AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Data Format for RFID Data Using a Data-on-Tag Approach

Sarita Pais
(Student ID: 0472377)

**A thesis
submitted in partial fulfilment of
Master of Computer and Information Sciences Degree
at the
Auckland University of Technology,
Auckland**

November 2010

Primary Supervisor: Dr. Judith Symonds

Acknowledgements

I thank my supervisor Dr. Judith Symonds for giving me the opportunity to do research on RFID technology. Dr Symonds has been a tremendous help and assistance in my thesis research over the last two years. I particularly appreciate her help in sourcing the appropriate tags for my research experiments.

I am grateful to Dr. Russell Pears, Dr. Andy Connor and Jim Buchan, my lecturers who guided me on the proper approach to academic research.

I appreciate the help provided by Grant Hugh and Rebecca Rowley from Tracient Technologies Ltd., Christchurch and Lyndon Baird from Intermec Technologies for providing me the RFID reader, tags and SDK to carry out my experiment; Dr. Nicholas Nicoloudis, Intermec Technologies for guidance on memory blocks and user memory.

Last but not least, I acknowledge the co-operation, patience and understanding of my family, friends and colleagues during my absence from work and home.

Table of Contents

Table of Figures

Table of Tables

Acronyms

API	Application Programming Interface
CML	Chemical Markup Language (CML)
CMOS	Complementary Metal Oxide Semiconductor
COTS	Commercial off the Shelf
CSV	Comma Separated Values
EDI	Electronic Data Interchange
EPC	Electronic Product Code
EEPROM	Electronically Erasable Programmable Read Only Memory
EPCIS	Electronic Product Code Information Service
FRAM	Ferro Magnetic Random Access Memory
HF	High Frequency
IC	Integrated Circuit
IEC	International Electrotechnical Commission
ISO	International Standard Organisation
LF	Low Frequency
LZW	Lempel–Ziv–Welch
PC	Personal Computer
PDA	Personal Digital Assistant
RFID	Radio Frequency Identification
SCMS	Supply Chain Management System
SD	Secure Digital
SDK	Software Development Kit
TID	Tag Serial Identification
UHF	Ultra High Frequency
UM	User Memory
XML	Extensible Mark-up Language
W3C	World Wide Web Consortium

Glossary

Air Interface Protocol: The process by which RFID tags and readers communicate.

Aggregation (Also known as containment): A hierarchical relationship to determine what physical objects are contained within another larger object.

API: Interface for the interaction with a set of functions used by components of a software system.

Chemical Markup Language (CML): An approach to manage molecular data using XML and Java.

Data-on-network Systems: Systems that specify minimal information on a tag, i.e. a unique identification, indexed to look up more details from a database via the enterprise network.

Data-on-tag Systems: Systems where complete data of the object is saved on the tag.

Electronic Product Code (EPC): Unique identifier for an RFID tag under the EPC scheme.

Heterogeneous Data Management: Management of data from a large number of sources such as semi-structured documents, databases and object repositories.

Huffman: Huffman's algorithm is for a symbol-by-symbol coding (i.e. a stream of unrelated symbols) with a known input probability distribution. It is a lossless data compression technique.

Data format: A way data or information is stored in computer file. Refers to data types like character, integer, floating point, boolean or any other data types.

Data structure: The organisation and storage of data in a computer. Used in computer programs to handle data as in arrays and other complex data structures like linked list and B-tree.

LZW: A lossless compression technique where input bytes are gathered into a sequence until the next character would make a sequence for which there is no code yet in the dictionary.

LZ77: An algorithm to compress data with references to matching data that have already passed through both encoder and decoder. Encoder keeps track of matches and decoder interprets the matches. The size of encoder and decoder is determined through a "sliding window" compression algorithm.

Middleware: Software that runs on a distributed network, collecting the RFID data read by the readers, cleansed before extracting meaningful information.

Personal Digital Assistant (PDA): A handheld portable device used to collect data anywhere in the environment and linked to wireless network for instant data transfer.

Reader /Interrogator: A device used to read an RFID tag. The reader has an antenna that emits radio waves; the tag responds by sending back its data.

Savant: An implicit federated application designed to update data simultaneously across the storage medium (Christian & Matthias, 2005).

Schema: The structure in a database system that defines the relationship between tables and fields. The term is also used with XML, describing its structure and data types.

SDK: A set of development tools that allows for the creation of software applications.

Tag: A microchip combined with an antenna in a compact package. The tag's antenna picks up signals from an RFID reader and returns the signal data.

User Memory: Additional memory on RFID tag apart from EPC ID where data can be written and read.

Wavelet: Compression technique mostly used for image compression.

Declaration

Attestation of Authorship

“I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning, except where due acknowledgement is made in the acknowledgements.”

Sarita Pais

Abstract

RFID tags can store more than just a tag ID. Data on an RFID tag can be updated through local processing. This is in contrast to the EPC global standard of data-on-network. In order to illustrate this concept of data-on-tag, a business scenario of a commercial laundry operation is considered. This thesis explores how data on an RFID tag can be manipulated (reading and writing) within the tag itself in the laundry bin operation of a commercial laundry. Tag manipulation in this research experiment attempts to use local processing within a tag to count and to calculate the weight of soiled linen tagged with RFID. The motivation of the thesis study is to process data at a local point and integrate it to the backend enterprise systems. Local processing avoids the need for expensive middleware, central database and network applications that are required in a data-on-network approach. Data captured from the linen and bin objects helps the laundry company with business intelligence at the enterprise level.

This research experiment explores the pros and cons of a data-on-tag approach and reflects on where the future in such an approach may lie.

The scope of this thesis study is to find a suitable data format for data stored in the linen and laundry bin tags. Two data formats viz CSV and XML along with compression techniques were discussed. A suitable format was chosen and a proof of concept prototype developed. The experiment conducted using the prototype examined how relevant data can be stored in the RFID tags and used in local processing without the need for a central database or network connectivity. The findings of the experiment results demonstrate sufficient proof of concept to suggest CSV data format. The experiment evaluated the accuracy of the data-on-tag approach of reading and writing data in CSV format in the tags. Issues encountered in the experiment are discussed, particularly related to writing data into the tag. The current research study opens new arenas for research study by academia in data-on-tag and in particular research on writing to tag. The research contribution aids industry and manufacturers for improved technology for writing more data into the tag accurately and efficiently. The conclusion explores the direction for future research on improving writing data on tag, using the data-on-tag approach.

1 Introduction

Mark Weiser (1993) envisioned a ubiquitous computing environment for the future where technology is omnipresent in the environment and ‘invisible’ to the user. Ubiquitous computing is also referred to as pervasive computing. Computing devices are small and well integrated into the environment to provide useful information where required at any time. Radio Frequency Identification (RFID) is one such promising technology in the field of pervasive computing. RFID is more advantageous than barcode technology, having no line of sight requirement for reading tags. RFIDs can read several tags simultaneously, store more data on the tag and data on the tag can be manipulated (Haas & Miller, 1997; Hardgrave, Armstrong, & Riemenschneider, 2007). The cost of implementing RFID technology is also reducing, making realisation of Return on Investment (ROI) in commercial deployment more achievable.

Diekmann, Melski, & Schumann (2007) suggest that generally commercial applications of RFID technology use passive tags and centralised data storage models. However Diekmann’s et al. (2007) claim is based on data from around 2006 and it is likely that the situation might have evolved from that time. RFID tags are mainly used for identification of physical objects and store an ID called an Electronic Product Code (EPC) in the tag. More details of the physical object are stored in database files that can be accessed via a centralised network. The EPC from the tag is used to retrieve more details about the physical object from the network database. This system of using data from an RFID tag in conjunction with data on the enterprise network is called as data-on-network approach.

RFID tags come with additional memory apart from that used for storing the EPC. This additional memory on the tag is of Electrically Erasable Programmable Read-Only Memory (EEPROM) type. Complete data about the physical object is stored on this EEPROM memory. This approach is known as a data-on-tag approach. In a data-on-tag approach, data is stored on the tag, with lesser reliance on the centralised network database. Thus data-on-tag data storage systems are a type of decentralised data storage (Melski, Thoroe, Caus, & Schumann, 2007). A decentralised data storage approach was envisioned by Gray (2004) where smart objects like smart dust stored data and local processing took place using data and some local algorithms.

If data from the tag is required to be used in a local context and there is no network connection as in a cold chain for temperature and humidity checks, then in such applications, data processing moves towards the periphery of the network not relying on a central database. Moreover, a network failure in a centrally controlled system results in an overall failure. However, in decentralised systems the stability is improved as the failure of one part of the system does not affect the system as a whole.

The application of data-on-tag approach is not popular because not enough is known about how to store data in a very small space and how to process that data with the limited capability of an RFID reader.

1.1 Research Motivation

The motivation for undertaking this research study is to determine how data can be used locally and to integrate it at the enterprise level without network connectivity. It will be advantageous if RFID technology can be used seamlessly with existing legacy enterprise systems.

1.2 Research Objectives

The main objective in this study is to determine a suitable data format to store data in the RFID memory for use in local processing.

Harmon (2006) suggested a single string written to the RFID tag in the user memory area. The application reading this string assigned a meaning to each character position within the string. An alternate approach to storing data was undertaken by Jiang & Xiang (2008) to compensate for the limited RFID memory where the application reading this data had an ontology to decode meaning from the coded data.

In order to achieve the main objective, it is important to understand how much data can be stored in a limited memory space on an RFID tag and how such data can be processed. It is also important to determine which data format is suitable to be incorporated in the limited RFID memory. As the RFID read rate is not 100% in commercial application deployment as reported by Tran, Sutton, Cocci et al. (2009) , it is essential to determine the write and read performance of the RFID tag memory with this data format.

Thus, the research focuses on finding a solution for the following questions:

> ***What data format is suitable for a data-on-tag approach?***
>
> ***How can data be stored in the RFID tag in a data-on-tag approach?***
>
> ***How data from RFID tags is read and written in a data-on-tag approach?***
>
> ***How accurate is the reading and writing of data using the data-on-tag approach?***

In order to undertake this research, the literature was first considered to understand RFID technology, RFID data characteristics, various data formats to store data in RFID memory and compression techniques to store more data in the limited memory of RFID tags. The knowledge gained was applied to a practical single business problem. The experiment undertaken explored read and write data in the selected data format with the limited user memory of RFID tags. The accuracy of reading and writing results are presented and analysed, and followed by a discussion and findings to address the research questions. Finally, the contribution, implications for academia and practice, limitations of the current research and future research in this direction are given in the conclusions.

1.3 Business Problem

In order to illustrate the concept of storing more data on RFID tags and processing it, a business scenario of a laundry bin for a hospital laundry was considered.

In current practice in the commercial hospital laundry industry, linen is supplied by the laundry company to the hospital. The supplier then regularly collects soiled linen from the laundry bins, and returns them cleaned and washed, to their original stock levels. The laundry company is disadvantaged by linen being lost or misplaced by the hospital, as they have no way of quantifying such losses. In a commercial situation, such losses can be substantial.

1.4 Research Design

The focus of this research is to identify a suitable data format to store data in an RFID tag and a data manipulation process where read and write operations take place locally without any integration to the backend enterprise system. In order to undertake this research study a business problem of a commercial laundry company is considered. This study explores how an RFID tag attached to each piece of linen can identify and quantify the soiled linen being picked up by the laundry company, using an information system (IS). This is achieved by researching the type of RFID and the data format that will be a suitable in this scenario.

The business process of the laundry bin scenario and the data in the tags are combined in building the IS. The IS facilitates the management of RFID data in the business process. The proposed IS is an integral part in addressing the research questions.

The prototype developed not only recognised the linen tag and maintained a count but also was able to update data by overwriting data into the linen tag. All data relating to a piece of linen is stored in the tag attached to that linen. The data collected from linen tags at the pickup point is analysed for read success rate and write success rate.

The prototype does not depend on a central database or middleware software for any meaningful information. Such an application suits situations where network connectivity is not available because local processing takes place close to the vicinity of the objects.

The cost of implementation of middleware and of changing legacy systems to suit RFID technology is very high. The prototype works without the need for expensive middleware or a change in the legacy system, giving the prototype a cost advantage.

1.5 Research Contribution

The main contribution of this thesis was to explore the current gap in the literature in areas on data-on-tag to identify a suitable data format for the storage of data in the tag. The experiment undertaken examines how this data format can be written to and read from the tag to be used in an application for local processing and is a proof of concept of the data-on-tag approach. The prototype developed in this research study identified the limitations of COTS (Commercial off the Shelf) hardware and software in reading and writing data to a tag. This research also identifies opportunities for researchers and hardware manufacturers to design improved write procedure on an RFID tag.

1.6 Thesis Structure

This chapter examines the motivation for the current research study and provides an overview of the research questions. The following chapters are structured to address the research objectives and questions and to find solutions through the experiments conducted. The results are discussed and a list of limitations and future recommendations are given.

Chapter 2 discusses the literature on RFID technology, main components necessary to implement RFID system, RFID data characteristics and the two approaches to using RFID; data-on-network and data-on-tag.
In order to understand how to store data on a tag, the user memory will be studied along with maximum possible storage capacity. Analysis of reported research on data formats used in similar situations and also data compression techniques used in similar areas will be discussed. Two data formats will be considered to store data on a tag. Suggestions will be made on the data format and compression techniques suitable for data storage on the tag.

Chapter 3 introduces research methodology and explains how the selected methodology of design science is applied in the current research study. Design science is a problem solving methodology. The guidelines in design science help in conducting research and set up the experiment for a single business problem of a laundry bin. The research process also explains the data collection process and analysis techniques used in this study.

Chapter 4 explains the set up for the selected data format with a pseudo code explaining how data from RFID tags is captured and processed in the prototype developed. The prototype was developed using the software provided by Tracient Technologies Ltd. The experiments used RFID tags of two different standards that were available to the researcher and which had sufficient data storage capacity. The two data formats discussed in Chapter 2 were evaluated for their suitability and the more suitable format was applied through the prototype to run the experiments. Two scenarios were considered in conducting the experiment with the RFID tags and the results presented.

Chapter 5 discusses the results from the set of experiments conducted. The accuracy of RFID read/write is evaluated. Challenges and issues are discussed.

Chapter 6 presents the research contributions on handling RFID data in a data-on-tag approach for academia and industry. The limitations of the current thesis and recommendations for future work in this field are presented. The thesis concludes with a brief summary of the whole research study.

2 Literature Review

This chapter reviews the available literature on the history of RFID technology. The components of the RFID system – the tag and reader were examined. Classification of tag standards and frequencies were examined from the available literature. Characteristics of the data collected from RFID system and the issues around these were studied. Data formats and compression techniques were studied. All of these factors helped the researcher in selection of an appropriate data format in the tag using the prototype to evaluate the research questions.

2.1 RFID History

RFID is not a new technology and its use dates back to the 1940s. Advancement in electromagnetic theory formed the basis for understanding radio frequencies which can reflect waves from objects (Landt, 2005). This understanding of radio frequencies was used in Radar during World War II. In 1948, Stockman (1948) first put forward the theory of RFID which was later developed by Los Alamos Scientific Laboratories in 1977 (Wu, Nystrom, Lin, & Yu, 2006). A brief overview of the development of RFID technology over the years is shown in Table 2.1.

Table 2.1: Growth of RFID Technology over the Years

Source: (Landt, 2005)

Decade	Event
1940–1950	Radar refined and used in a major World War II development effort. RFID invented in1948
1950–1960	Early explorations of RFID technology, laboratory experiments
1960–1970	Development of the theory of RFID. Start of applications field trials
1970–1980	Explosion of RFID development. Tests of RFID accelerate. Very early adopter implementations of RFID
1980–1990	Commercial applications of RFID become mainstream
1990–2000	Emergence of standards. RFID widely deployed. RFID becomes a part of everyday life
2000–	RFID explosion continues

Although RFID technology is a mature one, it has not been extensively used commercially in the past because of its uneconomical deployment cost. Although cost remains a key barrier it is being chipped away with falling prices making RFID deployment more economical. Apart from this, improvements in design of the chip and its encapsulation material are making the tag smaller with the only restriction on the antenna size. This makes it easier to integrate RFIDs into the environment attached to physical objects.

More details on RFID such as its components are studied in the following Section 2.2, which will inform the design of the experiment.

2.2 RFID System

The RFID system consists of two main physical components: the RF reader (interrogator) and RF tag (transponder). RFID tags are attached to physical objects and make them recognisable by the RFID reader through radio frequency communication. Tag ID is picked by the reader and subsequently is picked up by the application software which connects to more information about the physical object stored in a database.

2.2.1 RFID Tag

There are two main components in an RFID tag: an Integrated Chip (IC) where data can be stored, and an antenna which transmits and receives radio waves. The chip is very small and less than half a millimetre in size. The antenna is a flat conductive metallic coil. The IC and antenna are protected in an encapsulation material which is can be flat with adhesive on one side, making it convenient to attach the tag to physical objects. The antenna picks up the signal from the RFID reader (interrogator) and reciprocates, sending information from the chip to the reader. The RFID tag can come in various sizes and can be very small as shown in Figure 2.1. Tags suitable for laundry purposes are encapsulated in protective material which can withstand harsh temperatures and chemicals.

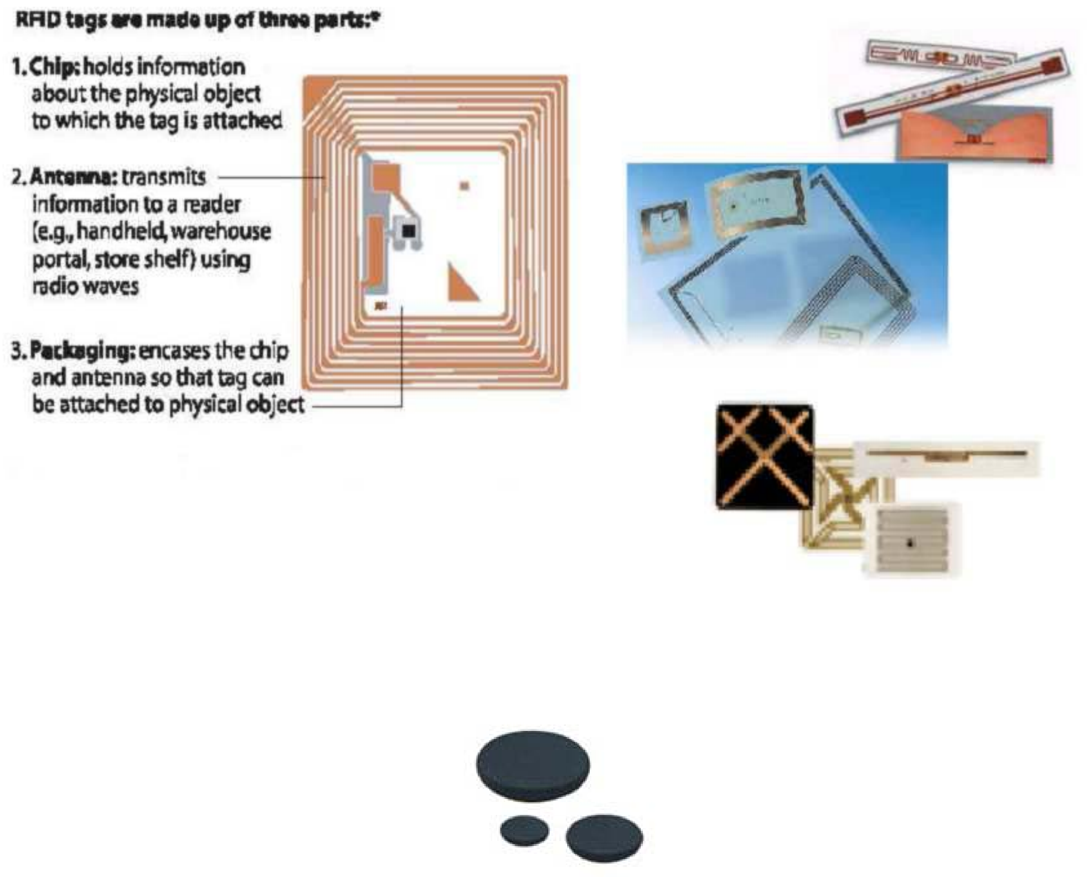

RFID tags are made up of three parts:*
1. Chip: holds information about the physical object to which the tag is attached
2. Antenna: transmits information to a reader (e.g., handheld, warehouse portal, store shelf) using radio waves
3. Packaging: encases the chip and antenna so that tag can be attached to physical object

Figure 2.1: RFID Tag Structure

Source: (Chawanthe, Krishnamurthy, Ramachandra, & Sarma, 2004, p. 1190; Gao-RFID, n.d.)

2.2.2 RFID Reader

The RFID reader is either mounted or handheld. Readers communicate with the RFID tags around them to read the tag ID numbers in tags through radio frequency communication. The RFID tag antenna activates the circuit within the tag and transmits the ID and other information to be picked by the reader. RFID readers can read and write data. Readers can be connected as an input device to a PC through RS232, USB port or wirelessly through Bluetooth. Readers can also be connected to handheld devices such as PDAs through a SD (Secure Digital) card interface. A sample of readers is shown in Figure 2.2.

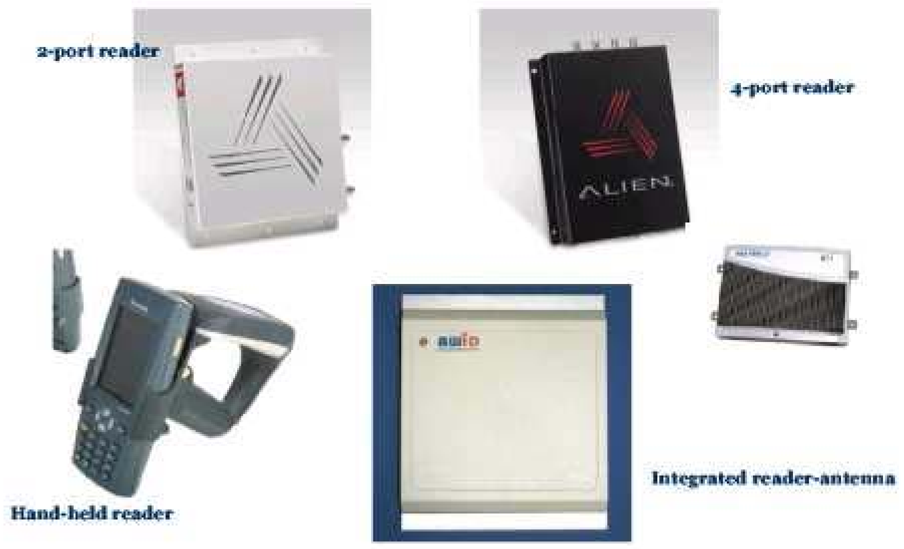


Figure 2.2: RFID Readers

Source: (Chawanthe et al., 2004, p. 1191)

2.3 RFID Tag Classification

RFID tag classifications determine their applications. RFID tags can be classified by functionality, frequency and standards.

Two different approaches on the communications between reader and tag (transferring power) - near field and far field (Want, 2006) are also discussed.

2.3.1 Functionality - Passive or Active

There are five classes of RFID tags falling into either passive or active tags (Chawanthe et al., 2004) as shown in Figure 2.3.

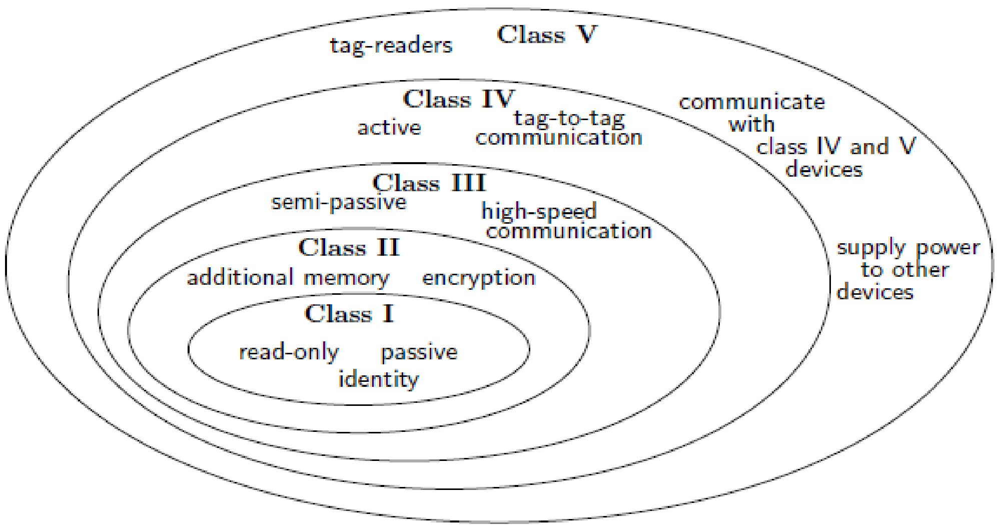


Figure 2.3: RFID Classes

Source: (Chawanthe et al., 2004, p. 1191)

Passive tags are read only while others have limited write capabilities. Passive tags have no battery and use the power reflected by electromagnetic radiation emitted by RFID readers to transmit data. Active tags have an inbuilt battery with a three to five years life span (Ni & Liu, 2004) and need an external power supply to transmit data autonomously.

Passive tags are generally cheaper than active tags. Passive tags have a limited read range and a reader can read anywhere from one tag to a few hundred tags at a time depending on the type of tag. Active tags on the other hand, are programmable and many tags can be read simultaneously from a bigger range more accurately.

Passive tags have been used in the same way as barcode technology, storing only product ID on the tag, with data manipulation done centrally in the database. Passive tags are generally widely used in applications like supply chain management, inventory management and asset tracking systems to identify physical objects for tracking. The use of active tags could store all required data of the product in the physical object and allow for data processing at the tag or reader level which gives the capability of local processing.

There are passive tags which have extended memory to store additional information to a maximum of 2KB. Active tags have memory between 16 bytes and 128 KB (Ward, Kraneneburg, & Backhouse, 2006). Vendors have recently developed UHF passive RFID tags with memory of 32 KB (Bacheldor, 2009).

The choice of active or passive tags will depend on various factors (Want, 2004), summarised in the Table 2.2.

Table 2.2: Comparison of Active and Passive tags

Source: (Want, 2004)

	Active	Passive
Memory	60 bytes to 128 KB	Limited, generally 512 bits up to 2 KB
Access	Read and write	Read and some writeable
Distance	20 to 100 meters	Up to 9.14 metres
Battery	On board	External
Frequency	455 MHz, 2.45 GHz, 5.8GHz	128 MHz
Cost	High	Low
Tag size (including antenna)	Large	Small
Lifetime	Limited battery life 4-5 years	Indefinite operation

From Table 2.2, it is seen that active tags have more memory and read from a greater distance. However the size, cost and requirement of an onboard battery are constraints to active RFID tags which is a crucial drawback in advancing their popularity. Active tags need continuous power supply to interact with the RFID reader. Active tags could be ideal for asset management in large business organisations. The choice for the current thesis study is for less expensive passive tags with some memory in it. More specific requirements on passive tags are discussed later as other classifications like frequencies and standards become part of the basis for choosing the appropriate RFID tags for the current thesis study.

Apart from the main classification as active or passive, RFID tags are also classified by their frequency which is discussed in Section 2.3.2.

2.3.2 RFID Frequency

RFID tags operate at different frequencies and the most common are summarised.

Low Frequency tags operate at 125 KHz and have a read range less than 0.5 meter. They are the least expensive of all the tag types but have a slow data read rate. Such tags are used in applications which require close range reading as in access control and item level tracking.

High Frequency tags operate at 13.56 MHz and have a read range of less than one meter. Although high frequency tags can read data faster than low frequency tags they are often deployed in similar applications to low frequency tags.

Ultra High Frequency tags operate at 433 MHz or 865 – 956 MHz and in long ranges as many as 100 tags can be concurrently read by a reader. As these tags operate at a frequency more than 900 MHz, it improves the read rate as data from the tag is read in a shorter time interval avoiding collision with data from other tags (Want, 2004).

Microwave tags operate at 2-30GHz and have a long read range of up to 10 meters.

A full list of RFID frequencies and their characteristics are listed in Table 2.3.

Table 2.3: RFID Frequencies and Characteristics

Source: (Ward et al., 2006)

Band	**Low Frequency**	**High Frequency**	**Ultra High Frequency**	**Microwave**
Frequency	30-300 kHz	3-30MHz	300 MHz – 3 GHz	2-30GHz
Typical RFID frequencies	125-134 kHz	13.56 MHz	433 MHz or 865 – 956 MHz 2.45 GHz	2.45 GHz
Read range	< 0.5 meter	<= 1.5 meters	433 MHz = up to 100 meters 865 – 956 mHz = 0.5 to 5 meters	Up to 10 meters
Typical data transfer rate	Less than 1 kilobit per second (kbit/s)	Approximately 25 kbits	433-956 = 30 kbits 2.45 = 100 kbits	Up to 100 kbits
Characteristics	Short-range, low data transfer rate, penetrates water but not metal	Higher ranges, reasonable data rate similar to GSM phone, penetrates water but not metal	Long ranges, high data transfer rate, concurrent read of < 100 items, cannot penetrate water or metals	Long range, high data transfer rate, cannot penetrate water or metal
Typical use	Animal ID, car immobiliser	Smart labels, contact less travel cards, access and security	Specialist animal tracking, Logistics	Moving vehicle toll

2.3.3 RFID Standards

As RFID applications gained commercial popularity, it was necessary to have some standards in place. This issue became visible when Wal-Mart, one of the foremost players in RFID, required that all goods in their supply chain be tagged using RFID at pallet and case levels (*Wal-Mart spells out RFID vision, RFID Journal 2003*). Thus all products from Wal-Mart suppliers and manufacturers are tagged at item and pallet levels and RFID is used in the complete supply chain. This is possible only when tags are compatible and readable by different companies in the supply chain. For tags to be read successfully there should be a uniform standard to identify tags. In the case of Wal-Mart, they took the initiative and asked all their trading partners to follow EPC Gen 2 standard. As RFID standards are still being developed, organisations are not willing to invest in RFID systems which may become obsolete. Thus if an internationally agreed standard is adhered to, different readers from different vendors can read any tag. However there is no one RFID standard today. Initially, RFID technology developed without an appropriate tag standard to store data about a physical object. The Auto-ID Centre in MIT came up with the Electronic Product Code (EPC) to uniquely identify each RFID tag (Harrison, 2003). The EPC contains the minimum information that needs to be carried on the tag, i.e. a unique identification. Use of EPC ensures that product information is unique across the company and can be indexed to *look up* more details from a database via the enterprise network.

The RFID standard is based on the ‘air interface’ protocol which takes care of the tags being recognised by the reader (Lin, Elmongui, Bertino, & Ooi, 2007). Two major organisations EPC Global (AutoID centre) and International Standard Organisations (ISO) have developed an international standard (Wu et al., 2006). However there continue to be issues of compatibility between the standards of different institutions, as a result of which readers and tags of different standards may not work together.

ISO had its own standards called 18000-6, which comprised of sub classes 18000-6A and 18000-6B. When EPC introduced a new Class1 – Gen 2 protocol, ISO approved this (O'Connor, 2006) and released its own updated standard called ISO 18000-6C which is the current standard on which tags are built.

The International Electro-technical Commission (IEC) is a non-profit international standard organisation which has worked together with ISO to jointly develop standards for electrical, electronic and related technologies products. An overview description of ISO/IEC 18000 is given in Appendix A.

This research experiment has used tags that comply with the ISO 1800-6B and ISO 18000-6C standards.

ISO 18000 6 Type A, B and C have different anti-collision protocols built into them. The two main protocols are by ALOHA algorithms for channel sharing by time and the other is 'tree-based protocols' which identifies tags by reader-machine questioning. ISO 18000 6A has ALOHA, ISO 18000 6B has Btree and ISO 18000 6C has random slotted (Q algorithm) anti-collision protocols (de Azambuja, Marcon, & Hessel, 2008). The existing anti-collision protocols were used without modification in this research experiment.

A more technical discussion on type A and B is given in ISO documentation (ISO, 2007) as in Appendix B.

2.3.4 Near and Far Field

Near field communication is based on magnetic induction. The reader and tag create a magnetic field between them and induce an electric current in the tag's antenna to activate the circuit within the tag to retrieve the tag ID. On the other hand, far field communication is based on electric radio waves reflection. The reader emits a continuous frequency which is reflected back by the tag along with the tag ID.

2.4 Comparison with Barcode Technology

Although barcode technology has been widely used for applications like Supply Chain Management System (SCMS) to identify and track products, RFID has many advantages. A barcode has a unique identity for all items of a particular type. For example all books of the same title and author have the same barcode. However, RFID on the other hand identifies each item using a unique identity. Thus each book copy has a unique product number stored on a RFID tag. Such unique identification further helps in tracing a particular book in the SCMS or in an asset tracking system such as for a public library.

Barcodes need a line of sight between the label and the scanner device. Each book for example, needs to be handled individually and in close proximity to the scanner to be identified. The process of individual product handling involves expensive labour processes in the supply chain. RFID tags on the other hand are automatically scanned by interrogators from a wider range (depending on the generation of tag and reader), offering scope for substantial reduction in handling costs.

The next section provides an overview of three ways to deploy RFID systems.

2.5 RFID System Deployment

Glover and Bhatt (2006) defined three ways of deploying RFID systems. In the first method of ***centralised deployment,*** data from all readers is moved to the central database and processing is done at enterprise level. Centralised deployment is currently the most common approach in most supply chain management systems. Here the whole burden is on the central location, where data traffic from all different sources is processed. Although managing a centralised server is relatively easier, it does not decrease data traffic. To ease the burden, ***distributed deployment*** is considered. It does some part of processing at the edge of the network but the major processing is done at the central location. The third type of deployment is ***self-organising*** where most of the computing moves to the edge and there is no data centre, and each edge is still a member of the network. With falling costs of RFID tags and related infrastructure, the latter two deployments could become viable in the future. Again, this is different from the current study, where RFID tags are not part of a network.

When processing control moves away from the centre, then middleware has to be deployed locally. Middleware filters erroneous and duplicate data before any edge processing is done.

Although RFID systems can be deployed in three ways as discussed in the Section 2.5, Diekmann et al. (2007) categorised them as data-on-network and data-on-tag described in the following Sections 2.6 and 2.7. Data-on-network is a general term describing centralised and distributed deployment whereas data-on-tag is partially a self organising type. Data-on-tag need not necessarily be part of the network and can process data from the tag locally in an application. In this study, the two terms described by Diekmann et al. (2007) will be used. Data-on-tag approach used in this study will not be part of any network and the data from the tag will be used in a prototype for local processing. The data thus processed can be used at the enterprise level for extracting business intelligence.

2.6 Data on Network

In a data-on-network approach, data about the physical object is stored in a central database and referenced through an identification attribute called an EPC. The RFID tag only stores the EPC and all additional data about the physical item is stored in the central database.

Examples of data-on-network are applications like SCMS where RFID technology can help stock level management and library systems where RFID technology is important in asset tracking and management. In the data-on-network approach, RFID data are voluminous; there could be duplicate readings, and/or erroneous readings which need to be cleansed by the middleware before any meaningful information is extracted from the data as discussed in Section 2.5. Data cleansing is done by Savant enterprise software developed by Auto-ID centre which runs on a distributed network, collecting the RFID data read by the readers. Data-on-network systems also strive for data independence so that the object is never invisible to the network (Joseph, 2003). This process is often referred to as data centric routing and ensures that each data node is always available to the system. The need for all objects to be visible on the network is a disadvantage of the data-on-network system because it clogs the system with unnecessary network traffic.

2.7 Data on Tag

As discussed in Section 2.6 data-on-network stores only an EPC in the RFID tag. However if more data can be stored in the RFID tag itself then the concept of data-on-tag can be applied to other technologies like smart cards and 2D barcode. Diekmann et al. (2007) highlighted this little researched aspect of 'data-on-tag' in the literature. The data-on-tag and data-on-network approaches are described in an architecture diagram in Figure 2.4. The data of the object saved on the tag is useful for local processing and does not depend on the backend data warehouse. Data-on-network depends on the backend data warehouse which creates bottlenecks (Kotak & Gruver, 2009) at the backend process. RFID applications using data-on-network approach are similar to traditional barcode systems as only the identification attribute is stored at the physical object. 2 D barcodes and RFID tags allow additional information to be stored with the physical objects which create opportunities for new applications and approaches.

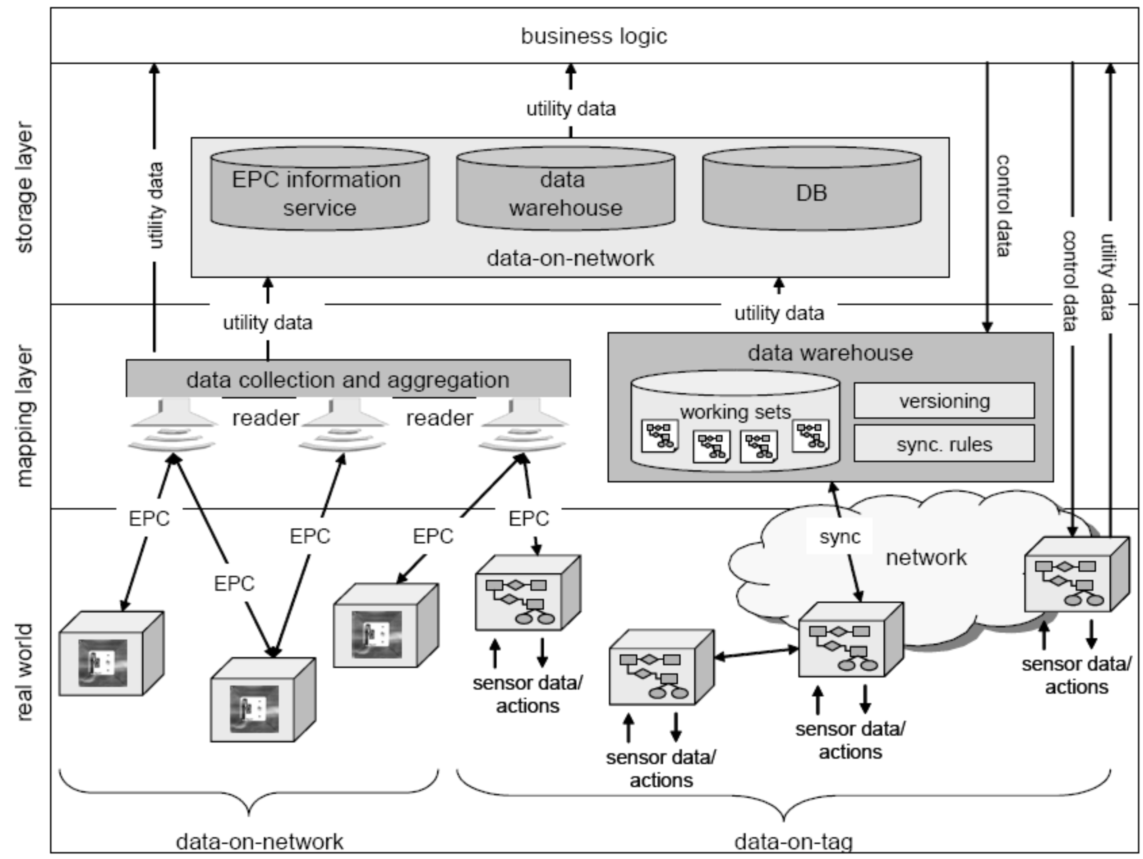


Figure 2.4: Integrated Data Management Architecture

Source: (Diekmann et al., 2007)

The concept of data-on-tag is not new. It has been used in production control and maintenance documentation in manufacturing, tracking patients and medical equipment in healthcare. In one example, additional information for garments was stored in the tag (Melski et al., 2007). The additional information was used to convey garment details along different stages of the supply chain. As these products travel long distances, accessing network details at different distribution centres is not possible as there is no central network connection at all times. In another example, in order to avoid bottlenecks at the central database, Ford manufacturing company stored customised details of the production process in active tags. The production process is not disrupted even if there is a network failure. Another advantage is keeping an update of maintenance documentation as in Frankfurt airport's fire shutters scattered at different locations. The maintenance person reads the maintenance instructions from the tag using a portable reader and updates with the current date and time. In this way, complete maintenance records can be kept and quality assurance targets can be met even where there is no central network access.

The data-on-tag approach can have applications similar to smart card technology where extra information is stored with the physical item. Chan, Cao, Chan, & Young (2001) proposed embedded processing inside a smart card. The smart card could carry patient information accessible by any hospital and did not require any sophisticated computer network to access the data from the backend database. The smart card in this case had a web-based application stored in a Java applet program. Thus any hospital computer with a smart card reader could access the patient data in the smart card through a web browser.

The smart card usually comes with a memory in the range 2 megabytes to 128 megabytes. With this memory it is possible to store a Java applet program in it. On the other hand passive RFID tags (Class 1 Gen 2) have very little memory on board. Because memory in passive RFID tags are in the range of 64 bits to 2 kilobytes, storing an applet program tag may not be possible. The RFID reader will need to read the applet code and run it on the reader device or on a PC/PDA. Active RFID tags have more memory than passive tags but lesser than smart card.

In order to overcome the limited memory in RFID tag, Romer, Schoch, & Mattern (2004) proposed storing an internet address (URL) in the tag through which the detail of the physical item could be looked up in the database server and not stored in the tag. No access to proprietary network was needed. The application was designed to access the tag's virtual counterpart on the internet.

However, it still needed an internet connection and application running on the web server in order to lookup data about the physical item.

Jiang & Xiang (2008) proposed compression techniques to store an image in a RFID tag of available memory size of one to two kilobytes. Compression was done through Wavelet and EZW encoding (discussed in Section 2.11). A similar approach for creating an indoor map of RFID tags in the physical environment was undertaken by Tsai (2008). Wavelet compression proved to be efficient in compressing data from an original size of 108 – 112 bytes to 68 – 71 bytes small enough to store within an RFID tag. Wavelet compression was ideal for image compression and involved numeric data and relative position of pixels in an image.

The concept of data-on-tag can be implemented in a distributed way by having data in RFID tags explained in case of super distributed database and tuple-based distributed data which are discussed in the next section.

2.7.1 Super Distributed Database

According to Sugawara, Yamaoka & Sakai (1997) a super distributed database is a network connecting many personal databases (PDBs). The data in these PDBs are not centrally controlled and can be edited locally. A super distributed database is created by RFID tags scattered in the environment as 'super-distributed' smart objects (Bohn & Mattern, 2004). The intention of the current research study is to explore potential ways to carry data in individual RFID tags not necessarily be part of a network. Data is used in a contextual manner when required and thus locally processed.

2.7.2 Tuple-based Distributed Data

Mamei, Quaglieri, & Franco Zambonelli (2006) used RFID tag memory to realise a system that worked through data distributed in the environment which they called the tuple-based distributed data approach. Mamei et al.(2006) used RFID tags to represent physical objects in the environment with the capability of storing digital data in them. Each data piece about the physical object was an attribute and all the attributes together were treated as a single tuple, coded as a simple data format in 512 bit memory of the RFID tag. For example a tuple with values ("flower", "red") was coded as (01, 14). The programming interface picked up the coded value and matched it with pre-defined values in the application to create more meaningful information from the data in the tag. The pre-defined values were stored in an application dependent ontology. This approach overcame the restricted memory of RFID tags and was an example of data format where a type entity could be used.

