A Framework for Auto-ID Enabled Business

by

Timothy Porter Milne

M.S. Mechanical Engineering, 1995
Brigham Young University

B.S. Mechanical Engineering, 1992
Brigham Young University

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 8, 2003
[September 2003]

Signature of Author:.....
Department of Mechanical Engineering
August 8, 2003

Certified by:.....
Sanjay E. Sarma
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by:.....
Ain A. Sonin
Chairman, Department Committee on Graduate Students

A Framework for Auto-ID Enabled Business

by

Timothy Porter Milne

Submitted to the Department of Mechanical Engineering
August 8, 2003
In partial fulfillment of the requirements for the degree of
Master of Science

ABSTRACT

Modern commerce is as much about trading information as it is about exchanging goods and money. With an Auto-ID enabled future, the expeditious exchange of information will be even more critical. The fundamental problem with e-commerce today is the soft connection between goods and their related information, which frequently results in breakdowns between the physical and information worlds.

The Auto-ID Center at MIT has proposed a system using Radio Frequency Identification (RFID) tags on objects, coupled with a distributed information system using the Internet, that will allow the tracking of physical objects and the automatic association of relevant data about those objects.

This thesis outlines a framework for a new method of using Auto-ID to connect the physical world to the information world using an example from commerce: Shipping and Receiving Verification. This thesis also presents a mapping of Core PML, a component of the Auto-ID system, into various EDI and XML based messaging systems including Simpl-eb and UBL. Finally, this thesis describes the results of a reference implementation based on my research, including a candidate platform and messaging channel, and then concludes with lessons learned and recommendations for future research in the field of Auto-ID enabled Business (a-Biz).

Thesis Supervisor: Sanjay E. Sarma
Title: Associate Professor of Mechanical Engineering

Acknowledgements

This work is not the result of one man's labor. For their contributions to this project I would like to thank the following (in no particular order):

First of all, I would like to thank the a-Biz team: Chris Clauss for the initial spark and the ongoing support that allowed a-Biz to flourish. Bob Ganley, who brought Sun's many resources to bear. Elwin Loomis for making his team available and Rick Schendel for assembling the group within Target that made this research both possible and enjoyable. Paul Rieger for the start and Tom Torre for the finish that kept the project rolling. Budi Saputra for his insights and ongoing feedback. Steve Rehling for his tireless and undaunted foray into all aspects of Auto-ID. Ted Osinski and Tom Heist for their help and feedback on the models. Amit Goyal for his exceptional contribution, it was a blast. Dan Engels and Tom Scharfeld for their help with readers and antennas. Tom Scharfeld and Heidi Schuster for a web page. Jim Clarke for gathering a great team, Sean Clark for the servers and moral support, and Rolla Grace for getting the servers online. Ben Griffin and his uncanny 6th sense for finding and fixing problems. Stu Stern and his team for creating reality from requirements. Dirk Heyman and his Zurich team for a great demo. And everyone else that contributed to a-Biz along the way.

There are also those from the Auto-ID team that I would like to single out: Robin Koh for his energy, direction, guidance and friendship. Christian Floerkemeier for his great work with PML Core, his help and his friendship. Dr. Duncan McFarlane for assembling a great UK team and all the great work they do. Humberto Moran for helping to congeal my thinking about Auto-ID use cases. Professor Sanjay Sarma for his advice, assistance and most of all enthusiasm. Dr. Brock for getting the whole thing started. Brendon Lewis, Sephen Ho, and Yun Kang for all the fun times. David Rodriguera and the sisters Skeete, Tracy and Carolyn, without whom nothing really happens at Auto-ID.

I would also like to thank some members of the MIT community: Dr. George Kocur for two of the best classes I took at MIT, and for his willingness to share his unique insight and industry expertise while proof reading this thesis. The ME department at MIT, for slamming some doors, but keeping others open. Leslie Regan and her staff for all of their help during my graduate career (keeping some of those doors open). Professor Ain Sonin for a comforting ear during troubled times.

To everyone who proof read this thesis: Sanjay Sarma, Tom Scharfeld, George Kocur and my wife Kim, it got better with each revision.

My wonderful children, Taylor, Jayden, Alyssa, and Amberly, for their patience and unconditional love, which truly makes make it all worthwhile.

