheless, the Fosstrak EPCIS implementation does support adding, modifying, and deleting master data via the capture interface.

EPCIS Query Application For the query procedure, fosstrak provides a java implementation called EPCIS Query Interface Client, through what is possible to select several conditions with wished field values and define a query. The query, by SOAP format is sent to the specified Query interface url and then there is a procedure to check the repository and obtain EPCIS event data that fit whit field values described.

3.2.2 How to work with Fosstrak EPCIS Repository?

EPCIS Repository fosstrak project allows beginner users to run their own EPCIS repository and interact with it on an easy way. Following the procedure that is described at Fosstrak webside, in User Guide repository, any user can download and install correctly all the interface needed for the correct and completed implementation.

EPCIS Repository module adds a section for more expert users who are interested in contributing to the project and a section that describes in extensive and technical way which is the whole architecture of the module for people who are interested on get more involved with the global fosstrak functionality. For this work, as a first instance of a bigger work, we will cover only the user part.

The first step is set up the global interface to run the repository. This part includes download Apache Tomcat servlet container and installs MySQL server. For more detail

about this part, see the section 5.2“Set up the interficie” in the next chapter.

After everything is installed in your computer, EPCIS repository feature can be download and integrated on it to be accessible following fosstrak webside instructions.

The next step is to download EPCIS Capture Client application and EPCIS Query Client application from the *download* section into fosstrak webside and install them. To connect with the database integrated in Apache Tomcat is necessary to place the WAR file contained in the archive in Tomcat's webapps directory. After restarting Tomcat, the WAR file will be exploded. This must be done for each module, either EPCIS repository.

After this, one can run the executable JAR file contained in the downloaded archives. This will launch the Fosstrak EPCIS Capture Client and the Fosstrak EPCIS Query Client.

For the Fosstrak EPCIS Capture Client you will find the window showed in figure 3.2. The showed example is filled by one of the possibilities that fosstrak offer to fill the window with an example of practical implementation, in this case is *“Three items are aggregated onto a barcode-labeled pallet”*. It is a good example in order to understand the meaning of EPCIS Events fields explained at section 2.4.3, in this case, for the AggregationEvent. Observe that:

- **Capture interface URL**: it needs to specify the URL of the EPCIS repository that you want to connect to. You can use fosstrak public repository (http://demo.fosstrak.org/epcis/capture or any other EPCIS repository. The option showed is for connect to the repository that we create just before, through Apache Tomcat servlet, in this case by port 8081.

- **Authentication Mode**: is used if you want to secure the access of your own repository. Both capture and query clients support HTTP Basic (username/password) client certificate-based authentication and are thus able to provide the client's identity to a requesting server. Also you can configure Tomcat to require client authentication before access to the EPCIS repository is granted (see fosstrak webside: UserGuide, *Securing your Repository*). It is possible to choose between basic authentication or X.509 Certificate.

After fill up all the fields, one only need to submit the EPCIS capture request by clicking "Generate event".

If one sets the box “Show debug window” the window showed at figure 3.3appears and he can check the XML coding for the EPCIS Event that one will generate. If then, he prefers to work by browser implementation, he can acces to the Tomcat capture space (http://localhost:8081/epcis-repository-x.x.x/capture) and fill up the available window with the Event XML code and submit the event. (see figure 3.4)

After all this process, the EPCIS Event is already stored into the database and someone can query it by EPCIS Query Client Interface. At figure 3.6 is showed the window interface

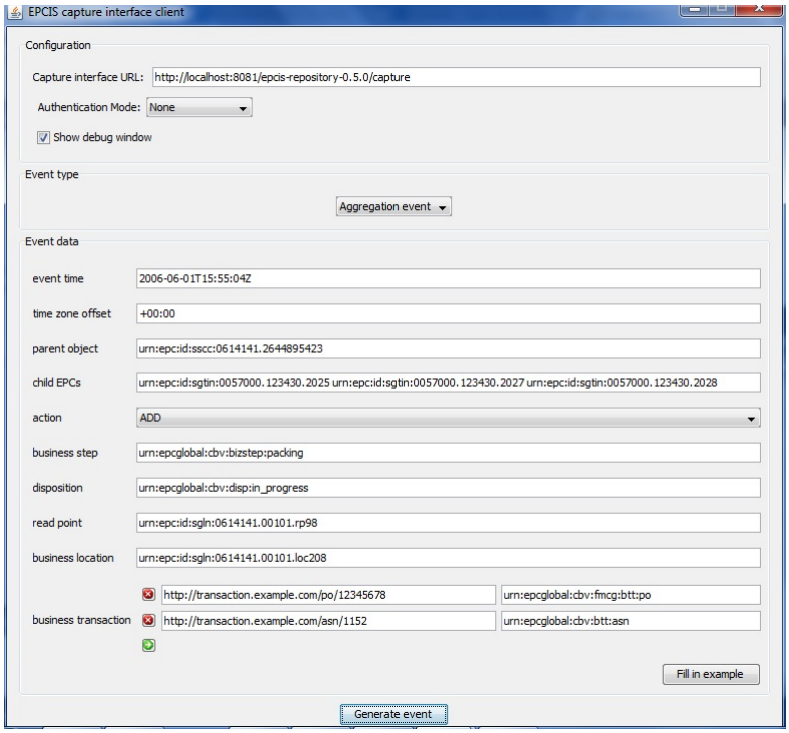


Figure 3.2: Fosstrak EPCIS Capture Client Interface

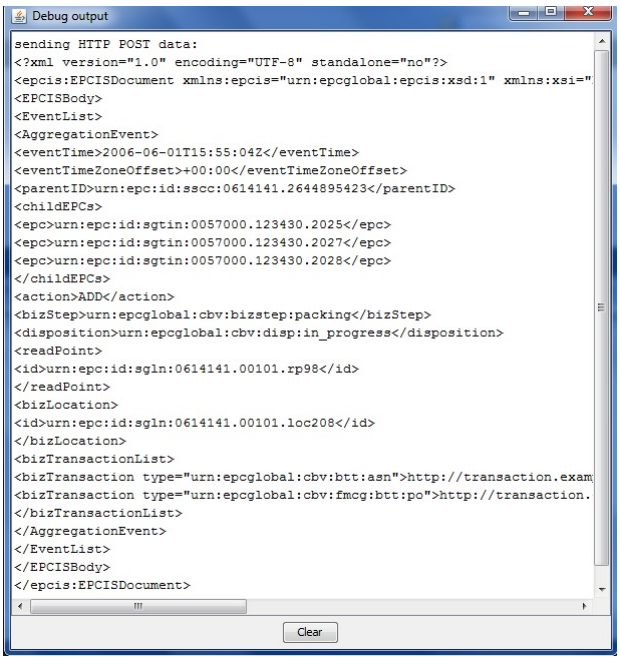


Figure 3.3: Debug window Capture Client Interface

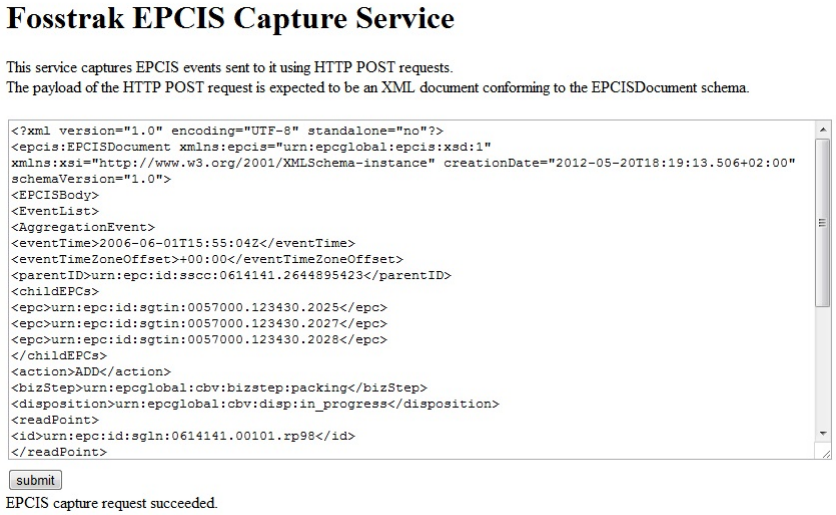


Figure 3.4: Fosstrak EPCIS Capture Client browser interface

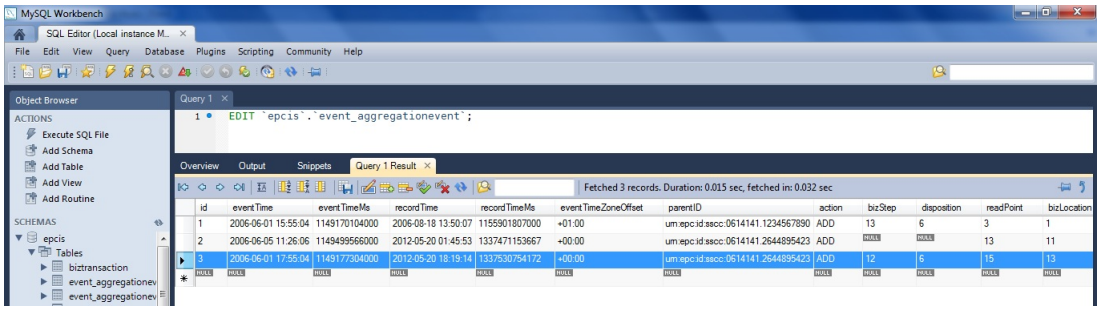


Figure 3.5: AggregationEvent stored into EPCIS repository by MySQL database.

for generate a query and how to fill up the fields to query, for example, the Aggregation Event generated previously. Observe that **Query Interface URL** and **Authentication Mode** fields match with the description given before.

So, one can now fill in the query parameters that he wants to consider for get the event (there is available every possible Event field as a condition) and submit the request to the repository by clicking "Run query". The results of the query will be displayed in a separate window as shows the figure 3.7.

3.3 LLRP Commander

3.3.1 What is LLRP Commander?

Fosstrak LLRP Commander free open-source module provides an intuitive interface to configure and control RFID readers that support EPCglobal's Low-Level Reader Protocol

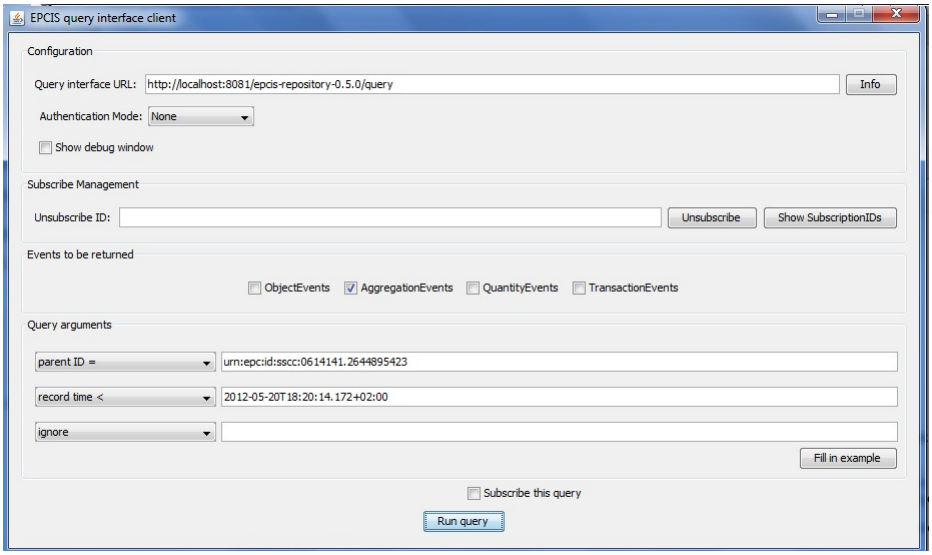


Figure 3.6: Fosstrak EPCIS Query Client interface (*www.fosstrak.org*)

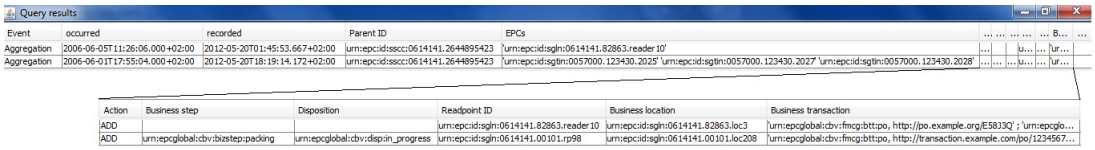


Figure 3.7: Query report

(LLRP) standard. It complements the Fosstrak RFID platform's three other open-source modules. The main idea is that, instead of assembling binary LLRP messages in custom code and project coordinators, an application engineer can use the LLRP Commander to build LLRP configuration messages in a convenient graphical editor.

After building a configuration LLRP message, the application engineer can utilize the LLRP Commander to distribute the configuration to any LLRP-compliant RFID interrogator, thereby eliminating the need to employ different proprietary tools for each reader type in an RFID deployment. Since all messages from and to the readers are captured and stored in a database, he notes, developers can easily inspect reader health and tag-read reports.

"The Fosstrak LLRP Commander thus has the potential to become a universal framework for reader clients—eliminating the need to have a different reader client for each reader type. This means that end users and systems integrators can use a single tool to configure all LLRP-compliant readers, and reader vendors do not have to invest in maintaining their own proprietary reader clients.[32]

It has been a goal of the LLRP Commander project to enrich the existing Filtering and Collection module (ALE middleware) with the ability to communicate with LLRP-enabled RFID readers. Most parts of the communication layer of the LLRP Commander can be used without modification directly by Fosstrak's Filtering and Collection module (see figure 3.8).

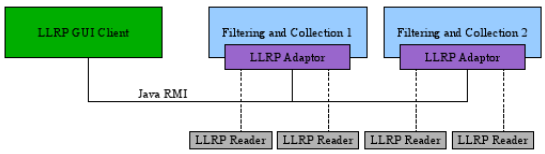


Figure 3.8: LLRP Commander role for Filtering and Collection data. [12]

3.3.2 How to work with LLRP Commander?

The LLRP Commander is an Eclipse plugin which is installed via the Eclipse Update mechanism, so one will need to have Eclipse 3.3 or later and JRE/JDK1.5 or later installed on your computer to run it. The installation via the Eclipse Update Manager procedure is described step by step within fosstrak webside (*fosstrak.org*).

Once the feature is installed, it has to be connected to the LLRP reader(s) in order to send and retrieve messages from it or them. For the purpose of test the correct installation and behavior of the application, could be useful to create a new LLRP reader at DEFAULT adaptor by "*Manually Add Local RFID reader*" instruction at Reader Explorer panel. The

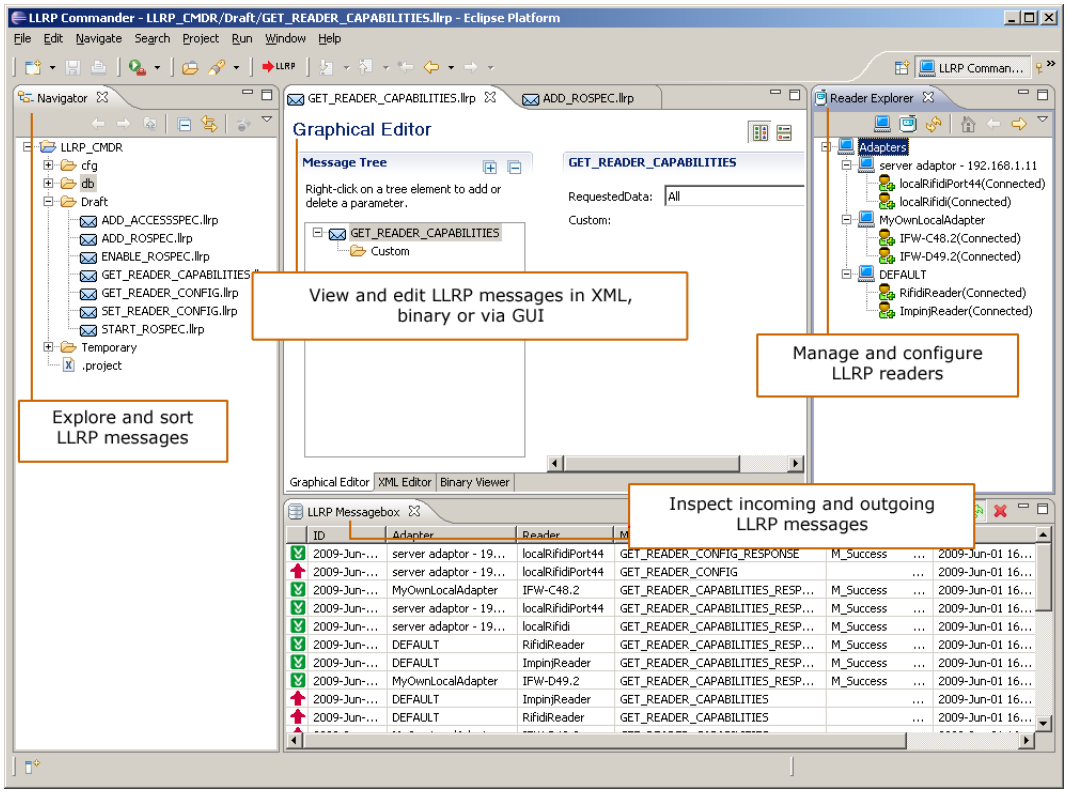


Figure 3.9: LLRP Commander GUI [13]

procedure of how to do connect with a physical reader can be find at the next section 5.2.4.

See below figure 3.9with general overview of the LLRP Commander GUI.