The data-on-network approach needs a secure network setup with middleware to filter data from the RFID reader as it reads the tags and before it is stored in the enterprise database. The data-on-tag approach on the other hand, is achieved locally by reading RFID tags from reader and storing data in a local database for accessing in computer applications. Furthermore, the data-on-tag approach does not depend on Savant or any other proprietary middleware. It is a decentralised way of managing data and continues to work even in case of a network failure.

The data-on-network approach stores limited information in the RFID tag like an EPC and additional information is securely stored in a central database as discussed in Section 2.6. A data-on-tag approach on the other hand has the additional information stored on the RFID tag itself. There is a need to reduce data size by compression for encryption and security. This requires a suitable data format to store the additional information in the limited memory space on the RFID tag.

With the data-on-network approach, it is only necessary to read the EPCs stored in the tag. In a data-on-tag approach the read and write process on the additional data in the tag occurs after the EPC is read.

Majority of the research reported on RFID data characteristics is at the enterprise level and relates to capturing voluminous data. The data characteristics discussed in those research studies also apply to the current research study. It considers whether the large amount of data collected can be handled to identify and eliminate erroneous, duplicate and missed readings.

Section 2.8 will detail the RFID data characteristics and issues that arise due to the way data is captured by the RFID reader.

2.8 RFID Data Characteristics

RFID tags are read by interrogators creating a continuous stream of data. These readings could be taken at different times and locations in the SCMS as items move along a conveyor belt for example. Hence, the literature is consulted to find out what data pieces are important and the issues and solutions currently available around reading RFID tags.

2.8.1 Voluminous RFID Data

The volume of RFID tag data would be in gigabytes of data a day in a moderate RFID deployment capturing location, date, time and tag ID (EPC) in continuous mode according to Derakhshan, Orlowska, & Li (2007). A large stream of data gets collected and this data needs to be cleaned. Savant middleware (Floerkemeier & Lampe, 2005) addresses most of the data cleansing before sending data to the server for extracting business intelligence. Palmer (2004) stressed the need to filter streaming data close to the source and send only relevant data to the centralised database avoiding congestion in the network infrastructure. In the case of the data-on-tag, data cleaning and filtering is considered locally. An interrogator could clean and filter data before uploading it into a PC/PDA to further clean, filter and upload required data to the data warehouse server to extract business intelligence.

2.8.2 Duplicate RFID Data

RFID tag data may be read by multiple readers in a single portal, to improve missed readings (Bai, Wang, & Liu, 2006). On the flip side, this will create duplicate readings. The other reasons for duplicates is the RFID tag is in the range of the reader for a long time or a single object is tagged with multiple tags with same EPC to improve reading accuracy (Bai et al., 2006).

2.8.3 Erroneous and Missed Readings

As stated earlier it is advantageous to read many times to reduce the error rate and by reading many times errors can be filtered out using mathematical algorithms based on probability. There is an observed read rate of 60-70% of tags by readers (Floerkemeier & Lampe, 2004) in most of RFID deployments. The problem not only relates to the reliable manufacture of tags, and also includes packaging considerations such as placement and tag orientation.

Clarke, Twede, Tazelaar, & Boyer (2006) in a study of shipping container loads comprising of cases loaded onto pallets packed to containers reported varying error rates with the best being 74-79% of container loads, with 100% of tags read accurately using UHF technology. Hence these factors have to be considered when processing data to retrieve accurate meaningful information.

The inaccuracies discussed are primarily due to false positive or false negative readings.

False positive

False positives occur when a tag continues to be read by the reader although it has exited the reader range or the same tag is read several times creating duplicates in the data stream.

False negative

False negatives occur when a tag is not recognised by the reader although it is in the read range (Jeffery, Garofalakis, & Franklin, 2006). Bai et al. (2006) suggested that the cross interference of reading multiple tags simultaneously could cause a whole set of tags to be missed.

After looking into the nature of RFID data, many researchers (Kabadayi, Pridgen, & Julien, 2006; Lin et al., 2007; Wang & Liu, 2005) have studied and suggested solutions to manage RFID data through heterogeneous data management, aggregation and containment.

2.8.4 Heterogeneous Data Management

Data captured from RFID streams is of interest not only to one application but to multiple applications. The data stream should be compatible with any application reading it, and also where data is represented in various formats in the case of data-on-tag approach. Many researchers have considered the issue of heterogeneous data management. Researchers have applied heterogeneous data management techniques to data collected from various sensor network sources (Kabadayi et al., 2006). Kabadayi et al. (2006) discuss the concept of a virtual sensor within the context of a crane with three physical sensors to measure load pressures while much of the processing of the raw data from the physical sensors is done locally within the crane.

2.8.5 Aggregation and Containment

Another issue considered was the importance of aggregation and containment (Lin et al., 2007; Wang & Liu, 2005). Aggregation and containment build the hierarchical relationship to determine what physical objects are contained within another (larger object). In a SCMS, a pallet contains cases of items tagged at pallet, case and item level. Logistical information like transport details on the pallet specify the location of the items in the supply chain as the items that are part of the pallet. Aggregation improves traceability of the item in the supply chain at all times.

After establishing the general characteristics of RFID data in a data-on-network approach, the next phase is to identify other characteristics of the tag that have an additional memory to store data, called user memory. The structure of user memory has to be understood to write data in the appropriate location for the reader to read it. Section 2.9 will examine the characteristics of the memory inside the UHF RFID tag and investigate how to write additional information to the tag as in data-on-tag approach.

2.9 User Memory

In addition to the EPC tag ID stored in an RFID tag, most tags are built with CMOS integrated circuits with EEPROM memory (Landt, 2005) which facilitates the reading and writing of additional data in the tag. This helps store additional information about an object on the object itself and does not depend on the network database for more information. Different RFID tags comply with different standards and their memory blocks could be of different sizes. The configuration of the memory on the tag is proprietary. The RFID reader should therefore know exactly which memory block address to read data from.

The tags for UHF standard have four memory banks as shown in Table 2.4 (Lin et al., 2007; Tracient, 2007a). In order to read the user memory, it is important to understand the memory structure and how data is stored in the different banks. Apart from the tag ID which is stored in one of the banks, there is also provision for 'access' and 'kill'[1] commands and user memory.

[1] 'Kill' command is a command by RFID readers sent to RFID tags, which disables reading data from RFID tags. This is useful where the information is no longer required. For example; In

Table 2.4: UHF Tag Memory Bank

Source: (Harmon, 2006)

Bank	Memory Definition
Bank 00	Reserved
Bank 01	EPC
Bank 10	TID
Bank 11	User

The reserved bank contains the ‘kill’ and ‘access’ passwords. When the reader transmits the ‘kill’ password, the tag responds by making the data unreadable and thus making the tag permanently unusable as a result. The ‘access’ password on the other hand changes the tag from open state to secured state. In a secure state data is protected (locked) but can be altered with the right ‘access’ password. If the application does not require the use of ‘kill’ or ‘access’ passwords, the corresponding memory locations in the tag should contain zero values.

The EPC bank contains the 12 byte EPC ID stored in blocks two to seven. This is a user defined ID.

The TID bank contains an additional tag serial ID of 4 bytes. This is a factory set unique ID and cannot be altered.

The user bank is a memory space where the user can read and write data.

To write data on the tag, the specific memory address of the bank and the block within this bank needs to be identified. All the data to be written to the tag is considered as a single data string and written entirely within the tag’s user memory location. It can then be read by an application program which retrieves the different pieces of information from that data string (Harmon, 2006).

Another approach to writing data on the tag is to identify specific blocks in the user memory to write different pieces of information to. Data written in this way

SCMS once the product is sold to the end user customer the information in the tag is destroyed to avoid illegitimate use of the product information.

is scattered over several blocks and may not be contiguous. The size of the memory block will depend on the tag standard and manufacturing specifications and will be different for different tags. When an object is being moved, different organisations along the supply chain, will find it hard to obtain the data from its tag unless the decoding key is known.

Harmon (2006) emphasized the need for a uniform standard to store and retrieve data from the user memory of an RFID tag. With a uniform standard in place, the reader will know the specific memory address in a tag where data can be stored. In order to save space and successfully read data from a specific memory address, data can be written as a string with the specific location of each character having a meaning that is understood by the application and which can be decoded. Similarly, a complete data set formatted, could be written starting from a specific block address. Data can be in Extensible Markup Language (XML) or Comma Separated Values (CSV) formats. The storage of data in the appropriate format and previous studies on instances where these data formats are used are discussed in Section 2.10.

2.9.1 Standards for User Memory

RFID technology requires a single standard to write non-EPC information to a tag and to enable it to be read across different organisations in the supply chain or between trading partners. The EPC Gen 2 standard was sent to the ISO in 2004 to be adopted as a common universal standard in a response to the need for uniformity. However there was no real importance given to how or what exactly is to be stored in user memory (Harmon, 2006) other than the 96-bit EPC(code). This was possibly due to several factors as identified by Harmon (2006). The purpose of standardising Gen 2 specifications was to promote the data-on-network concept where only the EPC was required to lookup further information from an enterprise-wide centralised database. It was also an initiative to promote RFID over barcode technology. To keep RFID tag costs to a minimum, only 96 bits in the EPC was incorporated to maintain only the bare number of transistors in the RFID chip.

An Electronic Product Code Information Service (EPCIS) has been defined EPC Gen 2 standard. EPCIS is supplementary to the EPC global standard. EPCIS determines data requirements for a product in the SCMS and facilitates the sharing of secure data in real time between trading partners (Asher, Morgan, Swan, & Traub, 2007). EPCIS contains important information like location, time stamp and stages of a product on its journey along the supply chain. EPCIS is a standard that stores different attributes of information which cannot be customised. Access to data between trading partners is often through web services. The use of EPCIS demonstrates a data-on-network approach and is an example of how standards for sharing data among various stakeholders of different organisations are developing, where data is maintained in a central enterprise database.

The process of standardising access to user memory is covered under the data protocol comprising of ISO/IEC 15961 – 1 (application interface), ISO/IEC 15961 – 2 (registration of RFID data constructs), ISO/IEC 15961 – 3 (RFID data constructs) and ISO/IEC 15962 (Data encoding rules and logical memory functions) as shown in Figure 2.5.

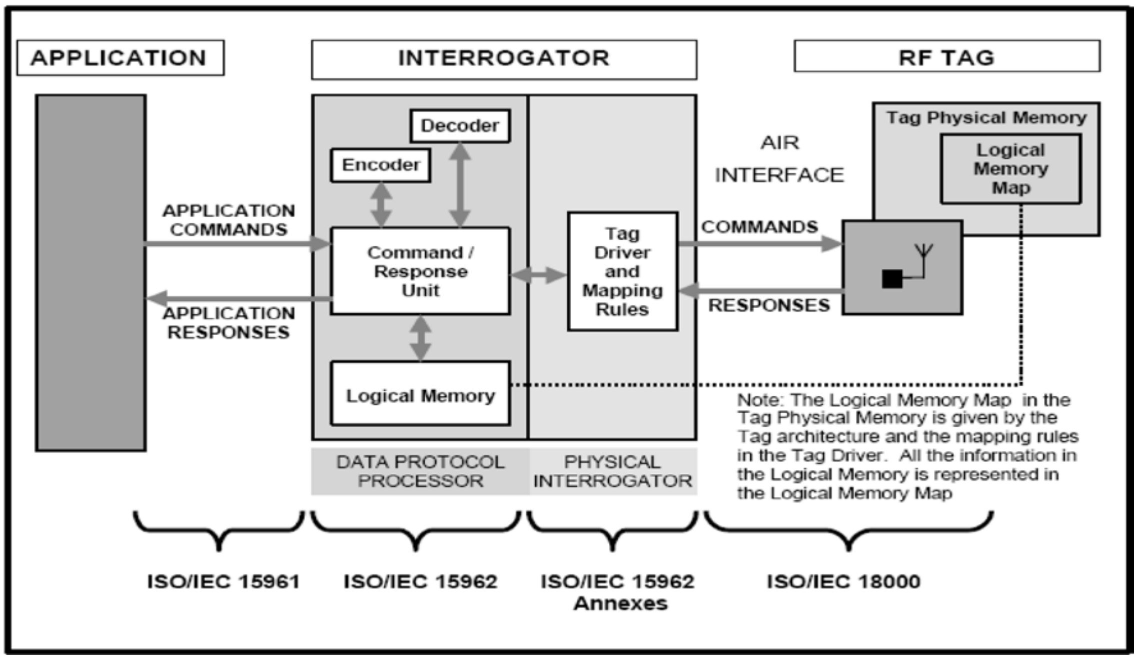


Figure 2.5: Data Protocol Standards of ISO/IEC 15961 and 15962 and the Air Interface Standards of ISO/IEC 18000

Source: (Harmon, 2006)

Although there are standards for the data-on-network approach for EPC and EPCIS, there are no standards for writing and reading data in user memory. It is therefore, important to understand the nature of the data in the user memory and how this data can be used for further processing.

The literature was reviewed to explore the writing of data on RFID tags with limited user memory as discussed in Section 2.10.

2.10 Data Format

In order to define a data format for RFID data, it is important to understand the nature of the data collected and what information is important to a given RFID application. RFID tags are read by readers (interrogators) creating a stream of data. This data stream consists of readings taken at different times by readers at different points as the product moves among the different business partners along the supply chain. In such a business scenario it is necessary to store a date and time stamp that records when it was picked up by the reader along the supply chain and an identifier to record the location of that product at a particular point along the supply chain. The different trading partners may require specific information about products along the supply chain. Electronic Data Interchange (EDI) are standardised documents to share data between organisations (Curtin, Kauffman, & Riggins, 2007). In the pre-internet era, data was represented in a specific format and shared among trading partners. Since the last few years internet enabled applications have used XML as a standard to facilitate inter organisational communication. Thus the literature is consulted to find out what information needs to be accounted for in an appropriate data format for the current research study and what other data characteristics need to be considered with RFID data. In the current research study, the data format to be selected should be able to share data between client organisations (hospital) and the laundry company. It should help in integration with the enterprise backend database of the laundry company.

Two possible data formats are considered – XML and CSV. The focus of the literature review is to explore whether these formats are suitable to read from and write to an RFID tag given its limited user memory.

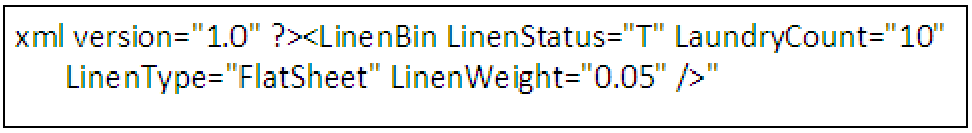
2.10.1 XML Format

XML is a simple text format to store data. It is both human readable and machine readable. Data in an XML file can be easily parsed and presented in a web browser and with a little formatting using style sheets can be made more meaningful to the user. XML coded data can be used to send data across web applications and databases. However its use is not limited to web applications; it is compatible across various database and spreadsheet applications and an XML document can be imported from and exported to various databases and spreadsheet tools. Chamberlin & Robie (2001) proposed a ‘quilt’ framework that incorporates a universal query language and a universal markup language XML to manage data from any one sources such as semi-structured documents, databases and object repositories. Haas & Miller (1997) investigated the use of database middleware as a tool for managing data from heterogeneous data sources. The schema involved in their research was XML.

Data to be stored in a tag should be in a format understood by other trading partners and XML format is a promising representation (Want, 2004). However the maximum size of the user memory on a passive tag is 2KB or less whereas the size of the XML file used to communicate the data about object the RFID tag is attached to would be a string of at least 107 characters which would be 107 bytes or longer.

Willis and Helal (2005) used HF RFID tags with a memory capacity of 2000 bits in their experiment. XML data of 869 bytes was compressed sufficiently enough to be stored in 250 bytes using Chemical Markup Language (CML) and Huffman compression techniques, discussed in Section 2.11.

In the current study, XML can be used to store a single record of linen in a tag. A simple XML file describing four attributes needs 107 characters which can be stored as 107 bytes as shown in Figure 2.6.



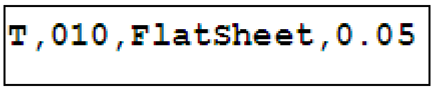
```
xml version="1.0" ?><LinenBin LinenStatus="T" LaundryCount="10"
    LinenType="FlatSheet" LinenWeight="0.05" />"
```

Figure 2.6: Example Linen Data in XML Format
Source: Author

2.10.2 CSV format

CSV is also a simple text format to store data values separated by commas. It is a very old technique from the earliest personal computer era. It was used in tape storage format for backing up data and exchanging data between different architectures. There is no internationally recognised standard for storing data in CSV generally. However for internet communications, there exists a standard called RFC 4180 (Shafranovich, 2005).

The CSV data as shown in Figure 2.7 has 20 characters. Data in different linen types (PatientGown, blanket) may vary with a possible upper limit of 30 characters, which should be sufficient in all cases to be stored in the limited RFID memory.



```
T,010,FlatSheet,0.05
```

Figure 2.7: Example Linen Data in CSV Format
Source: Author

This comparison of XML and CSV data format shows that XML takes 107 bytes whereas CSV takes 20 bytes to represent the same data. Clearly CSV format takes much fewer bytes to represent the same data when compared to XML. Further evaluation is done on their respective compression capabilities. Therefore, the literature was next researched for potential compression algorithms. Only compression algorithms suitable for CSV and XML and capable of being stored in the limited memory of RFID tags were considered.

2.11 Data Compression

There are various compression techniques for text and multimedia data like images, sound and video. Lossless data compression techniques are used where compressed data can be reconstructed to the original. This is necessary because compressed data stored in RFID tags needs to be reconstructed completely without losing any part of the original data. Lossy encoding achieves better compression than lossless encoding. The lossy technique is not appropriate if encoded data needs to be retrieved. The most common compression algorithms like Huffman and LZW are lossless data compression techniques and are further discussed.

Huffman Compression Technique

Huffman compression was proposed by Huffman (1952), where each character in a source data was stored along with probability of its occurrence. A binary tree was then constructed. Huffman compression is based on ASCII code and is used in text and image compression.

LZW Compression Technique

Lempel-Ziv-Welch (Ziv & Lempel, 1977) technique builds a coded dictionary scanning all the input data. Starting with a single character being represented in the dictionary, substrings are then searched in the input data to update the dictionary.

Huffman and LZW compression techniques are used internally in most common compression library functions available today in visual C# like Gzip and Deflate.

XML compression techniques are commonly used for sending data over the internet with low bandwidth and parsing time. XML is a better choice for sharing data among heterogeneous data types and the schema could be built in and stored along with the data. The next paragraph provides a specific overview of XML compression. Compression is achieved using different techniques such as lossy, lossless, schema-based or non schema-based.

Schema-based techniques give good compression if the format of the XML is known in advance. W3C has developed an XML binary format which reduces the verbosity of XML files, thus reducing processing overheads (Cokus & Pericas-Geertsen, 2005). This is ideal for devices like mobile phones and for processing data over low bandwidth as in the XML format discussed below with XMill and PPM techniques.

XMill Compression Technique

XMill is an XML compression technique used to compress large XML files of more than 20 KB (Liefke & Suciu, 2000). XMill is used for XML compression over the internet, with the objective of utilising lower bandwidth. According to Liefke and Suciu (2000) XMill has a better compression rate than Gzip. However, XMill is not ideal in this study where data is significantly less than the lower limit of 20KB.

XML PPM Compression Technique

XML PPM compression was proposed by Cheney (2006) where statistical text compression techniques on XML files were used. Again this was suitable only for large XML documents in the range of 1MB to 100MB.

There was insufficient literature on compressing small XML documents. Regardless of the compression algorithm employed, each character in the XML document takes 1 byte. A lossless compression is required in this experiment because of the need to recreate the original data. Thus RFID tags with low memory may not be ideal for storage in XML format which requires tags to have at least 1 KB of memory.

After reviewing CSV and XML data formats and the relevant compression techniques that can be used in the limited user memory of RFID tags, the literature was then reviewed to look into how data can be managed in a data-on-tag approach. Data structure and data manipulation are important aspects of building a data model. Hence Codds' requirements in the context of the data-on-tag approach are analysed in Section 2.12.

2.12 Requirements to build a Data Model

Codd (1980) proposed a combination of three components to build a data model.

1. Structural - Collection of all possible data structure types.
2. Manipulation - Collection of operators to retrieve and modify data stored.
3. Integrity - Necessary rules of integrity to insert-update-delete data.

2.12.1 Structural

After reviewing the two main data format types CSV and XML it was seen that CSV and XML formats were both suitable depending on the memory available on the tag for the particular application on the RFID tag. A CSV format has data values delimited by comma whereas an XML format can store details about the schema of the data in addition to the data values.

2.12.2 Manipulation and Integrity

Data in any information system is not static and needs to change according to the business process. Thus operations like adding, updating and deleting records are an important aspect of data management. In databases these operations are built into the database system which has a larger memory capacity. In addition, data integrity should also be taken care of and rules built into the database system. With the restricted user memory of RFID tags it may not be a possibility as of today. In the current study, the rules of manipulation and integrity are outside the data model and will be taken care of in the application accessing the data format in the user memory of RFID tags.

After reviewing the literature on RFID technology, RFID data characteristics, user memory, data formats and compression techniques, the research questions were restated to re-assess whether the relevant literature was covered.

2.13 Addressing Research Questions

What data format is suitable for a data-on-tag approach?

In the current chapter, the general literature on RFID technology, user memory and the two approaches of data-on-network and data-on-tag built the initial understanding of the data-on-tag concept. To store additional data other than EPC in the data-on-tag approach, CSV and XML data formats were discussed with the example of linen data to emphasis the size of data to be stored in the limited tag memory. Past literature on storing data were also presented.

How can data be stored in the RFID tag in a data-on-tag approach?

The need for a standard to store and share data among trading partners was discussed. Data was written as a string in the tag. More details on writing and reading data through a prototype are discussed in Chapter 4.

How data from RFID tags is read and written in a data-on-tag approach?

Writing and reading data in the selected format is covered in Chapter 4.

How accurate is the reading and writing of data using the data-on-tag approach?

RFID data characteristics covered in this chapter highlighted the voluminous, erroneous, duplicate data created through RFID implementation. In order to quantify the success of reading and writing through experimentation the evaluation criteria is discussed in Chapter 3. The experiment findings are presented and discussed in Chapter 5.

2.14 Summary

This chapter discussed the RFID system, its history, the classification of tags, RFID frequencies and standards which are a vital background to the research questions. A detailed analysis of the characteristics of RFID data and how data is handled has also been made.

The various ways in which RFID systems can be deployed and the two important approaches of data-on-network and data-on-tag were discussed. With the need for a data-on-tag approach, the user memory blocks and the block where user data is written and read were discussed. The two suitable data formats – CSV and XML were also discussed along with relevant compression algorithms and techniques.

The basic components of building a data model were discussed and the data format was considered in relation to this aspect. However data manipulation and integrity was not embedded into the data model being outside the scope of this research study. These were taken care of in the application software receiving the data.

The research questions were restated and a roadmap provided as to what was currently addressed and what will be covered in the next few chapters.

The next chapter discusses the research methodology which providing the guidelines to undertake the current thesis study. Thus it is important to discuss the methodology.

3 Research Methodology

3.1 Introduction

This chapter presents the research methodology used to fulfil the research objectives of this thesis study. It provides an overview of the research problem statements and the suitable methodology applied to undertake this research study. The research objective of this thesis study is to investigate an appropriate data format to store data in the limited memory of an RFID tag using the data-on-tag approach. The selected data format will then be implemented using RFID technology to evaluate its accuracy in reading and writing data using a data-on-tag approach. In order to undertake this research and answer the research questions, a proof of concept experiment involving RFID technology is undertaken and an appropriate methodology to guide the research process is sought.

3.2 Design Science as Research Approach

Design science is technology oriented and the main tasks involved are building an artefact/ framework/ model and evaluating it (March & Smith, 1995). Design science research produces an artefact that is partly or wholly created as compared to other research methodologies which involve an activity to understand a naturally occurring phenomenon. The artefact built in this thesis study was designed based on the research questions of storing and accessing data in a suitable format on RFID tags. The evaluation process determines how well the artefact has performed.

Several researchers like Hevner, March, Park, & Ram (2004), Vaishnavi and Kuechler (2008) and Peffers, Tuunanen, Rothenberger, & Chatterjee (2008) have proposed guidelines for the research requirements of design science. The current research study has followed these guidelines and discussed their importance with relevance to the research undertaken.

3.2.1 Design Science Guidelines

Hevner et al. (2004) stated that the order of the seven guidelines proposed by them are indicative only and that a researcher should know when and where to apply them. Vaishnavi and Kuechler (2008) proposed research guidelines in design science approach with five steps as shown in Figure 3.1. Peffers et al.

(2008) identified the dearth of design science research in IS literature. Peffers et al. (2008) proposed six steps after considering the guidelines proposed by earlier researchers like Hevner et al. (2004).

According to Vaishnavi and Kuechler (2008), the research study starts with the awareness of a problem, followed by suggestions for solutions which are supported with existing knowledge in the allied field to create a proposal and a tentative design. This proposal or design could be a creative or novel idea which will be further improved upon in the next stage of development. An artefact is then built and its performance evaluated through experiments. There could be informal evaluations taking place during the construction process. Vaishnavi and Kuechler (2008) says further formal evaluation takes place once the artefact is built. Suggestions will be given in an iterative process until an acceptable solution indicates conclusion by the researcher. Thus this process contributes new knowledge.

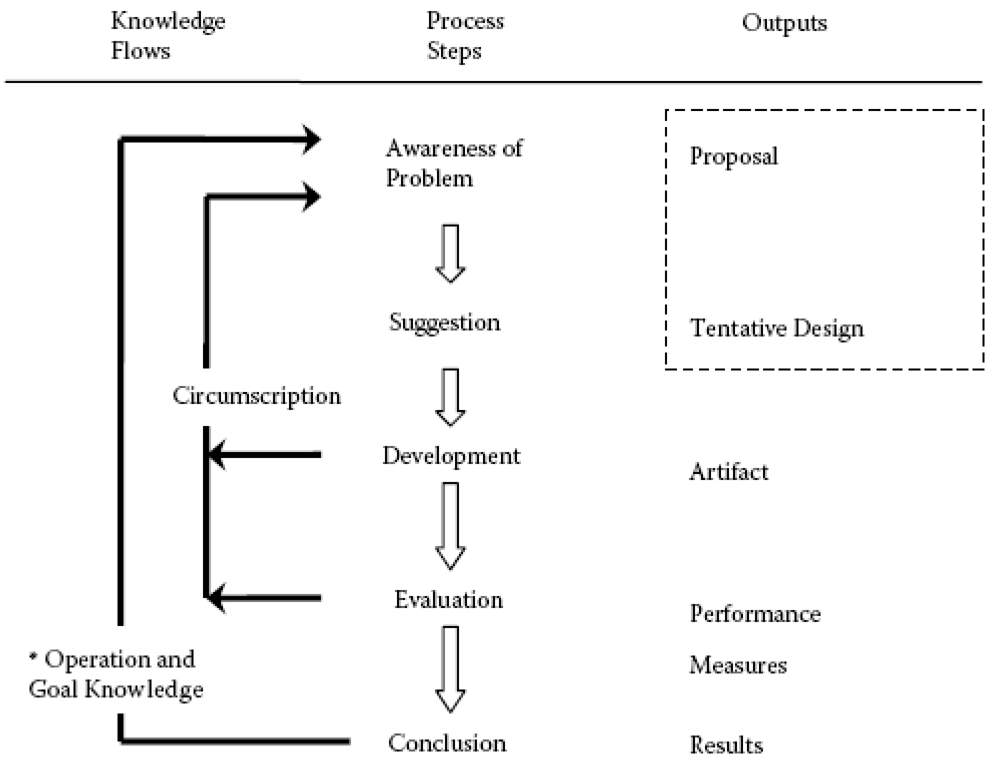


Figure 3.1: Design Science Research Methodology

Source: (Vaishnavi & Kuechler, 2008, p. 20)

Research Problem

The objective of design science research is to solve a previously unresolved business problem. A specific research problem is identified and the search for a solution justifies the creation of an artefact. The new artefact created needs to address this as the main requirement of the business problem. The rationale in developing a solution is of interest to the researcher and the audience of the research outputs.

Design an Artefact

Artefacts are not independent of people or organisations but are interdependent with them to satisfy business requirements. Artefacts constructed in design science are proof of concept work demonstrating the new innovative idea initially in a prototype stage. The prototype developed will require improvements if they are to be deployed in a business organisation.

Evaluation

Evaluation of the design is a major part of the research study which integrates the newly created artefact with the technical infrastructure of the business environment. IT artefacts can be evaluated in terms of performance and reliability. Experimenting in a controlled setting can be used to study the performance of the artefact. Evaluation criteria for the prototype have to be determined and the evaluation process requires metrics to measure the performance and efficiency of artefact built (March & Smith, 1995). Evaluation could be quantitative or qualitative (Peffers et al., 2008). Quantitative evaluation is used when the desired solution can be compared with a previous solution and qualitative evaluation is applied when an artefact is built to solve a problem not addressed earlier.

Iterative

At the end of the evaluation the researcher can decide whether to iterate further to improve the artefact or to conclude with a communication. The decision to iterate will depend on the feasibility to do so. According to Hevner et al. (2004) design is a search process to identify the most suitable solution to a given problem. It is a two stage iterative process of ‘generate design alternative’ and ‘test’ cycle. The design artefact is refined by changing requirements of the design problem. An optimal solution is achieved through good and feasible design.

Research Contributions

Research should be innovative and contribute new knowledge to a problem not solved earlier. The contribution of design science can be the design artefact itself which will add to and improve the existing knowledge base or use existing knowledge in an innovative way. Metrics used to evaluate the artefact are also crucial contributions which help in substantiating the design science artefacts to solve business problems.

Communications

Research needs to be presented to a target audience in a scholarly research paper or an academic journal which could help to further refine and improve the artefact in future.

Table 3.1 compares the different guidelines of design science by Vaishnavi & Kuechler (2008) and Peffers et al. (2008) against Hevner et al. (2004).

Table 3.1: Comparison of Guidelines of Design Science

Source: Author

Hevner et al. (2004)	**Vaishnavi & Kuechler (2008)**	**Peffers et al. (2008)**
Design as an artefact	Development	Design and development
Problem relevance	Awareness of problem	Problem identification and motivation
Design evaluation	Evaluate	Evaluation, demonstration
Research contributions	Part of conclusion	Part of communication
Research rigor	Suggestions	Demonstration
Design as search process	Suggestions and the iterative process	Demonstration, evaluation and iteration if necessary
Communication of research	Part of conclusion	Communication

The design of an artefact, research problem, evaluation and iterative process of design science is common to the three approaches. The steps outlined by Peffers et al. (2008) also coincide with the guidelines by Hevner et al. (2004) except for demonstration which could be embedded within the design evaluation of the artefact. Demonstration also gives rigor to research and aids the decision to iterate the research process. The guideline of research contribution as proposed by Hevner et al. (2004) is a part of the communication process (Peffers et al., 2008) or conclusion (Vaishnavi & Kuechler, 2008) in the other two approaches. Communication is a separate guideline by Hevner et al. (2004) and Peffers et al. (2008). However according to Vaishnavi & Kuechler (2008) communication is covered under conclusion. Peffers et al. (2008) have a separate guideline on demonstration which is part of evaluation in the other two approaches. Research rigor (Hevner et al., 2004) could be part of evaluation in other approaches and is brought about by suggestion (Vaishnavi & Kuechler, 2008) and demonstration (Peffers et al., 2008).

In the current research study, in order to build an artefact for designing a data format for writing and reading data on RFID tags, the guidelines of design science methodology are applied.

3.3 Business Problem

The foremost requirement to conduct research is a need for a research motivation and objectives which are based on a business problem in the current research study as stated in Section 1.3. The current Section 3.3 outlines the business problem.

The business problem of a commercial hospital laundry is considered. All linen belongs to the laundry company. It is known within the industry that linen within the hospital is not precisely accounted for and often lost, stolen or misplaced (Flynn, 1996). The laundry company is not able to bill the hospital for such lost linen. The current inventory management practice of allowing ‘n’ sets of linen for every bed serviced creates a burden of increased overhead costs and does not provide an incentive to trace lost or misplaced linen. A successful RFID deployment could reduce the number of sets of linen required for each serviced bed, allowing the laundry company to trace lost or misplaced linen. It thus reduces stock and cost overheads for the laundry company.

In the existing process, clean linen is replaced to the original order level and not by the number of soiled linen collected. The commercial laundry bears the cost of the lost linen. No manual count of soiled linen is made as counting soiled linen is labour intensive and a health hazard. RFIDs can be an ideal technology to keep track of linen collected and to count linen at the time of pickup as it requires no line of sight. Thus some local data storage and processing needs to be done at the laundry bin. The most important business requirement for the laundry company is to bill their clients for linen washed and to replenish linen supplies to the stock level *returned* rather than the stock level *ordered*.

The accounting system in the laundry company is quite dated and data structures are not able to be updated without major work in the backend database and full scale deployment of Savant middleware for RFID implementation. Therefore the devised system needs to interface easily with existing systems and cause little or no disruption to core business accountancy applications. Therefore, the prototype developed should be able to read and write to linen tags using RFID technology independently of the backend database and enterprise system. The log file of all processing generated by the prototype should be capable of being used by the enterprise system to update its records.

3.4 Design an Artefact

In this research study the artefact was developed as a proof of concept to read and write necessary data on tag with a suitable data format for the data-on-tag approach.

The current laundry system collects linen in laundry bins with no specific information on linen collected. The re-order is based on the original stock levels at the hospital. If any linen goes missing it is not accounted for. The proposed experiment in the current research study will collect linen along with the count of each linen type and weight of the total linen which will help the laundry company in billing the hospital.

The business process flow diagram for the proposed laundry bin is summarised in Figure 3.2.

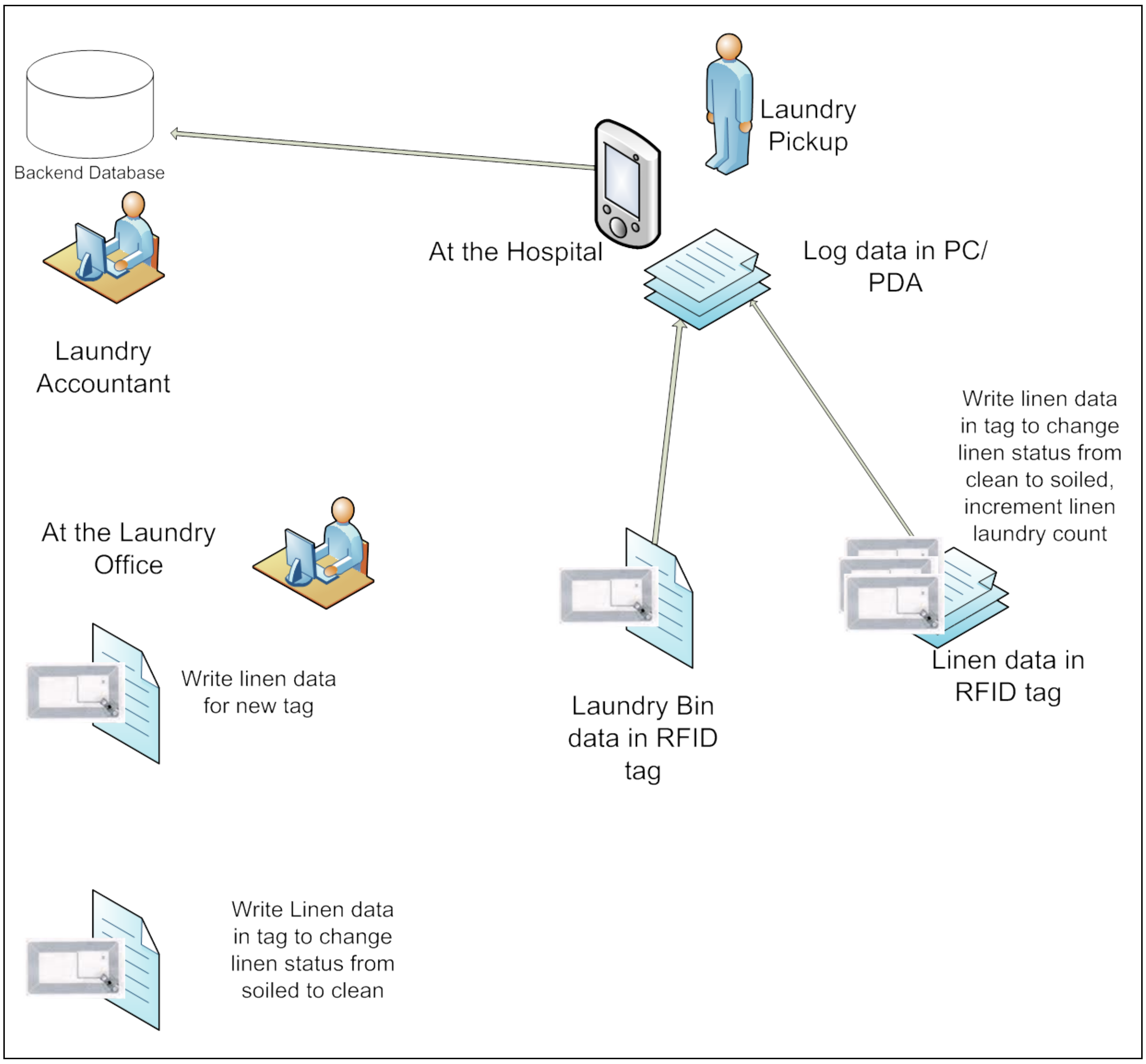

Backend Database
Laundry Pickup
At the Hospital
Log data in PC/ PDA
Laundry Accountant
Write linen data in tag to change linen status from clean to soiled, increment linen laundry count
At the Laundry Office
Laundry Bin data in RFID tag
Linen data in RFID tag
Write linen data for new tag
Write Linen data in tag to change linen status from soiled to clean

Figure 3.2: Proposed Laundry Business Process Flow Diagram

In the proposed laundry business process flow diagram in Figure 3.2, several important aspects of the system can be identified. Only a part of this proposed system is developed in this study as a proof of concept for the data-on-tag approach. Integrating the RFID data into the backend database is not part of the current research study. The main business scenario undertaken in this research study is collecting soiled linen from a hospital, and keeping a count and weight of linen collected. The proposed laundry system and its important aspects are described below. The major requirements of the laundry bin experiment are in the area of laundry company administration and at the linen pickup point.

Business Processes

The various business processes for the proposed laundry system are identified and explained: These are - At the hospital where soiled linen is identified and picked up when the laundry bin is collected and at the laundry office where new linen is added and status of linen changed from ‘soiled’ to ‘clean’ at the end of the wash process.

At the hospital, soiled linen is collected in the laundry bin. Data from the linen and laundry bin are collected and processed for billing purposes. Linen status in the linen tag should change from ‘clean’ to ‘soiled’ and other processing like calculating linen count and weight are done. Soiled linen is collected at the hospital premises in various linen bins. Each linen bin can carry a large number of linen pieces. Hence, at pickup, data from linen tags and bin tag have to be aggregated into the PC/PDA for billing. Data from the RFID tags identifying both the linen and laundry bin are brought together at the time of pickup for further processing. The person collecting the soiled linen first connects their PDA to the RFID reader attached to the laundry bin. Data about all the linen in the laundry bin is picked as a log file in the PC/PDA and processed to count the different linen by their type (Example: PillowCase, FlatSheet). Processing also collects total weight of dirty linen and prints a receipt which is attached to the laundry bin. The count of soiled linen is used to set the clean linen replacement level. The file stored in the PC/PDA can be useful for further processing at the enterprise level.

The data log is to be stored in the reader and in the actual implementation of the experiment is stored in the PC/PDA. Thus when the reader is connected to the PC/PDA it can download the data file for further local processing and for billing the hospital. The log file generated is an audit file of all activities which took place during the local processing and can be used for later scrutiny.

At the laundry office, there are two processes undertaken. Firstly the laundry company needs to replenish linen from time to time which requires a new tag to be created for each new piece of linen. Secondly at the end of a wash cycle in the laundry company, data on the tag needs to be updated, changing linen status from ‘dirty’ to ‘clean’. Deactivation of tags on worn-out linen is not included in the experiment because it is considered straight forward.

The aim of the experiment in the current research study is to investigate whether all necessary information about the linen or bin can be stored in an RFID tag, being the most suitable data format and the accuracy of such a data format in reading data from and writing data into the linen tag.

Linen is identified electronically and has an RFID tag attached to store information about it. The RFID tag attached to the linen is tough enough to withstand high temperature, detergents and water of the washing process.

Laundry bin is also identified using an RFID tag. This is required to relate linen to its laundry bin for billing purposes.

The data from linen and laundry bin tags is picked up by the prototype for local processing which can be a done by a legacy system in the laundry company.

Passive RFID tags are used in industry laundry applications at present, but the approach taken is data-on-network and only tag ID is necessary to process the data into useful information at the enterprise level. The approach in this thesis study is to have data stored on the tag attached to the linen to enable some level of local processing. Active RFID tags have large memory but need an on-board battery and it may not be possible to attach such tags to linen which goes through the harsh washing process. Passive RFID tags with memory capacity and no battery or power constraints are more suitable. However, the question is how much data can be stored in a passive tag. An RFID reader and tags of UHF frequency were used as their maximum read range is 1.5 meters and write range is 0.5 meter. This distance is theoretically sufficient in the current laundry bin experiment for the RFID reader attached to the laundry bin to read and write tags attached to linen as they were dropped into the bin.

As discussed in Section 2.12 regarding suitable data models, the three components of Codd’s theory – structure, manipulation and integrity are recalled. The format in either XML or CSV is considered. Data in XML or CSV format has information about the linen stored as a string. The other components of Codd’s theory about data manipulation and data integrity are not part of the data model for this research study. The prototype, through the RFID reader, can retrieve and read the string and then depending on the format used, translate the retrieved data into meaningful information with relevant data types like numeric and boolean. Thus data manipulation is done through the prototype by writing to tag and reading from the tag.

In the current study, data in the tag is stored as a single string with comma delimitation to store each of the fields necessary to include all the required data. This is to work around the limited user memory available in the tag. At pickup, data in the tag is updated and copied into the PC/PDA as a CSV file. Thus the CSV data from the linen tag will be included into a log file in the PC/PDA. The linen count and billing is done at this point. Duplicate tag data is eliminated at this stage. This file can be further uploaded at the laundry office to keep an audit trail of the transactions that take place each day at each location. Although creating an audit trail is an important feature of the system, this is out of the scope for the current research study.

The two data formats are discussed and possibilities of the different formats at different levels are examined in the following paragraph as shown in Figure 3.3.

Analysis of Data Format

After analysing data formats, compression techniques and requirements in building a data model suitable for the current research study, data in the RFID tag can be of two possible formats – XML and CSV. The two formats were detailed in Section 2.10.1 and Section 2.10.2 and illustrated in Figure 2.6 and Figure 2.7. CSV format required 20-30 bytes to represent the data in the current laundry bin process. The same data in XML format required 107 bytes.