But most of all thanks to my beautiful wife, Kim, for her unwavering support love and patience. What one accomplishes in this life is really the product of two souls joined in adventure, led by that God who quickens our intelligence, making all things both interesting and possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: Introduction

1.1 EDI and XML

Over the past half-century, many organizations have expended time and resources trying to connect their Order, Shipping Billing systems (OSB) to reduce or eliminate the many errors that occur while trading goods. Beginning in the 1950's, when large corporations began automating their routine paperwork, through the 1960's, when many core business functions were handed over to large data processing systems, and on through the 1970's, 80's and 90's, when many large corporations began using "standard" Electronic Data Interchange (EDI) using Value Added Network providers (VANs) [1], the lingua franca for business has undergone drastic changes. There are currently two Traditional EDI standards in widespread use, the ANSI X12 standard [2], maintained by the American National Standards Institute's [3] X12 Committee and widely used in North America, and the Electronic Data Interchange For Administration, Commerce, and Transportation (EDIFACT) standard [4], which is an international implementation of EDI sponsored by the United Nations [5] and the European Union and is used for international commerce and commerce within countries that have adopted EDIFACT as their standard.

Traditional EDI has brought commerce a long way towards the goal of connecting disparate systems between trading organizations, but it has failed to realize it's true potential. This is due in part to the fact that the two EDI "standards" are not really standards, but a set of rules from which a slew of proprietary implementations or industry specific "standards" have been based [6], as well as many other identified shortcomings including cost [7], and other factors [8, 9, 10]. The net effect of these barriers has been a lack of widespread EDI adoption.

With the explosive growth of the Internet, many industries have been moving away from traditional EDI transactions that use a dedicated VAN, and towards newer Extensible Markup Language (XML) [11] based technologies over the Internet. Even EDI is finding its way into the XML world as both the X12 Committee and EDIFACT have begun research and support for XML based initiatives [12, 13]. However, moving to XML is not the simple panacea that most would have you believe, and there remain many issues that will need to be addressed [14].

In addition to XML based EDI, there are other consortiums and industry groups that are deploying fundamentally new architectures aimed at addressing the issues surrounding electronically conducted business, and these include Simpl-eb [15], from the Global Commerce Initiative [16] and EAN.UCC [17], Rosettanet [18], and the ebXML [19] based Universal Business Language (UBL) [20] and many more. In fact, too many more. So many that the resulting standards proliferation has been an impediment to global commerce. A truly global standard is in order [21].

1.2 The Fundamental Problem

The problem, however, is that none of the initiatives or standards mentioned in the last section address the core problem of electronic data communication between organizations. There is a fundamental and physical disconnect between the information being transmitted via e-commerce, and the goods with which the information is associated.

This disconnect is a result of a combination of procedural, systemic and human errors. Data integrity and reliability are only as good as the systems and procedures in place to capture and convey them, and unfortunately most systems have a unreliable links, most involving humans, that lead to inaccuracies and a breakdown between the physical and information worlds. Until this gap can be bridged, the data being transmitted is only as good as the underlying assumptions and the labeling codes.

1.3 A Solution

The Auto-ID Center at MIT [22] has proposed a system using Radio Frequency Identification (RFID) tags on objects, coupled with a distributed information system using the Internet, that would allow for the tracking of physical objects and the automatic association of relevant data about those objects [23].

While directing the a-Biz Joint Project of the Auto-ID Center at MIT, I investigated how the Auto-ID infrastructure, including objects tagged with RFID, improves the process of shipping and receiving using an automatically generated and verified Despatch Advice. Goyal has summarized some of the initial work surrounding Shipping and Receiving Verification [24].

In this thesis I will consider the broader framework within which the Despatch Advice is an integral part [25], and give a detailed outline of an Auto-ID enabled method of connecting the physical and information worlds in the realm of e-commerce [26]. As part of the exposition, I will introduce various components of the Auto-ID system along with their integration into a framework.

In this thesis I will also present a mapping of Core PML, a component of the Auto-ID system, into various EDI and XML based messaging systems including Simpl-eb and UBL.

Finally, I will describe the results of a reference implementation based on my research, including a candidate platform and messaging channel, and then conclude with lessons learned and recommendations for future research in the field of Auto-ID enabled Business (a-Biz).

CHAPTER 2: Background

In this chapter I'll begin with a sampling of historical and emerging technologies used in marking and conveying information about physical goods. I will then proceed to introduce the major components and concepts necessary for an Auto-ID enabled framework for business. In the next chapter I'll use the building blocks presented in this chapter to build a complete picture of the framework.