Every part of the module, gives end-users different capabilities for the interaction with the reader:

- **Navigator**: use the built-in wizard to create new LLRP messages. Manage, rename and delete existing LLRP messages within the flexible and easily navigable tree structure.
- **Reader Explorer**: helps one to group readers into hierarchical structures and significantly enhances overview. The dynamic context menu gives one access to reader settings and allows one to send and retrieve LLRP messages to/from the LLRP readers.
- **LLRP Messagebox:** displays outgoing and incoming LLRP messages.
- **Editor View:** Three different editor views (XML, binary and graphical) allow one to conveniently inspect and modify LLRP messages.

3.4 ALE middleware

3.4.1 What is ALE middleware?

The Fosstrak ALE Middleware allows the end user to filter and collect data from RFID readers. It takes the EPC Network role of Filter & Collection software and develops the appropriate tools to interact with Filtering & Collection Interface and Reader Interface (by LLRP Commander software support).

The complete project functionality includes:

- Communicate with RFID reader.
- Filter, aggregate and consolidate raw RFID data from multiple physical readers into one LogicalReader.
- Deploy an RFID middleware that implements the Filtering and Collection role and supports EPCglobal's ALE 1.1 specification and LLRP 1.1 specification.
- Generate ECReports from LogicalReader integrated data.
- Means of support for generate events for an EPCIS repository based on ECReports from the ALE Filtering and Collection middleware by ALECapturingApp.

It supports out of the box any reader that support the EPCglobal LLRP protocol. This includes among others the Impinj Speedway reader, the Motorola FX7400, the Intermec IF61 and the open source Rifidi Tag Simulator. In addition, it also supports the proprietary FEIG protocol (ID ISC.LRU1000/ID ISC.MR101-A). For the readers which ALE middleware does not support its proprietary protocol, the user needs to implement his own HAL (Hardware Abstraction Layer) wrapper, as a support he will find the filtering, event generation, messaging and tool support that Fosstrak has already implemented.

3.4.2 Filtering & Collection Middleware with ALE server and LLRP Support

The Fosstrak ALE Middleware comprises three separate modules to implement EPCglobal's ALE 1.1 specification:

- The **filtering and collection server**: takes the filtering and collecting role as the standard defines; supports Logical Reader API that defines logical readers by Logical Reader Specification (*LRSpec*) and ALE Reading API that defines filtering and collections behavior using *ECSpecs.*

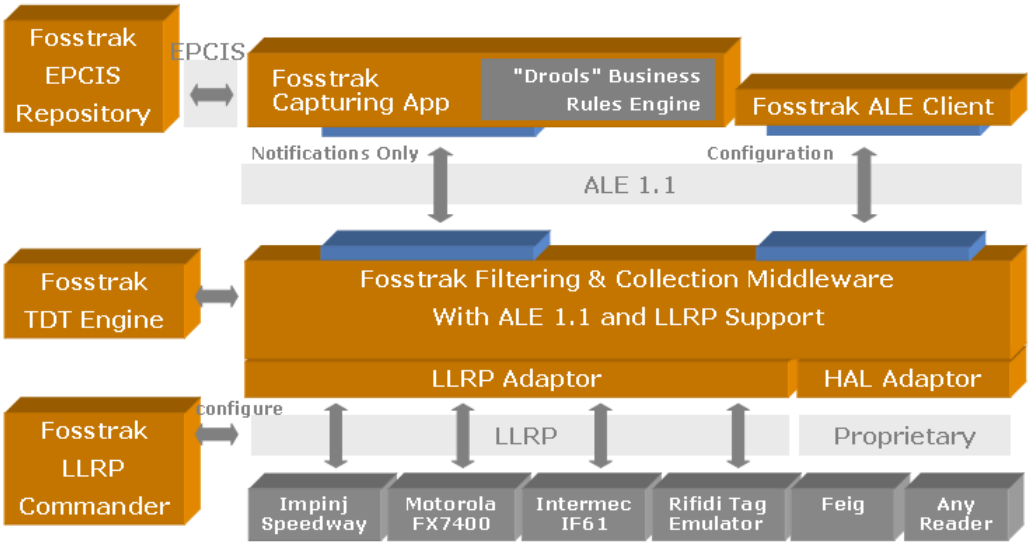


Figure 3.10: Filtering & Collection Middleware overview [14]

- A **standalone client** to configure filtering and collection servers by a Java GUI. It allows to define ECSpecs and LRSpecs on a filtering and collection server and to subscribe and unsubscribe reports generated by the server and specify the address where the reports should be sent to them.
- A **web-based client** to configure filtering and collection servers by a Java web application (WAR). It has the same possibilities as the standalone client does.

For the configuration and managing of RFID readers that support LLRP, Fosstrak ALE server uses **LLRP Commander**.

Figure 3.10 shows an overview of the global implementation of the Filtering & Collection Middleware, and is clear to see the role of each of these modules as well as the interaction with LLRP Commander and TDT modules.

Therefore, Fosstrak filtering & collection server does not implement full ALE 1.1 specification currently, event so is covering notable features as Logical Reader API and EventCycle. Refer to table3.2 for more detailed information.

3.4.3 What is...?

Logical Reader The main idea of LogicalReader is to give a software abstraction of one or more physical readers. It can be the connector from hardware but also from software (HAL) to software. It implements the behaviour of a reader in the context of ALE Client due to have a common view on a reader, without having to care about vendor versions, manufacturer and so on. This is achievable through the Logical Reader API.

Feature	Described in ALE Specification 1.1	Supported by Fosstrak ALE Middleware
ALE reading API	Sect. 8	yes
ALE writing API	Sect. 9	no
ALE LogicalReader API	Sect. 10	yes
Access Control API	Sect. 11	no
ECSpec	Sect. 8.2	yes
ECSpec	Sect. 8.2	yes
ECReports	Sect. 8.3	yes
Event Cycles	Sect. 5.2	yes
LRSpec	Sect. 10.4	yes
LRProperty	Sect. 10.5	yes
Tag Smoothing	Sect. 10.6	no

Table 3.2: Fosstrak ALE 1.1 specification coverage [20]

There are two different types of Logical Reader Definitions by Fosstrak:

- Dynamic Logical Reader Definitions: are read by fc-client and the fc-webclient and are used when we want to specify a logical reader at runtime through the Logical-Reader API.
- Static Logical Reader Definitions: are read/written by the Logical Reader Manager upon ALE deployment. They contain additional information for the Logical Reader Manager.

The Logical Reader Management is an interface through which clients may define logical reader names for use with the Reading API, which maps to one or more sources/actuators provided by the implementation, and there is the role of the Logical Reader Manager, which manages the list of Logical Readers which are part of a current setting.

Adaptor Adaptors allow to integrate a tag reader into the Logical Reader API of the fosstrak ALE. It is normal that each user likes to implement his own adaptor, depending on type of reader that they are working with. Fosstrak gives three options of adaptors for apply in Filtering and Collection ALE middleware and the means to help possible other developers, for each Logical reader definitions.

- **HALAdaptor**: is the most general one and provides an adaptor to the Hardware Abstraction Layer (HAL). It creates a module called SimulatorController that sets up the HAL device by the properties file. Fosstrak provides a HAL device as well, but does not allow polling mechanism, so it is possible to use any other HAL implementation.
- **RPAdaptor**: is an adaptor to the reader protocol (RP). It requires extra functionality to communicate with a rp-proxy and creates a module called InputGenerator

that sets up the connection between the RPAdaptor to the rp-proxy by two channels: command channel (connection settings) and notification channel (delivery of tags from physical reader to RPAdaptor).

- **LLRPAdaptor**: is an adaptor to the low level reader protocol (LLRP). For the integration of an LLRP enabled reader into the filtering and collection framework, Fosstrak implements a simple bridge to the reader management module of the Fosstrak LLRP Commander.

Event Cycle According to the EPC Standard, an EventCycle is the smallest unit of interaction between an ALE client and an ALE implementation through the ALE reading API. An EventCycle is an interval of time during which tags are read. The most important parameters specified by EventCycle are the time interval during which tags are read and the readers where tags shall be collected.

Whenever a client defines a new EventCycle through the ALE interface, a new ReportsGenerator will be created along an EventCycle. The ReportsGenerator acts as a gateway to the EventCycle for clients. A client does not subscribe on an EventCycle but on the associated ReportsGenerator. The ReportsGenerator ensures that the EventCycle is started/stopped and that subscribers (clients) receive the resulting tags. When there are subscribers for the EventCycle the ReportsGenerator starts the EventCycle. See the complete procedure at figure 3.11.

ECSpec and ECReport An ECSpec describes an event cycle and one or more reports that are to be generated from it. It contains a list of logical Readers whose read cycles are to be included in the event cycle, a specification of how the boundaries of event cycles are to be determined, and a list of specifications each of which describes a report to be generated from this event cycle.

An ECReport is the report generated after one EvenCycle and contains the information that the application got considering associated ECSpec.

To get a report an application has to subscribe itself to an already existing ECSpec using the Filtering and Collection client or webclient. The subscription has the form of an URI defining a host and a port for the later transfer of the ECReports.

A part of by subscription, there is 2 other methods to get a ECReport from the ALE, one is by **polling** and the other one is **immediate** notification. The method poll registers itself to an already defined ECSpec and waits until the first EventCycle is terminated and the ECReports are generated. Instead of registering itself with an URI, where the ECReports are should be sent to, the ECReports are returned directly. The immediate method actually activates a polling method not with a defined ECSpec but generating a

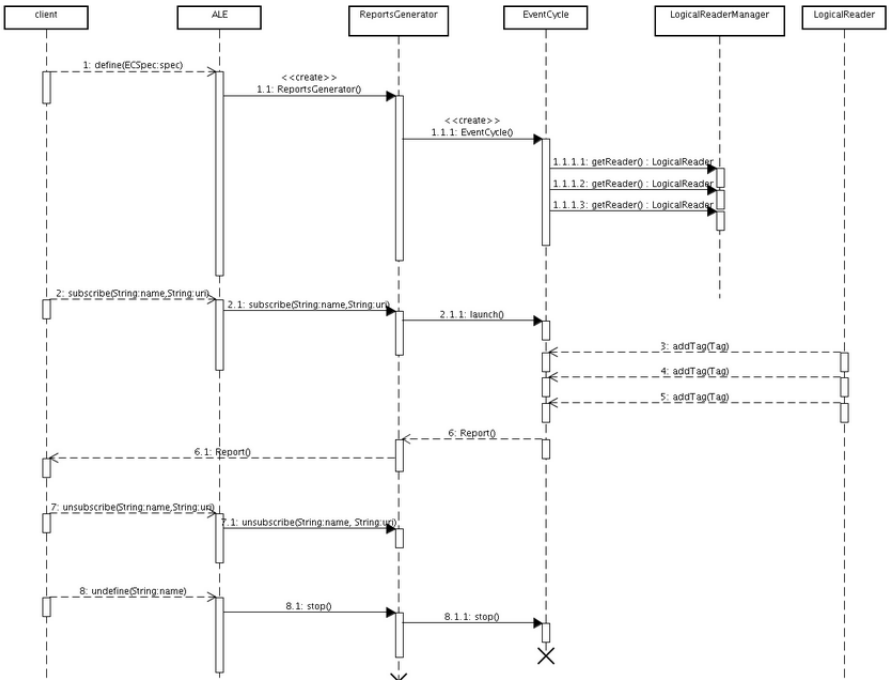


Figure 3.11: EventCycle behaviour *[15]*

new one

3.4.4 How to work with ALE middleware?

The features that Fosstrak ALE middleware offers to the user are:

- one server that interacts with the readers though Logical Reader Specifications (LRSpec) and the ALE Reading API.

- one client that is used for the server configuration. It is delivered as Java file format and as web-browser format and what it does is send to the server the LRSpec and ECSpec that it should follow.

- one tool that displays incoming HTTP requests.

- simulation of an example of a capture procedure with a HAL-simulator to show how to work with Fosstrak CaptureApplication due to "translate" from an ECReport to an EPCISEvent and store it to the EPCIS repository.

For the developers who wish to work with fosstrak and interact with the code, offers:

- three different implementation of Adaptors to integrate a tag reader into to the Logical Reader API: HALAdaptor (for a simulated hardware), RPAdaptor (for the reader are not LLRP, adapted to the reader protocol RP) and LLRPAdaptor (for the readers which support low level reader protocol). Moreover, Fosstrak gives the steps to create a new adaptor for the users who likes to create their own adaptor for their specific physical

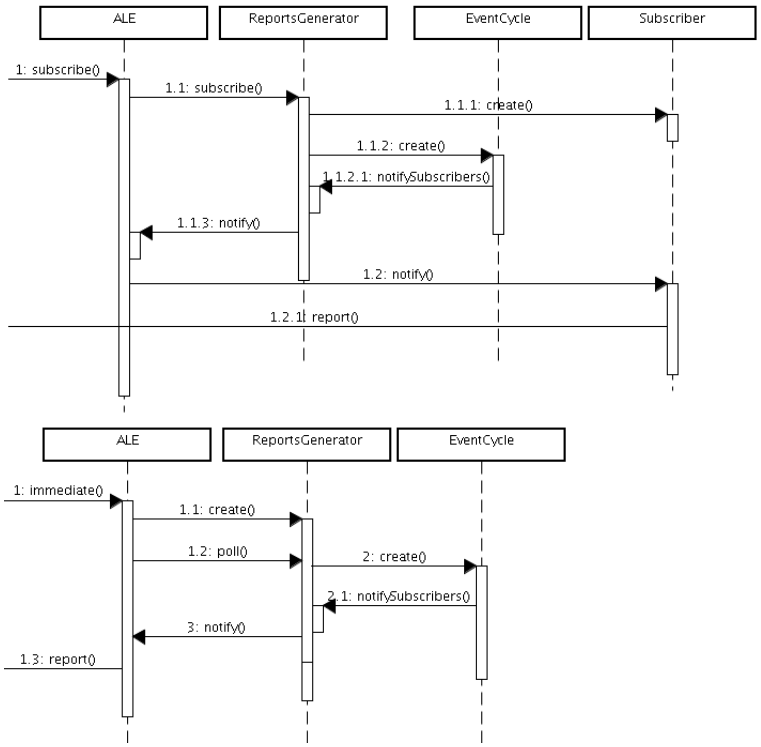


Figure 3.12: Subscription, poll and immediate notifications methods. *[16]*

reader. A part of them Fosstrak deliver a TestAdaptor that is only to make tests and simulate a reader that from time to time reads a tag.

- defined steps to implement a CaptureApplication. It gives two options: Generic Fosstrak Capturing App and a complete new Capturing App.

Apart from all of this, fosstrak includes all the required files (.xml files).

3.4.4.1 Fosstrak fc-server.

In order to work with Fosstrak ALE middleware, the firs step is have fc-server installed into the system. It is understood that Tomcat servlet is already installed and running into the system and fosstrak fc-server package downloaded, then the only thing that is needed is copy the fc-server WAR file into the Tomcat's *webapps* directory.

To acces to the application by web browser, the url is: *http://localhost:8081/fc-server-1.0.2/services/ALEService.*

For the configuration of the fc-server, the used files are InputGenerators.properties and LogicalReaders.xml, both of them can be found into the webapps/fc-server-1.0.2/WEB-INF/classes folder, into the Tomcat's directory.

3.4.4.2 Fosstrak standalone-client

The Filtering and Collection Standalone Client is a pure Java application. It is configured through the *ALEClient.properties* file. To get started quickly, it should be sufficient to adjust the EndPoint option to specify the server to communicate with. All other options can be left unchanged. The package can be downloaded from frosstak webside.

Ones the endpoint is defined, we can execute the .jar file and start the server configuration.

The application has two different windows, one for Reading API configuration (ECSpecs) and one for Logical Reader configuration (LRSpecs). From these applications are launched the commands to define,LogicalReaders, get ECSpecs, etc. through the *Command* section and the response from the server is displayed at *Result* section. To send the specified command click to *execute* button. See figure 3.13.

3.4.4.3 Fosstrak web-client

The Filtering and Collection Web Client is based on Java Server Pages (JSP). To set it up the only action that is needed, ones the package is downloaded into the system, is copy the WebClient WAR file into the Tomcat's webapps directory. After restart Tomcat will be possible to access to the WebClient GUI by connect to the Tomcat host place through the web browser. (eg. *http://localhost:8081/fc-webclient-1.0.2/services/ALEWebClient.jsp*).

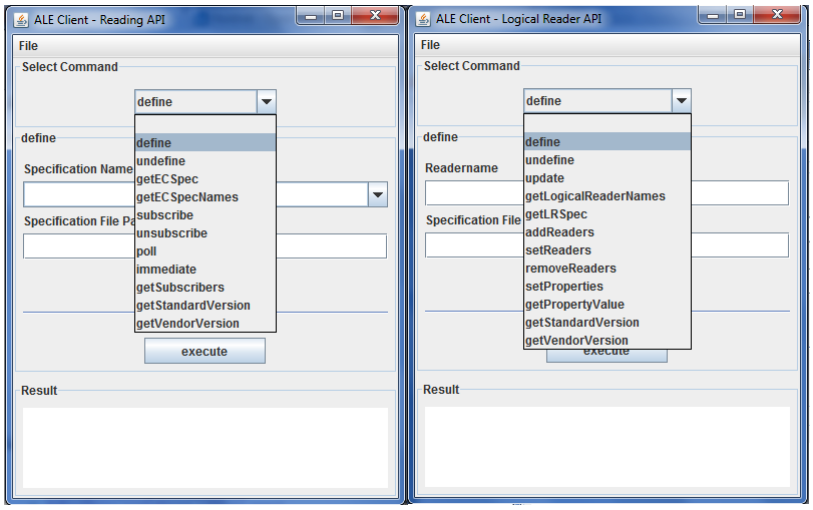


Figure 3.13: ALEClient GUI.