CSV or XML data format is written as a string in the RFID tag. Although it is theoretically possible to store long data strings in user memory with an XML data format, practically, long strings can be written in piecemeal over several memory blocks. However, writing in pieces in the user memory of the tag in the current research experiment is not suitable as multiple tags are involved and there is no guarantee of the reader reading and writing the same tag for the complete write procedure.

In either format, the RFID reader can read data and transfer the data to the prototype which then generates a data log that can be applied in processing at the enterprise level.

The data formats at different levels are covered in depth in Section 4.4.

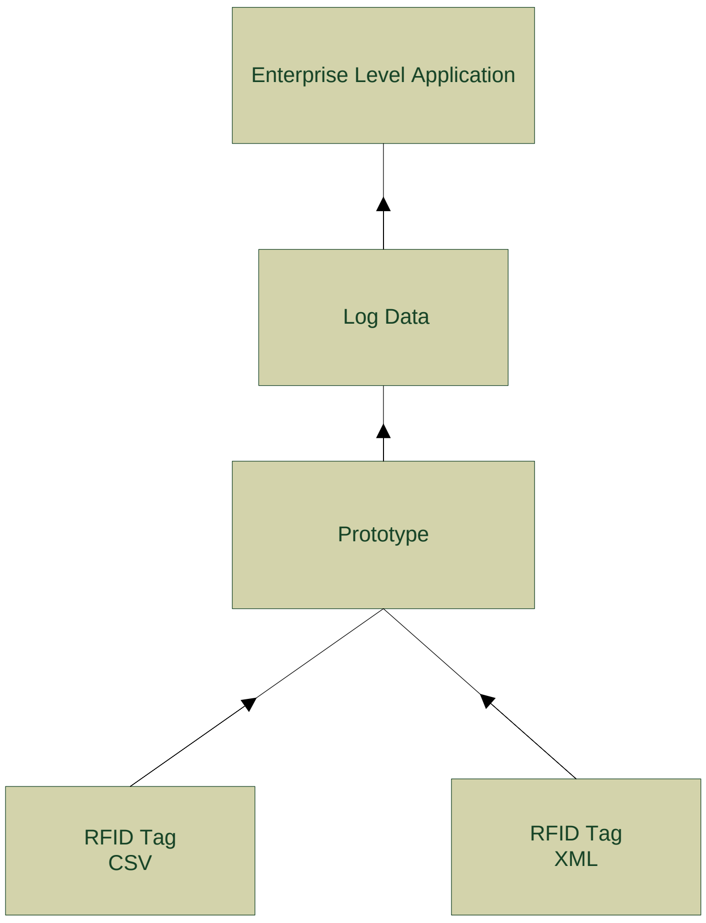


Figure 3.3: Data Formats in Different Levels

3.5 Demonstration and Evaluation

Demonstration shows the efficiency of the artefact built and is proven through experimentation according to Peffers et al. (2008). According to design science guidelines, evaluation criteria for the prototype has to be determined and the evaluation process requires metrics to measure the accuracy of the artefact built (March & Smith, 1995). The prototype discussed will need evaluation to validate the research process. The proposed data format is checked for technical feasibility as suggested by Moody (1998) through the prototype and how well the experiments perform in reading and writing data in the tag user memory. To measure the read Tag ID rate following formula is used:

> Read TAG ID percentage = Number of tags IDs read / Number of actual tags x 100

To measure the read and write user memory rate the following formulae are used:

> Read user memory percentage = Number of successful user memory read / Number of actual tags x 100
>
> Write user memory percentage = Number of successful user memory write / Number of actual tags x 100

RFID read rate reported in academic studies and industry implementations has not been 100% because of missed readings and was referred to in Section 2.8.3. According to Clarke et al. (2006) in a study involving UHF tags, there have not been instances of 100% successful read rates even with empty cartons. Bottles containing liquid had only a 25% success read rate. Rice filled jars had an 80.6% success read rate. There is no established read rate for linen in the literature to compare with. Moreover, the success rates in literature relate only to the reading of tag IDs. The current experiment goes beyond merely reading tags as the business scenario and proof of concept requires successful read and *write* to tag user memory.

The log file generated by the prototype has additional information of a date-time stamp to trace the date and time when the read and write process took place, in the simulated experiment. This log file is manually evaluated to calculate the successful read and write rate. The log file has details of the contents of the user memory and keeps a record of changes in data after write procedure. This is useful for quantitative analysis for recording write success rates. It was also used to evaluate the reasons for write failure.

The read and write success rates are used for quantitative analysis in Chapter 4 and the discussions are in Chapter 5.

3.6 Contributions and Communication

Hevner et al. (2004) stated that if the artefact is evaluated satisfactorily and research process is concluded, it should be communicated as a research output. Stating the research contributions and adding new knowledge is the final phase. The results from conducting the experiments give insights on how well the prototype performed. However, if it the research outputs are not satisfactory, it needs to iterate back with better suggestions to improve the artefact and evaluation. Otherwise it adds to future work and opportunities for improvements of the current research. This thesis resulting from the current research study is itself a communication of the research process and outcome.

3.7 Summary

In this chapter, the research approach and process for the current research study using design science was outlined and discussed. The research question was applied to a business scenario of a laundry company. The various processes in the business scenario were outlined. The research process was analysed using the steps proposed by Peffers et al. (2008). The research process was also compared with those proposed by Hevner et al. (2004) and Vaishnavi & Kuechler (2008).

The application of RFID system was described with this scenario. The main purpose of the research study through a data-on-tag approach was discussed.

The following chapters cover in more detail how the prototype is built using a suitable data format to store data in an RFID tag memory and its evaluation through empirical observations and research findings.

4 Experiment Design

The goal of conducting the experiment is to investigate the research questions as to how data can be managed in a data-on-tag approach.

The business scenario of a laundry bin is considered where each item of linen is identified by an RFID tag containing relevant data about that piece of linen. The laundry bin is also identified with an RFID tag containing information about the bin. The accuracy of reading and writing data using the selected data format is checked through an experiment using an RFID system comprising of a reader and tags.

4.1 Experiment Set Up

In the following sections, the complete RFID system used in the experiment will be described. It consists of one RFID reader and a SDK from Tracient, two types of tags from Skyetek and Intermec and one PC. Section 4.6 describes the prototypes developed using the SDK from Tracient to read and write data into the RFID tags.

4.1.1 Tracient RFID Reader

For the experiment, a handheld Padl-R-UHF RFID reader from Tracient Technologies Ltd. Christchurch was used as shown in Figure 4.1. The reader can be connected through Bluetooth, virtual port (RS232) or USB to a PC/PDA. The reader can also operate independently if the intention is only to collect data from RFID tags. The reader used in the experiment is able to read UHF tags. The UHF tag classification is defined in Section 2.3.2. The reader operates within 862-955 MHz frequency with a maximum reported read range of 1.5 meters and a write range of 0.5 meter.

RFID interrogators (readers) can operate in one of two modes. Readers can be set to either ‘read continuously’ or they can be set to ‘read once’. The ratio of missed readings can be improved if the read mode is continuous. Duplicate readings can then be eliminated before processing the raw data.

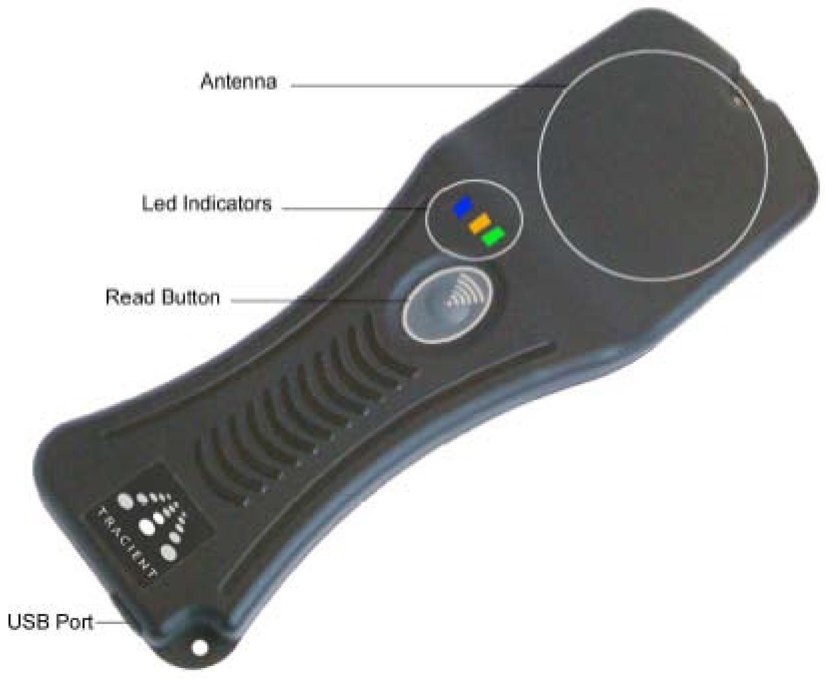


Figure 4.1: Padl-R UHF Tracient Reader
Source: (Tracient, 2007b)

Further technical details of the reader are summarised in Appendix C.

4.1.2 RFID Tags - Skyetek and Intermec IT67

The RFID tags to be used in the experiment are of passive type. Passive tags are defined in Section 2.3.1. Passive tags were chosen for their relatively lower cost and the fact that this type of tag does not need an external power supply. The reader supports UHF for a longer range and is able to read multiple tags simultaneously. Tags chosen were UHF to be compatible with the UHF reader. Because this research experiment takes a data-on-tag approach, the tags chosen were those that had the largest user memory available in the current market to store additional data apart from the tag ID.

Table 4.1 details the various RFID tags considered suitable for the experiment based on the required criteria of standards and having sufficient user memory.

Table 4.1: RFID Tags Suitability for Experiment

Source: Author

Tag Name	Frequency	User Memory	Standard	Acceptable Y/N
UPM Rafsec	UHF	None	ISO 18000 6C	N
Intermec 2”	UHF	None	ISO 18000 6C	N
Intermec 4”	UHF	None	ISO 18000 6C	N
Alien (ALN-9540 - "Squiggle™"	UHF	None	ISO 18000 6C	N
SkyeTek	UHF	1728 bits	ISO 18000 6B	Y
Intermec IT67	UHF	512 bits	EPC Gen 2/ISO 18000 6C	Y

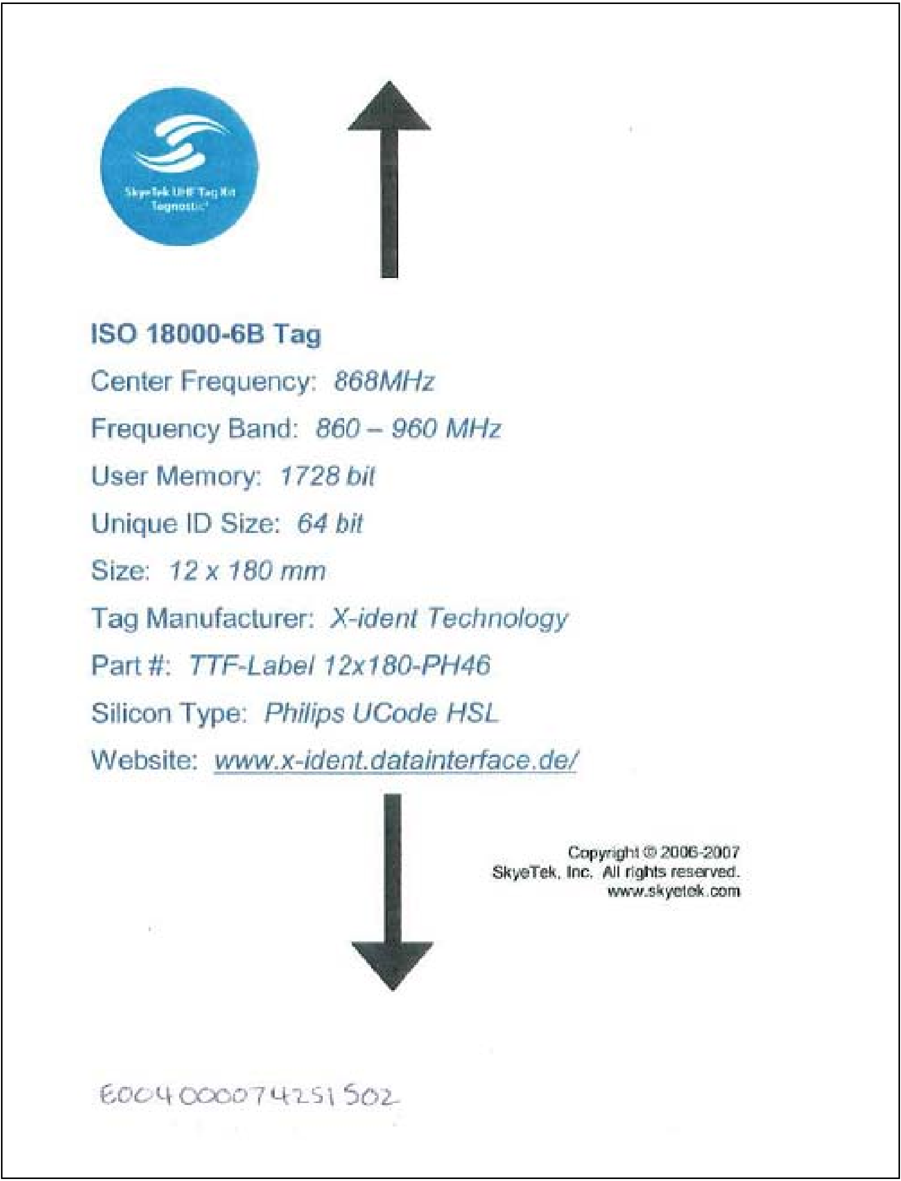
Tags like UPM Rafsec, Alien World Tags (ALN-9540 – ‘Squiggle™’) were not suitable as they did not have sufficient user memory. Skyetek UHF and IT67 UHF Intermec tags had acceptable user memory and were compatible with Tracient reader and the software provided by Tracient[2]. Thus only Skyetek and Intermec tags were found suitable for the experiment. Only 8 Skyetek tags were obtained through the University and 10 Intermec IT67 were obtained from the local Intermec vendor, all of which were used in the experiment. The technical details of the Skyetek and Intermec IT67 tags were analysed in greater detail to fully understand their relevance to the experiment.

Skyetek UHF Tag

Skyetek RFID tag complies with EPC Gen 2 / 18000-6B standard and operates within the 860 MHz to 960 MHz frequency range. Its dimensions are 12mm x

[2] Although there are tags like Tego (32 KB) with user memory of 1 KB or more do exist, they were not available at the time the experiment (September 2009).

180 mm. The user memory size is 1728 bit which translates to a storage capacity of 216 bytes. A picture of the Skyetek tag used in the experiment for the current thesis study is given in Figure 4.2.



ISO 18000-6B Tag

Center Frequency: *868MHz*

Frequency Band: *860 – 960 MHz*

User Memory: *1728 bit*

Unique ID Size: *64 bit*

Size: *12 x 180 mm*

Tag Manufacturer: *X-ident Technology*

Part #: *TTF-Label 12x180-PH46*

Silicon Type: *Philips UCode HSL*

Website: *www.x-ident.datainterface.de/*

Copyright © 2006-2007
SkyeTek, Inc. All rights reserved.
www.skyetek.com

E004 000074251502

Figure 4.2: Skyetek UHF Tag

IT67 UHF from Intermec

IT67 UHF tag as shown in Figure 4.3 complies with EPC Gen 2 / 18000-6C standard and operates within the 860 MHz to 960 MHz frequency range, similar to Skyetek tags. Its dimensions are 72.3 mm x 94.5 mm. It can withstand temperatures varying from -40 deg C to 85 deg C and operating temperatures are -40 deg C to 66 deg C. The IT67 Intermec tag has a 512 bit user memory chip. It is designed to provide both edge[3] and normal performance and has a range of nearly 8 metres. It has a tough external casing to withstand harsh environments but is not watertight and therefore not suitable for laundry purposes.

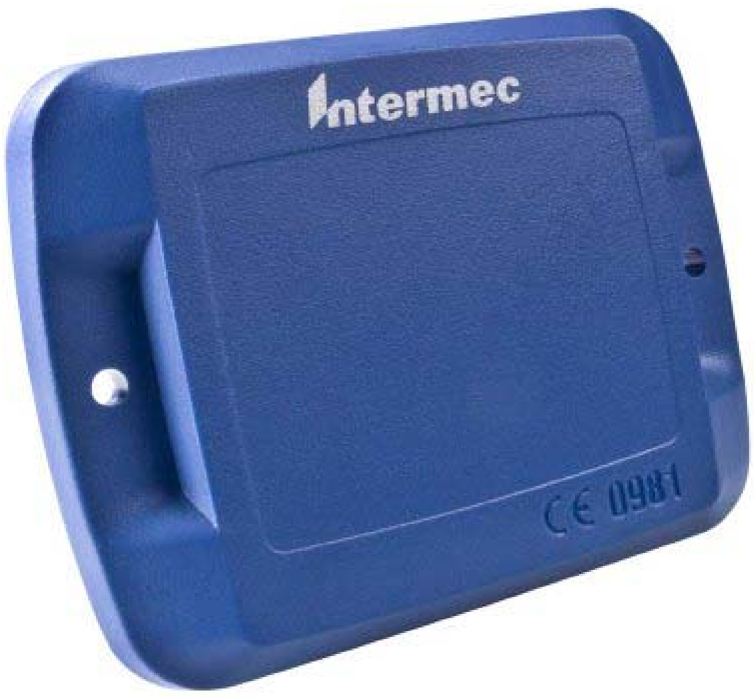


Figure 4.3: IT67 Intermec UHF Tag

Source: (*IT67 UHF tag)*

Both Skyetek and Intermec tags operate at the UHF frequency and are compatible with the Tracient reader. However the setting in the Control Panel of Tracient application is different for the two tags as they are of different standards. Skyetek tags are of EPC Gen 2 18000 6B and Intermec IT67 tags are of EPC Gen 2 18000 6C standard. Hence, the two types of tags cannot be used together in the laundry bin experiment.

The user memory in Intermec tag is only 512 bits which translates to 64 bytes which is smaller than the user memory of Skyetek tags.

[3] RFID tag data is captured close to the source and middleware software cleans and sends only required and meaningful data to the centralised enterprise system. This is called edge processing. Edge processing reduces network traffic.

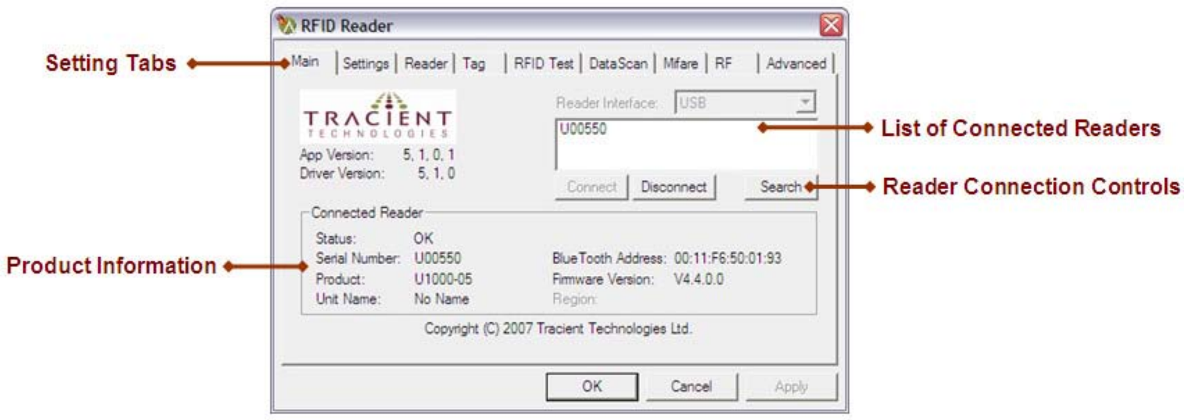
4.2 Tracient Application Software

The RFID discovery kit from Tracient has three main applications which allow configuration of the reader, collecting RFID tag data and downloading data to a secondary storage in a PC/PDA. These are respectively the Control Panel, RFID Wedge and RFID Sync. The SDK of the discovery kit has accessible source code to use Tracient RFID products and can be edited to write new applications.

Before using the SDK software provided by Tracient, the three applications were used with both the Skyetek and Intermec tags to verify their compatibility. Each application was tested to check if they were required in the experiment.

4.2.1 Control Panel

The Control Panel is an application to configure the reader to identify RFID tags of different standards. A screen shot from the Control Panel application is given in Figure 4.4. In the current thesis study, writing to and reading from user memory was necessary and hence the tag standard, start block to read user memory and number of bytes to be read from user memory were set in the Control Panel application. Various tags were tested to read tag ID, read user memory and write to user memory. The main functions of Control Panel are RFID test, datascan and reader log. The RFID test is used to test an RFID tag using the settings applied, and can set the reader to read/write in single mode or continuous mode. Datascan is used if user memory is required to be read while reading tag ID. Reader log can only store tag IDs. However if user memory needs to be stored then this information is stored externally in a file in a PC/ PDA. The Control Panel was used before the experiment started to set the standard of the RFID tags.


Setting Tabs
Product Information
List of Connected Readers
Reader Connection Controls
Main | Settings | Reader | Tag | RFID Test | DataScan | Mifare | RF | Advanced
Connect | Disconnect | Search
OK | Cancel | Apply

RFID Reader

App Version: 5, 1, 0, 1
Driver Version: 5, 1, 0
Reader Interface: USB
U00550

Connected Reader

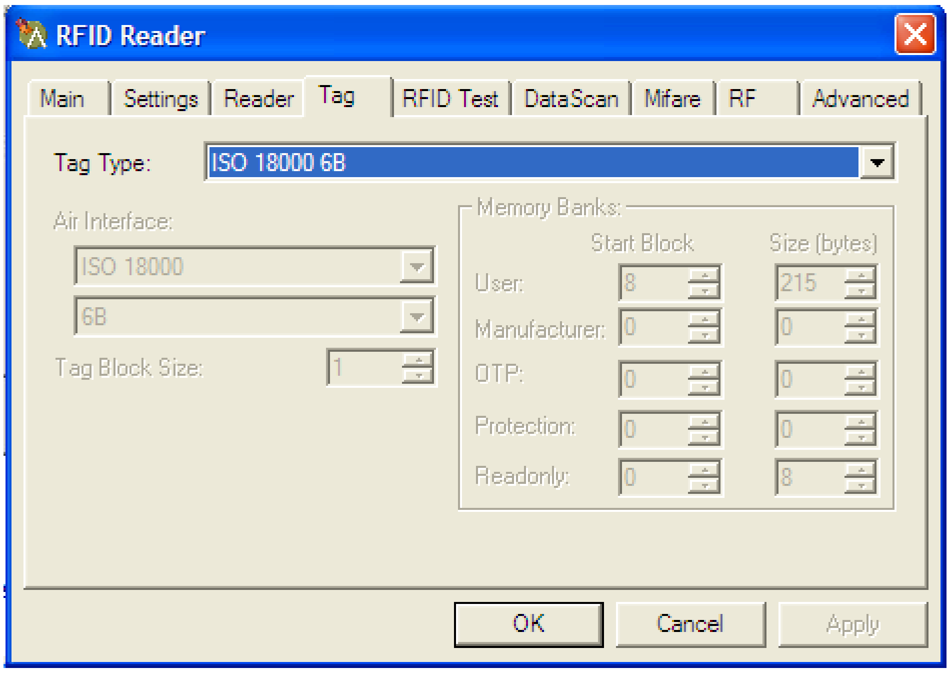
Status: OK
Serial Number: U00550
Product: U1000-05
Unit Name: No Name
BlueTooth Address: 00:11:F6:50:01:93
Firmware Version: V4.4.0.0
Region:

Copyright (C) 2007 Tracient Technologies Ltd.

Figure 4.4: RFID Control Panel

Source: (Tracient, 2007a)

The setting for Skyetek ISO 18000-6B through Control Panel is shown in Figure 4.5.


Main | Settings | Reader | Tag | RFID Test | DataScan | Mifare | RF | Advanced
OK | Cancel | Apply

RFID Reader

Tag Type: ISO 18000 6B

Air Interface: ISO 18000 / 6B

Tag Block Size: 1

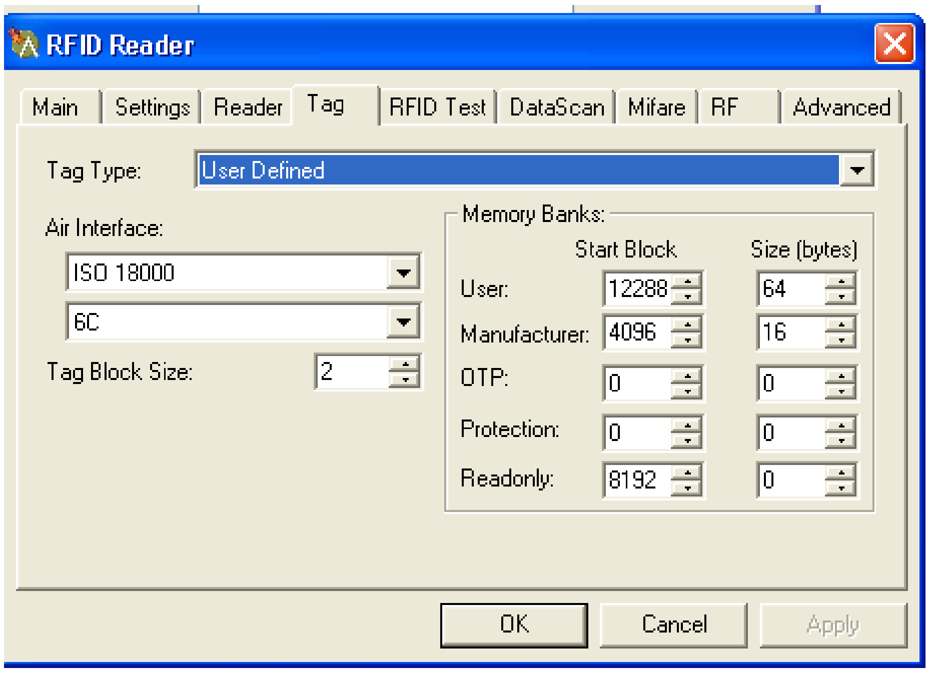
Memory Banks:

	Start Block	Size (bytes)
User:	8	215
Manufacturer:	0	0
OTP:	0	0
Protection:	0	0
Readonly:	0	8

Figure 4.5: Skyetek ISO 18000-6B Tag Setting in Control Panel

Source: Tracient Control Panel Software

The setting for Intermec IT67 ISO 18000-6C is given in Figure 4.6.


RFID Reader

Main | Settings | Reader | Tag | RFID Test | DataScan | Mifare | RF | Advanced

Tag Type: User Defined

Air Interface: ISO 18000, 6C

Tag Block Size: 2

Memory Banks:

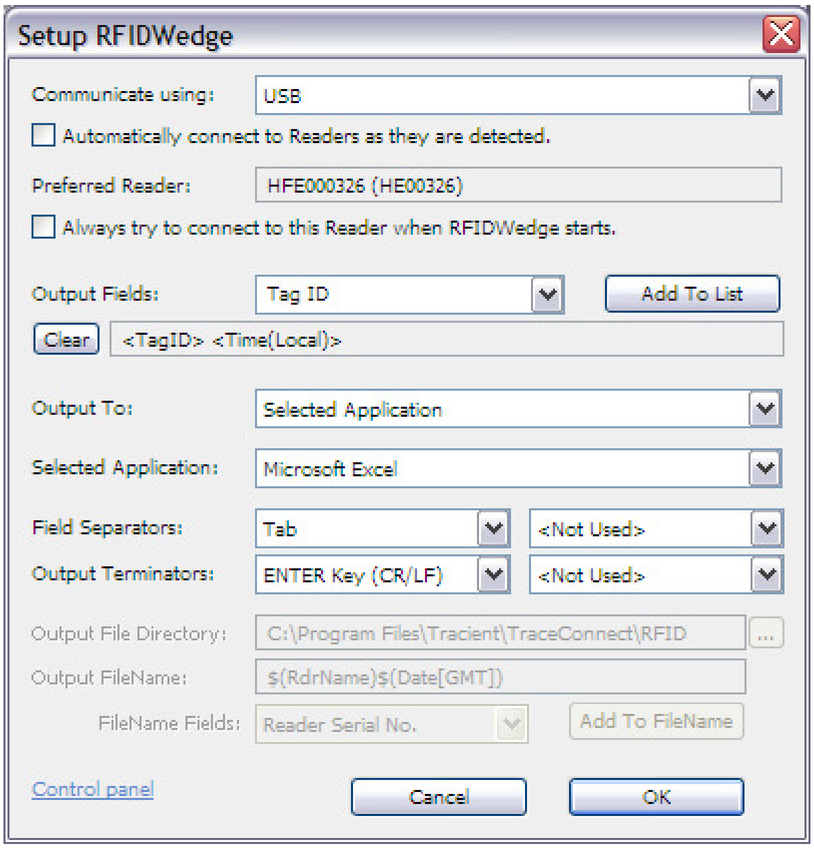
	Start Block	Size (bytes)
User:	12288	64
Manufacturer:	4096	16
OTP:	0	0
Protection:	0	0
Readonly:	8192	0

OK | Cancel | Apply

Figure 4.6: Intermec ISO 18000-6C Tag Setting in Control Panel
Source: Tracient Control Panel Software

4.2.2 RFID Wedge

The RFID Wedge application allows the transfer of tag data to an open foreground application or file that can display text or XML. Wedge can select a reader either connected through Bluetooth, virtual port (RS232) or USB. It can send selected fields to an output file as shown in Figure 4.7. The tag ID and timestamp showing when the tag was read by the reader are selected as fields in the output file. The output file can be in a different formats; text file, Excel or XML. If the datascan option is set on the reader, then data from the user memory can also be included as an output field in the output file. The data in the user memory is a set of information written as a string. However the Wedge application is designed for read only mode and it cannot write to the user memory. The laundry bin experiment requires data to be read and *written* to user memory. Therefore this application is not sufficient for this experiment.



Setup RFIDWedge

Communicate using: USB

☐ Automatically connect to Readers as they are detected.

Preferred Reader: HFE000326 (HE00326)

☐ Always try to connect to this Reader when RFIDWedge starts.

Output Fields: Tag ID

<TagID> <Time(Local)>

Output To: Selected Application

Selected Application: Microsoft Excel

Field Separators: Tab | <Not Used>

Output Terminators: ENTER Key (CR/LF) | <Not Used>

Output File Directory: C:\Program Files\Tracient\TraceConnect\RFID

Output FileName: $(RdrName)$(Date[GMT])

FileName Fields: Reader Serial No.

Figure 4.7: RFID Wedge Application Settings

Source:(Tracient, 2007a)

4.2.3 RFID Sync

The RFID Sync application allows the download of data collected in the reader log. This could include data like tag ID, user memory, reader ID, reader name and timestamp. The file could be in a different format like text or XML. RFID Sync cannot be used in this experiment as the solution requires data to be read and written into the RFID tag, for the same reason mentioned in Section 4.2.2. However the feature of collecting data about a tag and its user memory data has been incorporated in the prototype developed. The prototype kept a log of all data read from RFID tags into a text file.

4.2.4 Tracient SDK

If the experiment involved only reading tag IDs and user memory, then the three applications would be all that was needed. However the experiment involves reading and writing to tag user memory. Thus to read and write data in the tag user memory, Tracient provides a SDK with source code in various programming languages like C++, C# and ActiveX/Java/script. The SDK kit provides Dynamic Link Library (DLL) to support RFID reader to enable developers to build applications in C++, Visual C#.Net and Visual Basic.Net. Activex/Javascript provides a browser based interface for the reader to interact with tags.

In the current experiment of the laundry bin, Visual C# .Net 3.5 framework is used to build prototypes which read and write onto the RFID tags. The SDK libraries consist of methods to read tag ID, read from user memory and write to user memory. As the approach in the current experiment is data-on-tag, the library methods to read/write user memory need to be assigned with the correct input parameters to locate the block address and the number of bytes to write in that specific block.

4.2.5 Read/Write User Memory Functions

The read and write functions defined in the Tracient RFID SDK are described in Figure 4.8 and Figure 4.9 respectively. The main task of these functions is to identify the block in the memory bank where user memory resides and to write the data as a string. The string is called StringBuilder, a better approach to handle strings in C#.

```
DWORD RFID_ReadTag_UserMemory
 ( HANDLE cyDevice,
System.Text.StringBuilder lpIDBuffer,
LPVOID lpDataBuffer,
DWORD dwBlockAddress,
DWORD dwBytesToRead,
out DWORD lpdwBytesReturned,
    out DWORD lpdwBlocksReturned);
```

Figure 4.8: RFID_ReadTag_UserMemory()

Source: (Tracient, 2007c)

The starting block address has to be given in `dwBlockAddress` and is specific to the RFID tag's datasheet[4].

[4] Datasheet provides details of the RFID tag, user memory layout in banks and blocks of data.

```
DWORD RFID_WriteTag_UserMemory
(HANDLE cyDevice,
System.Text.StringBuilder lpIDBuffer,
LPVOID lpDataBuffer,
DWORD dwBlockAddress,
DWORD dwBytesToWrite,
out DWORD lpdwBytesWritten,
out DWORD lpdwBlocksWritten);
```

Figure 4.9: RFID_WriteTag_UserMemory()

Source: (Tracient, 2007c)

Data in the user memory was written as a string following the formats of CSV or XML supporting the earlier research described in Section 3.4.

4.3 Data Format Selection

As discussed in Section 4.2.4, the three applications supplied with the Tracient reader can only read data from a tag and store it in a file. In the laundry bin experiment the challenge was to read from and write to RFID tags. As storage space on the tags was limited to 1728 bits for Skyetek tags and 512 bits for Intermec tags the type of data and its format was crucial to the successful development of the prototype. Data could be written in various formats like CSV or XML as discussed in the Section 3.4. Data formats are discussed in more detail in Section 4.3.2.

4.3.1 Compression Techniques

In order to investigate the performance of various compression techniques on small files, C# library compression techniques like Gzip and Deflate were undertaken (as the prototypes developed for the experiment were in C# SDK provided by Tracient). Gzip and Deflate compressions techniques use industry standard algorithms for lossless compression. Both internally use Huffman and LZ77 coding (Microsoft, 2008). Gzip uses the same algorithm as Deflate and has other compression formats. Thus the size of Gzip compressed data is slightly larger than data compressed by Deflate. This was evident through compression trials with data of various sizes ranging from a few bytes to 1KB as demonstrated in Table 4.2.

Table 4.2: Comparison of Data Compression through Gzip and Deflate

Source: Author

Original Data Size	Gzip Compression	Deflate Compression
26 bytes	140 bytes	122 bytes
102 bytes	193 bytes	175 bytes
204 bytes	197 bytes	179 bytes
408 bytes	199 bytes	181 bytes
816 bytes	202 bytes	184 bytes
998 bytes	206 bytes	188 bytes
1996 bytes	222 bytes	204 bytes

From the examples in Table 4.2, it was evident that compression techniques available in C# successfully compressed data by 50% or more when the original data size was 408 bytes or more. It was further seen that compressing a small data file of 26 bytes actually increased the resulting file size enormously. The resultant size of the file compressed by Deflate was marginally less than that compressed by Gzip in all cases.

Further to the results from Table 4.2, an XML data that described four attributes with a data size of 127 bytes (as described in Section 2.10.1 for a linen data) was compressed. Deflate actually increased data size to 208 bytes. String compression techniques available in C# were not appropriate to compress a small string as Huffman coding is used internally and requires the 256 ASCII characters to build the compressed string. It was not possible to compress small strings through these techniques because an overhead was created to code each of the 256 ASCII characters regardless of the size of the string. When the string is small, the overhead makes up a larger proportion of the file overall.

4.3.2 Data Format Suitability

XML and CSV data formats were examined for their suitability in the current study involving the laundry bin experiment.

XML Format

An XML format would be suitable for a RFID tag with user memory of 1 KB or more. However informal experiments with the Tracient reader and Skyetek tags did not succeed in storing 107 bytes (as required for linen data in Figure 2.6, Section 2.10.1) by writing as a continuous string. Although Skyetek had 200 bytes, it could successfully write only 64 bytes of data at any one time. To write more bytes, the block address of the user memory had to be moved further to continue writing from where it had last written. Thus long strings were written in parts in the user memory. Compression techniques discussed in Section 2.11 and Section 4.3.1 were not suitable for small sized data.

CSV Format

Due to limited memory in the available RFID tags, the data format was kept as a string of characters, which has simple format for the different pieces of data attributes delimited by comma. A sample CSV linen data as described in Figure 2.7, Section 2.10.2 had 20 characters and was below the maximum 64 bytes of data which can be written as whole string. The data string was hard coded and written onto the user memory at the starting block set through the prototype and the entire data format was written through one command and read from its memory location. The application to retrieve meaningful data attributes from the data format was done through the prototype. The comma in the CSV format was useful to separate values into different attributes.

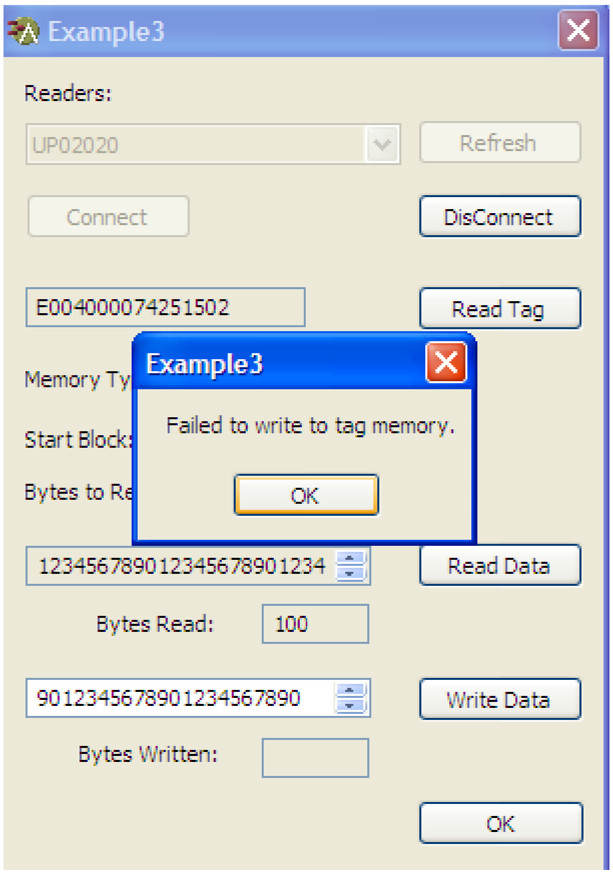
Therefore, considering XML and CSV, the only acceptable option was to store data in a CSV format.

Coded Data

Coded data in the RFID tag as discussed in Section 2.7.2 was not used in the current study since the application reading the coded data was not self explanatory and had the overhead of searching the meaning for it.

Limited Write Capability

The application provided by Tracient failed to write more than 64 bytes at a time. Figure 4.10 shows the error output caused by attempting to write more than 64 bytes to the tag. This application was modified in the prototype to run the experiment. To understand if this was a hardware or software problem, the following tests were done.


Example3

Readers: UP02020

E004000074251502

Example3

Failed to write to tag memory.

123456789012345678901234

Bytes Read: 100

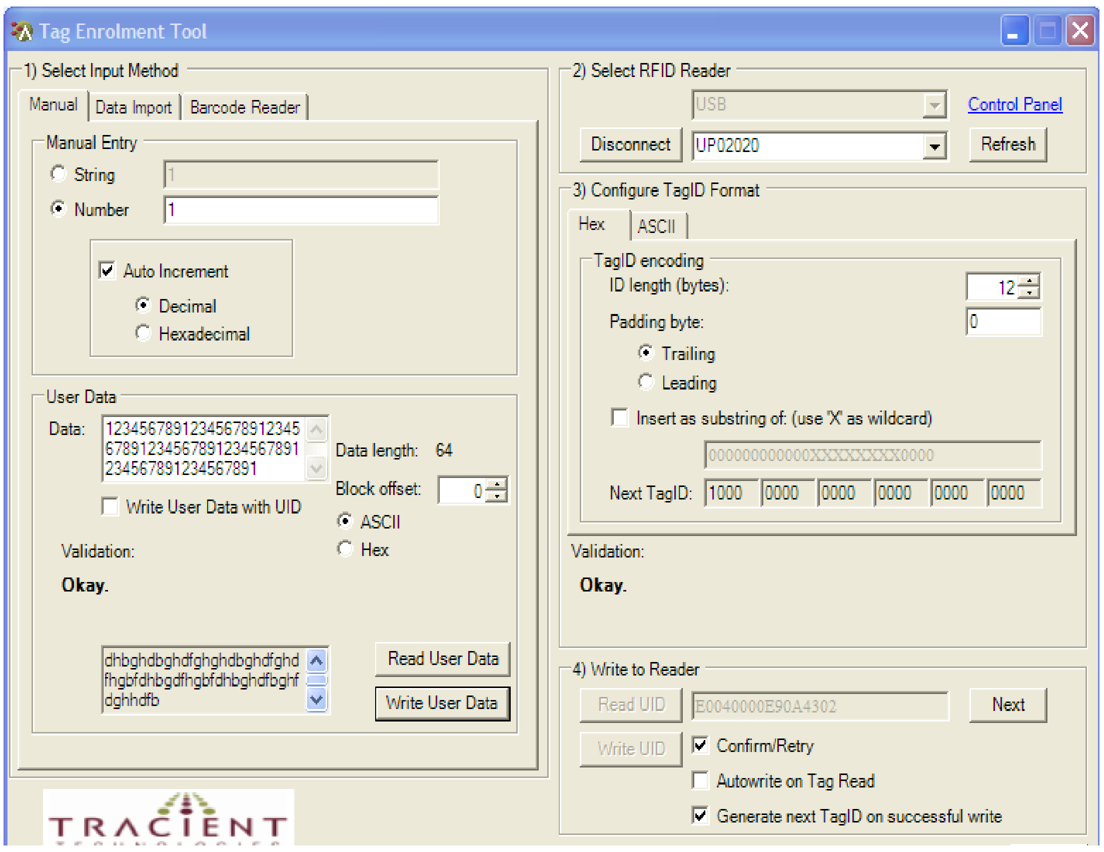
9012345678901234567890

Bytes Written:

Figure 4.10: Tracient Program Example3 - Attempt to write 100 bytes
Source: Tracient Software

Another application by Tracient called Enrolment Tool was used which could write 64 bytes at a time as shown in Figure 4.11. However, it failed to write more than 64 bytes as evident in Figure 4.12. To write data more than 64 bytes, the block offset had to be increased by 64 bytes to write long strings in parts of

64 bytes at a time. Thus the application could overcome the problem of writing more than 64 bytes.