2.1 Introduction

A brief review of related technologies is in order before covering the components of Auto-ID. This section begins with the UPC bar code and RFID, then turns to traditional EDI and newer XML based messaging standards. The subject matter in this section lays an introductory foundation for the Auto-ID components presented in the next section.

2.1.1 The UPC

Perhaps the most successful attempt to overcome the problem of associating information with an object to date is the Universal Product Code (UPC). For over 25 years the UPC has enabled many efficiencies in the supply chain, not the least of which are retail checkout and inventory management.

There are several versions of the UPC published by the EAN.UCC used worldwide, and a representative example of the UPC-12 is shown in the following figure [27]:

Figure 1 A Universal Product Code (UPC) with Manufacturer and Item [27]

As the figure shows, the UPC is broken down into manufacturer and item or product class codes, but there is no unique item serialization code. Thus, any

UPC labeled item is indistinguishable from any other similarly labeled item. There are, however, many advantages to be gained by unique serializations. The automotive industry has been using a uniquely serialized Vehicle Identification Number (VIN) for years to track individual vehicles [28]. This has implications for theft and product quality in the used car market.

But lack of serialization is only one of a UPC based system's shortcomings. Many other shortcomings have to do with the process of capturing the UPC code itself, including the line of sight restrictions this imposes [29]. The bar code must be properly aligned and visible for a scanner to successfully take a reading. For this reason, many systems designed around the traditional bar code expend a lot of time and energy assuring the alignment of code with scanner to assure proper reads.

But perhaps the most significant shortcoming of the UPC code is the limitations it imposes on the type and amount of information that it can convey. As noted, the standard UPC only contains manufacturer and product class. Other bar codes like the Serialized Shipping Container Code (SSCC) [30], which is used to label logistics units shipped between companies, contain a unique number that links to electronic information, but this information is static and conveyed in non-standard ways, making it difficult to reuse in other contexts. In fact, the SSCC, by design, is to be used only once, and is invalidated when any changes are made to the logistics unit. This system does not lend itself to use and re-use throughout the supply chain and lifetime of an individual object.

Recent research has focused on 2D and 3D barcodes, which can handle significantly larger amounts of information, and are referred to in the barcode industry as portable databases. With a portable database, all of the significant data necessary for a localized operation can be conveyed by and read from the barcode itself. There are several major drawbacks to a portable database approach, however, including slower read rates caused by the increased data transfer, error correction challenges, and the static nature of the data. If you don't use data redundancy or a good error correction scheme like Reed-Solomon error encoding [31, 32], then damaged, smudged or partially obscured barcodes become useless. More importantly, systems based on a portable database are inflexible because the information contained in the codes is static, again diminishing its usefulness in different contexts. If inappropriate information is selected for hard coding in the tag, it offers little or no benefit to other third parties trying to use or reuse the tag. Adams provides an excellent discussion of bar codes [33], including 2D barcodes [34].

2.1.2 RFID

In the last decade and a half, many industries have turned to traditional Radio Frequency Identification (RFID) technologies to overcome the data limitations and line of site restrictions attendant with 2D bar code technology. Though its use is still limited, RFID is being used in various commercial and industrial applications that range from the tracking of livestock to vehicles on the interstate [35].

The history of RFID actually goes back to the Second World War, when the British equipped their aircraft with transponders that were used to identify their aircraft [36]. This technology was dubbed identify friend or foe (IFF), and was the early precursor to the RFID developments in use today.

A typical RFID system includes a tag (or transponder), a reader (or interrogator), and a host system. The host system handles the information flow between the tag, reader and itself. The tag communicates to the reader over an air interface. Figure 2 shows examples of the various components of an RFID system.