The screen that appears is showed at figure 3.14. There are four differentiate sections, each one with different function:

- Filtering and Collection API (ALEServicePort): are listed all the possible commands that could be needed for the Reading API configuration: define, subscribe,... ECSpecs.

- LogicalReader API (ALELRService Port): are listed all the possible commands that could be needed for the LogicalReader API configuration: define, add, remove,... LRSpecs.

- Inputs: is where the filds needed to execute the command must be filled. To finally launch the command, button *Invoke* has to be press.

- Result: is where the answer from the server is displayed after invoke any command from the client.

The first command that must be invoked due to communicate server and web client, is *setEndpoint(String endPointName)* for Filtering and Collection API either for LogicalReader API, and insert the URL of the Filtering and Collection Server that is wished to communicate with. To check if the communication was correctly set, invoke *getVendorVersion()* command.

3.4.4.4 Fosstrak EventSinkUI

EventSinkUI is an user interface that can listen to a port and display incoming HTTP requests. It is delivered from Fosstrak into reader-rp-client package. To use it, it is necessary to download the reader-rp-client package, unpackage it and execute by console the following instruction:

java -cp <Reader_RP_Client_Version>.jar\org.fosstrak.reader.rp.client.EventSinkUI <PORT>

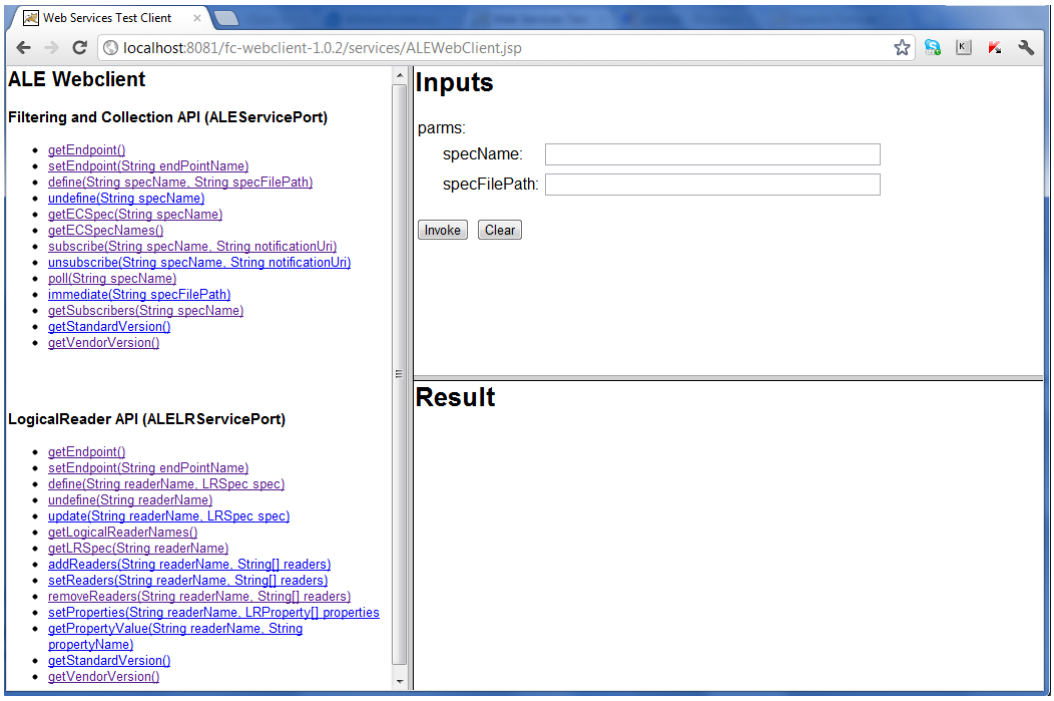


Figure 3.14: ALEWebClient GUI.

and make sure that the provided port is not used by any other application. The screen of the figure 3.15will appear.

3.4.4.5 Fosstrak fc-server configuration

Before to start the fc-server configuration make sure that there is any physical reader connected (or simulator) and Tomcat server is running with the appropriate .war files into the webapps folder.

To configure filtering and collection (fc) server through the filtering and collection clients the first step is to communicate each other. As it was explained before, by standalone-

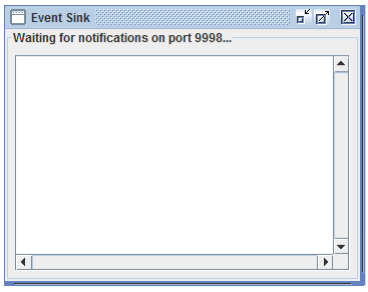


Figure 3.15: EventSinkUI

client it is done changing the ALEClient.properties file, and by web-client it is done invoking *setEndpoint(String endPointName)* and writing the URL of the server.

Ones this step is done, the next one is set the readers connected to ALE Middleware via the ALE Logical Reader API: first send the command **define** and specify *readername* and *specFilePath* with the chosen LRSpec file and its path. To verify that the reader has been created by sending **getLogicalReaderNames()** command and receiving as a result the reader just creates into the list of logical readers.

Next step is define the filtering and Collection Behavior via the ALE Filtering and Collection API (Reading API): first send the command **define** and specify *specname* and *specFilePath* with the chosen ECSpec file and its path. To verify that the Event Cycle has been created by sending **getECSpecNames()** command and receiving as a result the Event Cycle just creates into the list of ECSpecs.

Ones the Filtering and Collector Server knows the Logical Reader that is sending data and how the data will be sent, the next step is create a subscriber to the ECSpec, because when there is no subscriber for an ECSpec, the ECSpec is not executed. This is done by sending **subscribe**(String specName, String notificationUri) from ALE Filtering and Collection API (Reading API) and register the URL on which there is any tool that displays incoming HTTP requests listening (for example Fosstrak Event Sink GUI).

After all of these steps, if everything is working fine, the http requests monitor starts to display the reports with the aggregated data sent by the filtering and collection server.

3.4.4.6 ECSpec and LRSpec

At the end, the way what you are giving the information how the ALE Server must filter and collect the data is through the .xml files. Because this, it is very important to understand how they must be structured and how interact with them.

ECSpec The ECSpecs define how an specific Filtering and Collection Server shall generate reports. You can retrieve current tags, tags that have been added or deleted with respect to the last EventCycle or combinations of all. The .xml file specified things as Logical Reader name or the type of the report generated and the duration of the EventCycle. The ECSpec becomes active as soon as you subscribe with a notificationURI. Is important to consider that readTimeInterval should be smaller than the duration of the EventCycle.

Fosstrak deliver three examples of most usual ECSpecs: ECSpec_current (retrieve all the tags that are currently on the reader), ECSpec_additions (for the tags that have been added since the last EventCycle) and ECSpec_deletions (for the tags that have been deleted from the last EventCycle).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:LRSpec xmlns:ns2="urn:epcglobal:ale:wsdl:1" xmlns:ns3="urn:epcglobal:ale:xsd:1">
    <isComposite>false</isComposite>
    <readers/>
    <properties>
        <property>
            <name>ReaderType</name>
            <value>org.fosstrak.ale.server.readers.llrp.LLRPAdaptor</value>
        </property>
        <property>
            <name>Description</name>
            <value>LLRP reader</value>
        </property>
        <property>
            <name>PhysicalReaderName</name>
            <value>LogicalReader1</value>
        </property>
        <property>
            <name>ip</name>
            <value>localhost</value>
        </property>
        <property>
            <name>port</name>
            <value>5084</value>
        </property>
        <property>
            <name>clientInitiated</name>
            <value>true</value>
        </property>
    </properties>
</ns3:LRSpec>
```

Figure 3.16: Dynamic LLRPReader.xml [11]

```
<?xml version="1.0" encoding="UTF-8"?>
<LogicalReaders xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="/resources/LogicalReaders.xsd">
        <LogicalReader name="LogicalReader1">
                <LRSpec isComposite="false" readerType="org.fosstrak.ale.server.readers.llrp.LLRPAdaptor">
                        <LRProperty name="Description" value="LLRP reader"/>
                        <LRProperty name="PhysicalReaderName" value="LogicalReader1"/>
                        <LRProperty name="ip" value="localhost"/>
                        <LRProperty name="port" value="5084"/>
                        <LRProperty name="AdaptorClass" value="LLRPReader"/>
                        <LRProperty name="ImplClass" value="org.fosstrak.ale.server.readers.llrp.LLRPAdaptor"/>
                        <LRProperty name="clientInitiated" value="true"/>
                </LRSpec>
        </LogicalReader>
</LogicalReaders>
```

Figure 3.17: Static LLRPReader.xml [11]

LRSpec An LRSpec contains information about a reader so, there is one LRSpec definition per each kind of reader. Fosstrak delivers examples of .xml file for each kind of reader (RPReader, HALReader, Composite reader, LLRPReader and for a TestAdaptor).

When you want to define your own LogicalReader through an xml-file you need to obey some restrictions:

-The xml must contain exactly one LRSpec definition.

-It must be defined whether the reader is composite or not.

-The reader must contain at least the LRProperty of the ReaderType.

-If your reader is a composite reader, you must provide the list of the "subreaders".

-The xml must contain exactly one LogicalReaders tag.

-Whenever you define a LogicalReader you must specify an LRSpec and within that LRSpec you must specify if this reader is composite or not.

The .xml format changes depending on if it is a dynamic or static LogicalReader. See figure 3.16and figure 3.17as examples.

3.4.5 ALECapturingApp

Fosstrak ALE contains a schematic for a capturing application implementation. This means, how the Fosstrak Capturing Application can be used to link the Fosstrak ALE and the Fosstrak EPCIS repository by retrieving ECReports from the ALE and delivering EPCIS events to the EPCIS repository.

On one part, Fosstrak develops a simulated two different scenarios using a HAL simulator that models a simple store equipped with three RFID readers and uses the ALE filtering and collector server with a defined configuration. It is implemented one simple scenario and another more elaborate. The main idea is:

- first of all define a new capturing application that is listening on one port for incoming ECReports, and that is delivering EPCIS events to the EPCIS repository at http://localhost:8080/epcis-repository-0.5.0/capture by configuring webapps/capturing-app-VERSION/WEB-INF/classes/captureapplication.properties file (see figure 3.18)

- there is a multi reader simulator what is capable of simulating several readers in one application, To configure it fosstrak gives two files that must be included into the Tomcat's folder webapps\fc-server-VERSION\WEB-INF\classes\props: *SimulatorClient.xml* and S*imulatorController.xml.*

- After this, as the normal procedure to connect ALE server to some reader, set up EventCycle and LogicalReader per each reader that is wanted to work with through filtering and collection ALE.

- Then, configuration to deliver ECReports to the capturing application, that is invoke a *subscribe(String specName, String notificationUri)* in order that the notification URI will be the host that will deliver the reports.

- finally connect to the MySQL server and switch into the EPCIS database use epcis to see the result of what the simulator is sending.

On the other part, Fosstrak gives the user the opportunity to implement a particular capture application by two different way: using a generic capturing application provided by Fosstrak or developing the code for a completely new one as a developer user. The difference between them is that the generic fosstrak capturing application tries to reduce the coding amount to an absolute minimum. Retrieval and de-serialization of ECReports is handled automatically for the user, as well as the delivery of ECPIS documents to the EPCIS repository. (see section 5.2.5).

```
file: <TOMCAT_FOLDER>/webapps/capturing-app-VERSION/WEB-INF/classes/captureapplication.properties