Tag Enrolment Tool

1) Select Input Method

Manual | Data Import | Barcode Reader

Manual Entry
- String: 1
- [✓ Number]: 1

[✓] Auto Increment
- [✓ Decimal]
- Hexadecimal

User Data

Data: 1234567891234567891234567891234567891234567891234567891234567891

Data length: 64

Block offset: 0

[ ] Write User Data with UID

[✓ ASCII] Hex

Validation:

Okay.

dhbghdbghdfghghdbghdfghdfhgbfdhbgdfhgbfdhbghdfbghfdghhdfb

Read User Data

Write User Data

2) Select RFID Reader

USB

Control Panel

Disconnect

UP02020

Refresh

3) Configure TagID Format

Hex | ASCII

TagID encoding

ID length (bytes): 12

Padding byte: 0

- [✓ Trailing]
- Leading

[ ] Insert as substring of: (use 'X' as wildcard)

000000000000XXXXXXXX0000

Next TagID: 1000 0000 0000 0000 0000 0000

Validation:

Okay.

4) Write to Reader

Read UID

E0040000E90A4302

Next

Write UID

[✓] Confirm/Retry

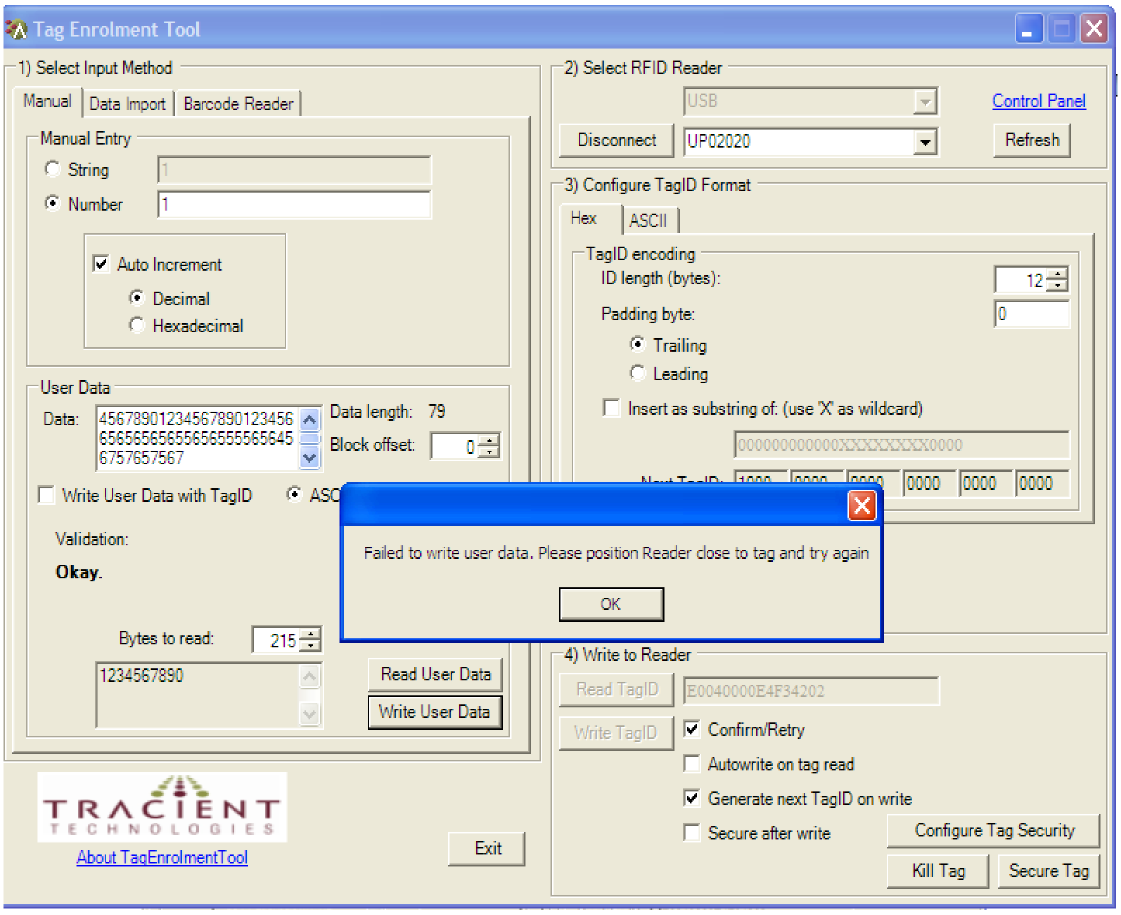
[ ] Autowrite on Tag Read

[✓] Generate next TagID on successful write

TRACIENT

Figure 4.11: Tracient Tag Enrolment Tool - Attempt to Write 64 bytes

Source: Tracient Software


Tag Enrolment Tool

1) Select Input Method — Manual | Data Import | Barcode Reader

Manual Entry
- String: 1
- [✓ Number]: 1
- [✓] Auto Increment — [✓ Decimal] Hexadecimal

User Data
- Data: 4567890123456789012345665656565655656555556564567576575 67
- Data length: 79
- Block offset: 0
- [ ] Write User Data with TagID — [✓ ASC]
- Validation: **Okay.**
- Bytes to read: 215
- 1234567890

Read User Data | Write User Data

About TagEnrolmentTool | Exit

2) Select RFID Reader
- USB
- Disconnect — UP02020
- Control Panel | Refresh

3) Configure TagID Format — Hex | ASCII

TagID encoding
- ID length (bytes): 12
- Padding byte: 0 — [✓ Trailing] Leading
- [ ] Insert as substring of: (use 'X' as wildcard) 000000000000XXXXXXXX0000
- Next TagID: 1000 0000 0000 0000 0000 0000

Failed to write user data. Please position Reader close to tag and try again

OK

4) Write to Reader
- Read TagID: E0040000E4F34202
- Write TagID
- [✓] Confirm/Retry
- [ ] Autowrite on tag read
- [✓] Generate next TagID on write
- [ ] Secure after write
- Configure Tag Security | Kill Tag | Secure Tag

Figure 4.12: Tracient Tag Enrolment Tool - Attempt to Write 79 bytes

Source: Tracient Software

4.4 Data Objects with Data Format

After considering the number of bytes which can be written at a time through the SDK provided by Tracient, the CSV format of linen data and bin data is considered.

4.4.1 Linen Data

Every RFID tag attached to linen carries data for that linen. Only one record for the linen is stored in the tag. The fields/attributes that describe the linen are linen status, laundry count, linen type and linen weight as shown in Figure 4.13.

Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight

Figure 4.13: Linen Tag Attributes

Source: Author

The Tag ID is stored as part of EPC in bank 01 as shown in Figure 2.6 and therefore the Tag ID from Figure 4.13 is not included in the user memory. The existing Tag ID is used to uniquely identify each tag and other additional data attributes are stored in the user memory. Hence, data stored in the user memory are ‘T’ or ‘F’ for linen status, a three digit laundry count to record the number of times the linen goes through the washing cycle and a maximum of 20 characters to describe linen type like ‘PatientGown’. Most of the data values are small except for linen type. Linen type values were not stored as codes because there could be overheads to get the full form of the data value from the application reading the data. Finally, four characters are allowed to store weight of linen in the form of one digit and two decimal places including decimal point. Each character could be stored as one byte which was sufficient to store one record of linen as characters in the user memory. The storage in terms of bytes is given below.

Linen Status	1 byte
Laundry Count	3 bytes
Linen Type	20 bytes (maximum)
Linen Weight	4 bytes
Comma to separate 3 bytes	
Total	31 bytes (where Total is not another data field)

Thus there are 31 bytes for data in a linen tag. An example with 19 bytes is shown in Figure 4.14. Most passive UHF tags have 512 bits or 64 bytes of user memory. The tags used in the current study were Skyetek tags with 1728 bits or 216 bytes of storage and Intermec IT67 tags with 512 bits or 64 bytes. Both tags had the memory capacity sufficient to accommodate the CSV format with the data attributes and values suggested.

E0040000E7204302	F	1	PillowCase	0.05

Figure 4.14: Linen Tag Values
Source: Author

4.4.2 Bin Data

In order to relate the laundry collected to the hospital it came from, an RFID tag is attached to the laundry bin. The laundry bin belongs to the hospital and the tag attached to it contains data about the hospital. The fields/attributes here are client (hospital) ID which is the same as Tag ID, tag type (to differentiate between bin and linen), client name and address as in Figure 4.15. The tag ID is stored as part of EPC in bank 01and is not part of the user memory.

Tag ID	Type	Client Name	Address1	Address2

Figure 4.15: Bin Tag Attributes
Source: Author

The data values to be stored in a bin tag are as shown in Figure 4.16. The storage in terms of bytes is given below.

Type	1 byte
Client Name	4 bytes
Address1	25 bytes (maximum allowed)
Address2	20 bytes (maximum allowed)
Comma	3 bytes
Total	53 bytes (where Total is not another data field)

Therefore, the maximum data size of the bin attributes are 53 bytes. However, the example for bin data taken for this experiment had 35 characters including commas totalling 35 bytes as in Figure 4.16. The Skyetek and Intermec IT67 tags used in the experiment had the capacity to store the entire data of 53 bytes.

E0040000B64B1502	B	C123	Wellesley Hospital	Australia

Figure 4.16: Bin Tag Values
Source: Author

4.4.3 PC/PDA Log Data

For every laundry bin and the linen dropped in it, the data about them has to be collectively put together. This connects the linen collected from a particular bin for billing purposes. Each linen tag carries one record about the linen and each bin carries a record about bin. These are brought together as a single record and stored as a log file in the PC/PDA.

A cartesian product of the two relations is considered (Codd, 1979), where every tuple from the first relation (Linen Table 4.3) joins with the only tuple from the second relation (Bin Table 4.4). The result of this operation on relations L(linen) and B(bin) with n and 1 number of tuples is a new relation Q (PC/PDA) with (n x 1) tuples and has the combined attributes of the input relations as in Table 4.5.

Table 4.3: Linen Tags - Relation L

Source: Author

Linen tag - Relation L

Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight
E0040000E7204302	F	1	PillowCase	0.05

Linen tag - Relation L

Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight
E004000031DD4202	F	1	Blanket	2

Linen tag - Relation L

Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight
E0040000E90A4302	F	1	FlatSheet	0.2

Table 4.4: Bin Tag - Relation B

Source: Author

Tag ID	Type	Client Name	Address1	Address2
E0040000B64B1502	B	C123	Wellesley Hospital	Australia

Table 4.5: Relation Q = L x B

Source: Author

Linen Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight	Bin Tag ID	Type	Client Name	Address 1	Address 2
E0040000E7204302	F	1	PillowCase	0.05	E0040000B64B1502	B	C123	Wellesley Hospital	12 Beach Road, Auckland
E004000031DD4202	F	1	Blanket	2	E0040000B64B1502	B	C123	Wellesley Hospital	12 Beach Road, Auckland
E0040000E90A4302	F	1	FlatSheet	0.2	E0040000B64B1502	B	C123	Wellesley Hospital	12 Beach Road, Auckland

Thus the records stored in each of unit of linen in the bin are brought together in a file in the PC/PDA. In practice however, all linen data may not be recorded or may include duplicates in the relation Q. Some tags may not be read by the RFID reader. This is because of the nature of RFID tag readers. In order to reduce the possibility of missing readings from the tag, the RFID reader is set on continuous read mode as discussed in Section 4.2.1. This will however result in duplication of tag data in the relation Q.

It is therefore necessary to erase or clean the duplicate data before processing. An approach of sorting all tuples at Q by tag ID and recording only unique data tuples while populating the data at the PC/PDA is considered.

After the data format was finalised, the read and write operations to the RFID tag were designed in the prototype to set up the experiment.

4.5 Experiment Design

Two separate experiments were carried out. Both the experiments were set up to read from and write to tags within the range of 0.5 meter as discussed in Section 4.1.1. In the first experiment the RFID tags were kept in close proximity to the RFID reader. This was to check if all RFID tags were recognised and read and write operations to the user memory of the RFID tags were successful or not. In the first experiment the tags were not attached to any linen. In the second experiment the tags were attached to linen and dropped into a box 29.7 centimetres long, 21 centimetres wide and 22 centimetres deep. An RFID reader was mounted on the box. The second experiment simulated a real setting to check whether the RFID tags were recognisable by the reader and the read and write operations to the tag user memory were successful. The purpose of the second experiment was to check if linen would impede RF signals. The second experiment also checked whether the distance, position in the box and orientation of tags affected the reading and writing capabilities in the RFID tags.

Skyetek and Intermec IT67 tags of different standards and user memory capacity were used in the experiment. Each experiment was carried out in two sub-steps as shown in the following paragraphs.

Experiment 1a was conducted with Skyetek RFID tags *unattached* to any linen as shown in Figure 4.24. Tags with linen data in CSV format were spread out around the reader. There was one other tag with bin data in the assortment.

Experiment 2a was conducted with Skyetek RFID tags *attached* to linen as shown in Figure 4.25. The same tags from Experiment 1a with the same linen data were used here. The linen was dropped into a box to simulate the laundry bin environment.

Experiment 1b was conducted with Intermec RFID tags *unattached* to any linen as shown in Figure 4.26. The same data as in the previous experiments with Skyetek tags was used.

Finally, Experiment 2b was conducted with Intermec RFID tags *attached* to linen as shown in Figure 4.27. The same tags from Experiment 1b with the same data were used. The setting was similar to Experiment 2a.

In Experiment 1a and Experiment 1b the tags were laid out on a flat surface around the reader within a radius of 15 centimetres and not attached to any linen. In Experiment 2a and Experiment 2b the tags were attached to linen and dropped into a box. Signal impedance and orientation will affect the results of these experiments and these factors were studied in repeated trials of the set of experiments.

4.5.1 Experiment Simulation

The experiment was set up with a Tracient reader connected to a PC/PDA and the necessary Tracient applications and prototype were loaded and active on the PC/PDA.

In a real setting, the bin with an RFID reader will be continuously active, assuming the laundry is collected every day. The reader’s memory will store data from the linen and bin as a cartesian product Q. The Padl-R UHF Tracient reader has 1MB memory and can store approximately 900 tag IDs. The firmware in the reader can also filter data and store unique tag IDs.

In the current simulated experiment, the Padl-R UHF Tracient reader was used to read tag IDs, read from and write to, user memory from the prototype. The reader log was not used in the current experiment but rather the log file with contents of user memory was stored in the PC/PDA. The existing reader firmware was not able to store user memory data as required in the experiment. The read from and write to user memory operations were controlled through the prototype. The experiment ran for 10 minutes with 7 Skyetek tags and later in another experiment with 10 Intermec tags as this was sufficient time to identify all tags. Each set of experiments were repeated five times for reliability and the resultant data collected.

While the experiment was running, the process of reading tag IDs and read/write user memory was observed on screen and a data log was also collected by the prototype and written as a text file to be used for later scrutiny.

The prototype developed involved read and write data operations into the user memory of the RFID tag described in detail in the next section.

4.6 Prototype

The prototype developed in this experiment explored the read and write process on tag using the Tracient SDK and four programs were written for the laundry bin to:

1. write data into a tag,
2. update tag data,
3. collect tag data at laundry pickup, and
4. clean tag data for local processing.

Programs 1 and 2 were used to prepare the tags for the four sets of experiment described in Section 4.4.1. Program 1 wrote the required data format into the tag user memory, whereas program 2 updated the linen status which changed in the course of running the experiment.

Program 3 was used in the experiment, to read and write data in the user memory and to collect all linen data and bin data. Program 4 was also used in the experiment to sort data collected from program 3 and to do some local processing to print the total count and weight of all unique linen collected from the laundry bin.

4.6.1 Writing on Linen Tag

Program 1 wrote a new linen tag with the necessary information as shown in Figure 4.17. The size of the user memory was limited as discussed earlier in Section 4.3, a single string with various attributes delimited by comma was written. Thus the complete string contained the different attributes of the linen. The prototype performed each of the following steps: connecting to the RFID reader, reading an RFID tag ID and where reading tag ID was successful then writing data into the user memory. As data to be written was in CSV format depicting all the attributes of linen as discussed in Section 4.4.1, a drop down list was used to select pre-defined values as shown in Figure 4.17. The program could also read data written in user memory to immediately check that the string was written correctly. To verify that all the processes of reading tag IDs, writing tag data and reading tag data worked as intended, a log was maintained and displayed on the right side of the user interface. The final step was to manually disconnect from the reader by using the disconnect button on the interface.

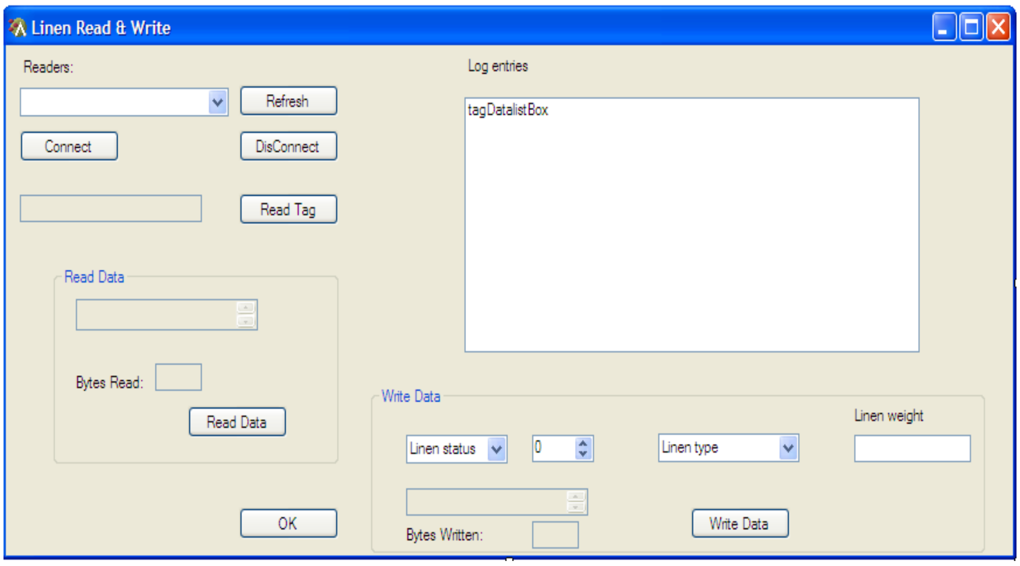


Figure 4.17: Interface for Writing New Linen Tag

Source: Author

The seven Skyetek tags as shown in Figure 4.18 had the necessary pre data stored in them through program 1 as shown in Table 4.6. All RFID tags came with a Tag ID stored as EPC which is not part of user memory. The data as defined is stored in the user memory of the tag in CSV format considering the memory limitations of the tag and writing procedure as discussed in Section 4.3.2.

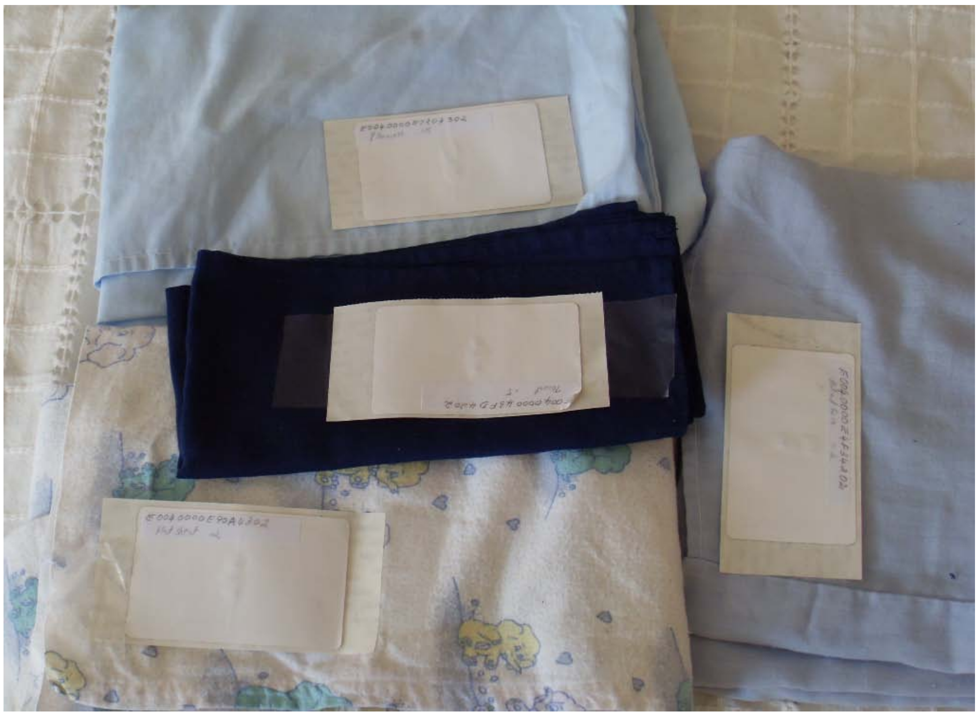


Figure 4.18: Skyetek Tags Attached to Linen

Source: Author

Table 4.6: Data in Seven Skyetek Tags

Source: Author

Linen Tag ID. (EPC)	Linen Status	Laundry Count	Linen Type	Linen Weight
E004000074251502	T	0	FlatSheet	0.2
E0040000E90A4302	T	0	FlatSheet	0.2
E004000031DD4202	T	0	Blanket	2
E0040000E4F34202	T	0	PatientGown	1.2
E0040000B34B1502	T	0	PillowCase	0.05
E0040000E7204302	T	0	PillowCase	0.05
E004000043FD4202	T	0	Towel	0.5

The ten Intermec tags were similarly configured to store data as shown in Table 4.7 through program 1(see Section 4.6.1).

Table 4.7: Data in 10 Intermec Tags

Source: Author

Linen Tag ID. (EPC)	Linen Status	Laundry Count	Linen Type	Linen Weight
3005FB63AC1F3681EC880401	T	0	FlatSheet	0.5
3005FB63AC1F3681EC880402	T	0	PillowCase	0.2
3005FB63AC1F3681EC880403	T	0	PatientGown	1
3005FB63AC1F3681EC880404	T	0	Blanket	2
3005FB63AC1F3681EC880405	T	0	Towel	0.5
3005FB63AC1F3681EC880406	T	0	FlatSheet	0.5
3005FB63AC1F3681EC880407	T	0	PillowCase	0.2
3005FB63AC1F3681EC880408	T	0	PatientGown	1
3005FB63AC1F3681EC880409	T	0	Towel	0.5
3005FB63AC1F3681EC880410	T	0	Blanket	2

4.6.2 Updating a Linen Tag

Program 2 to update linen tag was written to update the linen status from 'dirty' (F) to 'clean' (T). This process of changing linen status was done in the laundry office after the linen was washed as discussed in Section 3.4. The program read each linen tag and updated the status by changing one attribute in the complete string data as shown in Figure 4.18. To ensure data is correctly written to the tag, the read command reads data from the RFID tag after the write procedure and it displays it on the screen. At the same time a data log was maintained to keep track of all tag IDs read along with user memory data before and after updating the tag for the purpose of creating a trail for evaluation if necessary.

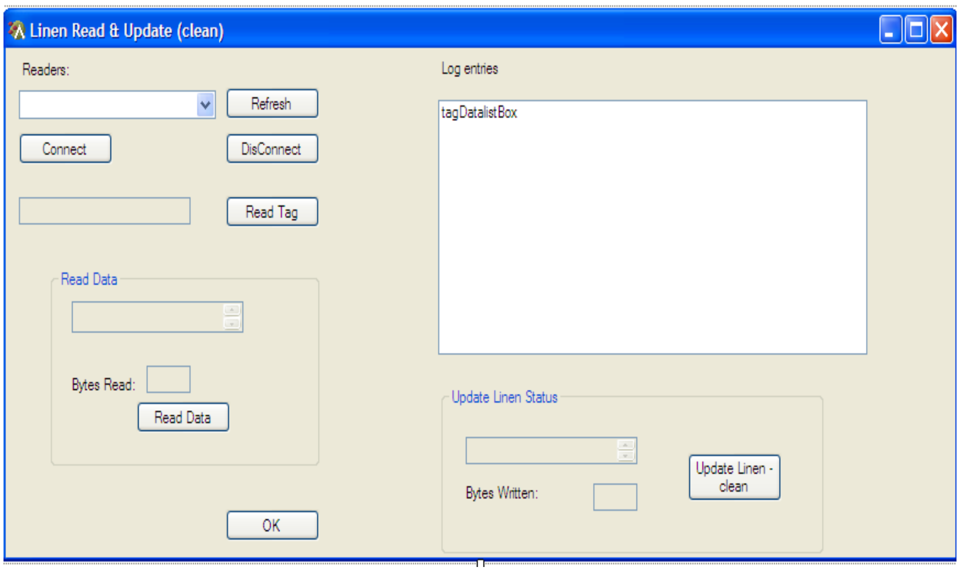


Figure 4.18: Interface for Updating Linen Tag

Source: Author

4.6.3 Collecting LinenTag Data

Soiled linen is collected at the client site in various laundry bins. Thus, at the pickup point data from linen tags and the bin tag had to be collated into the PC/PDA for billing.

Since linen carried an RFID tag containing only one record and the bin had one record in the bin tag, these had to be brought together as a single record in the PC/PDA as explained in Section 3.4. The process of reading data from linen and the bin performed by Program 3 is explained.

Program 3 was developed for reading and writing data to linen tag and reading bin tag process and executed in the PC/PDA. The RFID reader was connected to the PC/PDA in order to prepare the reader to read RFID tags from the bin and linen. At the start of the experiment, the ‘connect’ button was clicked and the bin tag was read. If the bin tag was successfully read, a timer was then started to process the reading and updating (writing) of data in the linen tags. The timer in the program was set to read and write to a tag every ‘n’ seconds. A date-time stamp was recorded for each read and write process. The linen tag and bin tag data were brought together as a single tuple and stored in a log file in the program. When the ‘disconnect’ button was clicked, the timer in the program was stopped and the reader stopped reading tags. Figure 4.19 shows the interface in the PC/PDA to connect to the reader and activate the timer.

Practically, at pickup time the reader is disconnected and the log data in the program stored in the CSV format file. As discussed in Section 4.5.1, the log file should ideally be in the reader, but is stored in the PC/PDA for the purpose of this experiment.

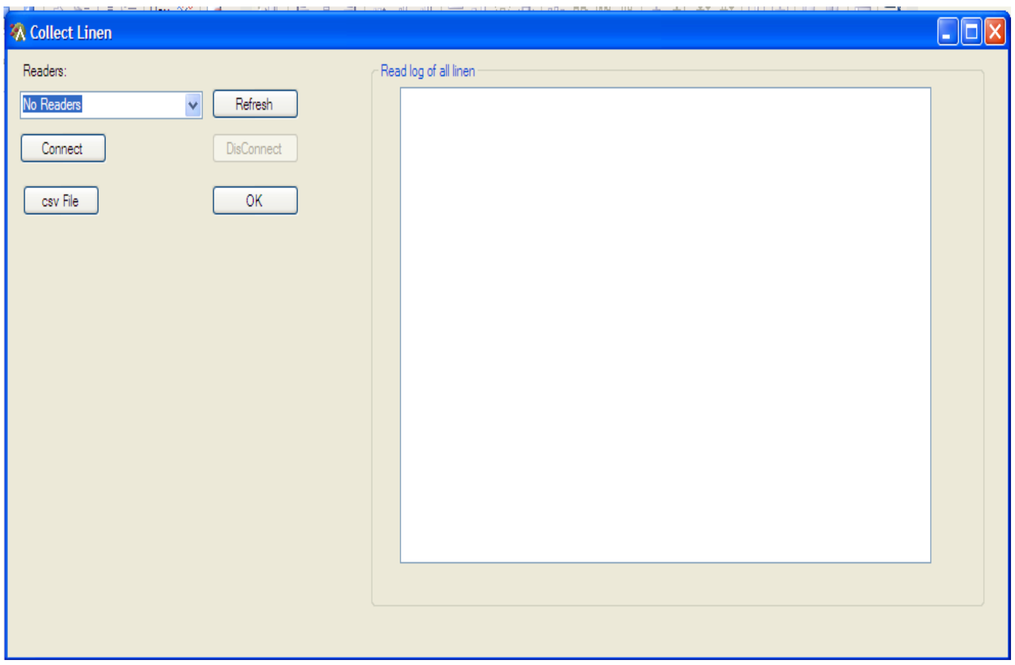


Figure 4.19: Interface to Collect Linen

Source: Author

The procedure of collecting data from RFID tags attached to linen is explained further. The RFID reader is set to read linen tags continuously. If the tag ID from the linen is successfully read, the reader will then read its user memory and extract information to the program. If user memory is successfully read, the program will then proceed to update the tag by over writing data into the user memory of the tag.

Local processing will also involve other checks which ensure that even if the same tag is read multiple times, only one unique tag is considered. If a tag needs a write procedure, the application for local processing needs to be coded so that data is written only once.

Thus each piece of linen data and data from the laundry bin tag are collected and stored as a cartesian product of bin and linen data as one tuple. This is further illustrated through the pseudocode provided in Figure 4.20.

```
Read data from Bin
If Read Bin TagID not successful
    Throw Read TagID exception
Else
    Read Bin user memory
    If Read Bin user memory not successful
    Throw Read User Memory exception
    Else
    Create Bin data b
    **Start Timer** to read Linen Tags continuously
    Read data from Linen
    If Read Linen TagID not successful
            Throw Read TagID exception
            Else
                    Read Linen user memory
                    If Read Linen user memory not successful
                            Throw Read User Memory exception
                    Else
                            Write data to user memory
                            If write user memory not successful
                                    Throw Write User Memory exception
                            Else
                                    Create Linen data l
                                    Join data from Bin and Linen b x l
                                    Place output in Q
    **Stop Timer**
    For each data in q do
            Write q to file
```

Figure 4.20: Pseudocode to Collect Linen Data

Source: Author

As discussed in Section 2.8.5, aggregation and containment help to associate individual products to their larger units and in turn to their pallet (unit- box-pallet) in the supply chain. In the current laundry bin experiment program 3 associates linen to their laundry bin through the concepts of Object Oriented Programming (OOP). In program 3 class ‘Linen’ and class ‘Bin’ are created and the object of class Linen is a member of class Bin. Thus aggregation and containment helped in processing data from the linen and bin as detailed in program 4 in Section 4.6.4.

4.6.4 Cleaning and Processing of Collected Linen Data

Program 4 was executed on PC/PDA owned by the pickup person collecting the soiled linen. The log file generated as CSV file from program 3 was transferred to program 4. This log data is the cartesian product Q giving details of all linen in the bin as described in Section 4.4.3.

The log file created through program 3 had many instances of repeated data about linen as the reader was set on continuous read mode. The process of removing duplicate records was done by program 4 and is explained in the pseudocode in Figure 4.22. Local processing of obtaining the total count and weight of linen was done at the end of the duplicate removal process and the total count and weight was printed as shown in Figure 4.21.

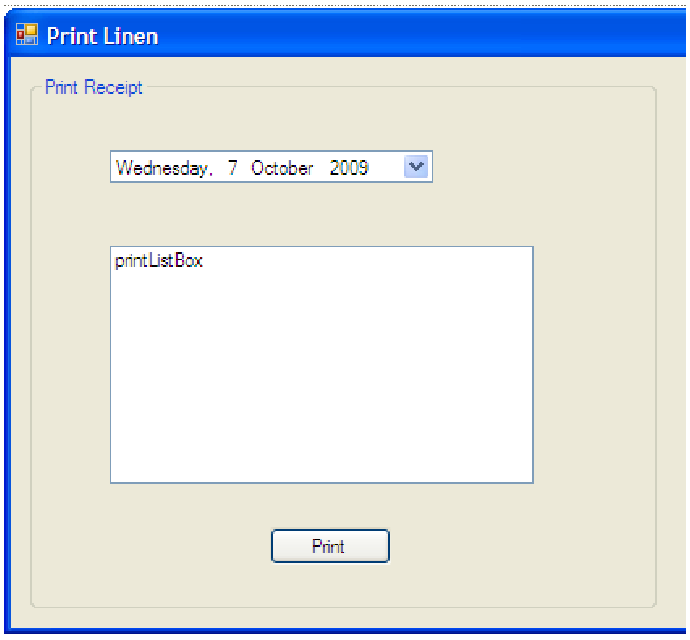


Figure 4.21: Interface to Print Linen Collected

Source: Author

The algorithm for linen data cleaning procedure is explained in Figure 4.22.

```
For each data q from file do
    Create data in an Array A1

Sort Array A1 by TagID of Linen
For each a1 from A1 do
    Write unique a1 into Array A2

Sort Array A2 by linentype
For each a2 from A2 do
    Keep linen counter
    Total linen weight

Print linen count for each linen Type
Print total linen weight
```

Figure 4.22: Pseudocode for Cleaning and Processing

Source: Author

As data is written as a single string in the tag, program 4 was used to process data into meaningful attributes. The algorithm will vary depending on whether XML or CSV formats are used. The experiment used CSV format and thus the program algorithm processes incoming data from the RFID tag (linen) in a loop to read the delimited comma string. Whenever it encounters a comma, the attribute thus realised is stored separately. This algorithm is explained in Figure 4.23.

The attribute values thus extracted were stored in appropriate data formats like integer for linen count, boolean for linen status and decimal for linen weight. This improved the processing to count linen numbers and total linen weight.

```
Declare an array of string to store all attribute values
Declare variables of int: lastPosition=0,   k=1
For each i in the incoming string
    If incoming string[i] == ","
        New attribute = substring (incoming string, lastPosition, k-1)
        k=1
        lastPosition = i
    else
        k=k + 1
Endfor
Last attribute = substring (incoming string,lastPosition,k-1)
```

Figure 4.23: Separating Different Attributes from a String with Comma Delimited
Source: Author

Once all the programs were written as attached in Appendix D, the experiment was ready to be run and the following experiment procedure was followed.

4.7 Experiment Procedure

As discussed earlier in Section 4.1.2, the experiments were set up with two types of tags – Skyetek and Intermec. Skyetek tags were used in Experiment 1a where tags were not attached to linen. In Experiment 2a the same tags were attached to linen to check if RF signals had any effect on read and write operations of data into the tag. Similarly Intermec tags were used in Experiment 1b without linen and in Experiment 2b the tags were attached to linen.

The experiment was conducted on a flat surface 90 centimetres long and 58 centimetres wide. The Tracient reader was connected to the PC and program 3 – 'Collect tag data' was run for 10 minutes. Informal experiments demonstrated that if all tags (7 or 10) were kept in the vicinity of the reader, most tags were recognised by the reader in the first three minutes. The experiment was run for 10 minutes to collect sufficient evidence of all read and write operations in the tags. Since all tags were recognised within the first three minutes, it was considered that 10 minutes was sufficient to fully test the technology.

The linen tag was prepared to receive the correct information through program 1 - Write data into new tag. On occasion when the read tag ID attempt failed, a second attempt was made. There were also times when the write to tag operation failed. A second attempt was made in this case too.

Each of the four experiments had five trials. These trials added reliability to the research study. The following steps were undertaken for each trial.

Step 1: Connect Tracient RFID reader to the PC/PDA. Tracient software Control Panel was used to configure the reader to a specific air interface standard. This can also help to indicate the starting block to write into user memory as shown in Figure 4.5 and Figure 4.6.

Step 2: Program 3 – Was run for 10 minutes in a simulated laboratory setting to collect tag data before laundry pickup. The aim of this step was to identify all the tags and to read and write to user memory the necessary data using program 3 and to obtain the result Q (refer Section 4.4.3) which is a cartesian product of linen tag data and bin tag data. While program 3 was running, tags unattached to linen were scattered around the reader in Experiment 1a and Experiment 1b and tags attached to linen were dropped in a box in Experiment 2a and Experiment 2b. The researcher observed on screen the process of reading tag IDs and read and write of user memory. A data log was generated and written as text file in the PC/PDA.

Program 3 was evaluated through a trial sheet attached as Appendix E. The trial sheet had observations for each trial of the experiment. It kept track of read tag ID failures, read user memory failures and write user memory failures as displayed on screen by running program 3, discussed in Section 3.5. Successful read tag ID and read user memory and write user memory attempts were also recorded through program 3.

Step 3: Data from the log file generated in program 3 is given as input to program 4 – Clean tag data for local processing. The final linen count which was actually the number of unique RFID tags recognised was the final output of program 4. Along with count, the total weight was also printed.

Step 4: For analytical purposes, the log data file was extracted and loaded onto an Excel spreadsheet to analyse the data read and written by the RFID reader. The Excel spreadsheet split the CSV file appropriately to efficiently analyse the information obtained and record this in the trial sheet manually by the researcher. Thus entries in the trial sheet confirmed whether or not all tags and strings stored in the user memory were successfully read and following this, were successfully written with the updated string.

Step 5: At the end of each of the five iterations of the experiment trial, the researcher manually checked through program 2 to confirm that the data was changed appropriately through program 3 (which changed the linen status from ‘T’ to ‘F’). This was recorded in the trial sheet. If the data on tag was changed appropriately then it indicated that a successful write had taken place through program 3.

Program 2 was then used to reset the Linen Status that had been written to the tags during the experiment. Program 2 changed the status of the washed linen from ‘F’ to ‘T’.

4.7.1 Experiment 1a without Linen - Using Seven Skyetek Tags

In Experiment 1a, Skyetek RFID tags were scattered around the reader randomly as shown in Figure 4.24. Some of the tags overlapped other tags.

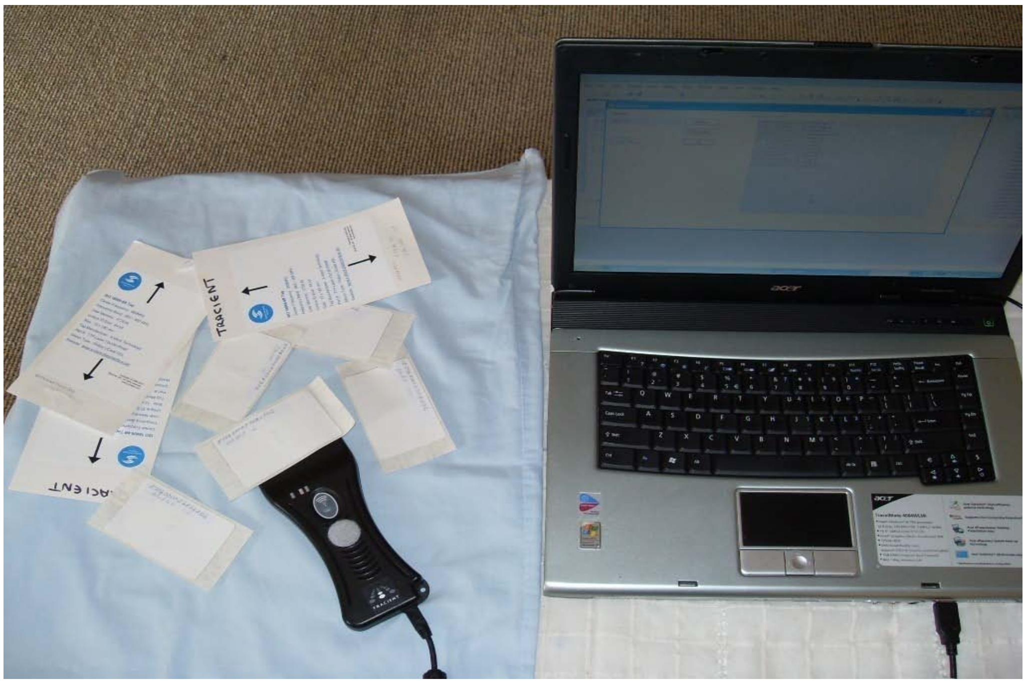


Figure 4.24: Experiment 1a without Linen - Using Seven Skyetek Tags

Source: Author

Read and write error messages on the screen were observed and recorded in the trial sheet attached as Appendix F. The number of read and write errors through program 3 were also recorded. The data generated on screen and through program 3 did not tally. This could be because of the interference with RF transmission between the various signals generated. Thus the exact number of read tag IDs, read user memory and write user memory error messages was not clear.

The log files with entries of read and write data into user memory were checked in step 5 of the experiment procedure. Each tag was checked through program 2 and successful read and write entries were recorded in the trial sheet. Data was also checked to see if it had the same status in the log file.

It was observed that not all tags were successfully read and written to user memory. If they failed to write or were not recognised by the reader then they would not be accounted for in the final count of linen. Details of tags recognised in each trial of the experiment were printed through program 4 and a sample from Experiment 1a Trial 2 is given in Table 4.8. The output from all other experiments and trials are in Appendix G. In Experiment 1a Trial 2, 5 tags were identified and their total weight was recorded as 3.95 Kg. Two tags were not recognised in this trial.

Table 4.8: Sample Output from Program 4
Source: Author

```
Client: C123 Wellesley Hospital Australia

Blanket   1

FlatSheet   1

PatientGown   1

PillowCase   1

Towel   1

Total Weight    3.95
```

The output of all five trials for the 7 Skyetek tags used without linen attached are given in Table 4.9. Program 4 prints the total linen count grouped by linen type and total weight of the linen. The missing tags will affect the final linen count and weight.

Table 4.9: Experiment 1a Results without Linen - Using Seven Skyetek RFID Tags
Source: Author

Experiment Program4 Results		Expected Count and Weight	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
Linen Count	Blanket	1	1	1	1	0	1
	FlatSheet	2	1	1	1	1	1
	PatientGown	1	1	1	1	1	1
	PillowCase	2	1	1	1	2	2
	Towel	1	1	1	1	0	0
Total Linen Weight(kg)		4.2	3.95	3.95	3.95	1.5	3.5

The five trials created log files, experiment trial sheets and the final output from program 4 which were retained for further analysis and is presented in Chapter 5 – Discussion.

4.7.2 Experiment 2a with Linen - Using Seven Skyetek Tags

The experiment procedure was repeated with tags attached to real linen and dropped into a box. This was to simulate a real setting with linen dropped into the laundry bin. The RFID reader was placed face down on top of the box. In this experiment the program recognised the tags as they were dropped into the box as shown in Figure 4.25. The same procedure described in Experiment 1a was used.

The outputs of all five trials for the 7 Skyetek tags used with linen attached are given in Table 4.10. Program 4 prints the total linen count grouped by linen type and total weight of the linen.

Table 4.10: Experiment 2a Results with Linen - Using Seven Skyetek Tags
Source: Author

Experiment Program4 Results		Expected Count and Weight	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
Linen Count	Blanket	1	1	1	1	1	0
	FlatSheet	2	1	1	1	0	1
	PatientGown	1	0	1	0	0	1
	PillowCase	2	1	1	2	2	1
	Towel	1	0	1	0	1	0
Total Linen Weight(kg)		4.2	2.25	4	2.3	2.6	1.45

Experiment 2a had tags which were not accounted for in the final output. In trials 1, 3, 4 and 5 not all linen was picked up by the reader. This could be because the RF signals were impeded by the linen causing the reader to fail to read and write to user memory. Another possible reason could be the distance of tags from the reader which was increased when compared to Experiment 1a, although it was still within the UHF range of less than 1.5 metres. Further, the reader was not in the same plane as the tags for the Experiment 2a as it was in Experiment 1a. A comparison of Experiment 2a with Experiment 1a is discussed in Chapter 5.