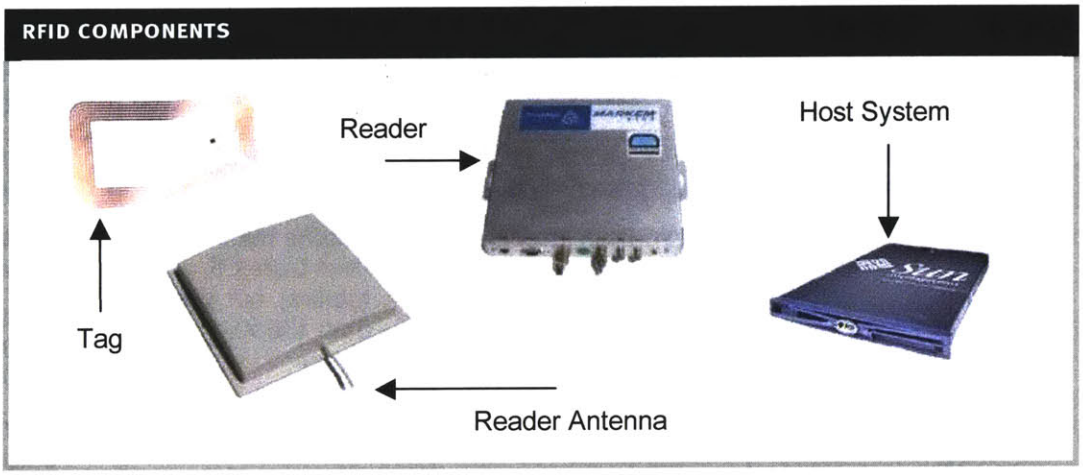


Figure 2 RFID Components

Tags can have onboard power, or derive power from the electromagnetic field emitted from the reader. They can also have active transmitters for communication to the reader. Tags are grouped in one of the following three categories shown in the following table [37].

TAG CATEGORIZATIONS

Tag Type	On-tag Power	Active Transmitter
Active	Yes	Yes
Passive	No	No
Semi-Passive	Yes	No

Table 1 Categorization of RFID Tags [37]

Commercially available passive RFID systems come in two main categories, near-field systems, that use inductive coupling of the tag to the reactive energy circulating around the antenna, and far-field systems, that couple to the real energy contained in propagating electromagnetic plane waves. Near-field systems operate in LF and HF bands with short reading distances, and far-field systems operate in UHF and microwave bands with longer read ranges [38, 39].

The following table shows the RFID frequencies in commercial use today with benefits and limitations [39, 40, 41].

RFID FREQUENCIES			
Frequency	**Benefits**	**Limitations**	**Common Uses**
Low frequency (less than 135 Kilohertz)	• Frequencies accepted worldwide • Works near metal • Least sensitive to liquids • Cheapest readers • In wide use	• Shortest read-range (less than 1.5 meters) • Slower tag read rate • Costliest tags	• Animal tracking • Beer keg tracking • Automobile anti-theft and key-and-lock systems
High frequency (13.56 Megahertz)	• Frequency accepted worldwide • Works in most environments • Less sensitive to liquids • Cheaper readers • In wide use	• Does not work near metal • Shorter read-range (less than 1.5 meters), but longer than 135 KHz range • Slower tag read rate • Costlier tags	• Library book tracking • Pallet/container tracking • Access control (buildings) • Airline baggage tracking
UHF (868 to 956 Megahertz)	• Longest read-range (more than 1.5 meters) • Faster tag read rate • Cheaper tags • Licensed for evaluation in Japan • Expanding commercial use	• More sensitive to liquids • Costlier readers	• Pallet and container tracking • Truck and trailer tracking
Microwave (2.45 Gigahertz)	• Longer read-range (more than 1.5 meters), but less than 915 MHz range. • Fastest tag read rate • Cheapest tags	• Frequency not licensed for commercial use in parts of Europe • Complex systems development • Most sensitive to liquids • Costliest readers	• Access control (vehicles)

Table 2 Commercial RFID Frequencies [39, 40, 41]

All of the frequencies listed in Table 2 experience detuning when the tags are in close physical proximity to varying degrees. As the table shows, the cost of

readers increases with increasing frequencies. Conversely the cost of tags decreases with increasing frequencies. This reflects current economics and does not take future volumes into consideration. The speed with which tags can be read tends to increase with increasing frequencies. As a point of reference, you can currently read about 200 tags/second at 13.56 MHz. The read range is more complex and dependent on the physics of the situation, and does not linearly relate to the frequency used. Of the frequencies listed in the table, the range tends to increase with frequency until you get to 2.45 GHz, which has a smaller range than 915 MHz. One final trend is the sensitivity to liquids or moist environments, which increases with the frequency used. These trends are simplifications of the true complexity involved. For example, read rates are not a simple function of frequency and range is affected by almost every component of an RFID system [39].

The following table shows some available frequencies with comments [41, 42].