n=1

cap.0.port=9999
cap.0.name=myFirstCaptureApp
cap.0.epcis=http://localhost:8080/epcis-repository-0.4.2/capture
```

Figure 3.18: captureapplication.properties file configuration [17]

3.5 Tag Data Translation (TDT)

3.5.1 What is TDT Engine?

The Fosstrak TDT Engine offers and easy way to convert between different EPC representations.

An EPC identifier may be expressed in a number of representations or encodings, such as BINARY, tag-encoding URI, pure-identity URI, legacy formats. The Fosstrak Tag Data Translation (TDT) Engine provides flexible translation (encoding/decoding) between these different representations of an EPC (see figure 2.11).

The Fosstrak TDT Engine makes use of the XML schema and instance files defined in EPCglobal Tag Data Translation v1.0 and represents a convenient way for Java developers to integrate TDT functionality into their software.

3.5.2 How to work with Fosstrak TDT Engine?

To cover EPCglobal Tag Data Translation v1.0 standard, Fosstrak gives a tool able to convert to EPC format several different kind of data that can be found into a RFID tag. The tool consists in different functions that read each field from a tag format and pack the information that contain into a EPC format. The functions are delivered by code, so that this code must be integrated into the general code of ALE server implementation.[11]

Possible usage scenarios include the following:

- Convert a binary EPC string read from a tag into a tag-encoding URI for use with ALE.

- Convert a binary EPC string read from a tag into a pure-identity URI for use with EPCIS.

- Convert a binary EPC string into a legacy format (e.g. GTIN + serial number) for use with legacy applications

- Convert an EPC into a hostname, to be looked up in order to perform an ONS query (N.B. ONS currently only supports SGTIN).

- Convert an EPC into a binary format, for writing an EPC onto a tag.

Chapter 4

Case of Study

Nowadays, the importance of sharing information is growing up vertiginously because everything is more and more connected. For the logistics and supply chain management it is not different and is starting to increase the idea of the Internet of things, what intents to has a global information about every single item around the wold. Such a huge project is tempting for all the companies that wish to get involved on it and so that, are appearing lots of projects attempting to contribute someway.

Supply Chain management has a close relationship with RFID technology as it offers a great contributions for make it easy, like, for example, that does not exist obstacles for a connection between the reader and the tag. For that reason, the RFID investigation teams have deep interest to work for get involved with all this movement.

But such huge and common project must follow lots of rules, to allow all the contributors understand each other. This reason is doing slower the process and it needs more effort.

This study has the aim to get contact with the present available tools that allow this interaction between RFID technology and EPCglobal standards.

At the beginning of this work, the department of Telecommunication Engineering of Czech Technical University of Prague just finish working into a couple of projects that developed a RFID inventory system, composed by one client and one server that connects from RFID reader to a database server to store the data read by the reader from a RFID tag. It was the first option to continue with this projects to improve the system, but after work for a months on this, it was discovered the existence of an open-source software that implements the same procedure following EPCglobal standards. This was the reason to decide invest the effort on that way, and change the project to discover how to manage the tool and which could be the profits.

The main target of this work then, is analyse deeply the tool, module by module, by comparing it with other applicable options, see how it fits with the EPCglobal Architecture Framework and give a list of conclusions that will help for next projects working on this

line.

The department has disposal a Motorola XR480 RFID reader and an AN480 RFID antenna, therefore might be analyzed as well if the software is compatible with these devices. The configuration of the reader will be very important to have all the system working.

Chapter 5

Related work

5.1 Tools

The complete system running is composed by several software and hardware components. Those are:

- The reader: Reader MotorolaXR480. [33]
- The antenna connected to the reader: AN480 RFID Antenna.
- The RFID tag: different RFID and GS1 tags.
- The Switch: to make the LAN for connect host with middleware software and reader.
- MySQL database: MySQL Server to configure the database and MySQL Workbench as a GUI to work with the database.[34]
- Tomcat servlet: to be running the web applications.[35]
- Fosstrak software packages.[11]
- LogicAlloy ALE Server.[36]
- Rfidi Emulator.[37]

At the end, the RFID system looks like figure.

Figure 5.1: Complete RFID System

5.2 Set up the interface

5.2.1 Run Apache Tomcat servlet

To have a servlet container running into the system is a paramount part for have the connection between the different modules and applications of our system. Apache Tomcat is an open source implementation of Java Servlet technology that allows us to interact with another different applications by web-browser interface.

To work with Apache Tomcat we need to have installed already into the system Java SE Runtime Environment (JRE) or full JDK package. Then, download package (tomcat.apache.org) defining a new folder for it, normally named apache-tomcat-(version) and install the application by Windows Service Installer. We may consider the port for the http connection, to be sure do not use any port that is being used already. We choose port 8081 (see figure 5.2). [38]

Include a user administration is optional, we have assigned: *username=CVUT, password=Prague* for the role manager-gui.

Into Apache Tomcat folder we can find several internal folders which we will have to work with and place application files: *catalina* (release directory), *webapps* (folder for the applications that can be displayed by web browser), etc.

To start up the servlet execute the command %catalina_home%\bin\startup.bat or initialize Apache Tomcat Windows service. Ones it is running we can access it by web browser calling: http://localhost:8081.

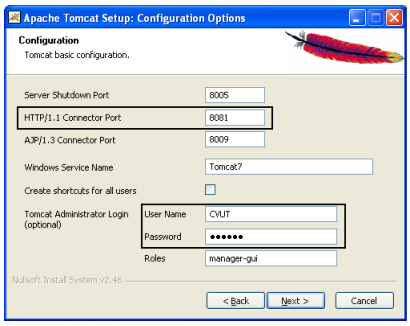


Figure 5.2: Tomcat servlet configuration

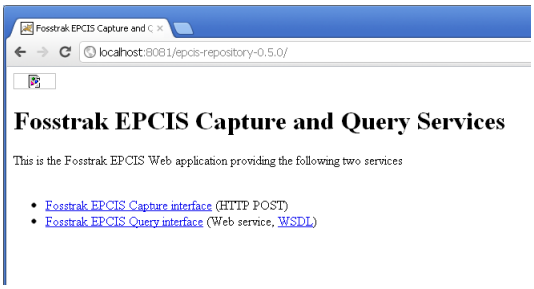


Figure 5.3: Fosstrak EPCIS Web Application

5.2.2 Create an EPCIS Repository

First of all download the epcis-repository package from fosstrak webside (EPCIS / Download) and copy the WAR file into the "tomcat_home"/webapps folder. Automatically, a new epcis-repository folder will be created and it is possible to access to the epcis-repository through Tomcat servlet by connecting http://localhost8081/epcis-repository-0.5.0/. See figure 5.3.

Next step is set up a new MySQL database. We will need first download, install and configure MySQL Server and MySQL Workbench. Both package are available from the MySQL webside [34]. For MySQL server configuration, it is needed to define during the installation process:

- hostname: localhost, 127.0.0.1
- tcp port to connect to: 3306
- username: root
- password: s3cret

and would be very useful to enable the MySQL Command Line Client.

To make sure that web applications deployed to Tomcat can access our MySQL database we will need to install MySQL Connector/J driver. Download it and copy the *mysql-connector-java-<version>-bin.jar* into Tomcat's lib directory.

Ones this is done, follow the steps defined below to create a database for EPCIS events by MySQL Command Line Client (or by console).

mysql> CREATE DATABASE epcis;

mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON epcis.* TO epcis IDENTIFIED BY 'epcis';

mysql> USE epcis; mysql> SOURCE <path-to-unpacked-download>/epcis_schema.sql

mysql> SOURCE <path-to-unpacked-download>/epcis_demo_data.sql (optional, to populate the database with some examples)

epcis_schema.sql gives the global structure of the database by tables and the fields that each data that will be stored must contain.

For the MySQL Workbench configuration go to Database–> Manage DB Connections and make sure all parameters are set correctly. Then click to Test Connection and see that the connection to the database created is working fine. See figure5.4.

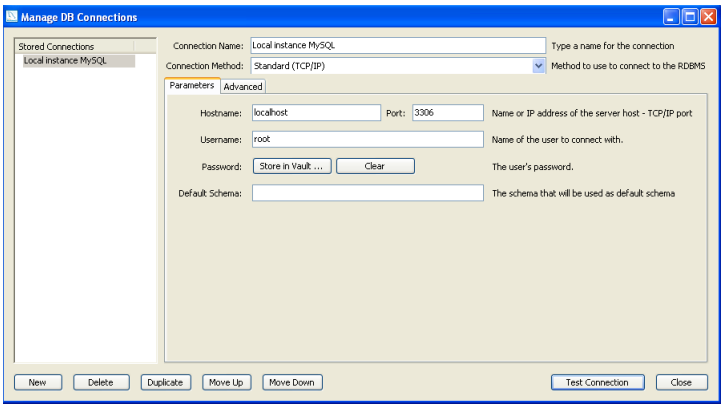


Figure 5.4: MySQL Workbench- manage DB connections

Check now if the application is running. In a default installation of Tomcat, the capture and query interfaces will now be available at http://localhost:8081/epcis-repository-<version>/capture and http://localhost:8081/epcis-repository-<version>/query, respectively. When you open the capture interface's URL in your web browser, you should see a short information page as is showed in figure 5.5.

To also check if the query interface is set up correctly, point your browser to its URL and append the string ?wsdl to it. The WSDL file of the query service should now be displayed in your browser. See figure 5.6.

Now, we are able to interact with the database by the Fosstrak JAVA capture and query application that there were presented in chapter 3, section 3.2.2.

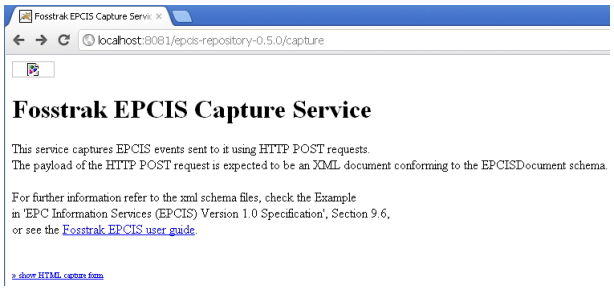


Figure 5.5: epcis-repository capture web application

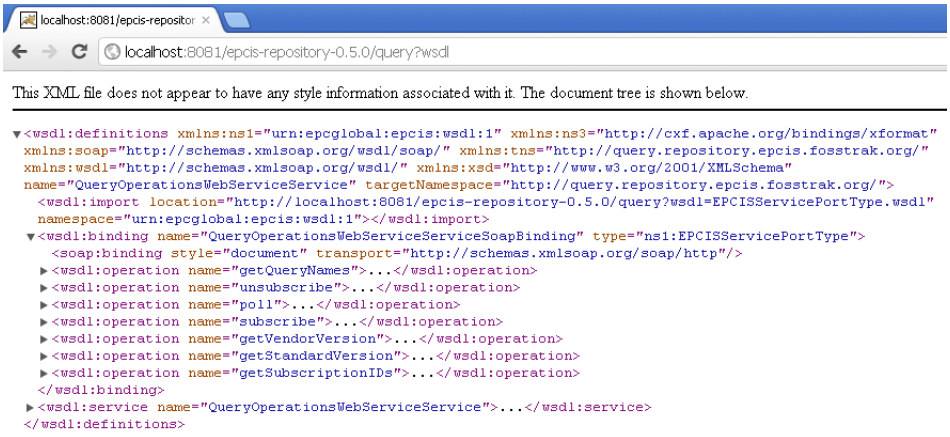


Figure 5.6: epcis-repository query web application

5.2.3 Reader configuration

For the connection between the reader and ALE Server, reader must be configurate properly. The reader is easily to reconfigure using the Administrator Console. For access to the Administrator Console there are two options:

- **RS232 Connection**: Connect a DB9 serial cable to the reader RS232 port and launch a terminal emulation program (such as HyperTerminal) on the host computer. Download the latest release of the software from the Motorola webside [39] and configure it as the manual says [33].

- **Ethernet Connection:** use an Ethernet connection (10/100Base-T Ethernet cable) from the host to the reader. If not connecting to an Ethernet network, connect one end of an Ethernet crossover cable (not provided) to the Ethernet card on the computer, and the remaining end to the TCP/IP port on the reader. On the host open an internet browser and enter the reader IP address. The IP address from factory defaults (after a reset) is 192.168.10.31. However, it is possible to use a