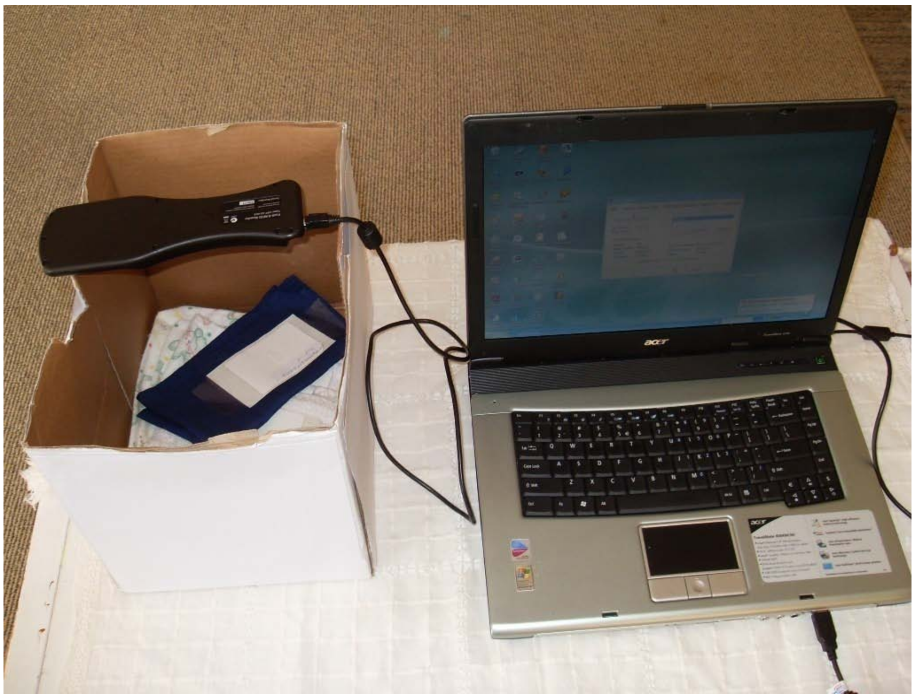


Figure 4.25: Experiment 2a with Linen – Using Seven Skyetek Tags
Source: Author

4.7.3 Experiment 1b without Linen - Using Intermec IT67 Tags

Experiment 1b was conducted with Intermec IT67 tags which had lower user memory than Skyetek. The air interface standard was ISO 18000-6C for Intermec IT67 tags. The experiment was set up similar to Experiment 1a as shown in Figure 4.26; the only difference being there were 10 tags instead of the 7 Skyetek tags considered in the former. The extra tags should not make more than a negligible difference for the time slot of 10 minutes. It was observed in log files of Experiment 1a that most tags were identified in the first 3 minutes and thereafter it was a repetition of the same data as the RFID reader was set to continuous read mode.

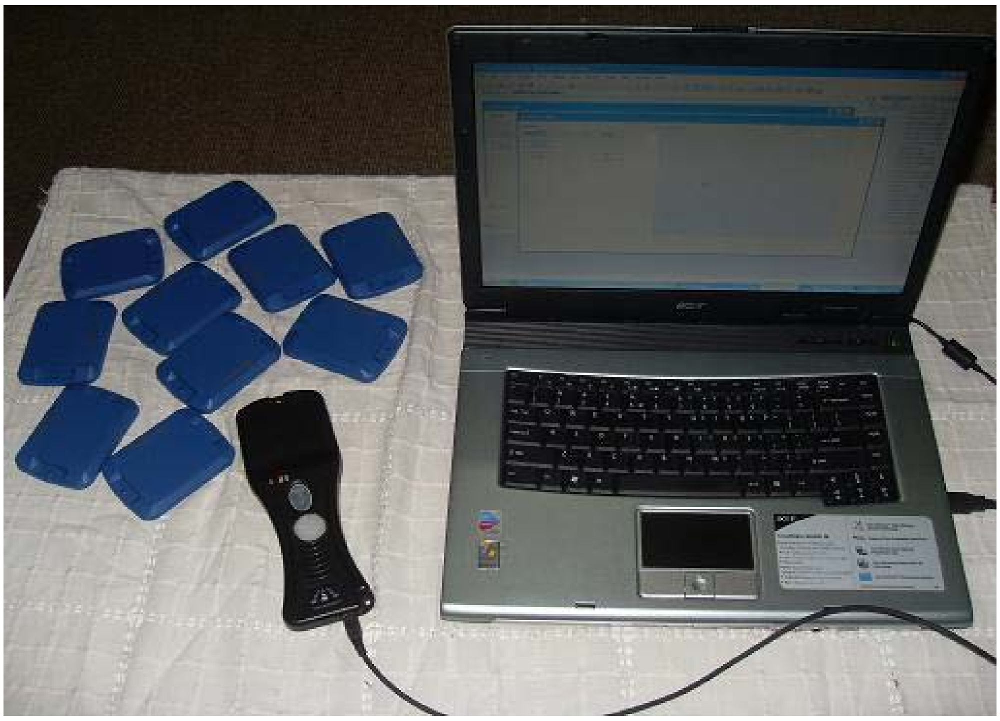


Figure 4.26: Experiment 1b without Linen - Using 10 Intermec IT67 Tags
Source: Author

The result of running program 3 and program 4 of Experiment 1b are recorded in Table 4.11. With the exception of a few tags missed being read or written, almost all tags were identified by the reader. A comparison with other experiments is done in Chapter 5 – Discussion.

Table 4.11: Experiment 1b Results without Linen – Using 10 Intermec Tags
Source: Author

Experiment Program4 Results		Expected Count and Weight	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
Linen Count	Blanket	2	1	1	2	1	2
	FlatSheet	2	1	2	2	2	2
	PatientGown	2	2	2	2	2	1
	PillowCase	2	2	1	2	2	2
	Towel	2	1	1	1	1	0
Total Linen Weight(kg)		8.4	5.4	6.2	7.9	6.4	6.9

4.7.4 Experiment 2b with Linen - Using 10 Intermec IT67 Tags

Experiment 2b was conducted with Intermec IT67 tags with linen attached. The set up was similar to Experiment 2a, but using Intermec tags as in Experiment 1b as shown in Figure 4.27.

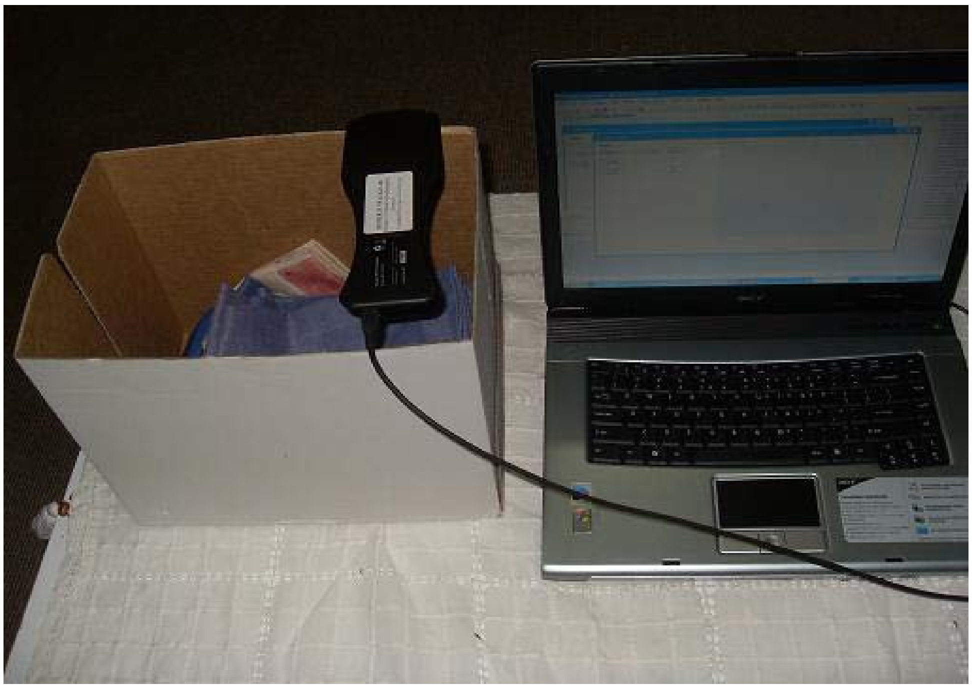


Figure 4.27: Experiment 2b with Linen - Using 10 Intermec IT67 Tags
Source: Author

The results of running program 3 and program 4 of Experiment 2b are recorded in Table 4.12. Trial 4 exhibited a higher degree of unreliability when compared to the other trial results in the same experiment. A more detailed evaluation is carried out after checking the log files in each trial, and the reason for any failures analysed.

Table 4.12: Experiment 2b Results with Linen – Using 10 Intermec Tags

Source: Author

Experiment Program4 Results		Expected Count and Weight	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
Linen Count	**Blanket**	**2**	1	1	2	0	2
	FlatSheet	**2**	1	2	1	2	2
	PatientGown	**2**	2	1	1	1	1
	PillowCase	**2**	2	2	2	1	2
	Towel	**2**	1	1	1	0	1
Total Linen Weight(kg)		**8.4**	5.4	5.4	6.4	2.7	7.4

4.8 Addressing Research Questions

What data format is suitable for a data-on-tag approach?

Mamei, Quaglieri and Zmabonelli (2006) proposed a simple data format to store data in an RFID tag as discussed in Section 2.7.2. As the memory in RFID tags is limited, initial investigation on the most promising data formats like CSV and XML were undertaken. XML, with one or two attributes describing some data took a minimum of 107 bytes. Further compressing small strings was not possible as most of the C#.Net libraries available were only able to compress strings of 300 characters or more. Compression algorithms like Huffman and LZW did not work satisfactorily when attempting to compress data of small size. In the current chapter, CSV was found more suitable than XML considering the limited data format and the ability of the Tracient RFID reader and SDK to write only 64 bytes at a time.

How can data be stored in the RFID tag in a data-on-tag approach?

Data was written as string in the CSV format in RFID tags using the read and write functions provided by Tracient SDK.

How data from RFID tags is read and written in a data-on-tag approach?

The prototype developed provided by Tracient SDK had three out of the four programs which demonstrated read and write procedures to tag. The prototype was developed to read the string and then separate it into the appropriate attributes for the CSV format.

How accurate is the reading and writing of data using the data-on-tag approach?

The experiment was executed using the prototype programs and results were presented. The success rate for read and write user memory will be presented and discussed in Chapter 5.

4.9 Summary

The experimental set up was undertaken to investigate the research question on handling data in a data-on-tag approach. The UHF tags available for these experiments were examined. Skyetek ISO 18000-6B had 1728 bits of user memory and Intermec IT67 ISO 18000-6C had 512 bits of user memory. The rationale for using CSV data format over XML format was discussed.

The RFID reader and supporting applications to set up the experiment provided by Tracient Technologies Ltd. were reviewed. The SDK for C# provided by Tracient was used to develop a prototype with 4 programs:

1. write data into a tag,
2. update tag data,
3. collect tag data at laundry pickup, and
4. clean tag data for further processing.

Program 1 and program 2 were respectively used to write and update data in a tag individually. Program 3 was used to read and write data in multiple tags to implement the true nature of pervasive computing of reading and writing data in the environment. Reliability and accuracy issues were checked through program 4 for desired output of linen count and weight. The results of program 4 were presented.

Research questions were presented and how well they were answered so far was discussed.

The data collected through the trials of the experiments is evaluated in Chapter 5 – Discussion.

5 Discussion

The current research study incorporates building theory, developing systems, conducting experiments and recording and evaluating observations in the multi-methodological design science framework as explained by Nunamaker, Chen, & Purdin (1991).

The aim of the prototype built as part of this research experiment was to capture RFID data in the selected CSV format, then clean and process it. In the prototype, program 3 generated a log file as discussed in Section 4.4.3. The log file thus generated was exported into an Excel spreadsheet for further evaluation, discussed in Section 4.7. The prototype was developed to read and write data in the user memory of the RFID tag and needed to be evaluated to determine the validity and reliability of the research and in particular whether it addressed the research questions.

5.1 Design Science Evaluation Criteria

The research questions as to how data from RFID tags can be better managed in a data-on-tag approach were addressed in Chapter 2 and Chapter 4. Section 4.6 explored the suitability of the chosen data format using a set of prototype programs. The current chapter evaluates data from the experiment to quantitatively analyse the results of the prototype programs.

5.2 Prototype Implementation and Results

The main purpose of building program 3 – ‘Collect tag data’ used at the laundry pickup point was to obtain the Cartesian product of data from linen tag and bin tag and to store them in a log file which was a text file format. The log file was then used for local processing in the PC/PDA to calculate the linen count and weight. It could also be further used for enterprise level processing (which was not a part of the experiment). Program 4 – ‘Clean tag data/processing’ processed the log file generated from program 3 giving the linen count and total weight of linen in the laundry bin.

In order to measure the reliability of the prototype it is important to check whether the data captured and filtered for unique tuples (records) is accurate. This will further answer the research question as to how data from RFID tags can be better managed and quantify the success rate for read and write data in a data-on-tag approach using the distributed data concept in a pervasive environment.

To check whether data was accurately captured through the read and write process, the log file generated from program 3 was manually checked by the researcher against each RFID tag ID to see whether the tag user memory was read and then written successfully. The log file contained the contents of user memory along with the date and time the read or write took place.

In some cases although the linen tag was recognised and user memory read, the write to user memory operation failed. There were also instances where a tag ID was not successfully read. Both these unsuccessful operations were not accounted for in the final linen count and weight generated by program 4.

5.3 Experiment and RFID Readings

In every trial of the experiment, each tag was observed. If a write was successful, it appeared in the final count, updating the linen count. This was manually observed in the trial data sheet.

When set in continuous read mode RFID tags are captured repeatedly by the reader as the tags are within range of the reader for a long time (Bai et al., 2006). Although this produces duplicate data, repeated data capture reduces the opportunity for false negative readings. In the current experiment duplicate readings were managed by maintaining only unique tags through program 4. However, false negatives still remained an issue in the experiment.

The possibility of false positives was not an issue of concern in this experiment as tags going out of the reader's range was not realistic as linen attached with tags were dropped into the laundry bin and remained immobile.

It was not possible to set or maintain a particular orientation of tags in the experiment, as linen with tags were dropped into the laundry bin. These issues could potentially affect the successful reading of tag IDs. The literature on erroneous RFID reading is discussed in Section 2.8.3. The experiment also had issues relating to read from and write to user memory. This experiment is primarily a proof of concept of storing data in tags and reading and updating them in a ubiquitous distributed environment using local processing.

Table 5.1 shows sample data from Experiment 1a Trial 2. It shows how the RFID reader picks RFID tags at random. The tags were first recognised by the reader and an attempt made through the prototype to read the contents of the user memory. If successful, the reader proceeded to write data to the user memory. Where the read was not successful, the tags were not updated (written). This was highlighted and colour coded in the table. Yellow indicates a successful read tag ID, read user memory and write to user memory. Pink represents a read and green represents a write to user memory respectively, that did not happen immediately and consecutively after a successful read user memory. These errors were seen even though the prototype program had read and write to tag in linear order as explained in the pseudocode in Figure 4.22. The reader was seen to read different tags in between the highlighted pink and green rows as shown in the log file Table 5.1. The log files generated from other experiment trials are attached in Appendix H.

Table 5.1: Sample Log Data File – Experiment 1aTrial 2 with Linen – Using Seven Skyetek Tags

Source: Author

Colour code	Explanation
🟪	User memory was read successfully for a tag A; however the write procedure did not take place immediately.
🟩	Meanwhile the reader reads other tags, later tag A write process takes place shown in green colour
🟨	Successful read and write user memory in linear sequence.

Linen Tag ID.	Linen Status	Laundry Count	Linen Type	Linen Weight	DateTimeStamp	Bin Tag ID	Client Name	Client Address1	Client Address2
E0040000E90A4302	TRUE	28	FlatSheet	0.2	7/11/2009 9:48	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	TRUE	30	PatientGown	1.2	7/11/2009 9:48	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:49	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	TRUE	35	Towel	0.5	7/11/2009 9:49	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:49	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:49	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:49	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:50	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:50	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	TRUE	31	PillowCase	0.05	7/11/2009 9:50	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	TRUE	47	Blanket	2	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	48	Blanket	2	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag ID.	Linen Status	Laundry Count	Linen Type	Linen Weight	DateTimeStamp	Bin Tag ID	Client Name	Client Address1	Client Address2
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	48	Blanket	2	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:53	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:53	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	48	Blanket	2	7/11/2009 9:53	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:53	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:53	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley	Australia

Linen Tag ID.	Linen Status	Laundry Count	Linen Type	Linen Weight	DateTimeStamp	Bin Tag ID	Client Name	Client Address1	Client Address2
								Hospital	
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	48	Blanket	2	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:58	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:58	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:58	E0040000B64B1502	C123	Wellesley Hospital	Australia

It was observed that not all tags were recognised in any one experiment trial. For example in Experiment 1a, while tag E004000074251502 (Flatsheet) was recognised by the reader and its user memory read, the write procedure was not recorded at any time in the log file. This tag was manually checked using program 1- Write data into tag, to verify whether data could be written into it. Though data was successfully written through program 1, this operation failed while running the experiment with program 3 - Collect tag data, when multiple tags were involved. It was concluded that the tag was faulty and this particular tag was eliminated from the log file as it could affect the outcome of the results.

Table 5.2: Successful Read Tag ID, Read User Memory and Write User Memory Extracted from Table 5.1

Source: Author

Linen Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight	DateTime Stamp
E0040000E4F34202	TRUE	30	PatientGown	1.2	7/11/2009 9:48
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:49

The tag ID was read through program 3 as explained in pseudocode Figure 4.22 and where successful it read user memory. On a successful read, the program 3 then proceeded to write to user memory. This is evident in Table 5.2.

It was noticed in Table 5.2 that tag E0040000E4F34202 (PatientGown) was successfully read showing old data in the first row and the updated data in the next row immediately below. A successful read and write in linear sequence order is indicated in yellow for evaluation purpose and this colour coding was done manually. It was observed that the attribute Linen Status changed from ‘True’ to ‘False’ and Laundry Count was incremented by one. The Date-Time Stamp is an additional attribute included in the cartesian product Q (log file) and is not stored in the tag.

Table 5.3: Read Tag ID, Read User Memory and Write User Memory Not Occuring in Sequence

Source: Author

Linen Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight	DateTimeStamp
E0040000E90A4302	TRUE	28	FlatSheet	0.2	7/11/2009 9:48
E0040000E4F34202	TRUE	30	PatientGown	1.2	7/11/2009 9:48
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:49
E004000043FD4202	TRUE	35	Towel	0.5	7/11/2009 9:49
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:49

In Table 5.3 while tag ID E0040000E90A4302 (FlatSheet) and its user memory was successfully read as indicated in pink in the first row, no data was written immediately. There were other tags that were read and written to before the first tag was updated and written to. The original tag E0040000E90A4302 updated, is identified in green in the Table 5.3.

There is no guarantee that the write operation will follow immediately after a read operation. A tag may never be written to. Such write failures are colour coded and recorded in Appendix H. Thus there were limitations in using the RFID equipment.

5.3.1 Experiment 1a– Result Discussion

Seven Skyetek tags unattached to linen were used in this experiment. Tag E004000074251502 (Flatsheet) was excluded from the experiment analysis as it was not written to while running the experiment with the reader in the range of multiple tags. Thus the number of tags used in Experiment 1a and Experiment 2a were reduced by one to a total of six. The performance percentages of read tag ID, read and write to user memory of the tags as discussed in Section 3.5 is shown in Table 5.4. In all trials, the average percentage of tag IDs recognised was 83% which appears to support the findings from other studies indicating a 60% – 70% success read rate (Derakhshan et al., 2007). It was also observed that where tag IDs were successfully read, the read user memory operation was also successful. However the subsequent write user memory operation was not similarly successful and was least in Trial 4. The average percentage of a successful write to user memory operation in this set up was 80%.

Table 5.4: Experiment 1a Findings without Linen – Using Six Skyetek Tags

Source: Author

	1	**2**	**3**	**4**	**5**	**Average**	**Std dev between trial values**
Read tag	83%	83%	100%	67%	83%	83%	12
Read UM	83%	83%	100%	67%	83%	83%	12
Write UM	83%	83%	83%	67%	83%	80%	7
Std dev between read & write	0	0	12	0	0		

Number of Skyetek UHF tags n = 6

The data in Table 5.4 is plotted as a bar graph as shown in Figure 5.1. The read user memory and write user memory performance coincides in all trials except trial 3 where the drop is clearly visible. The unsuccessful read tag IDs, read and write to user memory affected the overall results of Trial 4 for count and weight of linen, with both the read and write activities coming below the 80% trial average.

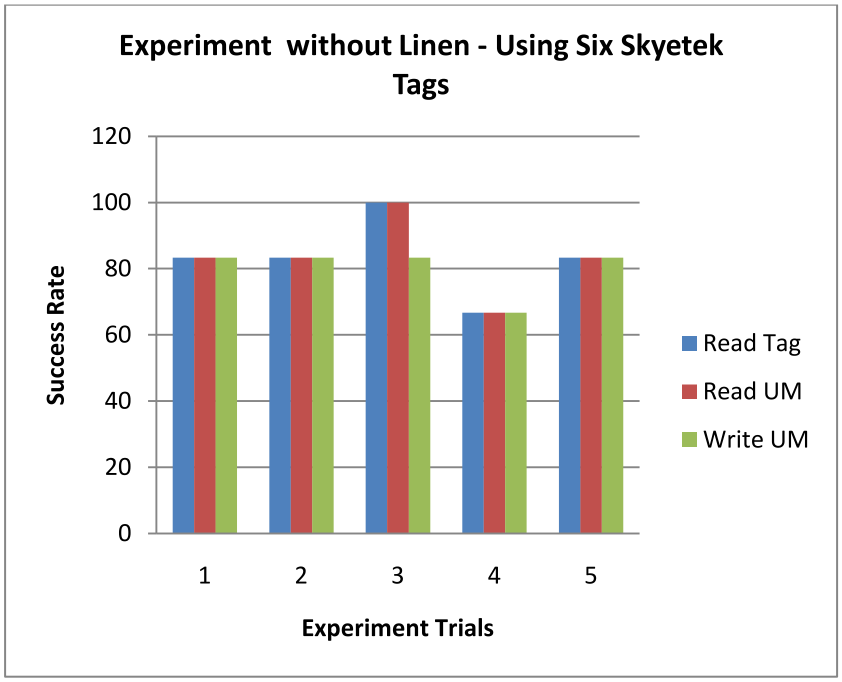


Figure 5.1: Bar Graph from Experiment 1a without Linen – Using Six Skyetek Tags

Source: Author

5.3.2 Experiment 2a - Result Discussion

Experiment 2a was conducted with Skyetek RFID tags attached to linen. The percentage of successful read tag IDs, read user memory and write to user memory operations was lower when compared to Experiment 1a as shown in Table 5.5. The average percentage of tags recognised in this experiment was 77% which was within the range of real world RFID deployment. The percentage of successful read tag IDs followed by successful read user memory in each of trials 1 to 5 were the same. However the percentage of successful write to user memory was lower for trials 3 and 4. The average percentage of successful write to user memory operation at 67% was lower than in Experiment 1a. Thus the performance of the experiment with RFID tags attached to linen was lower when compared to the same experiment without linen attached. The main reasons for the lower success rate was the many false negatives and write errors in trials 1 and 5 as observed through the trial sheets and log files of this experiment. One possible reason for this could be the impedance of RF signals caused by the linen.

Table 5.5: Experiment 2a Findings with Linen – Using Six Skyetek Tags

Source: Author

	1	**2**	**3**	**4**	**5**	**Average**	**Std dev between trial values**
Read Tag	50%	100%	100%	83%	50%	77%	25
Read UM	50%	100%	100%	83%	50%	77%	25
Write UM	50%	100%	67%	67%	50%	67%	20
Std dev between read & write	0	0	24	12	0		

Number of Skyetek UHF tags n = 6

Data in Table 5.5 is plotted as a bar graph and shown in Figure 5.2. The success rate of write to user memory is lower than that of read user memory in Trial 3 and Trial 4. The chance of write to user memory failure is greater overall, in Experiment 2a when compared to Experiment 1a.

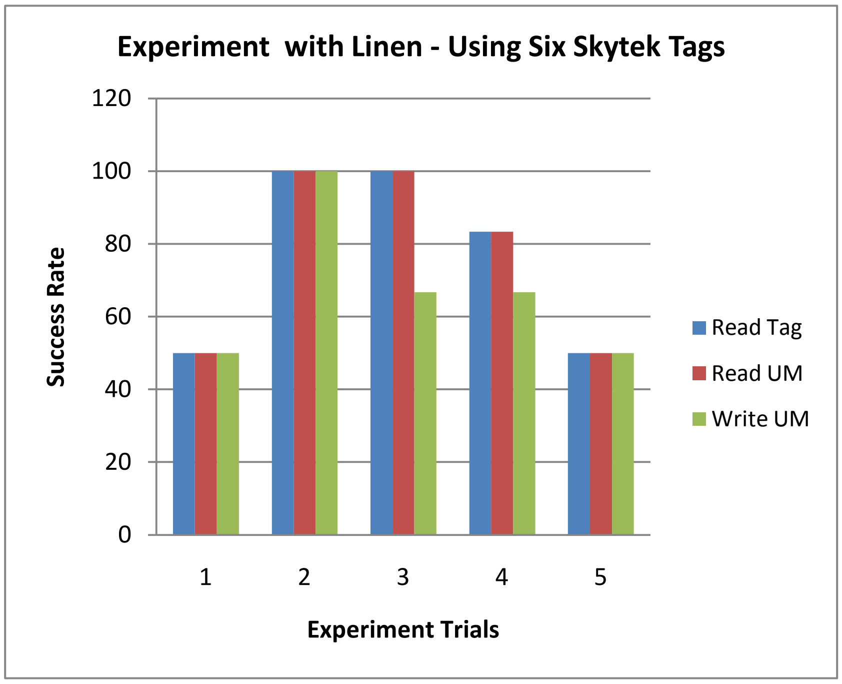


Figure 5.2: Bar Graph from Experiment 2a with Linen – Using Six Skyetek Tags

Source: Author

5.3.3 Experiment 1b– Result Discussion

Experiment 1b was conducted using Intermec tags. The set up was similar to Experiment 1a as no actual linen was involved. Intermec tags had lower user memory than Skyetek tags and informal experiments and trial sheets revealed many read and write to user memory failures. As the reader was set on continuous read mode and program 3 read tags randomly and continuously, the overall read tag rate improved as shown in Table 5.6 and Figure 5.3. However, the write to user memory rate was lower than that of read user memory in all trials.

Table 5.6: Experiment 1b Findings without Linen – Using 10 Intermec Tags

Source: Author

	1	**2**	**3**	**4**	**5**	**Average**	**Std dev between trial values**
Read Tag	100%	90%	100%	100%	90%	96%	5
Read UM	100%	90%	100%	100%	90%	96%	5
Write UM	70%	70%	70%	90%	70%	74%	9
Std dev between read & write	21	14	21	7	14		

Number of Intermec UHF tags n = 10

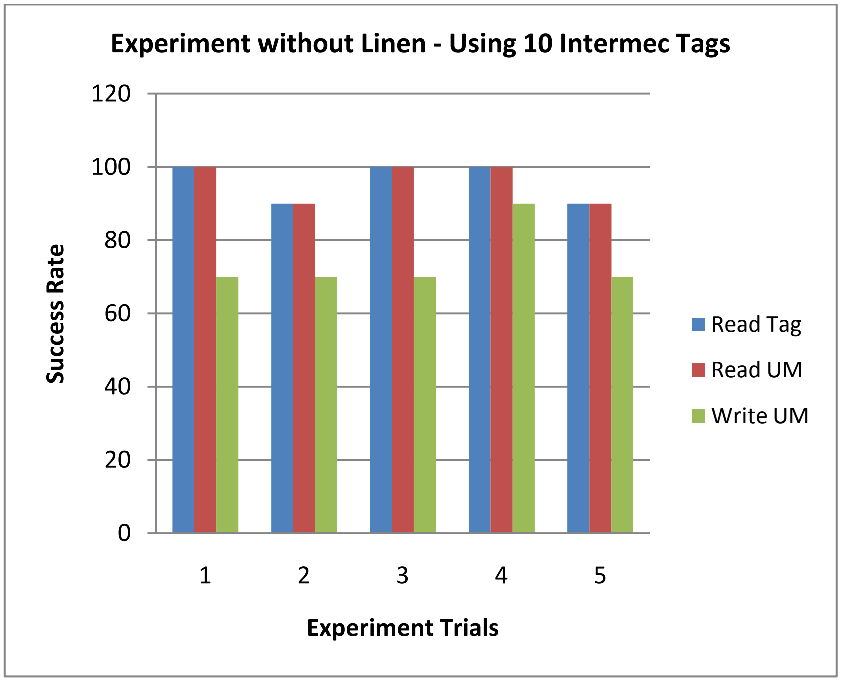


Figure 5.3: Bar Graph from Experiment 1b without Linen – Using 10 Intermec Tags

Source: Author

5.3.4 Experiment 2b – Result Discussion

Experiment 2b was conducted using 10 Intermec tags. The set up here was similar to Experiment 2a with the tags attached to linen. The success rate of write to user memory was lower than that of read user memory in all trials as shown in Table 5.7 and Figure 5.4. The average success rate for reading and writing user memory was lower than in Experiment 1b. Signal impedance caused by linen could be the main reason for the lower success rate.

Table 5.7: Experiment 2b Findings with Linen – Using 10 Intermec Tags

Source: Author

	1	**2**	**3**	**4**	**5**	**Average**	**Std dev between trial values**
Read Tag	90%	80%	90%	70%	100%	86%	11
Read UM	90%	80%	90%	70%	100%	86%	11
Write UM	70%	70%	70%	40%	80%	66%	15
Std dev between read & write	14	7	14	21	14		

Number of Intermec UHF tags n = 10

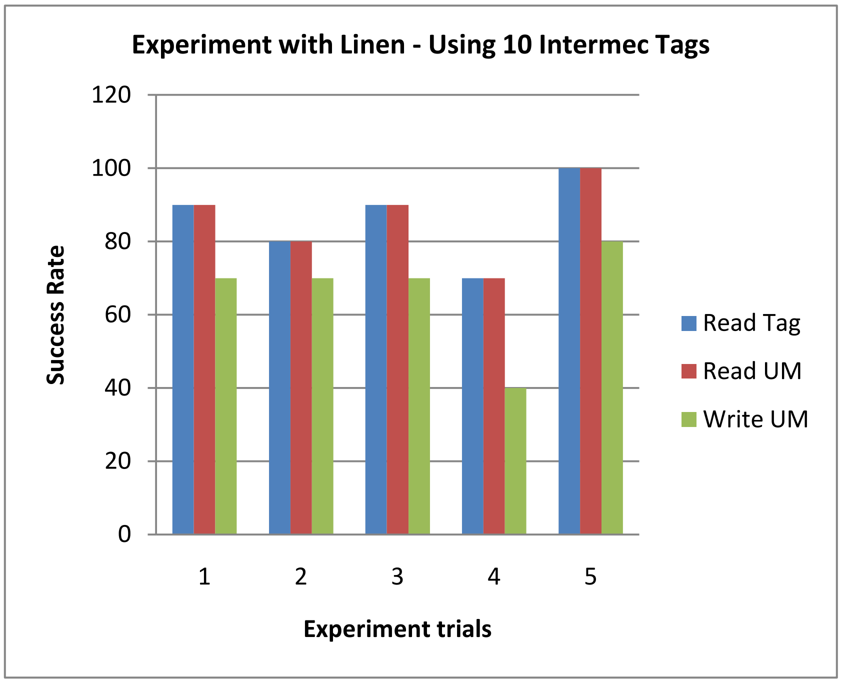


Figure 5.4: Bar Graph from Experiment 2b with Linen – Using 10 Intermec Tags

Source: Author

5.4 Overall Discussion of the Findings

Experiment 1a and Experiment 2a used Skyetek ISO 180000-6B tags and Experiment 1b and Experiment 2b used Intermec ISO 180000-6C tags.

Experiment 1a and Experiment 1b had no linen involved whereas Experiment 2a and Experiment 2b had linen attached to tags.

The performance of different tag types were observed through informal experiments and through experiment trial sheets. Although Intermec tags had many read ID and user memory failures, when compared to Skyetek tags, as evident in the log files attached as Appendix H, the overall read rate for Intermec tags was better than those of Skyetek as shown in Table 5.8 and graphically in Figure 5.5.

Write performance in both types of tags were lower when compared to their read user memory success rate. Informal experiments conducted observed that both Skyetek and Intemec tags could not be written with data size more than 64 bytes as described in Section 4.3.2. This is because the strength of radio frequency signals is sometimes not sufficient to complete a write procedure in a UHF passive tag.

The success rate for ‘write to user memory’ with linen attached was lower than those of experiments conducted without linen attached for both Skyetek and Intermec tags.

ISO 18000-6C using Intermec tags was an improvement over ISO 18000-6B using Skyetek tags for reading tag ID and user memory as evidenced in the Table 5.8. However, Intermec’s ISO 18000-6C was not better than Skyetek’s ISO 18000-6B as far as writing to user memory was concerned.

The specific reasons for varying tag performance cannot be identified. There could be several possible explanations for this. Some of these could be that the Tracient reader had limited capability to read from and write to user memory. The size of the antennae in the reader was not suitable to read and write many tags simultaneously. The Tracient reader used the existing and built-in anti-collision detection algorithm at the expense of the tags performance. The prototype was built using the SDK provided by Tracient, which had its limitations in writing data of more than 64 bytes to a tag user memory.

Tracient has recently developed an application called Enrolment tool (refer Section 4.3) which can control parameters to write 215 bytes. It requires data to be split and written in chunks of 64 bytes at a time in different block addresses. The energy is not sufficient enough to write all the data in the tag user memory in one go (Floerkemeier & Lampe, 2005). The application will have to split the data into a maximum of 64 bytes and written with the memory block offset moved to write in the appropriate position.

Table 5.8: Performance across Two Sets of Experiment – Using Skyetek and Intermec Tags

Source: Author

	Experiment 1a Skyetek tags only	**Experiment 2a Skyetek tags with Linen**	**Experiment 1b Intermec tags only**	**Experiment 2b Intermec tags with Linen**
Read tag	83%	77%	96%	86%
Read UM	83%	77%	96%	86%
Write UM	80%	67%	74%	66%

Comparing Experiment 2a with Experiment 1a and Experiment 2b with Experiment 1b, it is seen that the write performance decreased in experiments involving tags attached to linen for both Skyetek and Intermec tags as shown in Table 5.8 and Figure 5.5. This indicates that the presence of linen had some impedance effect obstructing the reader either while reading tag IDs or while reading and writing to user memory.

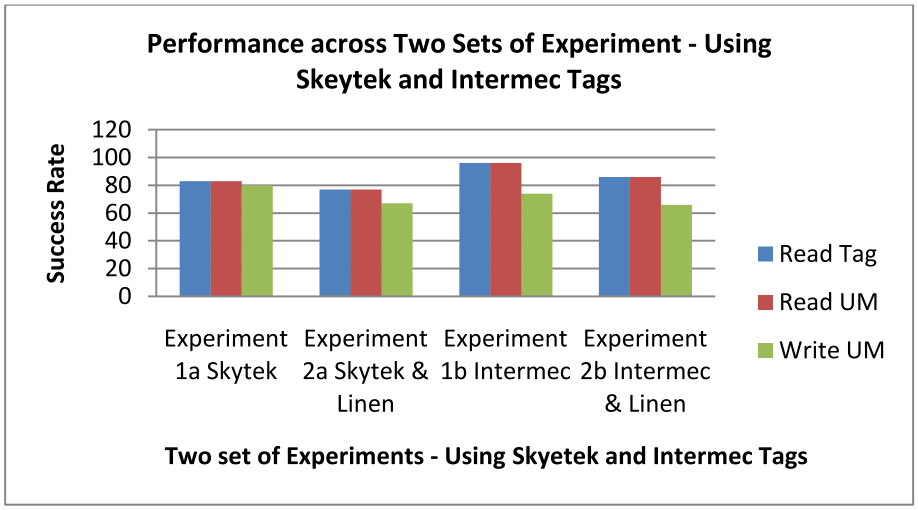


Figure 5.5: Bar Graph Showing Average Performance across Two Sets of Experiment – Using Skeytek and Intermec Tags

Source: Author

Log File Analysis

In order to further understand the reason for read and write user memory failure, the log data collected from the experiments was examined. Duplicate data in the log file was cleaned through an algorithm for sorting and considering unique records in the final processing to count the linen and their total weight. However, the log contained false negatives which continued to be an issue. If a tag ID was not recognised then the read and write to user memory operations were also failed. Thus the tag ID had to be recognised first to proceed with the data-on-tag approach.

More in depth findings and issues are discussed further in the following paragraphs.

Write Issue

Another issue seen generally was that read user memory was always successful if the tag ID was read. However write to user memory did not proceed in every case. A possible reason for this is that the UHF passive tags have no internal power supply and depend on the RFID readers power to read and write into it (Molnar, Soppera, & Wagner, 2006). This power is sometimes not sufficient to complete a write procedure into the tag if the tag has moved out of range. This was evident in some cases where data was partially written as in Experiment 1a Trial 2. While the linen status was changed from 'True' to 'False', the linen count was not incremented as observed in Table 5.9, where tag E004000043FD4202 laundry count has not changed and remained at 35.

Table 5.9: Partial write of data in tag extracted from Table 5.1

Source: Author

Linen Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight	DateTimeStamp
E004000043FD4202	TRUE	35	Towel	0.5	7/11/2009 9:49
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:50

Another example was with Intermec tags; data was corrupted and meaningless while writing into tag in Experiment 2b Trial 2 as observed in Table 5.10.

Table 5.10: Data Corrupted After Write Tag

Source: Author

Linen Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight	DateTimeStamp	
3005FB63AC1F3681EC880410	TRUE	0	Blanket	2	28/11/2009 9:55	9:55:04 p.m.
3005FB63AC1F3681EC880410	FALSE	00j9Blanket	2		28/11/2009 9:56	

Thus it is concluded that the tag needs to be in close proximity with the reader to write to user memory. Consequently, while the reader may read the tag ID and user memory in the UHF range, it may not be able to write to user memory with the same range used to read the tag. Writing to a tag requires more energy and therefore a lower range than that used to read a tag.

Distance

The performance of passive UHF RFID tags depends on the maximum distance (range) within which the RFID reader can read and write data into the tag (Nikitin & Rao, 2006). In the set up in Experiment 1a (Skyetek tags only) and Experiment 1b (Intermec tags only), tags were in close proximity to the RFID reader of approximately 12 centimetres while the tags in Experiment 2a (Skyetek tags with linen attached) and Experiment 2b (Intermec tags with linen attached) were further away as they were dropped into a box of height 22 centimetres. The overall performance of tags in Experiment 2a was lower than those in Experiment 1a. Similarly the overall performance of tags in Experiment 2b was lower than those in Experiment 1b. Thus distance could be seen as a reason for the lower performance.

Tags in Close Proximity

When tags were in close proximity with each other, there were chances of tags 'detuning' (Floerkemeier & Lampe, 2004) which could cause low read rates. Tag detuning occurs because of the external environment which decreases the resonance frequency within the tag from the operating frequency of the RFID system leading to less energy received by the tag from the reader. Further, Floerkemeier and Lampe (2004) confirmed that while smaller tag antennae decreased tag detuning, this also decreased their read range.

Orientation of Tags

Another reason for the reader not recognising the tags is possibly due to tag orientation. As the tags were scattered randomly around the reader or dropped into a box, there was no particular orientation and some tags overlapped others. In their experiments, Floerkemeier and Lampe (2004) experienced good performance in the range of 60-80% when tags were spread out or overlapped in a heap, similar to the current Experiment 1a and Experiment 1b. The performance however dropped to below 20% when tags were stacked on top of each other which could have happened in Experiment 2a and Experiment 2b as linen with tags were dropped in the box.

Signal Impedance

Another possible issue for reduced read tag ID could be signal impedance which was a possibility in Experiment 1b and Experiment 2b as actual linen like pillowcases and small hand towels were used as part of the simulation of a real laundry bin setting. Interference caused by linen could have reduced the read and write rate.

Tag Antenna

Although RFID technology allows contact-less, non line-of-sight and multiple reading, the reading efficiency is affected by the reader antenna and tag antenna (Wu et al., 2006). If the radio signals coming from tag antenna are not strong enough for the reader antenna to capture, the tag may not be recognised. It is also known that if the tag antenna is perpendicular to a reader antenna then the reader is not able to recognise the tag (Nikitin & Rao, 2006; Wu et al., 2006).

Tag Collision

It is an accepted fact in the literature studied that when multiple tags are scattered, their signals could collide and cancel out each other (Shih, Sun, Yen, & Huang, 2006). The tag would then wait to retransmit the tag ID if detected by the reader again. The tag collision would depend on the anti collision protocol discussed in Section 2.3.3 on RFID tag standard. This was beyond the scope of the current study.

5.5 Addressing Research Questions

All research questions except the last one were covered in Chapter 2 and Chapter 4. The last research question on success rate for reading and writing is covered in the current chapter.

How accurate is the reading and writing of data using a data-on-tag approach?

The success rate outlined in Section 3.5 was calculated and shown in tabular form and bar graphs. Reading tag IDs and user memory was in the expected range. However writing to user memory proved problematic. The possible reasons for read and write failure were analysed.

5.6 Summary

This chapter presented the research findings and analysis to answer the research question on accuracy of reading and writing data through a data-on-tag approach.

The results and evaluation of the performance of the experiments were explained through appropriate tables and bar graphs where necessary.

Although the results were not completely successful, the reasons for failure have also been considered. The performance of read tag ID and read user memory are at acceptable levels, and are similar to those in a practical real world situations, writing to passive tags did not achieve similar success rates.

6 Research Contribution, Limitation and Future Work

The laundry bin experiment was conducted to read and write data using RFID tags in the environment without human intervention. This chapter presents the contributions of this research along with the limitations of the current study and also makes recommendations for future work in this direction. It may be noted that the experiment results may vary using RFID readers from different vendors which will have variations in the associated software used in reading and writing the tags.

6.1 Research Contribution

Implications for industry: The COTS software and RFID readers and tags developed need further improvement to evolve ideas such as data-on-tag. Write procedure using passive UHF tags and UHF RFID reader need further development in the firmware at the manufacturing level.

Implications for academia: As write procedure is not accurate, more research is required on this aspect. Improvement in the application software to check data is correctly written could be carried out.

6.2 Addressing Research Questions

In this section the research questions are stated and an overall discussion is provided. The research questions were also addressed in earlier chapters 2, 4 and 5 to update the progress of the research study covering the research questions.

What data format is suitable for a data-on-tag approach?

Researching on active and passive tags, passive tags proved to be more suitable to the laundry application. Passive tags do not require external power supply to activate and works on power drawn from the reader. Further, research on data to be stored on the tag (user memory) asked for a data model to be designed. With limited memory in user memory, data format like CSV and XML were compared. CSV was recommended as a suitable format after considering the size of the data to be stored and the capacity of the COTS hardware and software available to write data in the user memory.

How can data be stored in the RFID tag in a data-on-tag approach?

Although CSV was identified as a suitable format for this study, data was actually stored as a string in the specific address block in the tag which indicated user memory. Even if XML format was chosen, the same process of writing data as a string would be considered in the COTS software provided by the vendor.