RFID FREQUENCY RANGES	
Frequency Range	**Comment**
< 135 kHz	low frequency, inductive coupling
6.765 - 6.795 MHz	medium frequency (ISM), inductive coupling
7.400 - 8.800 MHz	medium frequency, used for EAS (electronic article surveillance) only
13.553 - 13.567 MHz	medium frequency (13.56 MHz, ISM), inductive coupling, wide spread usage for contactless smartcards (ISO 14443, MIFARE, LEGIC, ...), smartlabels (ISO 15693, Tag-It, I-Code, ...) and item management (ISO 18000-3).
26.957 - 27.283 MHz	medium frequency (ISM), inductive coupling, special applications only
433 MHz	UHF (ISM), backscatter coupling, rarely used for RFID
868 - 870 MHz	UHF (SRD), backscatter coupling, new frequency, systems under development (Europe only)
902 - 928 MHz	UHF (SRD), backscatter coupling, several systems (USA/Canada only)
950 - 956 MHz	UHF (SRD), backscatter coupling, systems under development (for evaluation in Japan only)
2.400 - 2.483 GHz	SHF (ISM), backscatter coupling, several systems, (vehicle identification: 2.446 .. 2.454 GHz)
5.725 .- 5.875 GHz	SHF (ISM), backscatter coupling, rarely used for RFID

Table 3 Frequency Ranges for RFID Systems [41, 42]

Scharfeld gives a more detailed description of the allowable field strength, distances and transmission power associated with the frequencies listed in Table 3, which vary in different parts of the world [37].

The traditional RFID industry has been moving to include more and more information on the tag itself. Co-locating the information with the physical object

has its benefits, and certainly solves the fundamental problem posed in this thesis, but it has two important drawbacks. First, like the 2D and 3D barcodes mentioned in the last section, the data tends to be static and useful only within a narrowly defined scope or context. For this reason, many commercial RFID systems in use today are closed systems that are tailored more for a specific company's needs than for general reuse throughout the supply chain. The second problem is the cost and read rate of the tag are both adversely affected when the tag carries more information. Higher costs and slower read rates are particularly problematic when large volumes of inexpensive goods must be processed quickly in industries like Consumer Packaged Goods (CPG) and postal operations.

A central tenant of Auto-ID minimalism is the development and use of low cost, passive tags. This will allow the recurring and incremental cost of each tag to be absorbed within the unit cost of the items being tagged [37]. To accomplish this, the Auto-ID system aims to limit the amount of data stored on the tag itself. Decoupling the data from the object is similar to the concept of pointers in software engineering. Traditional RFID tags, with their on board data storage, are like pass by value, passing all of their data. Auto-ID tags that contain only an EPC code are similar to pass by reference, where the data is stored off tag and simply referenced. I will revisit this topic later in the chapter when I discuss the components of the Auto-ID infrastructure.

In addition to cost, probably the biggest limitations of traditional RFID technologies are the lack of worldwide data standards and available frequencies [40]. The frequency problem is one that affects not only the current RFID industry, but also any future Auto-ID implementations, and the regulatory environment is something that requires more work [37]. The size and type of antenna used is directly related to the frequency used as well [37], and is an important consideration in the design of product packaging.

The lack of worldwide standards is something that the Auto-ID center is hoping to address with the introduction of low cost, EPC based tags, with the necessary components to retrieve data about an object. This solves the fundamental problem posed by this thesis, and allows flexibility by allowing context sensitive and non-static information to be associated with an object.

2.1.3 XML

The Extensible Markup Language (XML) [43, 44] is a World Wide Web Consortium (W3C) [45] proposed recommendation of a file format to easily and cheaply distribute electronic documents on the World Wide Web. One of the features of XML is that it is extensible, not set like the Hypertext Markup Language (HTML) [46], so users can define their own tags using Document Type Definitions (DTD) [47] or the newer XML Schemas [48]. XML documents are self-describing and contain the rules to which the data must conform. This supports structure, like objects, hierarchies, and references, but more importantly, it supports validation and well formedness [44, 49].

A derivative of the Standard Generalized Markup Language (SGML) [50], XML and other markup languages, like HTML, allow you to "markup" elements with tags. These tags facilitate the separation of form from content. The same document can then be parsed and presented in many different ways, formats and contexts, allowing for data reuse. One important difference between HTML and XML is XML is much stricter in it rules of use, and only supports a small subset of the markup rules provided by SGML. It has been said that XML has 80% of the functionality of SGML with 20% of the complexity [49]. Indeed, the SGML specification is over 155 pages, while the XML specification is a compact 35 pages. All of the optional features of SGML have been removed from XML, and this aids in parsing, validating and easily guaranteeing that a document is well formed.