networking software (such as WireShark) to get the current IP address of the reader configuration. Then, the Administrator Console login window appears.

Following the next steps by Administration Console screens will be possible to make the communication line between the RFID LLRP reader with the LLRP Adaptor created by the ALE Server and interacts with LLRP Commander commands.

5.2.3.1 Log in

To log in into the reader configuration options insert the username and password (from factory defaults they are admin and symbol, but is possible to change it after get inside).

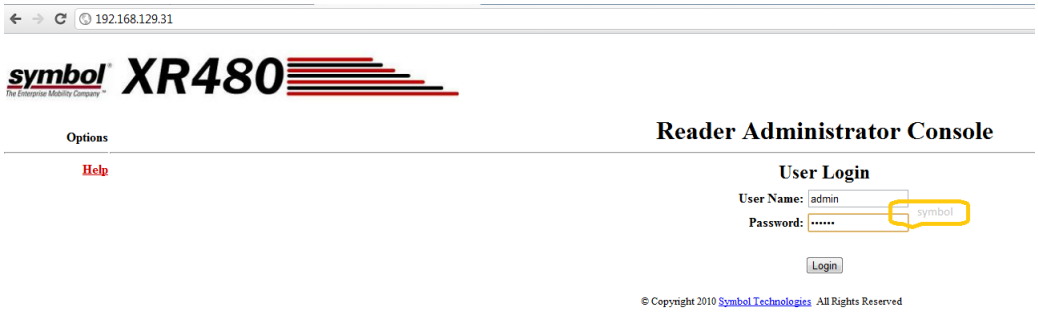


Figure 5.7: Access to the Administrator Console

5.2.3.2 Main panel

Once the access is successful the following panel appears and allow to navigate in through the options.

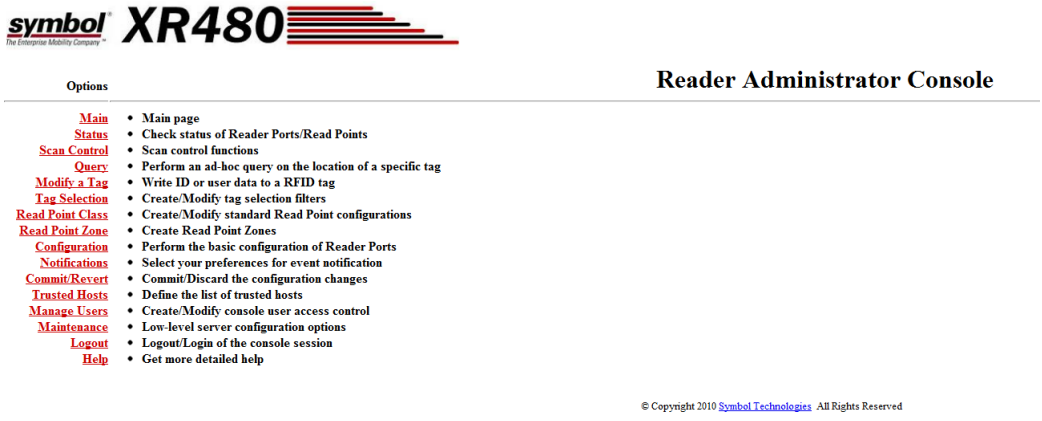


Figure 5.8: Main panel

5.2.3.3 Status

To check the current status of the reader: how many read points are enabled and working and the current status of the system. It is a information screen, not configuring one.

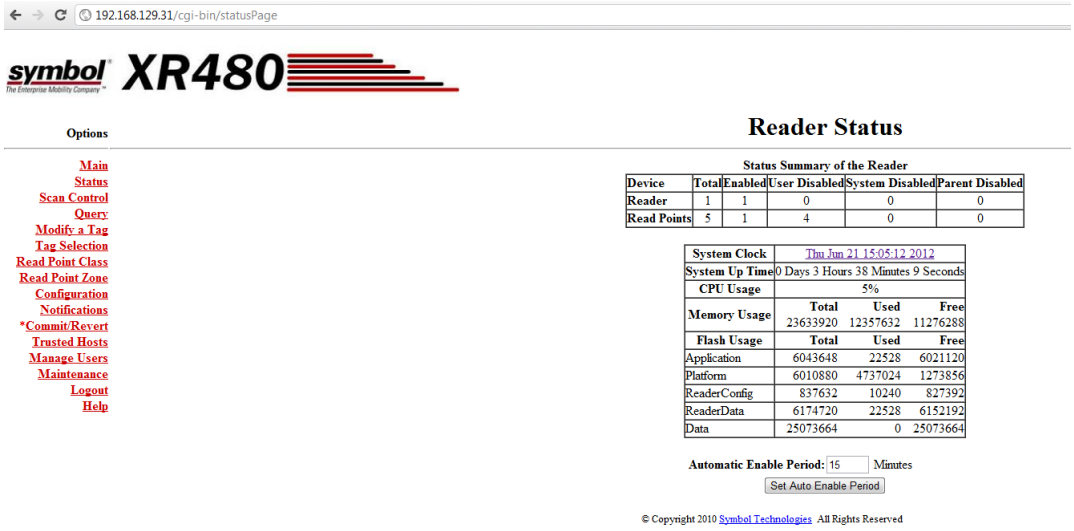


Figure 5.9: Reader status window

5.2.3.4 Tag selection

This screen is to select the kind of tags that is wished to detect. Select “EPC tags”. It is possible as well to create a personal filter, we will use the available one called “ALL”.

5.2.3.5 Scan control

To initiate an on demand scan and/or to enable/disable polled read points. Enable Polling by clicking the button.

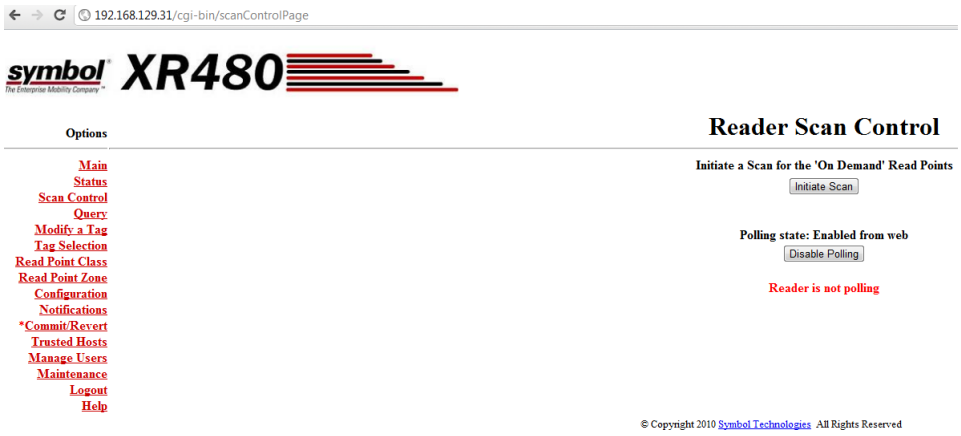


Figure 5.10: Scan Control window

5.2.3.6 Read Point Class

To schedule a periodic scan for a read point. Specify the tag filter used and the gain for the antenna. As we have available only one antenna, it is only necessary to configure one read point.

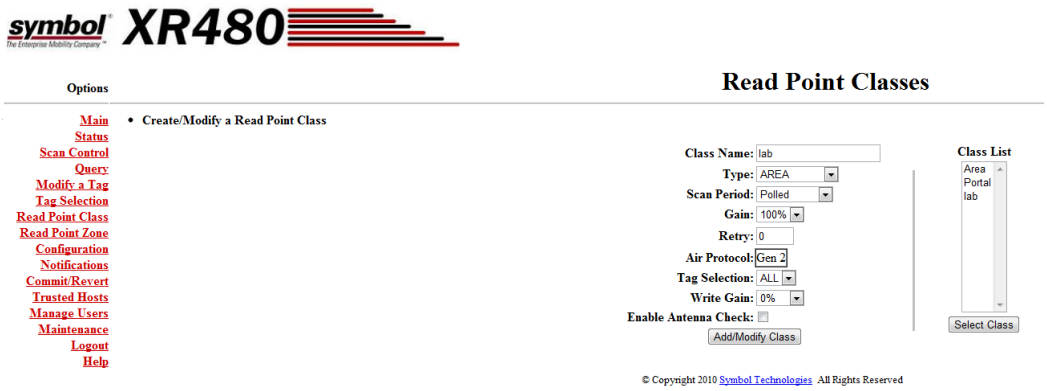


Figure 5.11: Read Point Class window

5.2.3.7 Read Point Zone

To create and manage zones. We will use the zone default because we do not need different zones for our study.

5.2.3.8 Reader Configuration

To configure the reader and all the read points (antennas) which we are using. Set first the reader properties and then define the read point. As we only have one antenna, enable

one read point and disable the others. For the read point configuration, use the zone and the class defined.

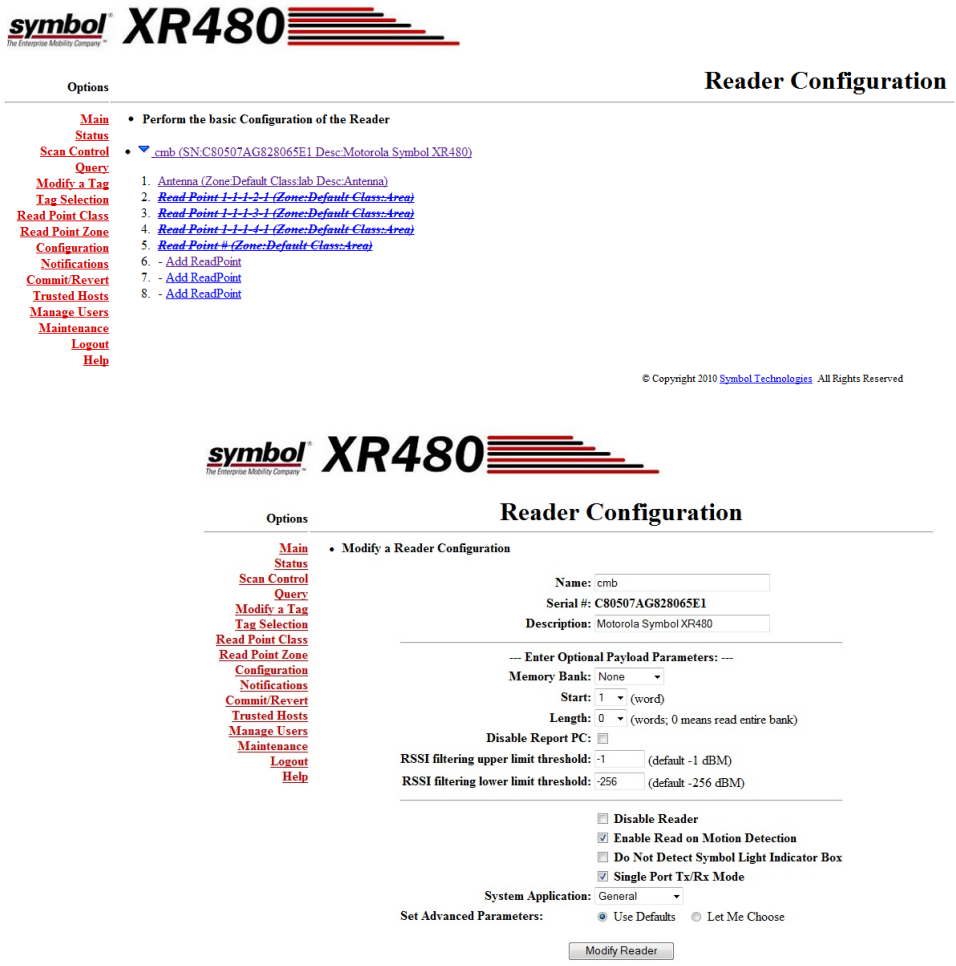


Figure 5.12: Reader and read point configuration windows

5.2.3.9 Query

To read tags and get status information. Check now if with the current configuration the reader is able to read tags. By clicking in "Submit Query" button, will be displayed all the tags visible by the reader. To check the tag ID and the time when it was detected, a report is generated as well by the reader in .xml format and displayed by navigator browser. To check it enter the following URL: http://[Reader IP Address] /cgi-bin/dataProxy?oper=queryTags.

Everytime that the reader detects a tag, includes it on a Internet database to be checkable for the user after time.

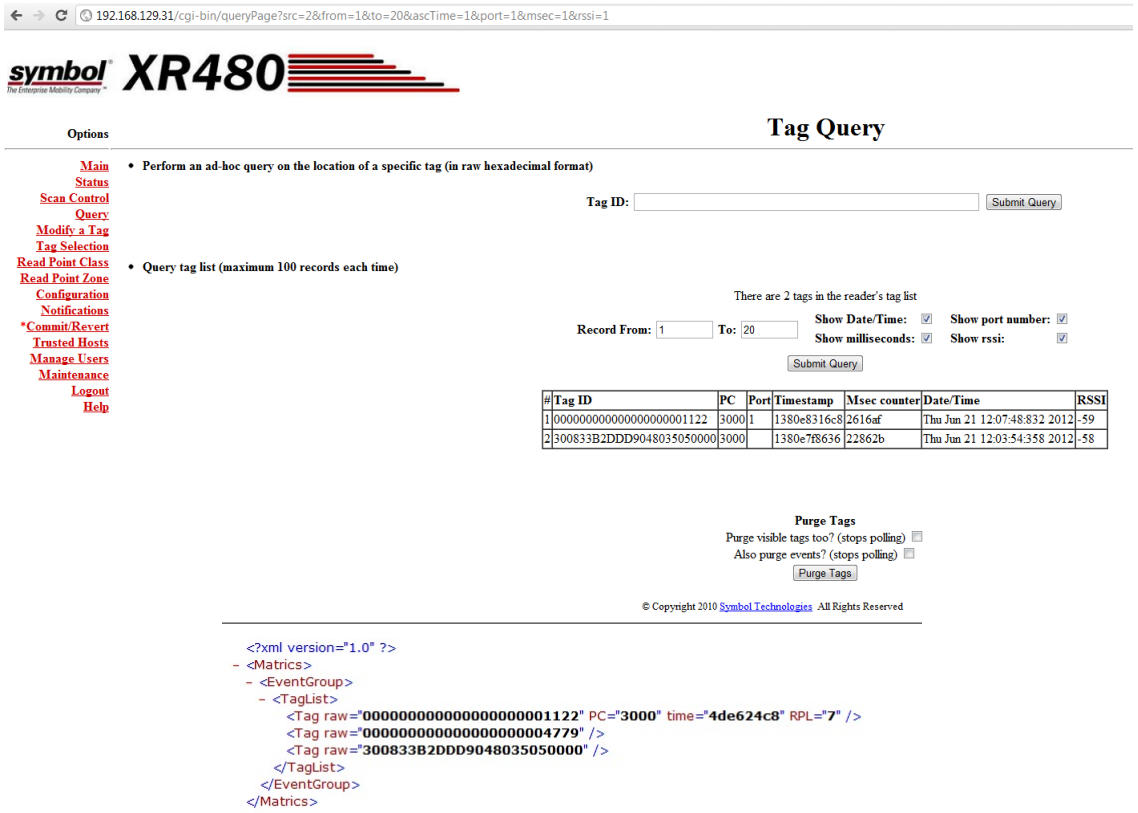


Figure 5.13: Query window and .xml report format

5.2.3.10 Notifications

It is possible through de Administrator Console to specify a host to send an event notification everytime that the reader detects one new tag, or that visibility changes, etc. The window Notifications has to be configured.

As we are asking to the reader for the information by a subscriber from ALE Server, is not necessary to configure this window.

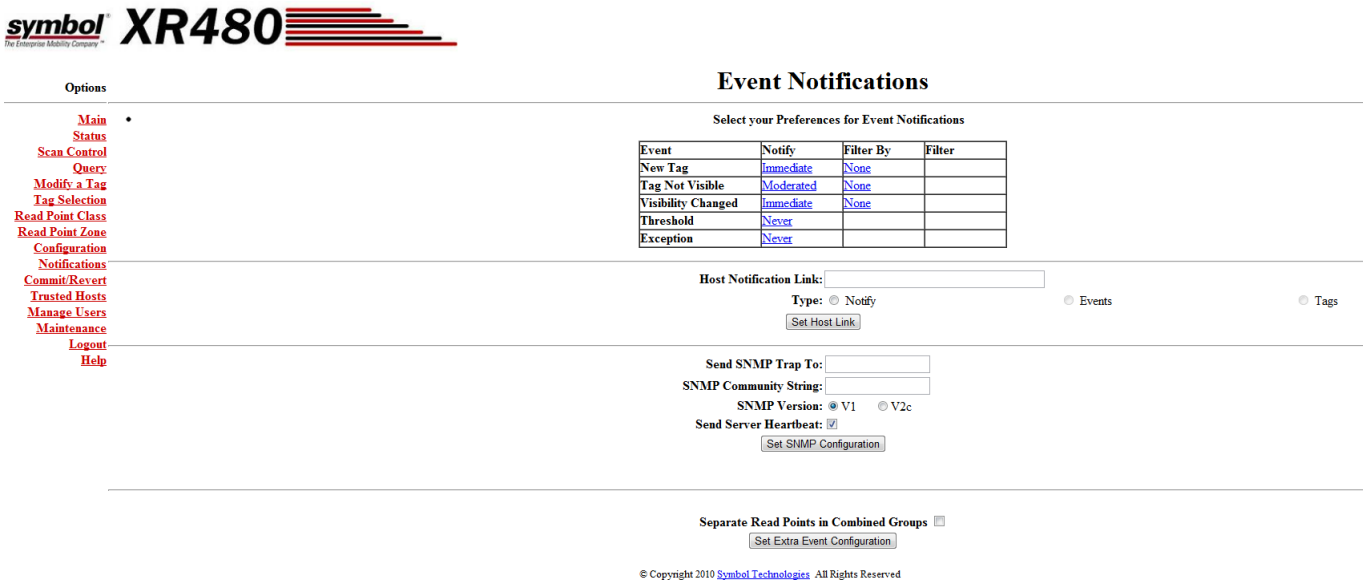


Figure 5.14: Notifications window

5.2.3.11 Commit/Revert

This window is for apply reader configuration modifications. For each modification in the configuration that you make, it is necessary to pass through this window and click to commit. Once this is done after some modification, the asterisk next to the Commit/Revert in left part of the screen, disappears.

Figure 5.15: Commit/Revers window

5.2.3.12 Trusted hosts

Be sure to specify here the IP of the host computer where is running Tomcat and ALE Server.

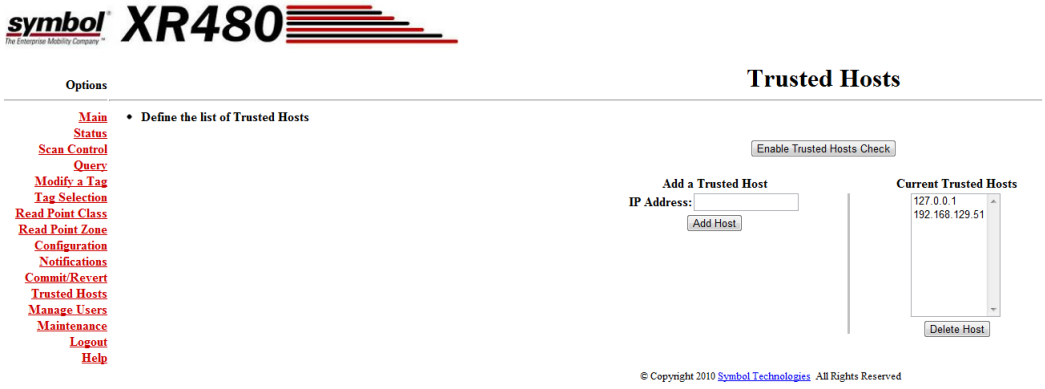


Figure 5.16: Trusted hosts window

5.2.3.13 Manage users

To crate and manage different users for the reader Administration Console. It is necessary that appears the admin user.

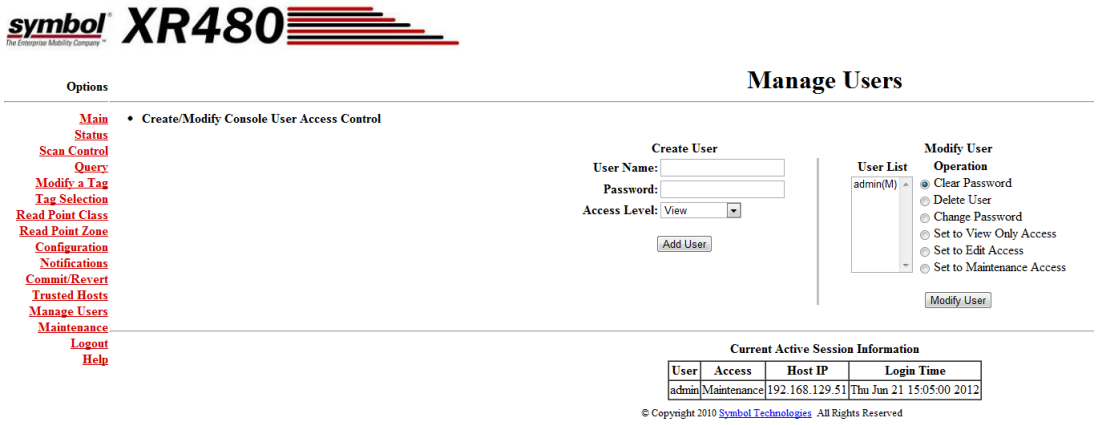


Figure 5.17: Manage users window

5.2.3.14 Maintenance

This will open a new screen with all the options for change configuration settings: Reader Maintenance Console. By changing this parameters the reader communication with the host will be configured.

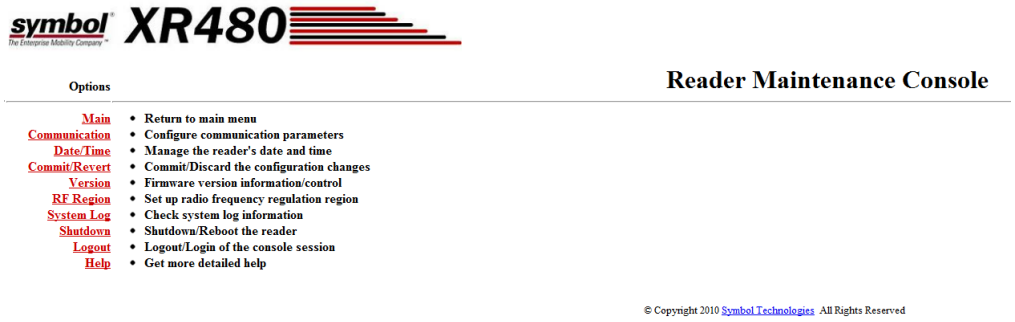


Figure 5.18: Maintenance window

Communication This is the most important parameters to be properly configured for having a good LLRP communication with the host. Set the parameters as appears in the next screen shot. Take in account that for all the reader configuration LLRP option must be deactivated, and activate it at the end.

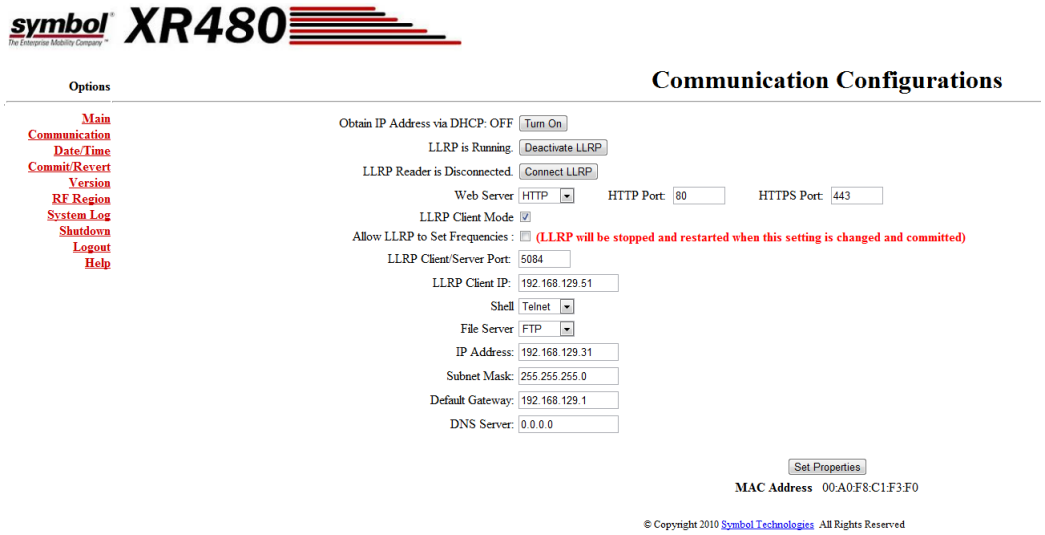


Figure 5.19: Communication window

Date/Time This window configuration is important in order to have correctly set the detection time of the tags on the reports.

Version This window displays the current firmware version and allows upgrading to new firmware. We don't have to configure anything in this window.

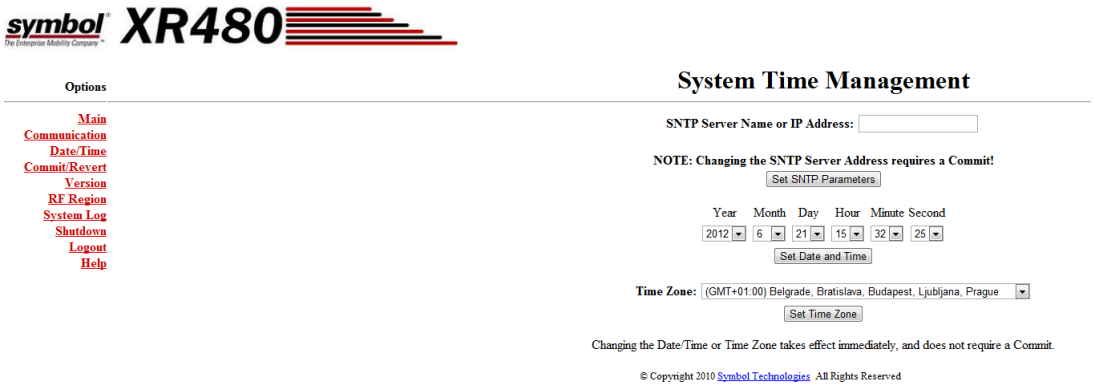


Figure 5.20: System Time Management window

RF Region This window is for the region control. To set here the current region where the reader is being used is very important for the RFID frequencies uses control.

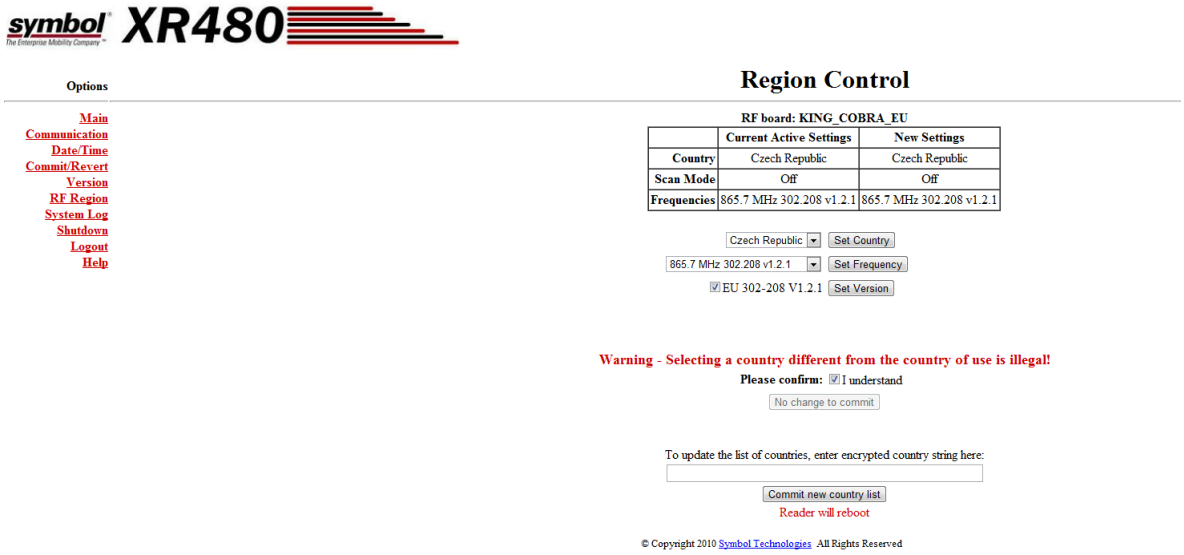


Figure 5.21: Region Control window

System Log This is a very useful window to check if there is some problem during the configuration process. It displays the log with all the actions done during configuration.

5.2.3.15 Log out

Before closing the browser, log out of the console. Click Logout from either the maintenance menu or the main menu. If unsaved changes are pending, the Commit / Revert window

appears.

If the browser is left idle (for 15 minutes) the session automatically logs out.

5.2.4 Set up ALE Server and capturing the data

For the Filtering and Collection role we need to connect an ALE Server to our reader. There are several options of open-source software that implements this functionality.We will work with ALE Middleware implemented by fosstrak because it cover the EPCglobal ALE 1.1 Specification and it approaches us to our target of to be as close as possible of the EPCglobal standards. Therefore, our reader supports LLRP and this is a important capability that facilitate to work with fosstrak. A part of this, if is not possible to work with LLRP reader, it is good to know that there is other options to work with RFID technology and EPCIS repository, as logicAlloy ALE Server.

To make the tests with ALE Servers, we will use a RFID reader simulator called Rifidi Emulator. The option of develop first the test with a simulator, will help to filter first all possible issues with the server services and once everything on the simulation works, connect the real reader changing the appropriate values on files and on the LLRP commander adaptor.

5.2.4.1 OPTION 1: Fosstrak ALE Middleware with LLRP Commander Support

Fosstrak ALE middleware is the option that Fosstrak implements and offers to all the developers community who like to work with EPCglobal. As well as all the options are opened for every single user by interact with the code, Fosstrak deliver for initial users an implemented platform what contains one module for the server, one module for a client to configure the server, one module what works as adaptor for any kind of possible RFID readers and one designable module to implement a personalized capture application. See section 3.4 for more information about.

As our Reader supports LLRP, we can use Fosstrak LLRP Commander, what will help us a lot with the interaction with our RFID reader. It will allow us to add as readers as we wish in the future without have to do important changes.

Firstly set up the reader simulator. Download the software from the webside [37] and install the application. Then, create a new LLRP reader following the steps by the windows appears up (see figure 5.22):

1. Add a new reader.

2. Select reader type: LLRPReader.

3. Enter a name, select number of antennas and configure port numbers: LLRP Administration IP Address and LLRP Connection IP Address.

4. Select the wished GPI/O.

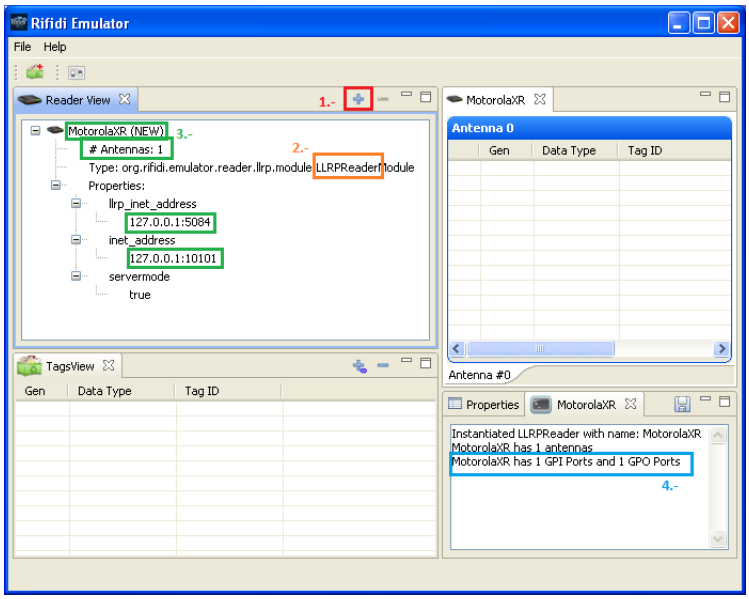


Figure 5.22: Create a Reader simulation with Rifidi Emulator.

Ones the reader is created and it simulates our Motorola Symbol XR 480 reader (LLRP enabled), it has to be abstracted as an LogicalReader in order end-user can manage it through the Reader Interface supporting LLRP standard. Define this logical reader is part of the Filtering and collection ALE Server, as well as to create an adaptor that can connect to the physical reader. This adaptor is for Fosstrak a simple bridge to the reader management module of the Fosstrak LLRP Commander.

Logical Reader Logical Reader is created by the server depending on a LRSpec.xml document that client sends to it through Logical Reader API with command *define(string 3, LRSpec spec)*. It must be specified: (see figure 5.23)

- The Logical Reader name.
- The physical reader associated name.
- The IP and port number from where is the physical reader connected.
- The properties that is wished that it has defined.

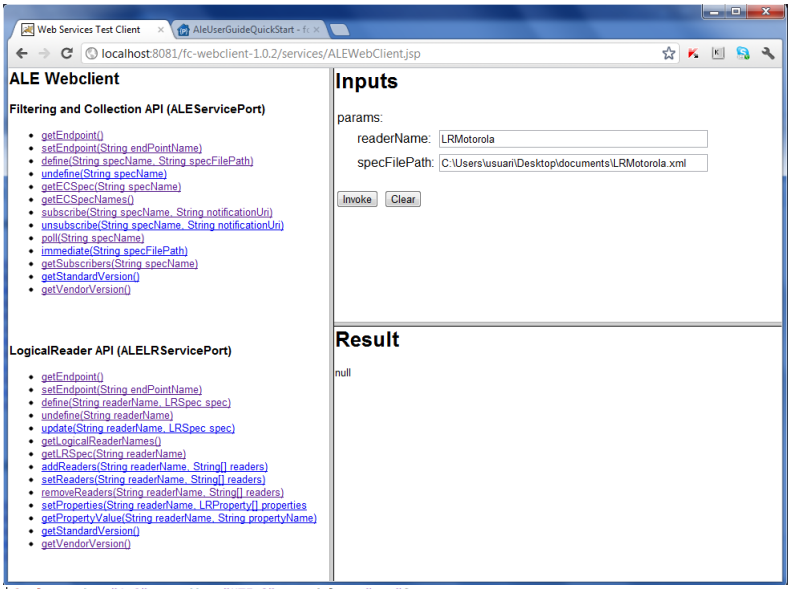
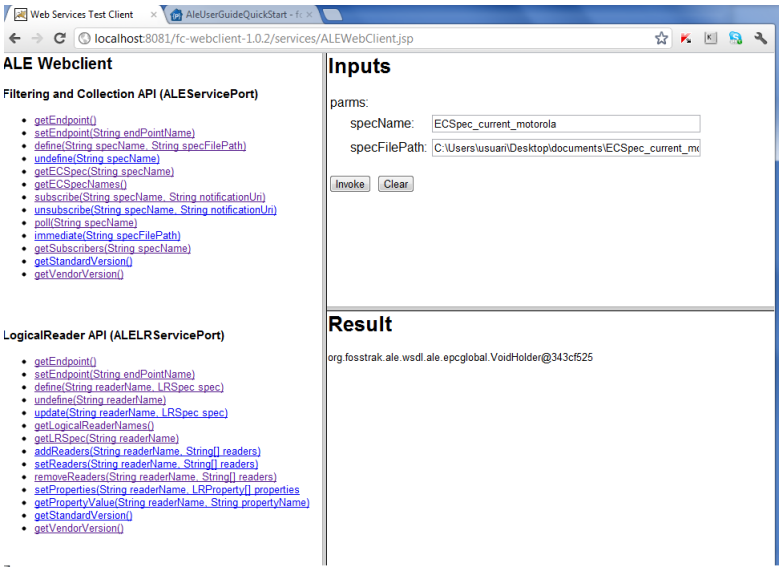


Figure 5.23: LRSpec.xml

*For the real implementation: ip= *192.168.129.31* (reader ip on the LAN network); PhysicalReaderName= *cmb* (reader name on the internal reader configuration)

Event Cycle One Event Cycle must be specified per each logical reader that we have. The ECSpec says to the server how to ask the reader for the information. This information is sent for the client to the server through Filtering and Collection API with command *define(string specName, string specFilePath)*. It must be specified: (see figure)

- The Logical Reader that applies.
- The period of repetitions time and the duration of each.
- The name of the report and what is wished that the report includes.