How is data from RFID tags read and written in a data-on-tag approach?

The SDK provided by Tracient had read and write functions to enable one to write data as a string in specific memory blocks in the tag. The prototypes had read and write functions provided by Tracient SDK. CSV data format was read and written as strings through the read and write functions in the prototype. The experiments were conducted by running the programs in the prototype.

How accurate is the reading and writing of data using the data-on-tag approach?

With the suggested CSV data format, and considering the limited user memory size and write capacity to the tag by the Tracient reader and SDK provided, two experiments were set up with two tags of different standards and user memory sizes. Different situations of 'tags only' and 'tags attached to linen' were considered. The overall results for reading tag IDs and reading user memory were within the range of real world deployment. However the accuracy of write to user memory was between 3% to 10% lower than read user memory for Skyetek tags and between 20% to 22% lower for Intermec IT67 tags. When the tags were attached to linen in the experiment, possible signal impedance effect reduced the accuracy of write to user memory by a further 13% and 8% respectively for Skyetek and Intermec IT67 tags.

6.3 Limitations of the Study

Various limitations of the study are presented in the following sections. The RFID technology used was from a specific vendor, Tracient Technologies Ltd., who also provided the reader and SDK software. The Tracient UHF reader was compatible with Skyetek and Intermec tags, working within a similar frequency range. Setting the anti-collision protocol in the reader was beyond the scope of this experiment. The prototype developed could read data from and write data into the tag memory. However the prototype was not checked with different readers from other vendors and other types of tags.

6.3.1 Limited Range and Moisture

The experiment was conducted in a simulated environment with a reduced size of the laundry bin giving the RFID reader a smaller range from which to read and write tags in the bin. This was a proof of concept experiment to simulate data in the environment as tuples to be put together and used for local processing using the data-on-tag approach. In a commercial situation, the bin is larger and the reader will have to read and write from a much bigger much range to recognise all the tagged linen in the bin. Write procedure was possible only in short distance close to the reader.

Soiled linen and linen with moisture was not involved in the experiment as the tags available for the experiment were not suitable to withstand moisture. Moisture can impede the RF signal and diminish the read range. Moisture could also increase the total weight of the linen.

6.3.2 Data Format

CSV data format was selected to write the entire data in a single procedure because of the limited user memory. Although XML format was considered, it was not suitable for writing strings with more than 64 bytes in a single write procedure with the SDK provided by the vendor. Although the alternate of moving the block offset could be used to write more than 64 bytes, the possibility of the tag moving out of the reader's range could disrupt the write procedure.

6.3.3 Read and Write

The laundry bin experiment experienced failures reading from and writing into tags. These problems were analysed in Section 5.4. Missed readings were anticipated and provisioned for by keeping the reader mode in continuous read mode and by then eliminating duplicate data in the log file.

6.3.4 Traceability and Visibility

In a data-on-tag approach~~,~~ the status of the object is not known at the central database. There is no way to track each item at any point in time. This is in contrast with SCMS applications, where, objects can be tracked through the system and the exact location of each object known.

6.3.5 Actual and Calculated Weights

If the linen count and linen weight of all linen in the laundry bin generated through program 4 matches the physical linen count and weight, then the process of reading and writing all the linen tag was correct. In other words the experiment has successfully recognised all the linen tags. This will help in establishing the reliability of RFID technology in the industry. However, if the two do not match, then this can alert the system that not all tags were read and written.

Although no water was involved in the current experiment, in reality soiled and/or wet linen may physically increase the weight of the linen bag. Although a small sample was considered in the current experiment, heavy linen like blankets can reduce the final weight dramatically if not recognised by the system. This can affect the laundry company if it bills its clients by weight.

6.4 Direction for the Future Study

The focus of this study was to achieve a basic understanding of the data-on-tag approach and how more information can be stored in a tag. The information was useful in a distributed application by reading and writing data in a tag. Although a business problem of hospital laundry was considered, this process can be applied to other situations where there is no access to a network database and application.

The limitations described Section 6.3 may present opportunities for future research as described in the sections between 6.4.1 to 6.4.4.

6.4.1 Limited Range

With improvements in the RFID technology an increased range to read and write may be possible in the near future. This research also identifies opportunities for researchers and hardware manufacturers to improve the write procedure on an RFID tag.

6.4.2 Data Format

In the future, as memory capacities in active and passive RFID tags increase, different data formats like the XML could be considered. It is still possible to have XML format in passive tags, with data written in parts as a long string may need more energy from the RFID reader to enable a successful write operation. An XML format will have the schema built into the data making it easy for any application reading it to build the schema on the fly.

With increased memory capacity in a tag, a portable database with all components of the data model (Codd, 1982) as in smart cards (Bobineau, Bouganim, Pucheral, & Valduriez, 2000) or flash memory (Kim, Baek, Lee, & Lee, 2006) could be considered. Another possibility is to store data in the tag as a Java applet which could be opened in any browser.

With greater awareness of the need of this concept of data-on-tag, manufacturers should be able to produce tags with larger user memory capacity at lower cost. Better compression techniques should be able to reduce the data size to fit into the memory of an RFID tag.

6.4.3 Read and Write

Skyetek and Intermec tags used in the experiment were of EEPROM type. Ferro Magnetic Random Access Memory (FRAM) uses lesser power to write to a tag (Finkenzeller, 2003, pp. 300-301). Thus FRAM can improve write performance.

In the current experiment, writing data of more than 64 bytes at a time was a limitation. This can be overcome by offsetting the block address and writing 64 bytes of data at a time. However, if the tag is moved during a write procedure, it will corrupt the data written.

An alternate way to improve write procedure is by incorporating integrity rules as in Codd’s theory. The application will record the tag data and if a write failure occurs, rewrite from the application into the tag will proceed.

6.4.4 Log File

The log file generated through prototype program 3 was manually evaluated. In future the log file could be used by the enterprise system to maintain an audit trail and keep track of the status of data (the linen) with date time stamp.

6.5 Conclusions

The current research study explored the applications using data-on-tag approach. The theoretical concept of data-on-tag approach is impressive. However the viability of data-on-tag approach is dependable on the RFID technology.

In the data-on-tag approach, CSV format appeared as the most relevant data format. CSV format stored the data in less bytes than XML. The write procedure through the COTS software was able to write only 64 bytes at a time which suited CSV format.

Experiment results using CSV format demonstrated sufficient proof of concept. Accuracy of the results was as expected for reading. However writing proved more problematic.

6.6 Summary

This research study was to explore RFID technology and how tags were used to identify objects in the physical world. Data-on-network could read tag ID (EPC) and look up more information in the network database server. The RFID system needs to be well integrated with the enterprise system and middleware software is required to filter and clean the stream of data collected from tags. However the other approach of data-on-tag can store required data about the physical object in the tag and does not need a network database server or direct integration into the enterprise system. This could be ideal for organisations with critical processes managed by legacy systems to adopt state of the art technologies like RFID, where RFID data can be processed locally, as all required data is stored in the tag. Processed RFID data can then be exported to the backend enterprise.

Although RFID technology is relatively expensive, its initial cost could be offset by savings in labour and replacement costs. In addition, using the data-on-tag concept, value addition could be achieved by obtaining accurate data on the number of times linen is laundered and thus information on the life of the linen, its durability and throughput estimates for the system. These added benefits should be taken into account when considering the cost of implementation of the system.

This thesis study outlined the RFID technology proposed for a laundry bin scenario. Passive tags with additional memory were used as they could write data into the tag. Local processing such as checking the total weight and calculating the total number of linen in the bin could take place without the infrastructure of a network, backend database or middleware. This laundry bin would be able to access RFID tagged linen without access to a network database and update data in the linen (clean or soiled), while the linen is processed, with no ambiguity of the linen status physically and in the database.

Although the cost of tags is relatively high, falling costs and eliminating expensive middleware should improve the economic viability of the concept. Since the data-on-tag approach was undertaken in this thesis study, a suitable data format was identified. The count and weight of soiled linen collected from the hospital was calculated close to source of collection. Later the transaction data captured at the PC/PDA was sent to the data warehouse for enterprise level computing. This helped the laundry company bill their clients efficiently taking care of lost linen. As writable tags are used with all relevant data written on it, this experiment is a novel approach compared to the regular data-on-network approach.

Literature discussion on data format and data compression techniques helped in exploring and understanding the data-on-tag approach. The performance of the laundry bin experiment was a proof-of-concept of this approach with acceptable results in most experiment trials; not withstanding the write procedure had a higher failure rate. The limitations of the experiment are analysed and future directions explored, for researchers to carry the baton to improve the data-on-tag approach. This research identifies opportunities for researchers and hardware manufacturers to improve the write procedure on an RFID tag.

References

Asher, C., Morgan, G., Swan, R., & Traub, K. (2007). *EPCIS (Electronic Product Code Information Service)*. Retrieved 19-05-2010, from http://www.epcglobalinc.org/standards/epcis/epcis_1_0-faq-20070427.pdf

Bacheldor, B. (2009). *Tego Launches 32-Kilobyte EPC RFID Tag*. Retrieved 3-3-2010, from http://www.rfidjournal.com/article/view/4578

Bai, Y., Wang, F., & Liu, P. (2006). Efficiently filtering RFID data streams. *VLDB Workshop on Clean DB, Seoul, Korea.*, 1-8.

Bobineau, C., Bouganim, L., Pucheral, P., & Valduriez, P. (2000). *PicoDBMS: Scaling down Database Techniques for the Smartcard.* Paper presented at the Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt. from

Bohn, J., & Mattern, F. (2004). Super-distributed RFID tag infrstructures. 1-12.

Chamberlin, D., & Robie, J. (2001). Quilt: An XML query language for heterogeneous data sources. *Lecture Notes in Computer Science*, 1-25.

Chan, A. T. S., Cao, J., Chan, H., & Young, G. (2001). A web-enabled framework for smart card application in health services. *Communications of the ACM, 44*(9), 77-82.

Chawanthe, S., Krishnamurthy, V., Ramachandra, S., & Sarma, S. (2004). Managing RFID data. *Proceedings of the VLDB, Toronto, Canada*, 1189-1195.

Cheney, J. (2006). Tradeoffs in XML database compression. *Proceedings / Data Compression Conference, Utah, U.S.A.*, 392 - 401.

Christian, F., & Matthias, L. (2005). *RFID middleware design: addressing application requirements and RFID constraints.* Paper presented at the Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies, Grenoble, France. from doi:http://doi.acm.org/10.1145/1107548.1107603

Clarke, R. H., Twede, D., Tazelaar, J. R., & Boyer, K. K. (2006). Radio Frequency Identification (RFID) performance: The effect of tag orientation and package contents. *Packaging Technology and Science, 19*(1), 45-54.

Codd, E. F. (1979). Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems, 4*(4), 397 - 434.

Codd, E. F. (1980). Data models in database management. *Proceedings of the workshop on Data Abstraction, Databases and Conceptual modelling, 11*, 112-114.

Codd, E. F. (1982). Relational database: a practical foundation for productivity *Communications of the ACM 25*(2), 109-117.

Cokus, M., & Pericas-Geertsen, S. (2005). *XML binary characterization properties*. Retrieved 2 March 2010, from http://www.w3.org/TR/xbc-properties/#compactness

Curtin, J., Kauffman, R. J., & Riggins, F. J. (2007). Making the 'MOST' out of RFID technology: a research agenda for the study of the adoption, usage and impact of RFID. *Inf Tecnol Manage, 8*, 87-110.

de Azambuja, M. C., Marcon, C. A. M., & Hessel, F. P. (2008). Survey of standardised ISO 18000-6 RFID anti-collision protocols. *The Second International Conference on Sensor Technologies and Applications, Cap Esterel, France*, 468-473.

Derakhshan, R., Orlowska, M. E., & Li, X. (2007). RFID data management: challenges and opportunities. *IEEE International Conference, Texas, USA*, 175-182.

Diekmann, T., Melski, A., & Schumann, M. (2007). Data-on-network vs. Data-on-tag: Managing data in complex RFID environments. *Proceedings of the 40th Annual Hawaii International Conference on System Sciences, Hawaii*, 224-233.

Finkenzeller, K. (2003). RFID handbook: fundamentals and applications in contactless smart cards and 300-301.

Floerkemeier, C., & Lampe, M. (2004). Issues with RFID usage in ubiquitous computing applications. *Pervasive Computing: Second International Conference, Vienna Austria*, 188-193.

Floerkemeier, C., & Lampe, M. (2005). RFID middleware design - addressing application requirements and RFID constraints. *Joint sOc-EUSAI conference, Grenoble*, 1-6.

Flynn, D. M. (1996). Laundry 'Par' could put you in the rough. *Cleaning & Maintenance Management, 33*(12).

Gao-RFID. (n.d.). *Gao RFID Inc - 860-960 MHz. Gen 2 Laundry Tag*. Retrieved 1-03-2010 from http://www.gaorfid.com/index.php?main_page=product_info&cPath=87&products_id=785&zenid=ed2af84578f820bae225ed3cfb4ce4a7

Glover, B., & Bhatt, H. (2006). *RFID essentials*: O'Reilly, US.

Gray, J. (2004). The next database revolution SIGMOD. *Paris, France*, 1-4.

Haas, L. M., & Miller, R. J. (1997). Transforming heterogeneous data with database middleware: Beyond integration. *Bulletin of the IEEE Computer Society Technical Committee on Data engineering*, 1-6.

Hardgrave, B. C., Armstrong, D. J., & Riemenschneider, C. K. (2007). RFID assimilation hierarchy. *Proceedings of the 40th Hawaii International conference on System Sciences, Hawaii*, 1-10.

Harmon, C. K. (2006). The necessity for a uniform organisation of user memory in RFID. *International Journal Radio Frequency Identification Technology and Applications, 1*(1), 41-51.

Harrison, M. (2003). EPC information service- data model and queries. *White Paper, Auto-d Centre Institute for Manufacturing, University of Cambridge, United Kingdom*, 1-20.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly, 28*(1), 75-105.

Huffman, D. A. (1952). A method for the contruction of minimum redundancy codes. *IRE*, 1098-1101.

ISO. (2007). *Information Technology - Radio Frequency Identification for item management*. Retrieved 1-1-2009, from http://www.iso.org

Jeffery, S. R., Garofalakis, M., & Franklin, M. J. (2006). Adaptive cleaning for RFID data streams. *VLDB, Seoul, Korea.*, 163-174.

Jiang, W., & Xiang, D. (2008). A compression framework for personal image used in mobile RFID system. *9th International Conference for Young Scientists, Zhang Jia Jie, China*, 769-774.

Joseph, M. H. (2003). Toward network data independence. *SIGMOD Rec., 32*(3), 34-40.

Kabadayi, S., Pridgen, A., & Julien, C. (2006). Virtual sensors: Abstracting data from physical sensors. *Proceedings of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks, IEEE Computer Society*, 587-592.

Kim, G., Baek, S., Lee, H., & Lee, H. (2006). LGeDBMS: a small DBMS for embedded system with flash memory. *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea*, 1255-1258.

Kotak, D. B., & Gruver, W. A. (2009). Distributed intelligent RFID systems. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, San Antonia, TX, USA*, 1214 - 1218.

Landt, J. (2005). The history of RFID. *Potentials IEEE, 24*(4), 8 – 11.

Liefke, H., & Suciu, D. (2000). XMill: An efficient compressor for XML data. *SIGMOD 29*(2), 153-164.

Lin, D., Elmongui, H. G., Bertino, E., & Ooi, B. C. (2007). Data management in RFID applications. *DEXA*, 434-444.

Mamei, M., Quaglieri, R., & Franco Zambonelli, F. (2006). Making tuple spaces physical with RFID tags. *Proceedings of the 2006 ACM symposium on Applied computing, Dijon, France*, 434 - 439.

March, S. T., & Smith, G. F. (1995). Design and natural science research in information technology. *Decision Support Systems, 15*, 251-266.

Melski, A., Thoroe, L., Caus, T., & Schumann, M. (2007). Beyond EPC – Insights from multiple RFID case studies on the storage of additional data on tag. *International Conference on wireless Algorithms, Systems and Applications, Chicago*, 281- 286.

Microsoft. (2008). *System.IO.Compression Namespace*. Retrieved 10-04-2009, from http://msdn.microsoft.com/en-us/library/3z72378a(v=VS.90).aspx

Molnar, D., Soppera, A., & Wagner, D. (2006). A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. 276-290.

Moody, D. L. (1998). Metrics for evaluating the quality of entity relationship models. *Springer-Verlag Berlin Heidelberg*, 211-225.

Ni, L. M., & Liu, Y. (2004). LANDMARC: Indoor location sensing using active RFID. *Wireless Networks, 10*, 701-710.

Nikitin, P. V., & Rao, K. V. S. (2006). Performance limitations of passive UHF RFID systems. *Proceedings of the IEEE Antennas and Propagations Society International Symposium*, 1011-1014.

Nunamaker, J., Chen, M., & Purdin, T. D. M. (1991). Systems development in information systems research. *Journal of Management Information Systems 7*(3), 89-106.

O'Connor, M. C. (2006). *Gen 2 EPC protocol approved as ISO 18000-6C*. Retrieved 14-05-2010, from http://www.rfidjournal.com/article/articleview/2481/1/1/

Palmer, M. (2004). Principles of effective RFID data management. *Enterprise Systems*, 1-8.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2008). A Design Science research methodology for information systems research. *Journal of Management Information Systems, 24*(3), 45-77.

Romer, K., Schoch, T., & Mattern, F. (2004). Smart identification frameworks for ubiquitous computing applications. *Wireless Networks, 10*, 689-700.

Shafranovich, Y. (2005). *Common format and MIME type for Comma-Separated Values (CSV) files*. Retrieved 17th Maarch 2010, from http://tools.ietf.org/html/rfc4180

Shih, D., Sun, P., Yen, D. C., & Huang, S. (2006). Taxonomy and survey of RFID anti-collision protocols. *Computer Communications, 29*, 2150–2166.

Stockman, H. (1948). Communications by means of reflected power. *Proceedings of IRE.*, 1196-1204.

Sugawara, K., Yamaoka, K., & Sakai, Y. (1997). A study on image searching method in super distributed database. *IEEE Global Telecommunications Conference, Phoenix, AZ, USA, 2*, 736-740.

Tracient. (2007a). *Tracient user manual*. from www.tracient.com

Tracient. (2007b). *Padl-R UHF RFID reader*. Retrieved 21-11-2008, from www.tracient.com

Tracient. (2007c). Tracient RFID SDK API: C# Programmer's Guide.

Tran, T., Sutton, C., Cocci, R., Nie, Y., Diao, Y., & Shenoy, P. (2009). Probabilistic inference over RFID streams in mobile enviornment. *25th International Conference on Data Engineering, Shanghai, China*, 1-12.

Tsai, T. (2008). *Map compression for a RFID-based two-dimensional indoor navigation*

System. unpublished thesis, Auckland University of Technology, Auckland.

Vaishnavi, V. K., & Kuechler, W. J. (2008). *Design Science research methods and patterns – Innovating information and communication technology*: Auerbach Publications, New York.

Wal-Mart spells out RFID vision, RFID Journal 2003. Retrieved 8-09-2010, from http://www.rfidjournal.com/article/purchase/463

Wang, F., & Liu, P. (2005). Temporal management of RFID data. *Proceedings of the 31st VLDB Conference, Trondheim, Norway*, 1128-1139.

Want, R. (2004). The magic of RFID. *Queue, 2*(7), 40-48.

Want, R. (2006). An introduction to RFID technology. *IEEE pervasive computing, 5*(1), 25-33.

Ward, M., Kraneneburg, R., & Backhouse, G. (2006). RFID: Frequency, standards, adoption and innovation. *JISC Technology and standards Watch*, 1-36. Retrieved from http://www.rfidconsultation.eu/docs/ficheiros/TSW0602.pdf

Weiser, M. (1993). Hot topics - Ubiquitous computing. *Computer 26*, 71-72.

Willis, S., & Helal , S. (2005). *RFID information grid for blind navigation and wayfinding.* Paper presented at the Proceedings of the ninth annual IEEE International Symposium on Wearable Computers, Osaka, Japan. from http://www.icta.ufl.edu/projects/publications/willis-RFID-ISWC%20v2.pdf

Wu, N. C., Nystrom, M. A., Lin, T. R., & Yu, H. C. (2006). Challenges to global RFID adoption. *Technovation, 26*(12), 1317-1323.

Ziv , J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory, 23*(3), 337–343.

Appendices

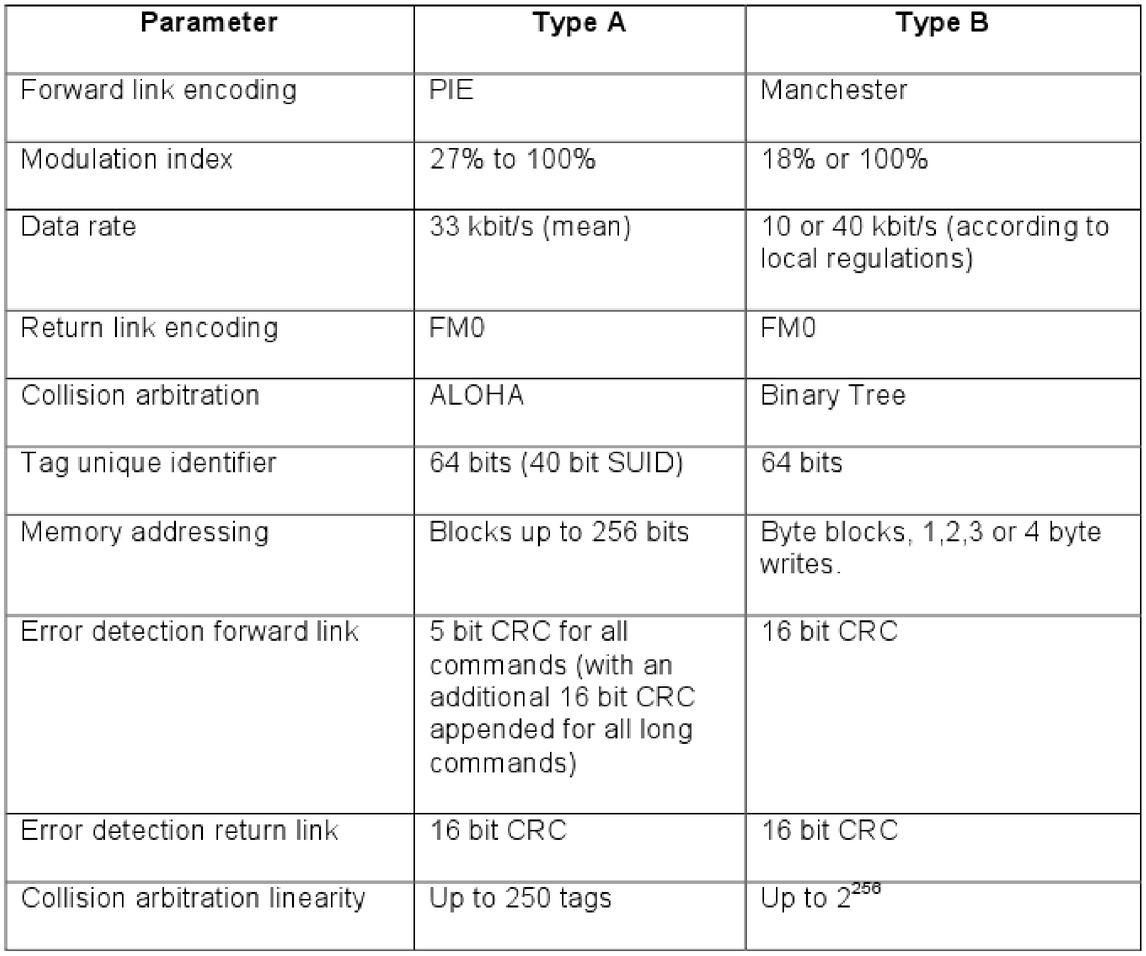
Appendix A - ISO/IEC 18000

Source: (ISO, 2007)

ISO/IEC 18000	RFID for Item Management Part 1: Defines the foundation for all air interface definitions in the ISO/IEC 18000 series. Part 2: Parameters for air interface communications below 135 kHz Type A (FDX): 125 kHz Type B (HDX): 134.2 kHz Part 3: Parameters for air interface communications at 13.56 MHz Part 4: Parameters for air interface communications at 2.45 GHz Passive tag operating as an interrogator talks first Battery assisted tag operating as a tag talks first. Part 6: Parameters for air interface communications at 860 MHz to 960 MHz Type A and type B with the primary difference being the anti-collision algorithm used. Type C - also known as EPCglobal Class 1 Gen 2. Part 7: Parameters for active air interface communications at 433 MHz

Appendix B - Comparison of ISO 18000-6A and ISO 18000-6B

Source: : (ISO, 2007)



Parameter	Type A	Type B
Forward link encoding	PIE	Manchester
Modulation index	27% to 100%	18% or 100%
Data rate	33 kbit/s (mean)	10 or 40 kbit/s (according to local regulations)
Return link encoding	FM0	FM0
Collision arbitration	ALOHA	Binary Tree
Tag unique identifier	64 bits (40 bit SUID)	64 bits
Memory addressing	Blocks up to 256 bits	Byte blocks, 1,2,3 or 4 byte writes.
Error detection forward link	5 bit CRC for all commands (with an additional 16 bit CRC appended for all long commands)	16 bit CRC
Error detection return link	16 bit CRC	16 bit CRC
Collision arbitration linearity	Up to 250 tags	Up to 2^{256}

Appendix C - Datasheet for Tracient Reader

Source: (Tracient, 2007b)

Size	210mm (h) x 70mm (w) x 15 mm (d) (approx.)
Weight	135 grams (approx.)
Operating temperature / storage temperature	-10 to +60º C
Ingress Protection	IEC 529 IP64
Approvals	CE, C-Tick, FCC
Drop specification	1m to plywood on all faces and corners.
User Interface	Single button for RFID reading, Multi-colour Leds, Audio beeper (multi tone). Configurable use case scenarios via USB or Bluetooth.
UHF Transponder	ISO ISO 18000-6A, ISO 18000-6B, ISO 18000-6C, EPC Class 0, EPC Class 1 (GEN2). Frequency 862-955 MHz
Approximate read range	1.5m
Data rate	~40 kbps (EPC C1G2), ~80 kbps (ISO18000 - 6C)
Maximum O/P Power	0.5 W
Power control range / Power control	3 mW - 0.5W (+5dBm - +27dBm) 1dBClass 2 (typically 10m

resolution / Bluetooth Interface	operation).
Antenna	Internal only
Additional Data communication Interface	USB – mini-B connector for wired applications
Power Supply	Via USB for high-capacity internal Li-Ion battery recharge
Firmware support	Field upgradeable via USB. Some configuration settings also available via Bluetooth interface.