```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ECSpec xmlns:ns2="urn:epcglobal:ale:xsd:1">
    <logicalReaders>
        <logicalReader>MotorolaXR</logicalReader>
    </logicalReaders>
    <boundarySpec>
       <repeatPeriod unit="MS">10000</repeatPeriod>
        <duration unit="MS">9500</duration>
        <stableSetInterval unit="MS">0</stableSetInterval>
    </boundarySpec>
    <reportSpecs>
        <reportSpec reportName="SubscribeSymbol Report">
            <reportSet set="CURRENT"/>
            <output includeRawHex="true" includeRawDecimal="true"
includeEPC="true" includeTag="true"/>
        </reportSpec>
    </reportSpecs>
</ns2:ECSpec>
```

Figure 5.24: Event Cycle Spec.xml

Subscriber As we know, the Event Cycle will not be launched if there is not a subscriber that ask for it, so the next step is define a subscriber. This is done launching the command *subscribe(string specName, string NotificationURI)*, where the specName is the name o the ECSpec that you define for your Logical Reader and notification URI is the address through the ALE Reports will be displayed. Before than that, we can execute the EventSinkUI as is showed at section 3.4.4.4to see immediately if it is working properly. If everything is ok, the window should show periodically empty reports (or without any information, only the name of the ECSpec and time,...).

LLRP Commander At this point the adaptor for the Logical Reader is already created for the server and we can see if the physical reader is connected to it by LLRP interface with LLRP commander application. Next step then is to open Eclipse and LLRP_CMD perspective and send a ROSpec.llrp file created before with the information that the report must contain and how it must be displayed. There are 2 works that should be done before: create the .llrp file and copy the xml code delivered for fosstrak and create a new adaptor instance called Server Adaptor -127.0.0.1. If everything is working fine at this point should appear on that adaptor a new Logical Reader called with the name that we specified inside LRSpec running. After check this, follow next steps to send properly the ROSpec to the reader (see figure 5.25):

- Send Get Reader Capabilities LLRP message.
- Send ROSpec LLRP generated message.
- Send ENABLE ROSpec LLRP message.

Once the ROSpec is sent to the reader, as soon as you add a new tag inside the antenna simulator, the reader starts generating ROSpec and we can see, in one side that LLRP Commander is receiving this as well, and in the other side, that ALE Server is displaying ALE Reports with tag information through the port that was defined as subscriber and it is showed by EventSinkUI windows.

> **Observation:** Before all the steps for set up ALE Server, should be created a server adaptor at LLRP interface as it is showed at figure 5.27. Then, close the eclipse and open it again after all the steps. The server adaptor will include the logical reader created running.

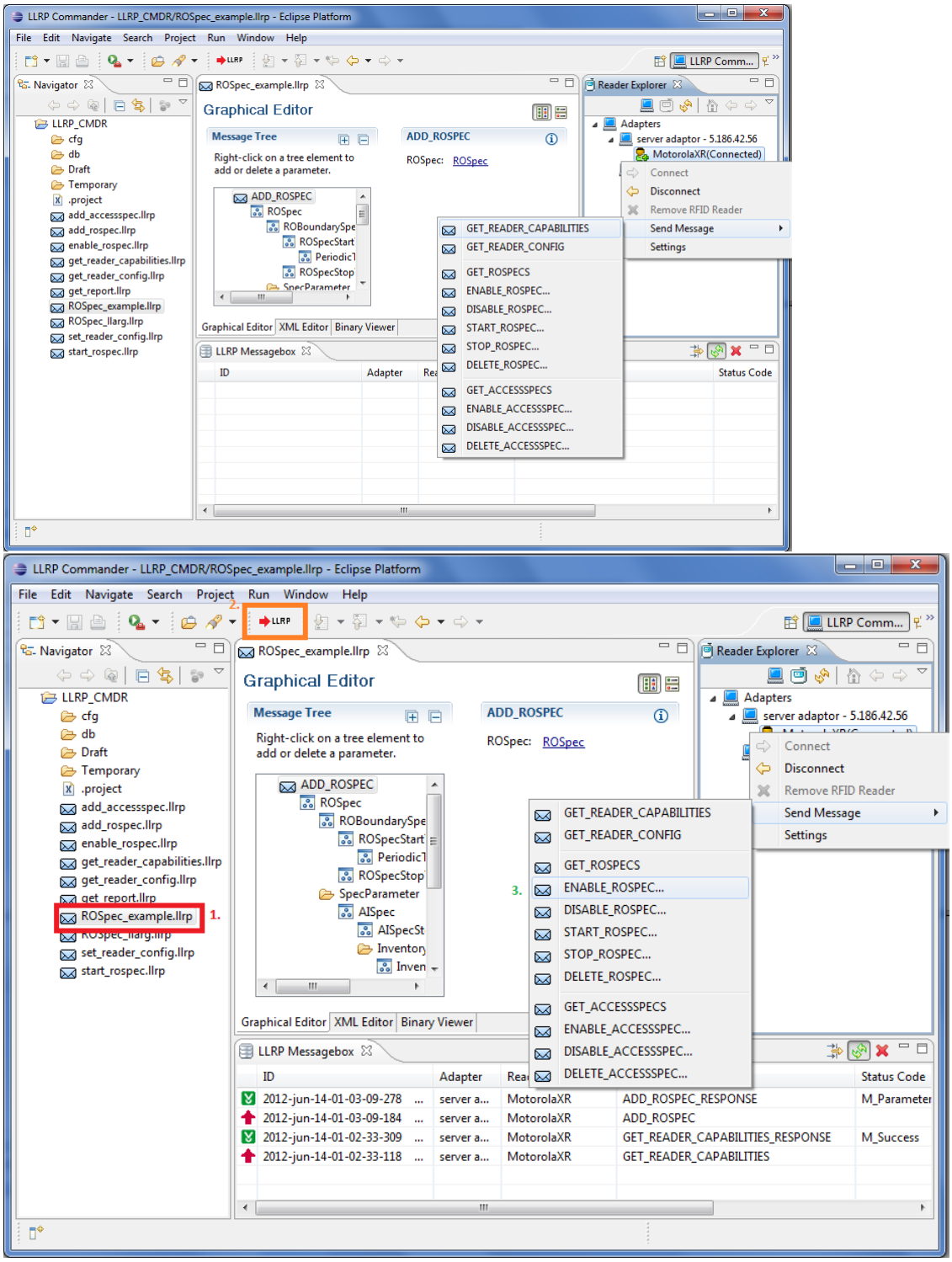


Figure 5.25: Send ROSpec to the reader

```
<?xml version="1.0" encoding="UTF-8"?>
<llrp:ADD_ROSPEC xmlns:llrp="http://www.llrp.org/ltk/schema/core/encoding/xml/1.0"
Version="1" MessageID="0">
<llrp:ROSpec>
<llrp:ROSpecID>1</llrp:ROSpecID>
<llrp:Priority>0</llrp:Priority>
<llrp:CurrentState>Disabled</llrp:CurrentState>
<llrp:ROBoundarySpec>
<llrp:ROSpecStartTrigger>
<llrp:ROSpecStartTriggerType>Periodic</llrp:ROSpecStartTriggerType>
<llrp:PeriodicTriggerValue>
<llrp:Offset>0</llrp:Offset>
<llrp:Period>5000</llrp:Period>
</llrp:PeriodicTriggerValue>
</llrp:ROSpecStartTrigger>
<llrp:ROSpecStopTrigger>
<llrp:ROSpecStopTriggerType>Null</llrp:ROSpecStopTriggerType>
<llrp:DurationTriggerValue>0</llrp:DurationTriggerValue>
</llrp:ROSpecStopTrigger>
</llrp:ROBoundarySpec>
<llrp:AISpec>
<llrp:AntennaIDs>0</llrp:AntennaIDs>
<llrp:AISpecStopTrigger>
<llrp:AISpecStopTriggerType>Null</llrp:AISpecStopTriggerType>
<llrp:DurationTrigger>0</llrp:DurationTrigger>
</llrp:AISpecStopTrigger>
<llrp:InventoryParameterSpec>
<llrp:InventoryParameterSpecID>1</llrp:InventoryParameterSpecID>
<llrp:ProtocolID>Unspecified</llrp:ProtocolID>
</llrp:InventoryParameterSpec>
</llrp:AISpec>
<llrp:ROReportSpec>
<llrp:ROReportTrigger>Upon_N_Tags_Or_End_Of_ROSpec</llrp:ROReportTrigger>
<llrp:N>1</llrp:N>
<llrp:TagReportContentSelector>
<llrp:EnableROSpecID>1</llrp:EnableROSpecID>
<llrp:EnableSpecIndex>1</llrp:EnableSpecIndex>
<llrp:EnableInventoryParameterSpecID>1</llrp:EnableInventoryParameterSpecID>
<llrp:EnableAntennaID>1</llrp:EnableAntennaID>
<llrp:EnableChannelIndex>1</llrp:EnableChannelIndex>
<llrp:EnablePeakRSSI>1</llrp:EnablePeakRSSI>
<llrp:EnableFirstSeenTimestamp>1</llrp:EnableFirstSeenTimestamp>
<llrp:EnableLastSeenTimestamp>1</llrp:EnableLastSeenTimestamp>
<llrp:EnableTagSeenCount>1</llrp:EnableTagSeenCount>
<llrp:EnableAccessSpecID>1</llrp:EnableAccessSpecID>
<llrp:C1G2EPCMemorySelector>
<llrp:EnableCRC>1</llrp:EnableCRC>
<llrp:EnablePCBits>1</llrp:EnablePCBits>
</llrp:C1G2EPCMemorySelector>
</llrp:TagReportContentSelector>
</llrp:ROReportSpec>
</llrp:ROSpec>
</llrp:ADD_ROSPEC>
```

Figure 5.26: Example of ROSpec LLRP message provided by Fosstrak.

Figure 5.28: Tag information reporting

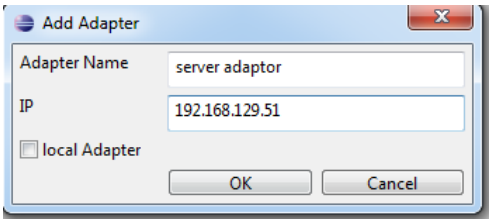


Figure 5.27: Create a server adaptor

Tag Information report At the end, the tag information is reported through the subscriber port (9999) and is showed for us EventSink window as is showed at figure 5.28

Capturing Application In this point, the next step is take the ALEReports from the notificationURI port (for us 9999) and convert them to a valid EPCIS Events. This is the most complicate part to implement because it depends on the internal administration and management of a company that has to decide which fields are interesting for them the EPCISEvent has contain.

Fosstrak does not deliver a feature for the capture application as it does for the rest of modules because this question. Instead, there is a package of documents that allow the

user to simulate an hypothetical supply chain example only for get involved on expected behavior of the capturing application. This work is not covering this part as is needed a developer work and it is out the scope. Please, refer to section below to see the option that Fosstrak offers for implement an own capture application.

5.2.4.2 OPTION 2: logicAlloy ALE Server

logicAlloy develops open source software engines that drive RFID (Radio Frequency Identification) applications. As an open source application with a zero-cost license, ALE Server gives the user the freedom and flexibility to use the software for no upfront cost. In addition, logicAlloy makes RFID easy by bundling a high-performance engine with a package of easy to use management tools including a web-based administration console and a SOAP API. [36]

It is fully compliant with EPCglobal's ALE Specification and is fully interoperable with the other components of the EPCglobal Network including EPCglobal's Tag Data Standards. Therefore, allows to "subscribe" to RFID data that ALE Server collects. Interested systems can tell ALE Server exactly what RFID tags they are interested in and how they want the data grouped and summarized.

ALE Server comes with a web-based administration console that's simple and easy to use. The administration console can configure every aspect of ALE Server and requires no special IT skills, only a basic understanding of Event Cycle Specifications and Reports.

First of all go to the logicAlloy webside [36]and download and install the ALE Server application. During the installation may be specified the Web Server Port. Ones it is installed, configuration of each part of the filtering and collection process is done only by Editing the screens through the Web Management Console (*http://localhost:8080/admin*): definition of LogicalReader, create the connection between Server and Reader and specification of EventCycle and subscribers. First step is go to Server/ Settings and define the name of the server (for example *urn:myco:aleserver:12345.67890*). To run the Server go to Services folder into your host and start logicAlloy Server.

Logical Reader After create an Alient 9-Series reader with Rifidi simulator and configure the Server Settings, configure the Logical Reader and the EPC Device as it is showed at figure 5.29. Server/Plugins folder contains all the plugins for readers that can be working with logicAlloy. Actually this is one the most important limitations of the Server, that can only work with specific readers.

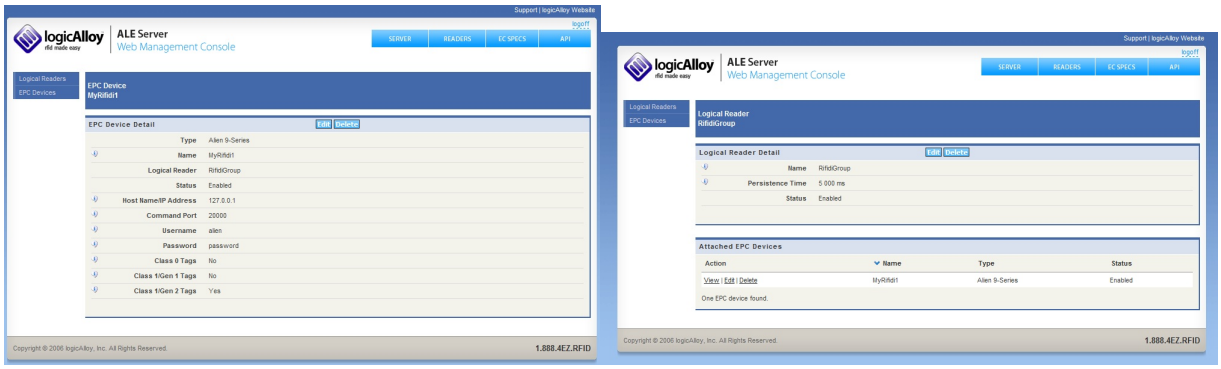


Figure 5.29: logicAlloy Reader Configuration

Event Cycle Next step is define the EventCycle specifications and add a subscriber to get the data from the reader. Here we have to specify the database subscriber, and add the direction of the capture API that we have connected to our repository, that we just create. It is possible to choose belong three types of Report Set: CURRENT, ADDITIONS or DELETIONS. See figure 5.30.

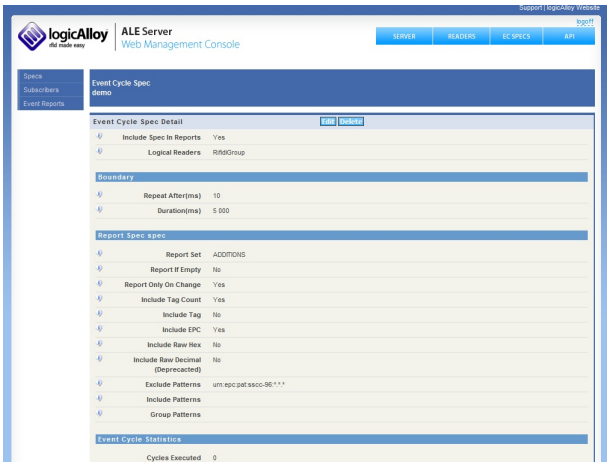


Figure 5.30: logicAlloy Event Cycle Configuration

Event Reports If everything work fine, when the configuration is done, the Rifidi reader simulator is connected to the server and, when we add a new tag inside the Antenna simulator, it will start reporting ECReports to the ALE Server. Go to EC Specs/Event Reports history to see the reports that are being generated.

Capturing data to EPCIS repository The data capturing and storing into the database is done automatically when the configuration had been done correctly. Tag information

from the reader has been filtered and collected by the ALE Server and included into a EPCIS standard structure that will be sent to the epcis-repository through the Tomcat servlet. Then the EPCIS Event will be included into the database and accessible for a query.

This process is easy and fast with logicAlloy, addressed to the companies who like to work with RFID technology and adapt their system into a EPCglobal standards, but this system is missing a lot of information on the way. It is recorded only fields: eventTime, recordTime, action and bizLocation (name of server).

Query EPCIS repository To check if the data has been aggregated correctly into the repository we have Fosstrak Query Java Application presented on section 3.2.2. We can define the query by the recordTime for example. We will receive the results through Query Results window or through a debug window if we have checked before "*Show debug window*" option.

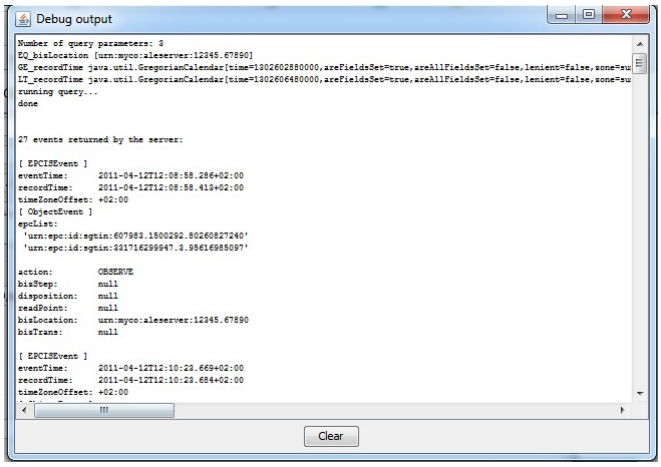


Figure 5.31: query for data from ALE Server stored into EPCIS- repository

5.2.5 Fosstrak Capture Application

The part of Capture Application is the most complicate one to implement not only because is complicate the procedure to change the format of a document from one standard to another one, as well as because is depending what each company wants to implement, how and they own vocabulary. It is exposed in the Background part that this is one of the big problems on the standardization process because there is not EPCglobal standard yet that covers the vocabularies part.

Due to this limitation, Fosstrak chooses the option of personal implementation for it. This means to get a developer role to interact with the code. Fosstrak presents 2 different

options to implement the package of rules that the capture application event generation must follows to fit with the company requirements: one is the implementation of set of rules with JBoss Drools and the other one to program directly by code all the instructions to get each part of the ALE Report and add to another part of the output EPCISEvent. As an example of second solution delivered by fosstrak see the figure **??**. Refer to fosstrak webside to find all the information about all the necessary documents and libraries to make a code source of capture application.

For the first solution it is needed to implement a set of JBoss rules (Drools), as the way to define any rule. To understand better what is this:

- **Drools** is a business rule management system (BRMS) with a forward chaining inference based rules engine, using an enhanced implementation of the Rete algorithm. It is written in Java, but able to run on Java and .NET and provides for declarative programming and is flexible enough to match the semantics of the problem domain with Domain Specific Languages (DSL) via XML using a schema defined for the problem domain. DSLs consist of XML elements and attributes that represent the problem domain.

- **JBoss rules** is a software system that executes one or more business rules in a runtime production environment and includes a forward chaining rule engine based on Drools. Actually, the JBoss Rules in the productised version of Drools.

Why this is useful? because the use of Drools offers the possibility to react more quickly and flexibly to modifications to be updated. Modifications to the source code of the wrappers with subsequent recompilation and deployment con be avoided in this way.

To implement the set of JBoss rules refer to the webside manuals[40]. There it can be found a first introduction of how to use it, as well as a more details to how write a rule files from the beginning.

The main idea is that if is wished to implement our own set of rules we have two options: starting from drools eclipse project or by using maven source (as fosstrak developer part suggest). Both of them will need to pass sometime through eclipse to compile the whole project.

The first thing that can be done is install into eclipse GEF (Eclipse Graphical Editing Framework) for Rule Workbench. Then install Drools plug-in unzipping the downloaded file in your main eclipse folder. Is easy by following the instructions in the webside manuals because the plug-in is inside the Eclipse available software.

Second thing to do is create a Drools runtime, what is a collection of jars on your file system that represents one specific release of the Drools project jars. This means that each Drools project must contain all or some of (depending on the difficulty of the project)

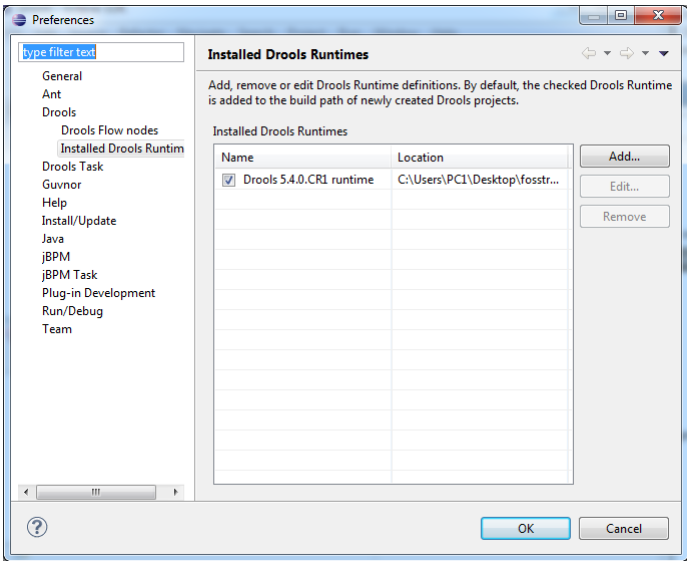


Figure 5.32: drools_runtime

these JAR fails and different releases are done by modifications on them. They work as the libraries for a drools project (see figure 5.32).

Finally you can create a drools project directly or by importing a java project and work with it with the Eclipse Drools perspective. If contrarily you made the code with maven, then you have to built the sources and import for eclipse project and link project dependencies by m2_repo classpath variable.

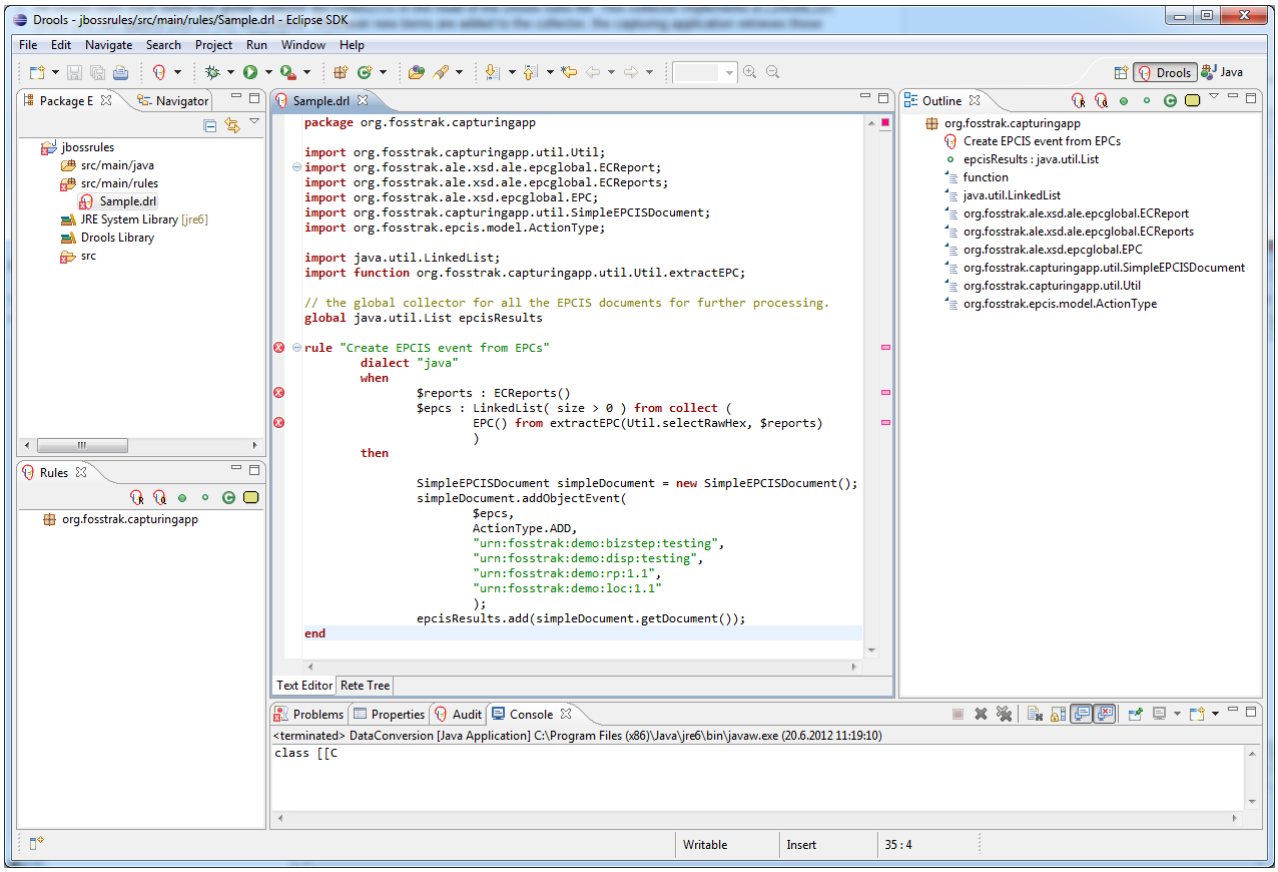


Figure 5.33: Eclipse Drools GEF with SimpleEPCISDocument.drl

Figure 5.34: Example of Capture Application Proposed by Fosstrak [18]

```
    public class CaptureApp {

        private int port;

        private String epcisRepository;

        private CaptureClient client = null;

        public CaptureApp(int port, String epcisRepository) {
                this.port = port;
                this.epcisRepository = epcisRepository;
        }

        private void handleReports(ECReports reports) throws IOException, JAXBException {
                System.out.println("Handling incomming reports");

                List<ECReport> theReports = reports.getReports().getReport();
                // collect all the tags
                List<EPC> epcs = new LinkedList<EPC>();
                if (theReports != null) {
                        for (ECReport report : theReports) {
                                if (report.getGroup() != null) {
                                        for (ECReportGroup group : report.getGroup()) {
                                                if (group.getGroupList() != null) {
                                                        for (ECReportGroupListMember member :
group.getGroupList().getMember()) {
                                                                if (member.getRawDecimal() != null) {

epcs.add(member.getRawDecimal());
                                                                }
                                                        }

                                                }
                                        }
                                }
                        }
                }

                if (epcs.size() == 0) {
                        System.out.println("no epc received - generating no event");
                        return;
                }

                // create the ecpis event
                ObjectEventType objEvent = new ObjectEventType();

                // get the current time and set the eventTime
                XMLGregorianCalendar now = null;
                try {
                    DatatypeFactory dataFactory = DatatypeFactory.newInstance();
                    now = dataFactory.newXMLGregorianCalendar(new GregorianCalendar());
                    objEvent.setEventTime(now);
                } catch (DatatypeConfigurationException e) {
                    e.printStackTrace();
                }

                // get the current time zone and set the eventTimeZoneOffset
                if (now != null) {
                    int timezone = now.getTimezone();
                    int h = Math.abs(timezone / 60);
                    int m = Math.abs(timezone % 60);
                    DecimalFormat format = new DecimalFormat("00");
                    String sign = (timezone < 0) ? "-" : "+";
                    objEvent.setEventTimeZoneOffset(sign + format.format(h) + ":" +
format.format(m));
                }
```

```
                EPCListType epcList = new EPCListType();
                // add the epcs
                for (EPC epc : epcs) {
                        org.fosstrak.epcis.model.EPC nepc = new org.fosstrak.epcis.model.EPC();
                        nepc.setValue(epc.getValue());
                        epcList.getEpc().add(nepc);
                }
                objEvent.setEpcList(epcList);

                // set action
                objEvent.setAction(ActionType.ADD);

                // set bizStep
                objEvent.setBizStep("urn:fosstrak:demo:bizstep:testing");

                // set disposition
                objEvent.setDisposition("urn:fosstrak:demo:disp:testing");

                // set readPoint
                ReadPointType readPoint = new ReadPointType();
                readPoint.setId("urn:fosstrak:demo:rp:1.1");
                objEvent.setReadPoint(readPoint);

                // set bizLocation
                BusinessLocationType bizLocation = new BusinessLocationType();
                bizLocation.setId("urn:fosstrak:demo:loc:1.1");
                objEvent.setBizLocation(bizLocation);