Appendix D - Partial Prototype Code

```
//Program : Read and write User Memory (SDK by Tracient Technologies
Ltd.)
//Updated by : Sarita Pais
//Date: 23 March 2009
//Updated Program: Read each tag Id. write linen data into the tag, read
linen data


using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;   // for Marshal

using Tracient.RFID;

namespace Example3
{
    public partial class Example3 : Form
    {

        RfidUsb usbRFID = new RfidUsb();
        IntPtr hReader;

        bool linenStatusBool = false;
        bool linenTypeBool = false;
        bool linenWeightBool = false;
        public Example3()
        {
            InitializeComponent();

            hReader = IntPtr.Zero;

            uint numDevices = usbRFID.GetNumDevices(false);
            this.RefreshList(numDevices);

            btnDisConnect.Enabled = false;
            btnReadTag.Enabled = false;
            btnReadData.Enabled = false;
            btnWriteData.Enabled = false;

        }

        private void btnRefresh_Click(object sender, EventArgs e)
        {   // Do a new search for the number of devices
            this.Cursor = Cursors.WaitCursor;

            uint numDevices = usbRFID.GetNumDevices(true);
            this.RefreshList(numDevices);

            this.Cursor = Cursors.Default;
        }
        private void btnConnect_Click(object sender, EventArgs e)
        {
            if (hReader == IntPtr.Zero)
            {
                int dev = this.cmbReaders.SelectedIndex;
                Object item = this.cmbReaders.SelectedItem;

                if (item.ToString() == "No Readers")
                {
                    MessageBox.Show("No readers available.");
                }
                else
                {
                    this.Cursor = Cursors.WaitCursor;

                    try
                    {   // Open the connection & subscribe to
notifications
                        hReader = usbRFID.Open((uint)dev);

                        // Set-up the UI
                        btnConnect.Enabled = false;
                        btnDisConnect.Enabled = true;
                        btnReadTag.Enabled = true;

   // Only allow read/write of tag data if have a tag ID is valid
                        if (txtTagID.Text.Length > 0)
                        {
                            btnReadData.Enabled = true;
                            btnWriteData.Enabled = true;
                        }
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.Message);
                    }
                    finally
                    {
                        this.Cursor = Cursors.Default;
                    }
                }
            }
            else
            {
                MessageBox.Show(this, "Reader already connected!",
"Example3", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

        }

        private void btnDisConnect_Click(object sender, EventArgs e)
        {
            this.Cursor = Cursors.WaitCursor;

            if (hReader != IntPtr.Zero)
            {
                usbRFID.Close(hReader);
                hReader = IntPtr.Zero;
                usbRFID.OnConnectionLost -= OnConnectionLost;
                usbRFID.OnAsyncRead -= OnAsyncRead;
            }
            btnConnect.Enabled = true;
            btnDisConnect.Enabled = false;
            btnReadTag.Enabled = false;
            btnReadData.Enabled = false;
            btnWriteData.Enabled = false;

            this.Cursor = Cursors.Default;
        }

        /// <summary>
        /// Read the Tag ID
        /// </summary>
        private void btnReadTag_Click(object sender, EventArgs e)
        {
            if (hReader != IntPtr.Zero)
            {
                this.Cursor = Cursors.WaitCursor;

                try
                {
                    string TagID;
                    // Read the Tag ID
                    TagID = usbRFID.Read(hReader);
                    if (TagID.Length > 0)
                    {
                        txtTagID.Text = TagID;
                        btnReadData.Enabled = true;
                        btnWriteData.Enabled = true;
                        this.tagDatalistBox.Items.Add(txtTagID.Text +
"...Tag read");
                    }

                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    this.Cursor = Cursors.Default;
                }
            }
        }

        /// <summary>
        /// Read Data from the tag
        /// </summary>
        private void btnReadData_Click(object sender, EventArgs e)
        {
            if (hReader != IntPtr.Zero)
            {
                this.Cursor = Cursors.WaitCursor;

                System.IntPtr lpTagData = IntPtr.Zero;

                try
                {
                    // Declare our variables
                    byte[] abTagDataBytes;
                    string strTagData;

                    uint uiBytesRead = 0;
                    uint uiAddress = 0;
                    uint uiBytes = 20;  // was 26 skytek

                    // Get the tag ID
                    StringBuilder strTagID = new
System.Text.StringBuilder(txtTagID.Text);


                        // Allocate unmanaged memory used to store tag
data
                        lpTagData = Marshal.AllocHGlobal((int)uiBytes);
                        Object itemMemType = "User";
                        // call the correct readtag function
                        switch (itemMemType.ToString())
                        {
                            case ("User"):
                                {
                                    uiBytesRead =
usbRFID.ReadTag_UserMemory(hReader, strTagID,

lpTagData, uiAddress, uiBytes);
                                    break;
                                }


                            default:
                                {
                                    break;
                                }
                        }

                        // Now display the bytes we read as text
                        // for this example we are going to assume the
data was ascii
                        abTagDataBytes = new byte[uiBytesRead];
                        Marshal.Copy(lpTagData, abTagDataBytes, 0,
(int)uiBytesRead);

                        System.Text.ASCIIEncoding asciiEncoder = new
System.Text.ASCIIEncoding();
                        strTagData =
asciiEncoder.GetString(abTagDataBytes);

                        txtReadData.Text = strTagData;
                        txtBytesRead.Text = uiBytesRead.ToString();
                        this.tagDatalistBox.Items.Add(strTagID + "   " +
txtReadData.Text + "...Data read");
                    }

                catch (Exception ex)
                {
                    // Error occurred - clear the data, and set bytes
read to zero.
                    txtReadData.Text = "";
                    txtBytesRead.Text = 0.ToString();

                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    // De-allocate unmanaged memory used to store tag
data
                    if (lpTagData != IntPtr.Zero)
                    {
                        Marshal.FreeHGlobal(lpTagData);
                    }

                    this.Cursor = Cursors.Default;
                }
            }
        }

        /// <summary>
        /// Write Data to the the Tag
        /// </summary>
        private void btnWriteData_Click(object sender, EventArgs e)
        {

            if (hReader != IntPtr.Zero)
            {
                this.Cursor = Cursors.WaitCursor;

                System.IntPtr lpTagData = IntPtr.Zero;

                try
                {
                    // Get the tag ID
                    StringBuilder strTagID = new
System.Text.StringBuilder(txtTagID.Text);

                    // Get the memory type they have selected
                    // Object itemMemType =
this.cmbMemoryType.SelectedItem;

                    // Declare our variables
                    byte[] abTagDataBytes;

                    uint uiBytesWritten = 0;
                    uint uiAddress = 0;
                    uint uiBytes = 0;
                    int iBytesConverted = 0;

                    // Get the address and number of bytes to write
                   // uiAddress = (uint)numStartBlock.Value;
                    uiBytes = (uint)txtWriteData.Text.Length;


                    // Set up our buffer for the bytes + a NULL
terminator
                    abTagDataBytes = new byte[uiBytes + 1];

                    // Get the bytes from the string the user has entered
                    System.Text.ASCIIEncoding asciiEncoder = new
System.Text.ASCIIEncoding();
                    // add text from read data and counter

                    if (linenStatusBool == true && linenTypeBool == true)
                    {
                        // proceed to write data to user memory
                        iBytesConverted =
asciiEncoder.GetBytes(txtWriteData.Text, 0,

(int)uiBytes, abTagDataBytes, 0);

                        // Add a NULL terminator
                        abTagDataBytes[uiBytes] = 0;
                        uiBytes++;
                        iBytesConverted++;

                        // Allocate unmanaged memory used to store data
to write
                        lpTagData =
Marshal.AllocHGlobal((int)iBytesConverted);

                        Marshal.Copy(abTagDataBytes, 0, lpTagData,
(int)iBytesConverted);
                        Object itemMemType = "User";
                       switch (itemMemType.ToString())
                        {
                            case ("User"):
                                {
                                    uiBytesWritten =
usbRFID.WriteTag_UserMemory(hReader, strTagID,

lpTagData, uiAddress, uiBytes);
                                    break;
                                }



                            default:
                                {
                                    break;
                                }
                        }


                        uiBytesWritten -= SUBTRACT_1_BYTE_FROM_WRITE_LEN;

                        // Now display the length of the string we wrote
                        txtBytesWritten.Text = uiBytesWritten.ToString();
                        linenStatusBool = false;
                        linenTypeBool = false;
                        linenWeightBool = false;
                        this.tagDatalistBox.Items.Add(strTagID + "   "
                        + txtWriteData.Text + "...Data written");
                        this.laundryCountNumericUpDown1.Value = 0;
                        this.linenStatuscomboBox.SelectedIndex = -1;
                        this.linenTypecomboBox.SelectedIndex = -1;
                        this.linenWeightTextBox.Clear();
                    }

                    else
                    {
                        MessageBox.Show("Data to be written to tag is not
complete. Go back and select all parts - Linen status, Linen type");

                    }
                }
                // }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    // De-allocate unmanaged memory used to store data to
write
                    if (lpTagData != IntPtr.Zero)
                    {
                        Marshal.FreeHGlobal(lpTagData);
                    }

                    this.Cursor = Cursors.Default;
                }
            }

        }

        private void Example3_Closing(object sender, FormClosingEventArgs
e)
        {
            if (hReader != IntPtr.Zero)
            {
                usbRFID.Close(hReader);
            }
        }

        /// <summary>
        /// Update the contents of the list box
        /// </summary>
        private void RefreshList(uint numDevices)
        {
            this.cmbReaders.Items.Clear();

            try
            {
                string desc;

                this.cmbReaders.BeginUpdate();
                if (numDevices == 0)
                {
                    desc = "No Readers";
                    this.cmbReaders.Items.Insert(0, desc);
                }
                for (uint dev = 0; dev < numDevices; dev++)
                {
                    desc = usbRFID.GetDeviceString(dev);
                    this.cmbReaders.Items.Insert((int)dev, desc);
                }
                this.cmbReaders.EndUpdate();
                this.cmbReaders.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }


        // Our method for updating the Example3 class
        private void ConnectionLostMethod(System.IntPtr cyDevice)
        {
            if (cyDevice == hReader)
            {
                hReader = IntPtr.Zero;
                MessageBox.Show(this, "Connection to Reader has been lost
or Reader has been turned off!", "Notification", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                usbRFID.OnConnectionLost -= OnConnectionLost;

                btnConnect.Enabled = true;
                btnDisConnect.Enabled = false;
                btnReadTag.Enabled = false;
                btnReadData.Enabled = false;
                btnWriteData.Enabled = false;
            }
        }





        private void btnOK_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void linenStatuscomboBox_SelectedIndexChanged(object
sender, EventArgs e)
        {
            linenStatusBool = true;
            buildLinenString();


        }

        private void laundryCountNumericUpDown1_ValueChanged(object
sender, EventArgs e)
        {
            buildLinenString();


        }

        private void linenTypecomboBox_SelectedIndexChanged(object
sender, EventArgs e)
```

```
        {
            linenTypeBool = true;
            buildLinenString();

        }

        /// <summary>
        /// Linen structure
        /// </summary>
        private void buildLinenString()
        {

            // linen count - 3 digits
            int linenCount = 
(this.laundryCountNumericUpDown1.Value).ToString().Length;
            string linenCountString = "";
            switch (linenCount)
            {
                case 1: linenCountString = "00" + 
this.laundryCountNumericUpDown1.Value.ToString();
                    break;
                case 2: linenCountString = "0" + 
this.laundryCountNumericUpDown1.Value.ToString();
                    break;
                case 3: linenCountString = 
this.laundryCountNumericUpDown1.Value.ToString();
                    break;
            }
            if (this.linenStatuscomboBox.Text == "Clean")

                txtWriteData.Text = "T".ToString() + "," + 
linenCountString + "," + this.linenTypecomboBox.Text.ToString();
            else
                if (this.linenStatuscomboBox.Text == "Dirty")
                    txtWriteData.Text = "F".ToString() + "," + 
linenCountString + "," + this.linenTypecomboBox.Text.ToString();
            txtWriteData.Text += "," + this.linenWeightTextBox.Text;


        }

        private void linenWeightTextBox_TextChanged(object sender, 
EventArgs e)
        {
            linenWeightBool = true;
            buildLinenString();
        }
    }

   }
```

```
//Program: Read and write User Memory (SDK from Tracient)
//Program updated by : Sarita Pais
//Date: 24 June 2009

//Updated Version:Read user memory and updates only one attribute
//This is done after wash when linen status is changed from soiled to
clean


using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;   // for Marshal

using Tracient.RFID;

namespace Example3
{
    public partial class Example3 : Form
    {
        // TTL: To account for NULL written, but not reported as part of
String length.
        private const int SUBTRACT_1_BYTE_FROM_WRITE_LEN = 1;

        RfidUsb usbRFID = new RfidUsb();
        IntPtr hReader;
       // flag variable to check if linen status and type are selected by
user
        // count is 0 by default

        public Example3()
        {
            InitializeComponent();

            hReader = IntPtr.Zero;

            uint numDevices = usbRFID.GetNumDevices(false);
            this.RefreshList(numDevices);
           // this.cmbMemoryType.SelectedIndex = 0;   //default to User
Memory

            btnDisConnect.Enabled = false;
            btnReadTag.Enabled = false;
            btnReadData.Enabled = false;
            btnWriteData.Enabled = false;

        }

        private void btnRefresh_Click(object sender, EventArgs e)
        {   // Do a new search for the number of devices
            this.Cursor = Cursors.WaitCursor;

            uint numDevices = usbRFID.GetNumDevices(true);
            this.RefreshList(numDevices);
            this.Cursor = Cursors.Default;
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            if (hReader == IntPtr.Zero)
            {
                int dev = this.cmbReaders.SelectedIndex;
                Object item = this.cmbReaders.SelectedItem;

                if (item.ToString() == "No Readers")
                {
                    MessageBox.Show("No readers available.");
                }
                else
                {
                    this.Cursor = Cursors.WaitCursor;

                    try
                    {   // Open the connection & subscribe to
notifications
                        hReader = usbRFID.Open((uint)dev);
                        // Assign the callback delegates we want to use
                        usbRFID.OnConnectionLost += new
RfidUsb.ConnectionLostEventDelegate(OnConnectionLost);
                        usbRFID.OnAsyncRead += new
RfidUsb.AsyncReadEventDelegate(OnAsyncRead);

                        // Set-up the UI
                        btnConnect.Enabled = false;
                        btnDisConnect.Enabled = true;
                        btnReadTag.Enabled = true;

                        // Only allow read/write of tag data if have a
valid tag ID
                        if (txtTagID.Text.Length > 0)
                        {
                            btnReadData.Enabled = true;
                            btnWriteData.Enabled = true;
                        }
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.Message);
                    }
                    finally
                    {
                        this.Cursor = Cursors.Default;
                    }
                }
            }
            else
            {
                MessageBox.Show(this, "Reader already connected!",
"Example3", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

        }
        private void btnDisConnect_Click(object sender, EventArgs e)
        {
            this.Cursor = Cursors.WaitCursor;

            if (hReader != IntPtr.Zero)
            {
                usbRFID.Close(hReader);
                hReader = IntPtr.Zero;
                usbRFID.OnConnectionLost -= OnConnectionLost;
                usbRFID.OnAsyncRead -= OnAsyncRead;
            }
            btnConnect.Enabled = true;
            btnDisConnect.Enabled = false;
            btnReadTag.Enabled = false;
            btnReadData.Enabled = false;
            btnWriteData.Enabled = false;

            this.Cursor = Cursors.Default;
        }

        /// <summary>
        /// Read the Tag ID
        /// </summary>
        private void btnReadTag_Click(object sender, EventArgs e)
        {
            if (hReader != IntPtr.Zero)
            {
                this.Cursor = Cursors.WaitCursor;

                try
                {
                    string TagID;
                    // Read the Tag ID
                    TagID = usbRFID.Read(hReader);
                    if (TagID.Length > 0)
                    {
                        txtTagID.Text = TagID;
                        btnReadData.Enabled = true;
                        btnWriteData.Enabled = true;
                        this.tagDatalistBox.Items.Add(txtTagID.Text +
"...Tag read");
                    }


                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    this.Cursor = Cursors.Default;
                }
            }
        }

        /// <summary>
        /// Read Data from the tag
        /// </summary>
        private void btnReadData_Click(object sender, EventArgs e)
        {
            if (hReader != IntPtr.Zero)
            {
                this.Cursor = Cursors.WaitCursor;

                System.IntPtr lpTagData = IntPtr.Zero;

                try
                {
                    // Declare our variables
                    byte[] abTagDataBytes;
                    string strTagData;

                    uint uiBytesRead = 0;
                    uint uiAddress = 0;
                    uint uiBytes = 20;  //skytek 26

                    // Get the tag ID
                    StringBuilder strTagID = new
System.Text.StringBuilder(txtTagID.Text);


                        // Allocate unmanaged memory used to store tag
data
                        lpTagData = Marshal.AllocHGlobal((int)uiBytes);
                        Object itemMemType = "User";
                        // call the correct readtag function
                        switch (itemMemType.ToString())
                        {
                            case ("User"):
                                {
                                    uiBytesRead =
usbRFID.ReadTag_UserMemory(hReader, strTagID,

lpTagData, uiAddress, uiBytes);
                                    break;
                                }


                            default:
                                {
                                    break;
                                }
                        }

                        // Now display the bytes we read as text
                        // for this example we are going to assume the
data was ascii
                        abTagDataBytes = new byte[uiBytesRead];
                        Marshal.Copy(lpTagData, abTagDataBytes, 0,
(int)uiBytesRead);

                        System.Text.ASCIIEncoding asciiEncoder = new
System.Text.ASCIIEncoding();
                        strTagData =
asciiEncoder.GetString(abTagDataBytes);
                    //convert strTagData (S)
                        txtReadData.Text = strTagData;
                        txtBytesRead.Text = uiBytesRead.ToString();
                        this.tagDatalistBox.Items.Add(strTagID + "  " +
txtReadData.Text + "...Data read");
                    }
                //}
                catch (Exception ex)
                {
                    // Error occurred - clear the data, and set bytes
read to zero.
                    txtReadData.Text = "";
                    txtBytesRead.Text = 0.ToString();

                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    // De-allocate unmanaged memory used to store tag
data
                    if (lpTagData != IntPtr.Zero)
                    {
                        Marshal.FreeHGlobal(lpTagData);
                    }

                    this.Cursor = Cursors.Default;
                }
            }
        }

        /// <summary>
        /// Write Data to the the Tag
        /// </summary>
        private void btnWriteData_Click(object sender, EventArgs e)
        {

            if (hReader != IntPtr.Zero)
            {
                this.Cursor = Cursors.WaitCursor;

                System.IntPtr lpTagData = IntPtr.Zero;

                try
                {
                    // Get the tag ID
                    StringBuilder strTagID = new
System.Text.StringBuilder(txtTagID.Text);

                    // Get the memory type they have selected
                    // Object itemMemType =
this.cmbMemoryType.SelectedItem;

                    // Declare our variables
                    byte[] abTagDataBytes;

                    uint uiBytesWritten = 0;
                    uint uiAddress = 0;
                    uint uiBytes = 0;
                    int iBytesConverted = 0;

                    // Get the address and number of bytes to write
```

```
                   // uiAddress = (uint)numStartBlock.Value;
                    // update linen data - clean
                    string updateString = txtReadData.Text;
                    updateString = "T" + updateString.Substring(1);
                    txtWriteData.Text = updateString;
                    uiBytes = (uint)txtWriteData.Text.Length;


                    // Set up our buffer for the bytes + a NULL
terminator
                    abTagDataBytes = new byte[uiBytes + 1];

                    // Get the bytes from the string the user has entered
                    System.Text.ASCIIEncoding asciiEncoder = new
System.Text.ASCIIEncoding();
                    // add text from read data and counter


                        // proceed to write data to user memory
                        iBytesConverted =
asciiEncoder.GetBytes(txtWriteData.Text, 0,

(int)uiBytes, abTagDataBytes, 0);

                        // Add a NULL terminator
                        abTagDataBytes[uiBytes] = 0;
                        uiBytes++;
                        iBytesConverted++;

                        // Allocate unmanaged memory used to store data
to write
                        lpTagData =
Marshal.AllocHGlobal((int)iBytesConverted);

                        Marshal.Copy(abTagDataBytes, 0, lpTagData,
(int)iBytesConverted);
                        Object itemMemType = "User";
                        // call the correct write tag function
                        switch (itemMemType.ToString())
                        {
                            case ("User"):
                                {
                                    uiBytesWritten =
usbRFID.WriteTag_UserMemory(hReader, strTagID,

lpTagData, uiAddress, uiBytes);
                                    break;
                                }


                            default:
                                {
                                    break;
                                }
                        }

                        // While we wrote a terminating NULL, we don't
report this, so as not to
```

```
                        // confuse the user who wrote a string of length
4 when by informing them
                        // that 5 bytes were written. Rather we are
reporting the string length written.
                        uiBytesWritten -= SUBTRACT_1_BYTE_FROM_WRITE_LEN;

                        // Now display the length of the string we wrote
                        txtBytesWritten.Text = uiBytesWritten.ToString();

                        this.tagDatalistBox.Items.Add(strTagID + "  "
                        + txtWriteData.Text + "...Data written");




                }
                // }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    // De-allocate unmanaged memory used to store data to
write
                    if (lpTagData != IntPtr.Zero)
                    {
                        Marshal.FreeHGlobal(lpTagData);
                    }

                    this.Cursor = Cursors.Default;
                }
            }

        }

        private void Example3_Closing(object sender, FormClosingEventArgs
e)
        {
            if (hReader != IntPtr.Zero)
            {
                usbRFID.Close(hReader);
            }
        }

        /// <summary>
        /// Update the contents of the list box
        /// </summary>
        private void RefreshList(uint numDevices)
        {
            this.cmbReaders.Items.Clear();

            try
            {
                string desc;

                this.cmbReaders.BeginUpdate();
                if (numDevices == 0)
                {
                    desc = "No Readers";
                    this.cmbReaders.Items.Insert(0, desc);
                }
                for (uint dev = 0; dev < numDevices; dev++)
                {
                    desc = usbRFID.GetDeviceString(dev);
                    this.cmbReaders.Items.Insert((int)dev, desc);
                }
                this.cmbReaders.EndUpdate();
                this.cmbReaders.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            this.Close();       }
       }}
```

```
//Author: Sarita Pais
//Date: 20-05-2009
//Class: Bin
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Example3
{
    class Bin
    {
        public string BinTag
        {
            get;
            set;
        }
        public string  TagType
        {
            get;
            set;

        }

        public string ClientId
        {
            get;
            set;
        }

        public string ClientName
        {
            get;
            set;
        }

        public string ClientAddress
        {
            get;
            set;
        }
        public Bin()
        {
            BinTag = "";
            TagType="";
            ClientId = "";
            ClientName = "";
            ClientAddress = "";
        }
        public Bin(string pbinTag,string ptagTYpe, string pid, string
pname, string paddress)
        {
            BinTag = pbinTag;
            TagType = ptagTYpe;
            ClientId = pid;
            ClientName = pname;
            ClientAddress = paddress;  }   }}
```

```
//Author: Sarita Pais
//Date: 20-05-2009
//Class:Linen
using System;
using System.Collections.Generic;
using System.Text;

namespace Example3
{
    class LinenBin
    {
        public string LinenTag
        { get; set;
        }
        public bool LinenStatus
        {
            get; set;
        }
        public int LinenCount
        {
            get; set;
        }
        public string LinenType
        {
            get; set;
        }

        public double LinenWeight
        {
            get; set;
        }
        public DateTime DataUpdate
        {
            get; set;
        }
        public Bin BinO
        {
            get; set;
        }
        public LinenBin()
        {
            LinenTag = "";
            LinenStatus = false;
            LinenCount = 0;
            LinenType = "";
            LinenWeight = 0.0;
            DataUpdate = DateTime.Now;
            BinO = new Bin();
        }

        public LinenBin (string tag, bool status, int count, string type,
double weight,DateTime dt,Bin bin)
        {
            LinenTag = tag;
            LinenStatus = status;
            LinenCount = count;
            LinenType = type;
            LinenWeight = weight;
            DataUpdate = dt;
            BinO = bin;
    }   }}
```

```
//Program : Read and Write user memory (SDK from Tracient)
//Updated by : Sarita Pais
//Date: 27-09-2009

//Updated Program: Read and write in multiple mode
//read bytes from tag user memory - split comma separated values.
//Build data for linen and bin tags
//update data in user memory of linen tag


using System;
using System.IO;
using System.Xml;
using System.Collections.Generic  ;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;   // for Marshal
using System.Linq;

using Tracient.RFID;
//Buffer to collect all linen tags
namespace Example3
{
    public partial class BufferLog : Form
    {
        // TTL: To account for NULL written, but not reported as part of
String length.
        private const int SUBTRACT_1_BYTE_FROM_WRITE_LEN = 1;
        string writeDataString, owriteDataString;
        string ReadDataString;
        //Linen and Bin data as Q relaton
        //Q = L x B
        List<LinenBin> bufferData = new List<LinenBin>();
        List<LinenBin> allBufferData = new List<LinenBin>();
        LinenBin theLinenBin = new LinenBin();
        Bin theBin ;
        string TagIDString;
        RfidUsb usbRFID = new RfidUsb();
        IntPtr hReader;


        int rfidIdCountSuccess = 0, rfidIdCountFailure = 0,
rfidUMReadCountSuccess = 0,
        rfidUMReadCountFailure = 0, rfidUMWriteCountSuccess = 0,
        rfidUMWriteCountFailure = 0;

        public BufferLog()
        {
            InitializeComponent();

            hReader = IntPtr.Zero;

            uint numDevices = usbRFID.GetNumDevices(false);
            this.RefreshList(numDevices);
```

```
                btnDisConnect.Enabled = false;


            }

            private void btnRefresh_Click(object sender, EventArgs e)
            {   // Do a new search for the number of devices
                this.Cursor = Cursors.WaitCursor;

                uint numDevices = usbRFID.GetNumDevices(true);
                this.RefreshList(numDevices);

                this.Cursor = Cursors.Default;
            }

            private void btnConnect_Click(object sender, EventArgs e)
            {
                if (hReader == IntPtr.Zero)
                {
                    int dev = this.cmbReaders.SelectedIndex;
                    Object item = this.cmbReaders.SelectedItem;

                    if (item.ToString() == "No Readers")
                    {
                        MessageBox.Show("No readers available.");
                    }
                    else
                    {
                        this.Cursor = Cursors.WaitCursor;

                        try
                        {   // Open the connection & subscribe to
notifications
                            hReader = usbRFID.Open((uint)dev);
                            // Assign the callback delegates we want to use
                            usbRFID.OnConnectionLost += new
RfidUsb.ConnectionLostEventDelegate(OnConnectionLost);
                            usbRFID.OnAsyncRead += new
RfidUsb.AsyncReadEventDelegate(OnAsyncRead);

                            // Set-up the UI
                            btnConnect.Enabled = false;
                            btnDisConnect.Enabled = true;
                            //read bin tag and store in a bin object
                            readTag();

                            if (theBin != null)
                            {//bin recognised and laundry bin is ready to
accept linen
                                MessageBox.Show("Bin is set up and ready to
accept linens", "Bin Set up", MessageBoxButtons.OK,
MessageBoxIcon.Information);

                                this.timer1.Enabled = true;

                            }
                        }
                        catch (Exception ex)
                        {
```

```
                        MessageBox.Show(ex.Message);
                    }
                    finally
                    {
                        this.Cursor = Cursors.Default;
                    }
                }
            }
            else
            {
                MessageBox.Show(this, "Reader already connected!",
"Reader ready", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

        }

        private void btnDisConnect_Click(object sender, EventArgs e)
        {
            timer1.Enabled = false;
            this.Cursor = Cursors.WaitCursor;

            if (hReader != IntPtr.Zero)
            {
                usbRFID.Close(hReader);
                hReader = IntPtr.Zero;
                usbRFID.OnConnectionLost -= OnConnectionLost;
                usbRFID.OnAsyncRead -= OnAsyncRead;
            }
            btnConnect.Enabled = true;
            btnDisConnect.Enabled = false;


            this.Cursor = Cursors.Default;
        }

        /// <summary>
        /// Read the Tag ID
        /// </summary>
        ///
        public void readTag()
        {
            if (hReader != IntPtr.Zero)
            {
                this.Cursor = Cursors.WaitCursor;

                try
                {
                    string TagID;
                    // Read the Tag ID
                    TagID = usbRFID.Read(hReader);
                    if (TagID.Length > 0)
                    {
                        TagIDString = TagID;
                        rfidIdCountSuccess++;
                        readData();
                    }

                }
                catch (Exception ex)
                {
                    rfidIdCountFailure++;
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    this.Cursor = Cursors.Default;
                }
            }
        }

        /// <summary>
        /// Read Data from the tag
        /// </summary>
        ///
        public void readData()
        {
            if (hReader != IntPtr.Zero)
            {
                this.Cursor = Cursors.WaitCursor;

                System.IntPtr lpTagData = IntPtr.Zero;

                try
                {
                    // Declare our variables
                    byte[] abTagDataBytes;
                    string strTagData;

                    uint uiBytesRead = 0;
                    uint uiAddress = 0;
                    uint uiBytes = 40;

                    // Get the tag ID
                    StringBuilder strTagID = new
System.Text.StringBuilder(TagIDString);

                    // Allocate unmanaged memory used to store tag data
                        lpTagData = Marshal.AllocHGlobal((int)uiBytes);
                        Object itemMemType = "User";
                        // call the correct readtag function
                        switch (itemMemType.ToString())
                        {
                            case ("User"):
                                {
                                    uiBytesRead =
usbRFID.ReadTag_UserMemory(hReader, strTagID,

lpTagData, uiAddress, uiBytes);
                                    break;
                                }


                            default:
                                {
                                    break;
                                }
                        }
                        // Now display the bytes we read as text
                        // for this example we are going to assume the
data was ascii
                        abTagDataBytes = new byte[uiBytesRead];
                        Marshal.Copy(lpTagData, abTagDataBytes, 0,
(int)uiBytesRead);

                        System.Text.ASCIIEncoding asciiEncoder = new
System.Text.ASCIIEncoding();
                        strTagData =
asciiEncoder.GetString(abTagDataBytes);

                        ReadDataString = strTagData;


                    //check which tag - linen or bin?
                        if (ReadDataString.Substring(0, 1) == "B" ||
ReadDataString.Substring(0, 1) == "b")
                        {
                            //bin
                           writeDataBin(strTagID,ReadDataString);

                        }
                        else
                        {
                            //linen
                            this.listBox1.Items.Add(TagIDString + "," +
ReadDataString + "," + DateTime.Now.ToShortTimeString() + "Reading tag id
& User memory" + "," + DateTime.Now.ToShortDateString() + "," +
DateTime.Now.ToLongTimeString());
                            writeDataAllLinenBuffer(TagIDString,
ReadDataString, DateTime.Now);
                            bool toWrite = false;
                            toWrite = readString();
                            rfidUMReadCountSuccess++;
                            if (toWrite == true)
                            // call write tag data
                            {
                                writeData();


                            }

                        }
                    }

                catch (Exception ex)
                {
                    // Error occurred - clear the data, and set bytes
read to zero.
                    rfidUMReadCountFailure++;
                    ReadDataString = "";
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    // De-allocate unmanaged memory used to store tag
data
                    if (lpTagData != IntPtr.Zero)
                    {
                        Marshal.FreeHGlobal(lpTagData);
                    }

                    this.Cursor = Cursors.Default;
                }
            }
        }

         /// <summary>
        /// Write Data to the the Tag
        /// </summary>
       public void writeData()
       {
            if (hReader != IntPtr.Zero)
            {
                this.Cursor = Cursors.WaitCursor;

                System.IntPtr lpTagData = IntPtr.Zero;

                try
                {
                    // Get the tag ID
                    StringBuilder strTagID = new
System.Text.StringBuilder(TagIDString);

                    // Get the memory type they have selected
                   // Object itemMemType =
this.cmbMemoryType.SelectedItem;
                    Object itemMemType = "User";
                    // Declare our variables
                    byte[] abTagDataBytes;

                    uint uiBytesWritten = 0;
                    uint uiAddress = 0;
                    uint uiBytes = 0;
                    int iBytesConverted = 0;

                    // Get the address and number of bytes to write
                    //uiAddress = (uint)numStartBlock.Value;

                    uiBytes = (uint)owriteDataString.Length;

                    // Sanity checks
                    if (itemMemType.ToString() == "Read Only")
                    {
                        MessageBox.Show("Cannot write to Read Only
memory.");
                    }

                    else
                    {
                        // Set up our buffer for the bytes + a NULL
terminator
                        abTagDataBytes = new byte[uiBytes + 1];

                        // Get the bytes from the string the user has
entered
```

```
                        System.Text.ASCIIEncoding asciiEncoder = new
System.Text.ASCIIEncoding();

                        iBytesConverted =
asciiEncoder.GetBytes(owriteDataString, 0,

(int)uiBytes, abTagDataBytes, 0);

                        // Add a NULL terminator
                        abTagDataBytes[uiBytes] = 0;
                        uiBytes++;
                        iBytesConverted++;

                        // Allocate unmanaged memory used to store data
to write
                        lpTagData =
Marshal.AllocHGlobal((int)iBytesConverted);

                        Marshal.Copy(abTagDataBytes, 0, lpTagData,
(int)iBytesConverted);

                        // call the correct write tag function
                        switch (itemMemType.ToString())
                        {
                            case ("User"):
                                {
                                    uiBytesWritten =
usbRFID.WriteTag_UserMemory(hReader, strTagID,

lpTagData, uiAddress, uiBytes);
                                    break;
                                }



                            default:
                                {
                                    break;
                                }
                        }

                        // While we wrote a terminating NULL, we don't
report this, so as not to
                        // confuse the user who wrote a string of length
4 when by informing them
                        // that 5 bytes were written. Rather we are
reporting the string length written.
                        uiBytesWritten -= SUBTRACT_1_BYTE_FROM_WRITE_LEN;

                        // Now display the length of the string we wrote
                        //txtBytesWritten.Text =
uiBytesWritten.ToString();
                        this.listBox1.Items.Add(TagIDString + "," +
owriteDataString + "," + "Write user memory" +  "," +
DateTime.Now.ToShortDateString ()+ "," +
DateTime.Now.ToShortTimeString());
                        writeDataAllLinenBuffer(TagIDString,
owriteDataString, DateTime.Now);
                        rfidUMWriteCountSuccess++;
```

```
                    }
                }
                catch (Exception ex)
                {
                    rfidUMWriteCountFailure++;
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    // De-allocate unmanaged memory used to store data to
write
                    if (lpTagData != IntPtr.Zero)
                    {
                        Marshal.FreeHGlobal(lpTagData);
                    }

                    this.Cursor = Cursors.Default;
                }
            }

        }

 public void writeDataLinenBuffer()
 {
        string[] attr = new string[] { "", "", "", "", "" };

 // separate into different attributes from a string with comma delimited
                        int lastPosition = 0;
                        int j = 0, k = 1;
                        for (int i = 0; i < owriteDataString.Length; i++)
                        {
                            if (owriteDataString.Substring(i, 1) == ",")
                            {
                                attr[j++] =
owriteDataString.Substring(lastPosition, k - 1);
                                k = 1;
                                lastPosition = i + 1;
                            }
                            else
                            {
                                k++;
                            }
                        }
                        attr[j++] =
owriteDataString.Substring(lastPosition, k - 1);
                        if (attr[0] == "T" || attr[0] == "t")
                            attr[0] = "true";
                        else if (attr[0] == "F" || attr[0] == "f")
                            attr[0] = "false";


                        theLinenBin = new LinenBin(TagIDString,
bool.Parse(attr[0]), int.Parse(attr[1]), attr[2], double.Parse(attr[3]),
DateTime.Now,theBin );
                        bufferData.Add(theLinenBin);


 }
```

```
      private void Example3_Closing(object sender, FormClosingEventArgs e)
        {
            if (hReader != IntPtr.Zero)
            {
                usbRFID.Close(hReader);
            }
        }



        private void btnOK_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
             readTag();

        }


        private bool readString()
        {
            writeDataString = ReadDataString;
            int laundryCount;
            string linenStatusString = writeDataString.Substring(0, 1);

            if (linenStatusString == "T")
            {

               // change linen from clean to dirty
                linenStatusString = "F";
                 // laundry count
                laundryCount = int.Parse(writeDataString.Substring(2, 3))
+ 1;

                if ( laundryCount.ToString ().Length == 1)

                owriteDataString = "F,00" + laundryCount.ToString() +
writeDataString.Substring(5);
                else if (laundryCount.ToString().Length == 2)
                    owriteDataString = "F,0" + laundryCount.ToString() +
writeDataString.Substring(5);
                else
                    if (laundryCount.ToString().Length == 3)

                        owriteDataString = "F," + laundryCount.ToString()
+ writeDataString.Substring(5);

                return true;
            }

            return false;
        }
        private void csvAllLinenButton_Click(object sender, EventArgs e)
        {
            this.csvAllLinenButton .Enabled = false;
            // write data from list array to  file
            string date = DateTime.Now.Day.ToString() + "-" +
DateTime.Now.Month.ToString() + "-" + DateTime.Now.Year.ToString();

            StreamWriter sw = File.CreateText("AllLinenBinL"+  date +
".txt");


            try
            {
          foreach ( string items in listBox1.Items   )
          {//write all linen to file
              sw.WriteLine(items );
              //write each linen object into allBufferLinen arraylist



          }

         sw.Close();



              }

            catch
            {
                MessageBox.Show("Error writing to file");
}
            StreamWriter sw1 = File.CreateText("AllLinenBinO" + date +
".txt");

            try
            {
                foreach (var lb in allBufferData )
                {//write all linen to file
                    sw1.WriteLine(lb.LinenTag + "," + lb.LinenStatus +","
+ lb.LinenCount + "," + lb.LinenType +"," + lb.LinenWeight + "," +
lb.DataUpdate  + ","+ lb.BinO .BinTag +"," + lb.BinO .ClientId + "," +
lb.BinO .ClientName +","+ lb.BinO .ClientAddress );

                }

               sw1.Close();


            }

            catch
            {
                MessageBox.Show("Error writing to file");
            }


            MessageBox.Show("RFID ID Read Count Success" +
rfidIdCountSuccess +
                " RFID ID Read Count Failure" + rfidIdCountFailure +
                " RFID User Memory Read Count Success" +
rfidUMReadCountSuccess +
                " RFID User Memory Read Count Failure" +
rfidUMWriteCountFailure +
                " RFID User Memory Write Count Success" +
rfidUMWriteCountSuccess +
                " RFID User Memory Write Count Failure" +
rfidUMWriteCountFailure);


        }




        public void writeDataAllLinenBuffer(string tagId, string items,
DateTime dt)
        {
            string[] attr = new string[] { "", "", "", "", "" };

            // separate into different attributes from a string with
comma delimited
            int lastPosition = 0;
            int j = 0, k = 1;
            for (int i = 0; i < items.Length; i++)
            {
                if (items.Substring(i, 1) == ",")
                {
                    attr[j++] = items.Substring(lastPosition, k - 1);
                    k = 1;
                    lastPosition = i + 1;
                }
                else
                {
                    k++;
                }
            }
            attr[j++] = items.Substring(lastPosition, k - 1);
            if (attr[0] == "T" || attr[0] == "t")
                attr[0] = "true";
            else if (attr[0] == "F" || attr[0] == "f")
                attr[0] = "false";


            theLinenBin = new LinenBin(tagId, bool.Parse(attr[0]),
int.Parse(attr[1]), attr[2], double.Parse(attr[3]), dt,theBin );

            allBufferData.Add (theLinenBin);

        }

        public void writeDataBin(StringBuilder tagId, string items)

        {
            string[] attr = new string[] { "", "", "", ""};

            // separate into different attributes from a string with
comma delimited
            int lastPosition = 0;
            int j = 0, k = 1;
            for (int i = 0; i < items.Length; i++)
            {
                if (items.Substring(i, 1) == ",")
                {
                    attr[j++] = items.Substring(lastPosition, k - 1);
                    k = 1;
                    lastPosition = i + 1;
                }
                else
                {
                    k++;
                }
            }
            attr[j++] = items.Substring(lastPosition, k - 1);
            theBin = new Bin(tagId.ToString (), attr[0], attr[1],
attr[2], attr[3]);

        }


    }

   }
```

```
// author: Sarita Pais
// Date :  June 2009
// Updated Date: Sep 2009
// Clean and process collected data from RFID tags
// Print count of each linen type and weight of all linen
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Text.RegularExpressions;

namespace PCCleaningSorting
{
    public partial class Form1 : Form
    {
        //to store all data from stream file into arraylist
        List<LinenBin> allBufferData = new List<LinenBin>();
        LinenBin theLinenBin = new LinenBin();
        Bin theBin ;
        // to store unique data from stream file into arraylist
        List<LinenBin> bufferData = new List<LinenBin>();
        public Form1()
        {
            InitializeComponent();
        }

        private void printButton_Click(object sender, EventArgs e)
        {
            // read data from file
            DateTime printDate = this.dateTimePicker1.Value ;
            string date = printDate.Day.ToString() + "-" +
printDate.Month.ToString() + "-" + printDate.Year.ToString();


            //clean allBufferLinen data and write to BufferData
            try
            {
                // read from file and put in allBufferData

                StreamReader sr = new StreamReader("AllLinenBinO" + date
+ ".txt");
                string strLine;
                string[] strArray;
                char[] charArray = new char[] { ',' };
                strLine = sr.ReadLine();


                while (strLine != null)
                {
                    //split row of string array into allBufferData
                    strArray = Regex.Split(strLine, ",");
                    theBin = new Bin(strArray[6], "B", strArray[7],
strArray[8], strArray[9]);
```

```
                    theLinenBin = new LinenBin(strArray[0],
bool.Parse(strArray[1]), int.Parse(strArray[2]), strArray[3],
double.Parse(strArray[4]), DateTime.Parse(strArray[5]), theBin);
                    allBufferData.Add(theLinenBin);
                    strLine = sr.ReadLine();
                }

                sr.Close();
                printButton.Enabled = false;
            }
            catch (IOException ex)
            {
                MessageBox.Show("An IO exception has been thrown" +
ex.ToString());
            }


            //sort allBbufferData
            var sortAllBufferData = from bd in allBufferData
                                    orderby bd.LinenTag
                                    select bd;


            IEnumerator<LinenBin> it = sortAllBufferData.GetEnumerator();


            string linenTag = "";
            LinenBin aLinenBin = new LinenBin();
            while (it.MoveNext())
            {
                theLinenBin = (LinenBin)it.Current;
                this.listBox1.Items.Add(theLinenBin.LinenTag +
theLinenBin.LinenType);

                if (linenTag == "")
                {
                    if (theLinenBin.LinenStatus == false)
                    {
                        linenTag = theLinenBin.LinenTag;

                        aLinenBin = theLinenBin;
                    }

                }
                else
                {
                    if (theLinenBin.LinenTag != linenTag)
                    {//last record of the linen tag
                        //write to buffer data
                        //check if status is F
                        if (theLinenBin.LinenStatus == false)
                        {
                            bufferData.Add(aLinenBin);
                            this.listBox2.Items.Add(aLinenBin.LinenTag +
aLinenBin.LinenType);// tested ok
                            //new tag
                            linenTag = theLinenBin.LinenTag;
                            aLinenBin = theLinenBin;
                        }
                    }

                }

            }

            bufferData.Add(aLinenBin);
            this.listBox2.Items.Add(aLinenBin.LinenTag +
aLinenBin.LinenType);
            this.listBox2.Items.Add("end");

            // store in arraylist

            //sort arraylist and print on listbox
            //print to a file

            var sortBufferData = from bd in bufferData
                                 orderby bd.LinenType
                                 select bd;


            IEnumerator<LinenBin> it1 = sortBufferData.GetEnumerator();


            StreamWriter sw = File.CreateText("PrintLinenBin" + date +
".txt");


            int count = 0;
            double totalWeight = 0;
            string heading = "";
            MessageBox.Show("Pause");



            while (it1.MoveNext())
            {
                theLinenBin = (LinenBin)it1.Current;
                this.listBox2.Items.Add(theLinenBin.LinenTag +
theLinenBin.LinenType);

                totalWeight += theLinenBin.LinenWeight;

                if (heading == "")
                {
                    heading = theLinenBin.LinenType;
                    count = 1;
                    sw.WriteLine("Client: " + theLinenBin.BinO.ClientId +
" " + theLinenBin.BinO.ClientName + " " +
theLinenBin.BinO.ClientAddress);
                }
                else
                {
                    if (theLinenBin.LinenType != heading)
                    {
                        MessageBox.Show(heading + count.ToString());
                        this.printListBox.Items.Add(heading + "    " +
count.ToString());
                        sw.WriteLine(heading + "    " + count.ToString());
                        heading = theLinenBin.LinenType;
                        count = 1;
                    }
                    else
                        count++;

                }

            }
            // process file to print receipt
            this.printListBox.Items.Add(heading + count.ToString());
            sw.WriteLine(heading + "    " + count.ToString());
            this.printListBox.Items.Add("Total Weight    " +
String.Format("{0:0.##}", totalWeight));
            sw.WriteLine("Total Weight    " + String.Format("{0:0.##}",
totalWeight));
            sw.Close();
        }
    }
}
```

Appendix E - Trial Sheet

Experiment Trial			
Tag type			
Number of tags		Time	
Date			

Through observation on screen – Program 3	
Read Tag id failure	
Read UM Failure	
Write UM failure	
Input string error	

	Through Program 3 count
Read Tag ID count success	
Read Tag ID count failure	
Read UM count success	
Read UM count failure	
Write UM count success	
Write UM count failure	

Tag ID	Linen Type	Status on Tag changed - checked though Program 2	Log Entries generated through Program 3	Print command through Program 4

Appendix F - Trial Sheets

Experiment 1aTrial sheet 1

Experiment	1		
Experiment Trial	1		
Tag type	Skyetek UHF ISO 18000-6B		
Number of tags	7		
Date	11/07/2009	**Time**	9.15 am - 9.25 am

Through observation on screen Program 3	
Read Tag id failure	
Read UM Failure	1
Write UM failure	
Input string error	6

	Through program 3 - count
Read Tag Id count success	42
Read Tag Id count failure	0
Read UM count success	31
Read UM count failure	1
Write UM count success	5
Write UM count failure	1

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
E004000074251502	FlatSheet	N	R	
E0040000E90A4302	FlatSheet	Y	RW	Y
E004000031DD4202	Blanket	Y	RW	Y
E0040000E4F34202	PatientGown	Y	RW	Y
E0040000B34B1502	PillowCase	Y	RW	Y
E0040000E7204302	PillowCase	N	NONE	
E004000043FD4202	Towel	Y	RW	Y

Experiment 1aTrial sheet 2

Experiment	**1**		
Experiment Trial	2		
Tag type	Skyetek UHF ISO 18000-6B		
Number of tags	7	**Time**	9.48 am – 9.58 am
Date	07-11-2009		

Through observation on screen –m Program 3	
Read Tag id failure	
Read UM Failure	3
Write UM failure	2
Input string error	

	Through program 3 count
Read Tag Id count success	58
Read Tag Id count failure	0
Read UM count success	53
Read UM count failure	2
Write UM count success	3
Write UM count failure	2

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
E004000074251502	FlatSheet	N	NONE	N
E0040000E90A4302	FlatSheet	Y	RW	Y
E004000031DD4202	Blanket	Y	RW	Y
E0040000E4F34202	PatientGown	Y	RW	Y
E0040000B34B1502	PillowCase	Y	RW	Y
E0040000E7204302	PillowCase	N	NONE	N
E004000043FD4202	Towel	Y	RW	Y

Experiment 1aTrial sheet 3

Experiment	1		
Experiment Trial	3		
Tag type	Skyetek UHF ISO 18000-6B		
Number of tags	**7**		
Date	11/07/2009	**Time**	10.51 am - 11.01 am

Through observation on screen Program 3	
Read Tag id failure	
Read UM Failure	5
Write UM failure	1
Input string error	

	Through program 3 - count
Read Tag Id count success	40
Read Tag Id count failure	0
Read UM count success	32
Read UM count failure	1
Write UM count success	5
Write UM count failure	1

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
E004000074251502	FlatSheet	N	NONE	
E0040000E90A4302	FlatSheet	Y	RW	Y
E004000031DD4202	Blanket	Y	RW	Y
E0040000E4F34202	PatientGown	Y	RW	Y
E0040000B34B1502	PillowCase	Y	R	
E0040000E7204302	PillowCase	Y	RW	Y
E004000043FD4202	Towel	Y	RW	Y

Experiment 1aTrial sheet 4

Experiment	1		
Experiment Trial	4		
Tag type	Skyetek UHF ISO 18000-6B		
Number of tags	7		
Date	11/07/2009	**Time**	11.26 am - 11.36 am

Through observation on screen Program 3	
Read Tag id failure	
Read UM Failure	5
Write UM failure	
Input string error	2

	Through program 3 - count
Read Tag Id count success	55
Read Tag Id count failure	0
Read UM count success	41
Read UM count failure	1
Write UM count success	4
Write UM count failure	1

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
E004000074251502	FlatSheet	N	R	
E0040000E90A4302	FlatSheet	Y	RW	Y
E004000031DD4202	Blanket	N	NONE	
E0040000E4F34202	PatientGown	Y	RW	Y
E0040000B34B1502	PillowCase	Y	RW	Y
E0040000E7204302	PillowCase	Y	RW	Y
E004000043FD4202	Towel	Y	NONE	

Experiment 1aTrial sheet 5

Experiment	1		
Experiment Trial	5		
Tag type	Skyetek UHF ISO 18000-6B		
Number of tags	**7**		
Date	11/07/2009	**Time**	12.09 pm - 12.19 pm

Through observation on screen Program 3	
Read Tag id failure	
Read UM Failure	10
Write UM failure	1
Input string error	4

	Through program 3 - count
Read Tag Id count success	58
Read Tag Id count failure	0
Read UM count success	34
Read UM count failure	1
Write UM count success	4
Write UM count failure	1

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
E004000074251502	FlatSheet	N	R	
E0040000E90A4302	FlatSheet	Y	RW	Y
E004000031DD4202	Blanket	Y	RW	Y
E0040000E4F34202	PatientGown	Y	RW	Y
E0040000B34B1502	PillowCase	Y	RW	Y
E0040000E7204302	PillowCase	Y	RW	Y
E004000043FD4202	Towel	N	NONE	

Experiment 2aTrial sheet 1

Experiment	2 (with linen)		
Experiment Trial	1		
Tag type	Skyetek UHF ISO 18000-6B		
Number of tags	7		
Date	11/02/2009	**Time**	2.31 pm - 2.41 pm

Through observation on screen Program 3	
Read Tag id failure	
Read UM Failure	6
Write UM failure	2
Input string error	4

	Through program 3 - count
Read Tag Id count success	58
Read Tag Id count failure	0
Read UM count success	40
Read UM count failure	1
Write UM count success	3
Write UM count failure	1

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
E004000074251502	FlatSheet	N	R	
E0040000E90A4302	FlatSheet	Y	RW	Y
E004000031DD4202	Blanket	Y	RW	Y
E0040000E4F34202	PatientGown	N	NONE	
E0040000B34B1502	PillowCase	Y	RW	Y
E0040000E7204302	PillowCase	N	NONE	
E004000043FD4202	Towel	N	NONE	

Experiment 2aTrial sheet 2

Experiment	2 (with linen)		
Experiment Trial	2		
Tag type	Skyetek UHF ISO 18000-6B		
Number of tags	7		
Date	11/02/2009	**Time**	3.27 pm - 3.37 pm

Through observation on screen Program 3	
Read Tag id failure	
Read UM Failure	2
Write UM failure	2
Input string error	8

	Through program 3 - count
Read Tag Id count success	59
Read Tag Id count failure	0
Read UM count success	29
Read UM count failure	2
Write UM count success	6
Write UM count failure	2

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
E004000074251502	FlatSheet	N	R	
E0040000E90A4302	FlatSheet	Y	RW	Y
E004000031DD4202	Blanket	Y	RW	Y
E0040000E4F34202	PatientGown	Y	RW	Y
E0040000B34B1502	PillowCase	Y	RW	Y
E0040000E7204302	PillowCase	Y	RW	Y
E004000043FD4202	Towel	Y	RW	Y

Experiment 2aTrial sheet 3

Experiment	2 (with linen)		
Experiment Trial	3		
Tag type	Skyetek UHF ISO 18000-6B		
Number of tags	7		
Date	11/03/2009	**Time**	2.15 pm - 2.25 pm

Through observation on screen Program 3	
Read Tag id failure	
Read UM Failure	3
Write UM failure	3
Input string error	2

	Through program 3 - count
Read Tag Id count success	57
Read Tag Id count failure	0
Read UM count success	38
Read UM count failure	3
Write UM count success	4
Write UM count failure	3

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
E004000074251502	FlatSheet	N	R	
E0040000E90A4302	FlatSheet	Y	RW	Y
E004000031DD4202	Blanket	Y	RW	Y
E0040000E4F34202	PatientGown	N	R	
E0040000B34B1502	PillowCase	Y	RW	Y
E0040000E7204302	PillowCase	Y	RW	Y
E004000043FD4202	Towel	N	R	

Experiment 2aTrial sheet 4

Experiment	2 (with linen)		
Experiment Trial	4		
Tag type	Skyetek UHF ISO 18000-6B		
Number of tags	7		
Date	11/03/2009	**Time**	3.12 pm - 3.22 pm

Through observation on screen Program 3	
Read Tag id failure	1
Read UM Failure	7
Write UM failure	3
Input string error	19

	Through program 3 - count
Read Tag Id count success	57
Read Tag Id count failure	1
Read UM count success	17
Read UM count failure	2
Write UM count success	4
Write UM count failure	2

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
E004000074251502	FlatSheet	N	R	
E0040000E90A4302	FlatSheet	N	NONE	
E004000031DD4202	Blanket	Y	RW	Y
E0040000E4F34202	PatientGown	N	R	
E0040000B34B1502	PillowCase	Y	RW	Y
E0040000E7204302	PillowCase	Y	RW	Y
E004000043FD4202	Towel	Y	RW	Y

Experiment 2aTrial sheet 5

Experiment	2 (with linen)		
Trial	5		
Tag type	Skyetek UHF ISO 18000-6B		
Number of tags	7		
Date	11/04/2009	**Time**	3.18 pm - 3.28 pm

Through observation on screen Program 3	
Read Tag id failure	
Read UM Failure	8
Write UM failure	2
Input string error	3

Through observation on screen Program 3	
Read Tag Id count success	50
Read Tag Id count failure	0
Read UM count success	27
Read UM count failure	2
Write UM count success	3
Write UM count failure	2

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
E004000074251502	FlatSheet	N	R	
E0040000E90A4302	FlatSheet	Y	RW	Y
E004000031DD4202	Blanket	N	NONE	
E0040000E4F34202	PatientGown	Y	RW	Y
E0040000B34B1502	PillowCase	Y	RW	Y
E0040000E7204302	PillowCase	N	NONE	
E004000043FD4202	Towel	N	NONE	

Experiment 1bTrial sheet 1

Experiment	Experiment 3 without linen		
Experiment Trial	1		
Tag type	Intermec IT67 UHF ISO 18000-6C		
Number of tags	10		
Date	28-11-2009	**Time**	9.51 pm - 10.01 pm

Through observation on screen	
Read Tag id failure	
Read UM Failure	21
Write UM failure	3
Input string error	3

	Through program 3 - count
Read Tag Id count success	66
Read Tag Id count failure	0
Read UM count success	33
Read UM count failure	3
Write UM count success	6
Write UM count failure	3

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
3005FB63AC1F3681EC880401	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880402	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880403	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880404	Blanket	Y	RW	Y
3005FB63AC1F3681EC880405	Towel	N	R	
3005FB63AC1F3681EC880406	Flatsheet	N	R	
3005FB63AC1F3681EC880407	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880408	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880409	Towel	Y	RW	Y
3005FB63AC1F3681EC880410	Blanket	N	R	

Experiment 1bTrial sheet 2

Experiment	Experiment 3 without linen		
Experiment Trial	2		
Tag type	Intermec IT67 UHF ISO 18000-6C		
Number of tags	10		
Date	28-11-2009	**Time**	10.59 pm - 11.09 pm

Through observation on screen				
Read Tag id failure				
Read UM Failure	30			
Write UM failure	3			
Input string error	7			
	Through program 3 - count			
Read Tag Id count success	62			
Read Tag Id count failure	0			
Read UM count success	19			
Read UM count failure	4			
Write UM count success	7			
Write UM count failure	3			
Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
3005FB63AC1F3681EC880401	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880402	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880403	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880404	Blanket	N	R	
3005FB63AC1F3681EC880405	Towel	N	R	
3005FB63AC1F3681EC880406	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880407	Pillowcase	N	NONE	
3005FB63AC1F3681EC880408	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880409	Towel	Y	RW	Y
3005FB63AC1F3681EC880410	Blanket	Y	RW	Y

Experiment 1bTrial sheet 3

Experiment	Experiment 3 without linen		
Experiment Trial	3		
Tag type	Intermec IT67 UHF ISO 18000-6C		
Number of tags	10		
Date	28-11-2009	**Time**	11.51 pm - 12.01 am

Through observation on screen	
Read Tag id failure	
Read UM Failure	13
Write UM failure	3
Input string error	10

	Through program 3 - count			
Read Tag Id count success	59			
Read Tag Id count failure	0			
Read UM count success	35			
Read UM count failure	3			
Write UM count success	9			
Write UM count failure	3			
Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
3005FB63AC1F3681EC880401	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880402	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880403	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880404	Blanket	Y	RW	Y
3005FB63AC1F3681EC880405	Towel	N	R	
3005FB63AC1F3681EC880406	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880407	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880408	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880409	Towel	Y	RW	Y
3005FB63AC1F3681EC880410	Blanket	Y	RW	Y

Experiment 1bTrial sheet 4

Experiment	Experiment 3 without linen		
Experiment Trial	4		
Tag type	Intermec IT67 UHF ISO 18000-6C		
Number of tags	10		
Date	29-11-2009	**Time**	8.44 am - 8.54 am

Through observation on screen	
Read Tag id failure	
Read UM Failure	27
Write UM failure	
Input string error	7

	Through program 3 - count
Read Tag Id count success	59
Read Tag Id count failure	0
Read UM count success	27
Read UM count failure	2
Write UM count success	8
Write UM count failure	

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
3005FB63AC1F3681EC880401	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880402	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880403	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880404	Blanket	N	NONE	
3005FB63AC1F3681EC880405	Towel	N	R	
3005FB63AC1F3681EC880406	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880407	Pillowcase	N	RW	Y
3005FB63AC1F3681EC880408	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880409	Towel	Y	RW	Y
3005FB63AC1F3681EC880410	Blanket	Y	RW	Y

Experiment 1bTrial sheet 5

Experiment	Experiment 3 without linen		
Experiment Trial	1		
Tag type	Intermec IT67 UHF ISO 18000-6C		
Number of tags	10		
Date	29-11-2009	**Time**	9.55 pm - 10.03 pm

Through observation on screen	
Read Tag id failure	
Read UM Failure	27
Write UM failure	4
Input string error	9

	Through program 3 - count
Read Tag Id count success	62
Read Tag Id count failure	0
Read UM count success	24
Read UM count failure	6
Write UM count success	7
Write UM count failure	

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
3005FB63AC1F3681EC880401	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880402	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880403	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880404	Blanket	Y	RW	Y
3005FB63AC1F3681EC880405	Towel	N	R	
3005FB63AC1F3681EC880406	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880407	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880408	Patientgown	N	R	
3005FB63AC1F3681EC880409	Towel	N	NONE	
3005FB63AC1F3681EC880410	Blanket	Y	RW	Y

Experiment 2bTrial sheet 1

Experiment	Experiment 4 with linen		
Experiment Trial	1		
Tag type	Intermec IT67 UHF ISO 18000-6C		
Number of tags	10		
Date	12/01/2009	**Time**	12.09 pm - 12.19 pm

Through observation on screen	
Read Tag id failure	
Read UM Failure	23
Write UM failure	5
Input string error	6

	Through program 3 - count
Read Tag Id count success	65
Read Tag Id count failure	0
Read UM count success	28
Read UM count failure	5
Write UM count success	7
Write UM count failure	

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
3005FB63AC1F3681EC880401	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880402	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880403	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880404	Blanket	Y	RW	Y
3005FB63AC1F3681EC880405	Towel	N	R	
3005FB63AC1F3681EC880406	Flatsheet	N	NONE	Y
3005FB63AC1F3681EC880407	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880408	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880409	Towel	Y	RW	
3005FB63AC1F3681EC880410	Blanket	N	R	

Experiment 2bTrial sheet 2

Experiment	Experiment 4 with linen		
Experiment Trial	2		
Tag type	Intermec IT67 UHF ISO 18000-6C		
Number of tags	10		
Date	12/01/2009	**Time**	1.40 pm - 1.50 pm

Through observation on screen	
Read Tag id failure	
Read UM Failure	18
Write UM failure	5
Input string error	8

	Through program 3 - count
Read Tag Id count success	57
Read Tag Id count failure	0
Read UM count success	21
Read UM count failure	5
Write UM count success	7
Write UM count failure	

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
3005FB63AC1F3681EC880401	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880402	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880403	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880404	Blanket	Y	RW	Y
3005FB63AC1F3681EC880405	Towel	N	R	
3005FB63AC1F3681EC880406	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880407	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880408	Patientgown	N	NONE	
3005FB63AC1F3681EC880409	Towel	Y	RW	Y
3005FB63AC1F3681EC880410	Blanket	N	NONE	

Experiment 2bTrial sheet 3

Experiment	Experiment 4 with linen		
Experiment Trial	3		
Tag type	Intermec IT67 UHF ISO 18000-6C		
Number of tags	10		
Date	12/01/2009	**Time**	12.11 pm - 2.21 pm

Through observation on screen	
Read Tag id failure	
Read UM Failure	10
Write UM failure	5
Input string error	6

	Through program 3 - count
Read Tag Id count success	56
Read Tag Id count failure	0
Read UM count success	33
Read UM count failure	5
Write UM count success	7
Write UM count failure	5

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
3005FB63AC1F3681EC880401	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880402	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880403	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880404	Blanket	Y	RW	Y
3005FB63AC1F3681EC880405	Towel	N	R	
3005FB63AC1F3681EC880406	Flatsheet	N	NONE	
3005FB63AC1F3681EC880407	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880408	Patientgown	N	R	
3005FB63AC1F3681EC880409	Towel	Y	RW	Y
3005FB63AC1F3681EC880410	Blanket	Y	RW	Y

Experiment 2bTrial sheet 4

Experiment	Experiment 4 with linen		
Experiment Trial	4		
Tag type	Intermec IT67 UHF ISO 18000-6C		
Number of tags	10		
Date	12/01/2009	**Time**	2.44 pm - 2.54 pm

Through observation on screen	
Read Tag id failure	
Read UM Failure	14
Write UM failure	5
Input string error	7

	Through program 3 - count
Read Tag Id count success	56
Read Tag Id count failure	0
Read UM count success	26
Read UM count failure	5
Write UM count success	4
Write UM count failure	5

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
3005FB63AC1F3681EC880401	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880402	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880403	Patientgown	N	R	
3005FB63AC1F3681EC880404	Blanket	N	NONE	
3005FB63AC1F3681EC880405	Towel	N	R	
3005FB63AC1F3681EC880406	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880407	Pillowcase	N	NONE	
3005FB63AC1F3681EC880408	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880409	Towel	N	NONE	
3005FB63AC1F3681EC880410	Blanket	N	R	

Experiment 2bTrial sheet 5

Experiment	Experiment 4 with linen		
Experiment Trial	5		
Tag type	Intermec		
Number of tags	10		
Date	12/01/2009	**Time**	3.49 pm - 3.59 pm

Through observation on screen	
Read Tag id failure	
Read UM Failure	20
Write UM failure	6
Input string error	1

	Through program 3 - count
Read Tag Id count success	60
Read Tag Id count failure	0
Read UM count success	32
Read UM count failure	7
Write UM count success	8
Write UM count failure	7

Tag ID	**Linen Type**	**Status on Tag changed - checked though Program 2**	**Log Entries generated through Program 3**	**Print command through Program 4**
3005FB63AC1F3681EC880401	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880402	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880403	Patientgown	N	R	
3005FB63AC1F3681EC880404	Blanket	Y	RW	Y
3005FB63AC1F3681EC880405	Towel	N	R	
3005FB63AC1F3681EC880406	Flatsheet	Y	RW	Y
3005FB63AC1F3681EC880407	Pillowcase	Y	RW	Y
3005FB63AC1F3681EC880408	Patientgown	Y	RW	Y
3005FB63AC1F3681EC880409	Towel	Y	RW	Y
3005FB63AC1F3681EC880410	Blanket	Y	RW	Y

Appendix G - Print Receipt

Experiment 1a Trial sheet 1

Client: C123 Wellesley Hospital Australia
Blanket 1
FlatSheet 1
PatientGown 1
PillowCase 1
Towel 1
Total Weight 3.95

Experiment 1a Trial sheet 2

Client: C123 Wellesley Hospital Australia
Blanket 1
FlatSheet 1
PatientGown 1
PillowCase 1
Towel 1
Total Weight 3.95

Experiment 1a Trial sheet 3

Client: C123 Wellesley Hospital Australia
Blanket 1
FlatSheet 1
PatientGown 1
PillowCase 1
Towel 1
Total Weight 3.95

Experiment 1a Trial sheet 4

Client: C123 Wellesley Hospital Australia
FlatSheet 1
PatientGown 1
PillowCase 2
Total Weight 1.5

Experiment 1a Trial sheet 5

Client: C123 Wellesley Hospital Australia
Blanket 1
FlatSheet 1
PatientGown 1
PillowCase 2
Total Weight 3.5

Experiment 2a Trial sheet 1

Client: C123 Wellesley Hospital Australia
Blanket 1
FlatSheet 1
PillowCase 1
Total Weight 2.25

Experiment 2a Trial sheet 2

Client: C123 Wellesley Hospital Australia
Blanket 1
FlatSheet 1
PatientGown 1
PillowCase 2
Towel 1
Total Weight 4

Experiment 2a Trial sheet 3

Client: C123 Wellesley Hospital Australia
Blanket 1
FlatSheet 1
PillowCase 2
Total Weight 2.3

Experiment 2a Trial sheet 4

Client: C123 Wellesley Hospital Australia
Blanket 1
PillowCase 2
Towel 1
Total Weight 2.6

Experiment 2a Trial sheet 5

Client: C123 Wellesley Hospital Australia
FlatSheet 1
PatientGown 1
PillowCase 1
Total Weight 1.45

Experiment 1b Trial sheet 1

Client: C123 Taylor AKL
Blanket 1
FlatSheet 1
PatientGown 2
PillowCase 2
Towel 1
Total Weight 5.4

Experiment 1b Trial sheet 2

Client: C123 Taylor AKL
Blanket 1

FlatSheet	2
PatientGown	2
PillowCase	1
Towel	1
Total Weight	6.2

Experiment 1b Trial sheet 3

Client: C123 Taylor AKL

Blanket	2
FlatSheet	2
PatientGown	2
PillowCase	2
Towel	1
Total Weight	7.9

Experiment 1b Trial sheet 4

Client: C123 Taylor AKL

Blanket	1
FlatSheet	2
PatientGown	2
PillowCase	2
Towel	1
Total Weight	6.4

Experiment 1b Trial sheet 5

Client: C123 Taylor AKL

Blanket	2
FlatSheet	2
PatientGown	1
PillowCase	2
Total Weight	6.9

Experiment 2b Trial sheet 1

Client: C123 Taylor AKL

Blanket	1
FlatSheet	1
PatientGown	2
PillowCase	2
Towel	1
Total Weight	5.4

Experiment 2b Trial sheet 2

Client: C123 Taylor AKL

Blanket	1
FlatSheet	2
PatientGown	1
PillowCase	2
Towel	1
Total Weight	5.4

Experiment 2b Trial sheet 3

Client: C123 Taylor AKL

Blanket	2
FlatSheet	1
PatientGown	1
PillowCase	2
Towel	1
Total Weight	6.4

Experiment 2b Trial sheet 4

Client: C123 Taylor AKL

FlatSheet	2
PatientGown	1
PillowCase	1
Total Weight	2.7

Experiment 2b Trial sheet 5

Client: C123 Taylor AKL

Blanket	2
FlatSheet	2
PatientGown	1
PillowCase	2
Towel	1
Total Weight	7.4

Appendix H - Experiment Log Files

Experiment 1a Trial 1 Log sheet

Linen Tag ID	Linen tatus	Laundry Count	Linen Type	Linen Weight	DateTimeStamp	Bin Tag ID	Client Name	Client Address1	Client Address2
E0040000E90A4302	TRUE	27	FlatSheet	0.2	7/11/2009 9:15	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:15	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	TRUE	29	PatientGown	1.2	7/11/2009 9:15	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	30	PatientGown	1.2	7/11/2009 9:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	TRUE	34	Towel	0.5	7/11/2009 9:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:17	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	30	PatientGown	1.2	7/11/2009 9:17	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	TRUE	30	PillowCase	0.05	7/11/2009 9:17	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:17	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	TRUE	46	Blanket	2	7/11/2009 9:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	47	Blanket	2	7/11/2009 9:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	TRUE	30	PillowCase	0.05	7/11/2009 9:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	31	PillowCase	0.05	7/11/2009 9:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:19	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag ID	Linen tatus	Laundry Count	Linen Type	Linen Weight	DateTimeStamp	Bin Tag ID	Client Name	Client Address1	Client Address2
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	30	PatientGown	1.2	7/11/2009 9:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	47	Blanket	2	7/11/2009 9:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:22	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	30	PatientGown	1.2	7/11/2009 9:23	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:23	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:23	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	28	FlatSheet	0.2	7/11/2009 9:24	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	31	PillowCase	0.05	7/11/2009 9:24	E0040000B64B1502	C123	Wellesley Hospital	Australia

Experiment 1a Trial 2 Log sheet

Linen Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight	DateTimeStamp	Bin Tag ID	Client Name	Client Address1	Client Address2
E0040000E90A4302	TRUE	28	FlatSheet	0.2	7/11/2009 9:48	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	TRUE	30	PatientGown	1.2	7/11/2009 9:48	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:49	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	TRUE	35	Towel	0.5	7/11/2009 9:49	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:49	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:49	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:49	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:50	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:50	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	TRUE	31	PillowCase	0.05	7/11/2009 9:50	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight	DateTimeStamp	Bin Tag ID	Client Name	Client Address1	Client Address2
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	TRUE	47	Blanket	2	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	48	Blanket	2	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:51	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	29	FlatSheet	0.2	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	48	Blanket	2	7/11/2009 9:52	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:53	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:53	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	48	Blanket	2	7/11/2009 9:53	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:53	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:53	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	31	PatientGown	1.2	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag ID	Linen Status	Laundry Count	Linen Type	Linen Weight	DateTimeStamp	Bin Tag ID	Client Name	Client Address1	Client Address2
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	48	Blanket	2	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	35	Towel	0.5	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:58	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:58	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	32	PillowCase	0.05	7/11/2009 9:58	E0040000B64B1502	C123	Wellesley Hospital	Australia

Experiment 1a Trial 3 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000E7204302	TRUE	43	PillowCase	0.05	7/11/2009 10:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	44	PillowCase	0.05	7/11/2009 10:54	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	TRUE	31	PatientGown	1.2	7/11/2009 10:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	TRUE	35	Towel	0.5	7/11/2009 10:55	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E004000043FD4202	FALSE	36	Towel	0.5	7/11/2009 10:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	TRUE	29	FlatSheet	0.2	7/11/2009 10:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	30	FlatSheet	0.2	7/11/2009 10:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:55	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	44	PillowCase	0.05	7/11/2009 10:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:56	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	44	PillowCase	0.05	7/11/2009 10:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	30	FlatSheet	0.2	7/11/2009 10:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	36	Towel	0.5	7/11/2009 10:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	30	FlatSheet	0.2	7/11/2009 10:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:57	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:58	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:58	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	TRUE	32	PillowCase	0.05	7/11/2009 10:58	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	44	PillowCase	0.05	7/11/2009 10:58	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	32	PatientGown	1.2	7/11/2009 10:58	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	44	PillowCase	0.05	7/11/2009 10:59	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	44	PillowCase	0.05	7/11/2009 10:59	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	TRUE	48	Blanket	2	7/11/2009 10:59	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	49	Blanket	2	7/11/2009 10:59	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	44	PillowCase	0.05	7/11/2009 10:59	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	49	Blanket	2	7/11/2009 10:59	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	49	Blanket	2	7/11/2009 11:01	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	36	Towel	0.5	7/11/2009 11:01	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E004000031DD4202	FALSE	49	Blanket	2	7/11/2009 11:01	E0040000B64B1502	C123	Wellesley Hospital	Australia

Experiment 1a Trial 4 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000B34B1502	TRUE	33	PillowCase	0.05	7/11/2009 11:27	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:27	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:27	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:27	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:28	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:28	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	TRUE	44	PillowCase	0.05	7/11/2009 11:28	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:28	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	TRUE	32	PatientGown	1.2	7/11/2009 11:28	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	33	PatientGown	1.2	7/11/2009 11:28	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:28	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:29	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:29	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:29	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:29	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:29	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:29	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	TRUE	30	FlatSheet	0.2	7/11/2009 11:30	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	31	FlatSheet	0.2	7/11/2009 11:30	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:30	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:30	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	33	PatientGown	1.2	7/11/2009 11:31	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag id	Linen status	Laundr y count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:31	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:31	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:31	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:31	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:31	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:32	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:32	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:32	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	33	PatientGown	1.2	7/11/2009 11:33	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:33	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:33	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	33	PatientGown	1.2	7/11/2009 11:33	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	31	FlatSheet	0.2	7/11/2009 11:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	31	FlatSheet	0.2	7/11/2009 11:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	31	FlatSheet	0.2	7/11/2009 11:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	33	PatientGown	1.2	7/11/2009 11:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	45	PillowCase	0.05	7/11/2009 11:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	34	PillowCase	0.05	7/11/2009 11:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	31	FlatSheet	0.2	7/11/2009 11:36	E0040000B64B1502	C123	Wellesley Hospital	Australia

Experiment 1a Trial 5 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000E90A4302	TRUE	31	FlatSheet	0.2	7/11/2009 12:10	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	32	FlatSheet	0.2	7/11/2009 12:10	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	32	FlatSheet	0.2	7/11/2009 12:10	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	TRUE	49	Blanket	2	7/11/2009 12:10	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	50	Blanket	2	7/11/2009 12:10	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	32	FlatSheet	0.2	7/11/2009 12:11	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	32	FlatSheet	0.2	7/11/2009 12:11	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	32	FlatSheet	0.2	7/11/2009 12:11	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	TRUE	33	PatientGown	1.2	7/11/2009 12:11	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	TRUE	34	PillowCase	0.05	7/11/2009 12:11	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	35	PillowCase	0.05	7/11/2009 12:11	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	34	PatientGown	1.2	7/11/2009 12:12	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	32	FlatSheet	0.2	7/11/2009 12:12	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	32	FlatSheet	0.2	7/11/2009 12:12	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	35	PillowCase	0.05	7/11/2009 12:12	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	35	PillowCase	0.05	7/11/2009 12:12	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	50	Blanket	2	7/11/2009 12:13	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	34	PatientGown	1.2	7/11/2009 12:13	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E004000031DD4202	FALSE	50	Blanket	2	7/11/2009 12:13	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	50	Blanket	2	7/11/2009 12:14	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	TRUE	45	PillowCase	0.05	7/11/2009 12:14	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:14	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:14	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:15	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	34	PatientGown	1.2	7/11/2009 12:15	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	34	PatientGown	1.2	7/11/2009 12:15	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	34	PatientGown	1.2	7/11/2009 12:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	34	PatientGown	1.2	7/11/2009 12:17	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:17	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	35	PillowCase	0.05	7/11/2009 12:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	34	PatientGown	1.2	7/11/2009 12:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	46	PillowCase	0.05	7/11/2009 12:19	E0040000B64B1502	C123	Wellesley Hospital	Australia

Experiment 2a Trial 1 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address1	Address 2
E0040000B34B1502	TRUE	25	PillowCase	0.05	2/11/2009 14:32	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:32	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	TRUE	42	Blanket	2	2/11/2009 14:33	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:33	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:33	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	25	FlatSheet	0.2	2/11/2009 14:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:36	E0040000B64B1502	C123	Wellesley	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address1	Address 2
								Hospital	
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:37	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:37	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:37	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:37	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:38	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:38	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:38	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:38	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:38	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:38	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:39	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:39	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:39	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:39	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:39	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:40	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:40	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:40	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:40	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	43	Blanket	2	2/11/2009 14:40	E0040000B64B1502	C123	Wellesley	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address1	Address 2
								Hospital	
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:40	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:41	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	26	PillowCase	0.05	2/11/2009 14:41	E0040000B64B1502	C123	Wellesley Hospital	Australia

Experiment 2a Trial 2 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000E90A4302	TRUE	25	FlatSheet	0.2	2/11/2009 15:28	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	25	FlatSheet	0.2	2/11/2009 15:29	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	25	FlatSheet	0.2	2/11/2009 15:29	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	25	FlatSheet	0.2	2/11/2009 15:29	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	25	FlatSheet	0.2	2/11/2009 15:29	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	25	FlatSheet	0.2	2/11/2009 15:30	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	25	FlatSheet	0.2	2/11/2009 15:30	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	TRUE	30	Towel	0.5	2/11/2009 15:30	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	31	Towel	0.5	2/11/2009 15:30	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	TRUE	26	PillowCase	0.05	2/11/2009 15:31	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	27	PillowCase	0.05	2/11/2009 15:31	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	27	PillowCase	0.05	2/11/2009 15:31	E0040000B64B1502	C123	Wellesley	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
								Hospital	
E0040000B34B1502	FALSE	27	PillowCase	0.05	2/11/2009 15:31	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	27	PillowCase	0.05	2/11/2009 15:32	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	31	Towel	0.5	2/11/2009 15:32	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	TRUE	43	Blanket	2	2/11/2009 15:32	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	44	Blanket	2	2/11/2009 15:32	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	TRUE	40	PillowCase	0.05	2/11/2009 15:32	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	40	PillowCase	0.05	2/11/2009 15:33	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	31	Towel	0.5	2/11/2009 15:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	44	Blanket	2	2/11/2009 15:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	TRUE	26	PatientGown	1.2	2/11/2009 15:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	27	PatientGown	1.2	2/11/2009 15:34	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	27	PillowCase	0.05	2/11/2009 15:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	31	Towel	0.5	2/11/2009 15:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	40	PillowCase	0.05	2/11/2009 15:35	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	27	PillowCase	0.05	2/11/2009 15:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	40	PillowCase	0.05	2/11/2009 15:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	31	Towel	0.5	2/11/2009 15:36	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	31	Towel	0.5	2/11/2009 15:37	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	44	Blanket	2	2/11/2009 15:37	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E004000031DD4202	FALSE	44	Blanket	2	2/11/2009 15:37	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	FALSE	31	Towel	0.5	2/11/2009 15:37	E0040000B64B1502	C123	Wellesley Hospital	Australia

Experiment 2a Trial 3 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000E7204302	TRUE	40	PillowCase	0.05	3/11/2009 14:17	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	41	PillowCase	0.05	3/11/2009 14:17	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	TRUE	27	PatientGown	1.2	3/11/2009 14:17	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	41	PillowCase	0.05	3/11/2009 14:17	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	41	PillowCase	0.05	3/11/2009 14:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	TRUE	27	PillowCase	0.05	3/11/2009 14:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	28	PillowCase	0.05	3/11/2009 14:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	28	PillowCase	0.05	3/11/2009 14:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	28	PillowCase	0.05	3/11/2009 14:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	TRUE	25	FlatSheet	0.2	3/11/2009 14:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	28	PillowCase	0.05	3/11/2009 14:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:20	E0040000B64B1502	C123	Wellesley	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
								Hospital	
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	28	PillowCase	0.05	3/11/2009 14:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	TRUE	44	Blanket	2	3/11/2009 14:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	28	PillowCase	0.05	3/11/2009 14:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	45	Blanket	2	3/11/2009 14:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	28	PillowCase	0.05	3/11/2009 14:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	45	Blanket	2	3/11/2009 14:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:22	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:22	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:22	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:23	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:23	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:23	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:23	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	28	PillowCase	0.05	3/11/2009 14:23	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:24	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:24	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:24	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:24	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:24	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	28	PillowCase	0.05	3/11/2009 14:25	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	45	Blanket	2	3/11/2009 14:25	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	TRUE	31	Towel	0.5	3/11/2009 14:25	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	26	FlatSheet	0.2	3/11/2009 14:25	E0040000B64B1502	C123	Wellesley Hospital	Australia

Experiment 2a Trial 4 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000E7204302	TRUE	41	PillowCase	0.05	3/11/2009 15:13	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	42	PillowCase	0.05	3/11/2009 15:13	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	TRUE	28	PatientGown	1.2	3/11/2009 15:14	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	TRUE	45	Blanket	2	3/11/2009 15:14	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	46	Blanket	2	3/11/2009 15:14	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	TRUE	28	PillowCase	0.05	3/11/2009 15:15	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	29	PillowCase	0.05	3/11/2009 15:15	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	46	Blanket	2	3/11/2009 15:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000043FD4202	TRUE	33	Towel	0.5	3/11/2009 15:16	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E004000043FD4202	FALSE	34	Towel	0.5	3/11/2009 15:16	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	46	Blanket	2	3/11/2009 15:18	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	46	Blanket	2	3/11/2009 15:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	29	PillowCase	0.05	3/11/2009 15:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	46	Blanket	2	3/11/2009 15:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	46	Blanket	2	3/11/2009 15:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	46	Blanket	2	3/11/2009 15:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	46	Blanket	2	3/11/2009 15:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	46	Blanket	2	3/11/2009 15:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E004000031DD4202	FALSE	46	Blanket	2	3/11/2009 15:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	FALSE	42	PillowCase	0.05	3/11/2009 15:22	E0040000B64B1502	C123	Wellesley Hospital	Australia

Experiment 2a Trial 5 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000B34B1502	TRUE	29	PillowCase	0.05	4/11/2009 15:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:19	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:20	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000E90A4302	TRUE	26	FlatSheet	0.2	4/11/2009 15:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E90A4302	FALSE	27	FlatSheet	0.2	4/11/2009 15:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E7204302	TRUE	42	PillowCase	0.05	4/11/2009 15:20	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	TRUE	28	PatientGown	1.2	4/11/2009 15:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:21	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:22	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:23	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	29	PatientGown	1.2	4/11/2009 15:23	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:24	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:24	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:24	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	29	PatientGown	1.2	4/11/2009 15:25	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:25	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:25	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:26	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:26	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:26	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	29	PatientGown	1.2	4/11/2009 15:26	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000E4F34202	FALSE	29	PatientGown	1.2	4/11/2009 15:26	E0040000B64B1502	C123	Wellesley Hospital	Australia

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:26	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:27	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:27	E0040000B64B1502	C123	Wellesley Hospital	Australia
E0040000B34B1502	FALSE	30	PillowCase	0.05	4/11/2009 15:27	E0040000B64B1502	C123	Wellesley Hospital	Australia

Experiment 1b Trial 1 Log sheet

Linen Tag id	Linen status	Laund ry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681 EC880401	TRUE	0	FlatSheet	0.5	28/11/2009 21:51	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880401	FALSE	1	FlatSheet	0.5	28/11/2009 21:51	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880401	FALSE	1	FlatSheet	0.5	28/11/2009 21:52	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880401	FALSE	1	FlatSheet	0.5	28/11/2009 21:52	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880407	TRUE	0	PillowCase	0.2	28/11/2009 21:53	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880407	FALSE	1	PillowCase	0.2	28/11/2009 21:53	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880407	FALSE	1	PillowCase	0.2	28/11/2009 21:53	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880404	TRUE	0	Blanket	2	28/11/2009 21:53	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880404	FALSE	1	Blanket	2	28/11/2009 21:53	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880403	TRUE	0	PatientGown	1	28/11/2009 21:54	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880403	FALSE	1	PatientGown	1	28/11/2009 21:54	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880402	TRUE	0	PillowCase	0.2	28/11/2009 21:54	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681	FALSE	1	PillowCase	0.2	28/11/2009 21:54	3005FB63AC1F3681E	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
EC880402						C880468			
3005FB63AC1F3681 EC880402	FALSE	1	PillowCase	0.2	28/11/2009 21:54	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880404	FALSE	1	Blanket	2	28/11/2009 21:54	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880410	TRUE	0	Blanket	2	28/11/2009 21:55	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880403	FALSE	1	PatientGown	1	28/11/2009 21:55	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880407	FALSE	1	PillowCase	0.2	28/11/2009 21:55	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880408	TRUE	0	PatientGown	1	28/11/2009 21:56	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880408	FALSE	1	PatientGown	1	28/11/2009 21:56	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880404	FALSE	1	Blanket	2	28/11/2009 21:56	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880409	TRUE	0	Towel	0.5	28/11/2009 21:56	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880409	FALSE	1	Towel	0.5	28/11/2009 21:56	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880404	FALSE	1	Blanket	2	28/11/2009 21:57	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880409	FALSE	1	Towel	0.5	28/11/2009 21:57	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880408	FALSE	1	PatientGown	1	28/11/2009 21:57	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880407	FALSE	1	PillowCase	0.2	28/11/2009 21:57	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880404	FALSE	1	Blanket	2	28/11/2009 21:57	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880408	FALSE	1	PatientGown	1	28/11/2009 21:58	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880403	FALSE	1	PatientGown	1	28/11/2009 21:58	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880407	FALSE	1	PillowCase	0.2	28/11/2009 21:58	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880404	FALSE	1	Blanket	2	28/11/2009 21:58	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681	TRUE	0	FlatSheet	1	28/11/2009 21:58	3005FB63AC1F3681E	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
EC880406						C880468			
3005FB63AC1F3681 EC880407	FALSE	1	PillowCase	0.2	28/11/2009 21:59	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880409	FALSE	1	Towel	0.5	28/11/2009 21:59	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880407	FALSE	1	PillowCase	0.2	28/11/2009 21:59	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880407	FALSE	1	PillowCase	0.2	28/11/2009 22:00	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880407	FALSE	1	PillowCase	0.2	28/11/2009 22:00	3005FB63AC1F3681E C880468	C123	Taylor	AKL
3005FB63AC1F3681 EC880408	FALSE	1	PatientGown	1	28/11/2009 22:00	3005FB63AC1F3681E C880468	C123	Taylor	AKL

Experiment 1b Trial 2 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880408	TRUE	1	PatientGown	1	28/11/2009 23:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	2	PatientGown	1	28/11/2009 23:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	TRUE	1	Towel	0.5	28/11/2009 23:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	2	Towel	0.5	28/11/2009 23:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	TRUE	0	FlatSheet	1	28/11/2009 23:01	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	1	FlatSheet	0.5	28/11/2009 23:01	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	2	Towel	0.5	28/11/2009 23:02	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	TRUE	1	Blanket	2	28/11/2009 23:02	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	1	FlatSheet	0.5	28/11/2009 23:02	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	2	Towel	0.5	28/11/2009 23:03	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	2	Towel	0.5	28/11/2009 23:03	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880402	TRUE	1	PillowCase	0.2	28/11/2009 23:03	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	2	PillowCase	0.2	28/11/2009 23:03	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	TRUE	0	FlatSheet	1	28/11/2009 23:03	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	1	FlatSheet	1	28/11/2009 23:03	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	1	FlatSheet	1	28/11/2009 23:04	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	TRUE	0	Blanket	2	28/11/2009 23:04	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	FALSE	1	Blanket	2	28/11/2009 23:04	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	FALSE	1	Blanket	2	28/11/2009 23:04	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	1	FlatSheet	0.5	28/11/2009 23:04	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	2	FlatSheet	0.5	28/11/2009 23:04	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	2	PatientGown	1	28/11/2009 23:06	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	2	FlatSheet	0.5	28/11/2009 23:07	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	TRUE	1	PatientGown	1	28/11/2009 23:07	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	28/11/2009 23:07	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	2	PatientGown	1	28/11/2009 23:08	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Experiment 1b Trial 3 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880403	TRUE	2	PatientGown	1	28/11/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	TRUE	2	Towel	0.5	28/11/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	3	Towel	0.5	28/11/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880407	TRUE	1	PillowCase	0.2	28/11/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	2	PillowCase	0.2	28/11/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	2	PillowCase	0.2	28/11/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	2	PillowCase	0.2	28/11/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	2	PillowCase	0.2	28/11/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	TRUE	2	PatientGown	1	28/11/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	2	FlatSheet	0.5	28/11/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	3	FlatSheet	0.5	28/11/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	28/11/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	3	FlatSheet	0.5	28/11/2009 23:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	28/11/2009 23:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	3	FlatSheet	0.5	28/11/2009 23:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	28/11/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	TRUE	2	PatientGown	1	28/11/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	3	PatientGown	1	28/11/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	28/11/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	TRUE	1	FlatSheet	1	28/11/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	2	FlatSheet	1	28/11/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	28/11/2009 23:56	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	3	Towel	0.5	28/11/2009 23:56	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880404	TRUE	1	Blanket	2	28/11/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	28/11/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	28/11/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	TRUE	2	PillowCase	0.2	28/11/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	28/11/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	28/11/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	28/11/2009 23:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	TRUE	1	Blanket	2	28/11/2009 23:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	FALSE	2	Blanket	2	28/11/2009 23:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	28/11/2009 23:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	28/11/2009 23:59	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	28/11/2009 23:59	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	28/11/2009 23:59	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	29/11/2009 0:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	29/11/2009 0:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	2	FlatSheet	1	29/11/2009 0:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	29/11/2009 0:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	29/11/2009 0:01	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	29/11/2009 0:01	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Experiment 1b Trial 4 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880403	TRUE	2	PatientGown	1	11/28/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	TRUE	2	Towel	0.5	11/28/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	3	Towel	0.5	11/28/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	TRUE	1	PillowCase	0.2	11/28/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	2	PillowCase	0.2	11/28/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	2	PillowCase	0.2	11/28/2009 23:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	2	PillowCase	0.2	11/28/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	2	PillowCase	0.2	11/28/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	TRUE	2	PatientGown	1	11/28/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	2	FlatSheet	0.5	11/28/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	3	FlatSheet	0.5	11/28/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	11/28/2009 23:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	3	FlatSheet	0.5	11/28/2009 23:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	11/28/2009 23:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	3	FlatSheet	0.5	11/28/2009 23:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	11/28/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	TRUE	2	PatientGown	1	11/28/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	3	PatientGown	1	11/28/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	11/28/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL

3005FB63AC1F3681EC880406	TRUE	1	FlatSheet	1	11/28/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	2	FlatSheet	1	11/28/2009 23:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	2	PatientGown	1	11/28/2009 23:56	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	3	Towel	0.5	11/28/2009 23:56	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	TRUE	1	Blanket	2	11/28/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	11/28/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	11/28/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	TRUE	2	PillowCase	0.2	11/28/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	11/28/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	11/28/2009 23:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	11/28/2009 23:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	TRUE	1	Blanket	2	11/28/2009 23:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	FALSE	2	Blanket	2	11/28/2009 23:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	11/28/2009 23:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	11/28/2009 23:59	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	11/28/2009 23:59	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	11/28/2009 23:59	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	11/29/2009 0:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	11/29/2009 0:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	2	FlatSheet	1	11/29/2009 0:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	2	Blanket	2	11/29/2009 0:00	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	11/29/2009 0:01	3005FB63AC1F3681EC880468	C123	Taylor	AKL

3005FB63AC1F3681EC880402	FALSE	3	PillowCase	0.2	11/29/2009 0:01	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Experiment 1b Trial 5 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880407	FALSE	2	PillowCase	0.2	29/11/2009 8:45	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	TRUE	3	Towel	0.5	29/11/2009 8:47	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	4	Towel	0.5	29/11/2009 8:47	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	TRUE	3	PatientGown	1	29/11/2009 8:47	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	4	Towel	0.5	29/11/2009 8:47	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	4	Towel	0.5	29/11/2009 8:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	4	Towel	0.5	29/11/2009 8:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	4	Towel	0.5	29/11/2009 8:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	TRUE	3	PatientGown	1	29/11/2009 8:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	4	PatientGown	1	29/11/2009 8:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	TRUE	2	Blanket	2	29/11/2009 8:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	FALSE	3	Blanket	2	29/11/2009 8:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	4	Towel	0.5	29/11/2009 8:50	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	TRUE	2	PatientGown	1	29/11/2009 8:51	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	3	PatientGown	1	29/11/2009 8:51	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	3	PatientGown	1	29/11/2009 8:51	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	TRUE	3	PillowCase	0.2	29/11/2009 8:51	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	4	PillowCase	0.2	29/11/2009 8:51	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880409	FALSE	4	Towel	0.5	29/11/2009 8:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	3	PatientGown	1	29/11/2009 8:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	4	PillowCase	0.2	29/11/2009 8:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	3	PatientGown	1	29/11/2009 8:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	4	PillowCase	0.2	29/11/2009 8:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	3	FlatSheet	0.5	29/11/2009 8:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	4	PillowCase	0.2	29/11/2009 8:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	4	FlatSheet	0.5	29/11/2009 8:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	TRUE	2	FlatSheet	1	29/11/2009 8:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	3	FlatSheet	1	29/11/2009 8:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	4	PillowCase	0.2	29/11/2009 8:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	4	Towel	0.5	29/11/2009 8:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	4	Towel	0.5	29/11/2009 8:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	3	FlatSheet	1	29/11/2009 8:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	3	FlatSheet	1	29/11/2009 8:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Experiment 2b Trial 1 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880409	TRUE	5	Towel	0.5	1/12/2009 12:09	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	6	Towel	0.5	1/12/2009 12:09	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	6	FlatSheet	0.5	1/12/2009 12:10	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880401	FALSE	7	FlatSheet	0.5	1/12/2009 12:10	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	6	Towel	0.5	1/12/2009 12:11	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	TRUE	3	PillowCase	0.2	1/12/2009 12:11	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	TRUE	3	Blanket	2	1/12/2009 12:11	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	4	Blanket	2	1/12/2009 12:11	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	TRUE	4	PatientGown	1	1/12/2009 12:11	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	6	Towel	0.5	1/12/2009 12:12	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	4	PatientGown	1	1/12/2009 12:12	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	6	Towel	0.5	1/12/2009 12:12	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	TRUE	5	PillowCase	0.2	1/12/2009 12:13	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 12:13	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 12:13	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 12:13	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	TRUE	4	PatientGown	1	1/12/2009 12:13	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	5	PatientGown	1	1/12/2009 12:13	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	5	PatientGown	1	1/12/2009 12:14	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	4	PatientGown	1	1/12/2009 12:14	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	5	PatientGown	1	1/12/2009 12:14	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	4	PatientGown	1	1/12/2009 12:15	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	5	PatientGown	1	1/12/2009 12:15	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 12:15	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 12:16	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	TRUE	3	PillowCase	0.2	1/12/2009 12:16	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	4	Blanket	2	1/12/2009 12:17	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	4	PillowCase	0.2	1/12/2009 12:17	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	TRUE	4	Blanket	2	1/12/2009 12:17	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 12:18	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	4	PillowCase	0.2	1/12/2009 12:18	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 12:19	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Experiment 2b Trial 2 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880407	TRUE	4	PillowCase	0.2	1/12/2009 13:42	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	TRUE	6	Towel	0.5	1/12/2009 13:42	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	7	Towel	0.5	1/12/2009 13:42	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	7	Towel	0.5	1/12/2009 13:43	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	TRUE	4	Blanket	2	1/12/2009 13:43	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	5	Blanket	2	1/12/2009 13:43	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	7	FlatSheet	0.5	1/12/2009 13:43	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	TRUE	5	PatientGown	1	1/12/2009 13:43	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	6	PatientGown	1	1/12/2009 13:44	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	8	FlatSheet	0.5	1/12/2009 13:44	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	TRUE	5	PillowCase	0.2	1/12/2009 13:45	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 13:45	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	TRUE	4	PillowCase	0.2	1/12/2009 13:46	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	5	Blanket	2	1/12/2009 13:46	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	TRUE	4	FlatSheet	1	1/12/2009 13:46	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	5	FlatSheet	1	1/12/2009 13:46	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	8	FlatSheet	0.5	1/12/2009 13:47	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	4	PillowCase	0.2	1/12/2009 13:47	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	8	FlatSheet	0.5	1/12/2009 13:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880401	FALSE	8	FlatSheet	0.5	1/12/2009 13:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	5	FlatSheet	1	1/12/2009 13:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	5	FlatSheet	1	1/12/2009 13:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	4	PillowCase	0.2	1/12/2009 13:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	5	FlatSheet	1	1/12/2009 13:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Experiment 2b Trial 3 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880403	TRUE	6	PatientGown	1	1/12/2009 14:12	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:12	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:13	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	TRUE	5	PillowCase	0.2	1/12/2009 14:13	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	TRUE	4	PatientGown	1	1/12/2009 14:13	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	TRUE	5	Blanket	2	1/12/2009 14:13	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	8	FlatSheet	0.5	1/12/2009 14:14	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	TRUE	5	PillowCase	0.2	1/12/2009 14:14	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	TRUE	5	Blanket	2	1/12/2009 14:14	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	6	Blanket	2	1/12/2009 14:14	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	5	PillowCase	0.2	1/12/2009 14:15	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	9	FlatSheet	0.5	1/12/2009 14:15	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 14:15	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880410	FALSE	6	Blanket	2	1/12/2009 14:15	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	FALSE	6	Blanket	2	1/12/2009 14:16	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	FALSE	6	Blanket	2	1/12/2009 14:16	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 14:16	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:16	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:17	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	9	FlatSheet	0.5	1/12/2009 14:17	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	5	PillowCase	0.2	1/12/2009 14:17	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:17	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	5	PillowCase	0.2	1/12/2009 14:17	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 14:17	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	9	FlatSheet	0.5	1/12/2009 14:18	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 14:18	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:18	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:19	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:19	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	TRUE	7	Towel	0.5	1/12/2009 14:19	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	8	Towel	0.5	1/12/2009 14:19	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:19	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	9	FlatSheet	0.5	1/12/2009 14:20	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880409	FALSE	8	Towel	0.5	1/12/2009 14:20	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:20	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	FALSE	7	PatientGown	1	1/12/2009 14:21	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Experiment 2b Trial 4 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880403	TRUE	7	PatientGown	1	1/12/2009 14:46	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	TRUE	5	PillowCase	0.2	1/12/2009 14:47	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	TRUE	5	FlatSheet	1	1/12/2009 14:47	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	6	FlatSheet	1	1/12/2009 14:47	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	6	FlatSheet	1	1/12/2009 14:47	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	TRUE	5	PatientGown	1	1/12/2009 14:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	6	PatientGown	1	1/12/2009 14:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 14:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 14:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	6	PatientGown	1	1/12/2009 14:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	6	FlatSheet	1	1/12/2009 14:48	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	6	PatientGown	1	1/12/2009 14:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	6	FlatSheet	1	1/12/2009 14:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	TRUE	6	Blanket	2	1/12/2009 14:49	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	TRUE	6	Blanket	2	1/12/2009 14:50	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880408	FALSE	6	PatientGown	1	1/12/2009 14:50	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 14:50	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	6	PatientGown	1	1/12/2009 14:50	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	6	PatientGown	1	1/12/2009 14:51	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	9	FlatSheet	0.5	1/12/2009 14:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	10	FlatSheet	0.5	1/12/2009 14:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	6	PatientGown	1	1/12/2009 14:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	6	PatientGown	1	1/12/2009 14:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	6	PatientGown	1	1/12/2009 14:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	6	FlatSheet	1	1/12/2009 14:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 14:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	5	PillowCase	0.2	1/12/2009 14:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	6	PatientGown	1	1/12/2009 14:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Experiment 2b Trial 5 Log sheet

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880402	TRUE	5	PillowCase	0.2	1/12/2009 15:50	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	6	PillowCase	0.2	1/12/2009 15:50	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	TRUE	5	PillowCase	0.2	1/12/2009 15:51	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	TRUE	6	PatientGown	1	1/12/2009 15:51	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	TRUE	6	Blanket	2	1/12/2009 15:51	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880406	TRUE	6	FlatSheet	1	1/12/2009 15:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	7	FlatSheet	1	1/12/2009 15:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	5	PillowCase	0.2	1/12/2009 15:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	7	PatientGown	1	1/12/2009 15:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	6	PillowCase	0.2	1/12/2009 15:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	TRUE	10	FlatSheet	0.5	1/12/2009 15:52	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	7	FlatSheet	1	1/12/2009 15:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	5	PillowCase	0.2	1/12/2009 15:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	6	PillowCase	0.2	1/12/2009 15:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	6	PillowCase	0.2	1/12/2009 15:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	5	PillowCase	0.2	1/12/2009 15:53	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880403	TRUE	7	PatientGown	1	1/12/2009 15:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880406	FALSE	7	FlatSheet	1	1/12/2009 15:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	7	PatientGown	1	1/12/2009 15:54	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880402	FALSE	6	PillowCase	0.2	1/12/2009 15:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	5	PillowCase	0.2	1/12/2009 15:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	TRUE	6	Blanket	2	1/12/2009 15:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	5	PillowCase	0.2	1/12/2009 15:55	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	TRUE	8	Towel	0.5	1/12/2009 15:56	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	7	Blanket	2	1/12/2009 15:56	3005FB63AC1F3681EC880468	C123	Taylor	AKL

Linen Tag id	Linen status	Laundry count	Linen type	Linen weight	DateTimeStamp	Bin Tag id	Client name	Address 1	Address 2
3005FB63AC1F3681EC880402	FALSE	6	PillowCase	0.2	1/12/2009 15:56	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	7	Blanket	2	1/12/2009 15:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880410	FALSE	6	Blanket	2	1/12/2009 15:57	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880408	FALSE	7	PatientGown	1	1/12/2009 15:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	10	FlatSheet	0.5	1/12/2009 15:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880409	FALSE	8	Towel	0.5	1/12/2009 15:58	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880401	FALSE	10	FlatSheet	0.5	1/12/2009 15:59	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880407	FALSE	5	PillowCase	0.2	1/12/2009 15:59	3005FB63AC1F3681EC880468	C123	Taylor	AKL
3005FB63AC1F3681EC880404	FALSE	7	Blanket	2	1/12/2009 15:59	3005FB63AC1F3681EC880468	C123	Taylor	AKL