                // create the EPCISDocument containing a single ObjectEvent
                EPCISDocumentType epcisDoc = new EPCISDocumentType();
                EPCISBodyType epcisBody = new EPCISBodyType();
                EventListType eventList = new EventListType();
                eventList.getObjectEventOrAggregationEventOrQuantityEvent().add(objEvent);
                epcisBody.setEventList(eventList);
                epcisDoc.setEPCISBody(epcisBody);
                epcisDoc.setSchemaVersion(new BigDecimal("1.0"));
                epcisDoc.setCreationDate(now);

                int httpResponseCode = client.capture(epcisDoc);
                if (httpResponseCode != 200) {
                    System.out.println("The event could NOT be captured!");
                }
        }

        public void run() {
                client = new CaptureClient(epcisRepository);

                ServerSocket ss = null;
                try {
                        ss = new ServerSocket(port);
                        while(true) {
                                try {
                                        Socket s = ss.accept();
                                        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));


                                       String data = in.readLine();
                                        String buf = "";
                                        // ignore the http header
                                        data = in.readLine();
                                        data = in.readLine();
                                        data = in.readLine();
                                        data = in.readLine();
```

```
                                        while (data != null) {
                                                buf += data;
                                                data = in.readLine();
                                        }
                                        System.out.println(buf);

                                        // create a stream from the buf
                                        InputStream parseStream = new
ByteArrayInputStream(buf.getBytes());

                                        // parse the string

                                        ECReports reports =
DeserializerUtil.deserializeECReports(parseStream);
                                        if (reports != null) {
                                                handleReports(reports);
                                        }
                                } catch (Exception e) {
                                        System.out.println("ERROR: " + e.getMessage());
                                }
                        }
                } catch (IOException e1) {
                    System.out.println(e1.getMessage());
                }
        }

        public static void help() {
                System.out.println("You need to specify the port where to listen and the
url of the epcis repository");
                System.out.println("Example: ");
        }

        /**
         * starts the CaptureApp.
         *
         * @param args the first command line parameter is the TCP port. if omitted port
9999 is used.
         */
        public static void main(String[] args) {
        CaptureApp client;
        int port;
        String epcisRepository;
        // check if args[0] is tcp-port
        // and args[1] is epcis repository
        if (args.length == 2){
                port = Integer.parseInt(args[0]);
                epcisRepository = args[1];
            client = new CaptureApp(port, epcisRepository);

        } else {
                help();
                return;
        }

        client.run();
        }
}
```

Chapter 6

Evaluation

The main questions this work aimed to answer are: Is it possible to integrate and merge the Supply Chain Management from an organization using the Fosstrak implementation to follow EPCglobal standards? How to use it? Which advantages presents comparing some other options? And finally if could be Motorola XR480 RFID reader compatible with this integration?

6.1 Fosstrak assessment

As it is stated in the Case of Study section, one of the initial objectives was to analyze how to integrate Fosstrak in a work with RFID system, in the way of following the current tendency of adapting to the EPCglobal standards to get involved with the EPCglobal Network.

The work with Fosstrak has been resulted very agreeable, realizing that is possible to integrate a RFID internal system from a company or a investigation work with the RFID technology, only with some limitations.

The modules have been analyzed one by one from a novice user perspective, to check the accessibility for people that start learning the basics of EPCglobal without the experience of working with these kind of concepts, that in first place are abstract; and it has been observed that integrating all the Architecture Framework is possible.

The search for rigor with the standards of EPCglobal may turn the exchange of information between modules into an involved task, but in the end it becomes the potential distinctive of the tool.

As it has been exposed before, the EPCIS module repository already has the EPCglobal accreditation, and the rest of the modules are working to achieve it as well.

Fosstrak offers more tools than this document covers, as it just tries to find the basic coverage that a user needs to get the basic implementation of a RFID system. One of

this tools is the EPCIS WebAdapter Interface, a not standardized interface that adds a RESTful, fully Web-enabled API to an standard EPCIS repository. It neither implements the Fosstrak module TDT Engine, as it is not necessary with the tag that is available for the developed work, even though it is presented in chapter 3.

6.2 Fosstrak and EPCglobal standards integration

According to the standards the main integration point of any EPCglobal based solution with the business processes are the capture applications.

Fosstrak presents 2 different implementations for the EPC data capture in a database: a Java application with GUI that allows users to easily define the wished fields of an EPCIS Event and store it directly to the EPCIS repository, and besides, a set of files that are implemented inside the structure of the ALE Server module, that perform the function of turning the ALE Reports generated by the ALE Server under petition into an EPCIS Events format, and saving it to the EPCIS Repository.

Obviously the second option is the most desirable because it works automatically, but it is also the hardest option to implement due to the lack of vocabularies definition and shared structures of the standard.

This is the first limitation of the EPCIS standard that we face up to, and this limitation is making this EPCglobal Network development difficult. The second limitation is that the standard still doesn't define means to capture masterdata information into the EPCIS Repository.

In the other hand, we realized that Fosstrak ALE Server integrates almost fully EPCglobal ALE 1.1 standard, and it becomes a really good solution for the filtering and collection EPC tags.

6.2.1 The need for description of a capture application

The main differences from one capture application and another one are the business rules and particularities of the situations and contexts that must be interpreted to correctly process an ALE Report int an EPCIS Event.

From an organization's perspective, the perfect Capture Application would be one where analysts would specify the business rules and the required interactions in a standardized format, relating: readers, rules, locations, and systems. By loading these rules on the capture application, it would be able to interpret and perform the business logic automatically.

Unfortunately for companies and developers, neither the EPCglobal standards define such a format, nor is Fosstrak leading this way.

At the end, still working with Fosstrak, in any case of implementation that it offers is needed to implement a particular capture application, and no standard is applied. The implementation of a new capture application was out of the scope of this project, as there was not wished structure defined, this could be the next step.

This is absolutely the most complicate part to have working by use Fosstrak for the RFID system.

6.3 Alternatives

As a Fosstrak alternatives, there are other software interfaces that allow reading information from RFID tags and storing it in an EPCIS repository, like the ALE Server logic Alloy , or the ASPIRERFID from OW2 Consortium.

In this work the ALE Server logic Alloy is presented as an Fosstrak alternative option, and it has been stated as a conclusion that, although it achieves the milestone of tag storage inside the repository, the process to this goal is far from the compliance of the EPCglobal Standards, for example, the ALE 1.1 specification.

Another of the great limitations that these alternatives use to have is the compatibility with real readers. For example, with the logic Alloy is not possible to work with the Motorola Symbol XR480.

Also in this direction, Fosstrak offers wide possibilities, directing towards the LLRP readers support. This is the line that EPCGlobal seems to be pointing to, according to the latest standard updates. Therefore the compatibility with other readers (not supporting RP or LLRP) could be done by developing a wrapper using the Fosstrak Hardware Abstraction Layer (HAL). But this is a tough task, involving low-level programming and good knowledge of the equipments.

6.4 Our own implementation

For the implementation of the RFID system where Fosstrak ALE Middleware has to connect through Reader Interface, there have been used a RFID MotorolaSymbolXR480 reader, a Motorola AN480 RFID antenna and a EPC Class1 Gen2 test tag.

The fact that the reader supports LLRP has made it's configuration to connect with Fosstrak ALE Server much easier. If it had not supported LLRP this work would have been much more complicated, as much in the implementation of the Logical Reader Adapter as the Reader Management, because it wouldn't have been possible to use LLRP Communication.

In order to connect this system to the PC and keep having Internet access at disposal,

it has been necessary to create a LAN at the working place. We used a simple switch at the final implementation because was what we could have available, but it was considered to implement an internal LAN in order to protect the material from any external attack using a MicroTik, which allows the implementation of two different interfaces, to obtain the desired security.

6.5 Problems

Fosstrak is a platform that has been conceived and documented to become a shared project between all it's developer users, with the final objective of implementing a software suitable to be used for the control and management in a supply chain environment, following as much as possible the EPCglobal standards.

From the initial idea that started this project, to see if it was possible to use the tool by any user to adapt it to an own RFID system, this aim to be rigorous with the standards presents some problems.

First of all, the fact of working with standards that are still not fully defined usually results in the impossibility of adaptation to a complex technology as RFID. In the second place, and really outstanding, the fact of being focused for developers, from a user point of view, makes really hard to reach the insights of the entire platform way of working.

All this work of discovering the tool has been very complicated, starting with getting involved with all the required support software like Tomcat servlet, MySQL database, Drools and even maven platform, in the attempt to introduce the tool as another support for the user.

All this complexity in the form added to the initial ignorance of the whole EPCglobal institution, has implied a previous work way too extensive compared to the limited time available for the project achievement, and has minimized the time available for the experimentation with the reader and the real system.

For this reason, aspects as the implementation of a Capture Application, must have been excluded from the initial scope of the project.

Must be said, as well, that the fact of being a new tool, and the standardization for the sharing of EPC data being an emergent movement, the information at the web is scant and the search for answers is really difficult and sometimes frustrating.

Chapter 7

Conclusions

The first and most important to be mentioned is that the website change, that has made disappear www.fosstrak.org, referred during the whole work, changing it for http://code.google.com/p/fosstrak/ . According to what Fosstrak members said, the change of platform from standard url to code google is in the aim to make the code more open and more accessible for the developers that want to take active part in the project, as for facilitating the exchange of doubts and personal issues between them.

EPCglobal and their proposed Architecture Framework gives a proposition to standardize the RFID relationship between different supply chains environments and inside themselves. This current preposition will enable the companies easily to capture and exchange RFID data following a standard way.

As we have seen, the standards still don't fulfill all the possibilities of the RFID technology, and only time will tell if it will end covering them.

In any way, the force that the RFID technology is getting, coming in possible applications for the real world that would have been unimaginable a short time ago, for sure will force the acceleration of all this process standardization.

7.1 Fosstrak

- The documentation supplied by Fosstrak is still mainly aimed to developers that want to interact with the project and, therefore, already have a wide knowledge about the tool after a previous experience. This fact causes that novel users that want to make a basic use of the tool easily get lost.

- It lacks a more extensive documentation about how the different modules interact for a practical and complete implementation of the tool.

- One of the most important comments that should be taken from this thesis for the

next students involved, is the difficulty of Fosstrak integration for a business logic. Capture application was until last June 2009 out of Fosstrak scope, and even now the current implementation does not provide a trivial way to process the business logic properly with technical concepts like items and packages.

- The Capture Application part offers a simulation for the basic users that can't be used with a real reader. If one tries to use the Generic Capture Application it results complex due its strong dependence of libraries that aren't very accessible for the user as well as the need to know how to work with Drools. This causes the possible bugs to become less traceable and much harder to solve. And yet, this is a good offer to users that are not developers and want to work with Fosstrak.

- ALE server with LLRP Commander results a very useful tool, with a really complete and intuitive user interface.

- Fosstrak gives the possibility to implement an own system for the companies as is much more personalized than any offered by any other tool.

7.2 Contributions

The contributions of this work are:

- For the interested in introducing themselves in the work with RFID and EPCglobal, a good and compact biography that will save them a lot of time of reading and understanding all the EPCglobal standards and all the interactions between them.

- For the novel users of Fosstrak, this document will be useful as a guide of the basic steps to become familiar with the tool and the whole software surrounding it for its correct functioning. In addition it introduces a new tool to work in a much easier way with the database MySQL.

- For the university, this work can be used as a starting documentation for future students wanting to contribute with its effort to create a own system of data capture from RFID tags with an EPCIS repository.

7.3 Future work

This work tries to encourage new students to work deeply with this tool, that has a lot of potential, and doing it as developers, because it is the best way to be able to interact, change and adapt the tool to the own needs. As well as encourage the companies to adapt

their own effort to Fosstrak tool because it will serve themselves to join these movement that is starting to emerge from the RFID technology with the Internet of Things.

Bibliography

[1] Stephan Haller; Stamatis Karnouskos; Christoph Schroth. ***The internet of things in an enterprise context. Technical report, SAP Research CEC***, http://www.springerlink.com/content/p2j04gn134021nk4/, 2009.

[2] URL: http://2.bp.blogspot.com/_hQreHX_aOfc/TKbuJB8-HkI/AAAAAAAAABM/9ppWmJ1kqwg/s1600/untitled1.bmp.

[3] EPC global GS1, http://www.gs1.org/epcglobal.

[4] EPC competence centre de AECOC, 2010. http://www.epcglobalsp.org.

[5] URL: http://www.ubiu.co.kr/eng/content/sol.asp?code=14.

[6] Synchronize your supply chain with rfid. Technical report, Motorola, http://www.motorola.com/web/Business/Products/RFID/RFID.

[7] Fosstrak webside. Free and open source software for track and trace. http://www.fosstrak.org

[8] Quick introduction to rfid. http://www.polygait.calpoly.edu//tutorial.htm.

[9] Dipole. Rfid readers. http://www.lectoresrfid.com/Lectores_RFID/Lectores_RFID_Clasificacion.html

[10] The RF in RFID: physical layer operation of passive UHF tags and Readers. 4.RFID Frequency Bands, Daniel M. Dobkin, October 2005. http://www.enigmatic-consulting.com/Communications_articles/RFID/RFID_frequencies.html

[11] URL: http://www.scansource.eu/en/education.htm?eid=8&elang=en

[12] EPCglobal®. The EPCglobal Architecture Framework version 1.4. Technical report, EPC- global®, December 2010.

[13] EPCglobal®. The EPCglobal Architecture Framework version 1.2. Technical report, EPC- global®, September 2007.

[14] EPCglobal US. http://www.epcglobalus.org.

[15] EPCglobal US. http://www.epcglobalus.org/ContactUs/GetanEPCManagerNumber/tabid/270/Default.aspx.

[16] EPCglobal ®. EPCtm radio-frequency identity protocols Class-1 Generation-2 UHF RFID Protocol for communications at 860 mhz – 960 mhz version 1.2.0. Technical report, EPCglobal Inc, October 2008.

[17] EPCglobal Inc. Low level reader protocol (LLRP), version 1.1. Technical report, EPCglobal ®, October 2010.

[18] GS1. http://www.gs1tw.org/twct/web/EPC/images/ALE-1.gif.

[19] EPCglobal Inc. The application level events (ALE) Specification, version 1.1.1, Part I: Core specification. Technical report, EPCglobal ®, March 2009.

[20] EPCglobal ®. EPC tag data Standard version 1.5. Technical report, EPCglobal ®, 2010.

[21] EPCglobal Inc. Gs1 epcglobal tag data translation (TDT) 1.6. Technical report, EPCglobal ®, October 2011.

[22] EPCglobal Inc. EPCglobal object name service (ONS) version 1.0.1. Technical report, EPCglobal®, May 2008.

[23] EPCGlobal. EPC information services (EPCIS) version 1.0.1 Specification. Technical report, EPCglobal Inc, September 2007.

[24] RFID Jornal. Fosstrak releases open-source LLRP software, http://www.rfidjournal.com/article/view/4968, June 2009.

[25] Fosstrak webside. LLRPbackground, https://code.google.com/p/fosstrak/wiki/LlrpBackground.

[26] Fosstrak LLRP Commander, https://code.google.com/p/fosstrak/wiki/LlrpMain.

[27] Fosstrak. Fosstrak ALEmain, https://code.google.com/p/fosstrak/wiki/AleMain.

[28] Fosstrak. Ale 1.1 Specification covered, https://code.google.com/p/fosstrak/wiki/AleFeatures.

[29] Fosstrak. Event cycle, https://code.google.com/p/fosstrak/wiki/AleDevGuideEventCycle .

[30] Fosstrak. ALE Developers guide Reports Notifications, https://code.google.com/p/fosstrak/wiki/AleDevGuideReportsNotification.

[31] Fosstrak. ALE Capturing Appliation Quick Start, https://code.google.com/p/fosstrak/wiki/AleCapturingAppQuickStart .

[32] XR Series RFID Readers Integration Guide. Motorola Solutions, December 2008.

[33] Mysql- the world's most popular open source database, http://www.mysql.com/.

[34] Apache tomcat, http://tomcat.apache.org.

[35] LogicAlloy RFID made easy, http://www.logicalloy.com/.

[36] Rifidi emulator software, http://www.rifidi.org/download.html.

[37] HOW TO: Create an open source EPCglobal Network. http://www.behrend.psu.edu/outreach/rfid/howto/HowTo.html.

[38] Motorola. Motorola support software downloads, http://support.symbol.com/support/product/softwaredownloads.do.

[39] Drools Introduction and General User Guide. jboss.org, http://docs.jboss.org/drools/release/5.4.0.Final/droolsjbpm-introduction- docs/html/index.html.

[40] Fosstrak webside. Capturing Application: Developer Guide, https://code.google.com/p/fosstrak/wiki/AleCapturingAppDevGuide.

[41] Fosstrak Blog. Open Source Software for Track & Trace, http://fosstrak.wordpress.com/.

[42] Republic Polytechnic, Supply Chain Management TDC, http://www.rp.edu.sg/tdc/scm/RFID%20for%20Supply%20Chain%20Mgmt.html

[43] Guilherme Pereira , ***Assessment of an open-source, standards-based RFID supply chain integration***, Instituto Superior Técnico, Universidad Técnica de Lisboa, Phd Thesis, October, 2009.

[44] Aysegul SARAC, Nabil ABSI, St ´ephane DAUZERE-PERES**, A literature review on the impact of RFID technologies on supply chain Management**, Phd Thesis, 2009.

[45] Thorsten Blecker; George Q. Huang, ***RFID in Operations and Supply Chain Management,*** Erich Schmidt Verlang GmbH & Co., 2008.

[46] EPCglobal Inc. The application level events (ALE) Specification, version 1.1.1, Part I: Core specification. Technical report, EPCglobal ®, March 2009.