In actual practice, however, the extensibility of XML proves to be a stumbling block. As many schemas are developed, the Internet community is inundated with varying and overly specialized industry standards. It is important for major standards bodies and organizations (like ANSI, the International Standards Organization (ISO) [51], and the UN) to set forth standards, and for the user community to adopt them. This is in fact exactly what is happening, and I will discuss several XML based standards later in the chapter (Simpl-eb, UBL).

Standards aside, XML is human readable, machine readable, and document ready. This makes it perfect for data interchange between computers, and in fact many applications are now based on it. A common architecture uses a web server as a data channel to connect two databases. XML is a common formatting tool for Web, EDI, and paper based documents [49].

Before leaving this topic, I would like to give a more formal definition of what it means for an XML document to be valid and well-formed.

A valid document:

- Refers to or includes an XML schema or DTD and follows all the formatting rules in that schema or DTD [44]

A well-formed document:

- Follows all of the XML syntax rules, but may not be valid [44]

A browser can accept a well-formed XML document that has already been validated by a server (there is no need to download the DTD or schema to revalidate).

2.1.4 Traditional EDI

In Chapter 1 I provided a brief introduction and history of traditional Electronic Data Interchange (EDI). Interestingly enough, like RFID, EDI also traces its roots to the late 1940's and early 50's. Combined with a VAN, EDI offers the content and the channel for Business-to-Business (B2B) communication. As previously mentioned, however, both the one-time setup and recurring costs of this infrastructure are prohibitive for most medium and small companies. This

also leads to a stove-piped Supply Chain that is very inflexible and incurs a very high switching cost when a link in the chain is replaced. Less than 80,000 of 6.3 million US businesses use EDI, and only 125,000 businesses worldwide [49]. For these reasons and others, the use of traditional EDI has peaked, and new users of Internet e-commerce will forgo EDI and use cheaper XML based platforms instead.

The recent proliferation of the Internet for commercial use has provided an infrastructure backbone that can replace the dedicated VAN. And with the emergence of extensible, XML based replacements for the EDI messages themselves, we have the opportunity to build a truly open standard that will replace both the channel and messages of traditional EDI, at a much lower cost. The opportunity for a true Supply Network also emerges by avoiding the stove piping mentioned above with EDI. As mentioned in Chapter 1, both major EDI platforms are moving to an XML based messaging scheme.

One disadvantage of XML based messaging over a traditional EDI message, however, is the XML markup adds a lot of overhead to the message. Please refer to the appendix, where examples of the same message encoded in both traditional EDI and various XML based schemas can be compared. These disadvantages aside, the extensible nature of XML based messaging allows for easy modification and extension of the messages to accommodate business needs.

Two common EDI messages are the EDI 214 and 856. The EDI 856 is the Advanced Shipping Notification (ASN), or Despatch Advice (DA). The DA includes quantity counts of the items being shipped and other information, including the Purchase Order (PO) and Bill of Lading (BOL). The BOL is a legal document that is shipped with the goods and is a manifest of the shipment.

The EDI 214 is the Shipment Status Message, and is sent by a transportation carrier to the company receiving a shipment on three different occasions:

- When the shipment is ready to be picked up
- When the shipment has been picked up
- When the shipment is expected to arrive

Like the EDI 856, the EDI 214 also includes the PO and BOL numbers.

There are many other defined EDI documents, but I will only draw upon the EDI 856 and 214 in this thesis.

2.1.5 Simpl-eb

As I mentioned in Chapter 1, Simpl-eb is a redesign of business methods, and, among other things, it defines the messaging standards for the business messages. The UCC uses a method called Global Standards Management Process (GSMP) in the development of their standards [52]. The GSMP uses a global network of end users to develop and validate the business models.

Rather than translating existing EDI messages into XML, the UCC began their modeling effort from scratch. They began by modeling the transactions necessary for business messaging by putting together a basic framework for a business transaction between a buyer and a seller, which they call Simpl-eb [53, 54]. The core sequence of messages includes: party introductions, exchange item request and price, place an order, despatch advice (DA), and invoice. The first two messages are grouped under alignment, and the last three under trade or commerce. Alignment includes the exchange and coordination of static data before any transaction to minimize the amount of information exchanged during the actual transaction. The DA includes minimal information about the quantities being shipped and date of movement to the receiver of that shipment. This message is kept simple by aligning the product catalogue data prior to the sending of the advice message. The DA then refers back to the more complete and previously aligned catalogue data. The following figure is a Unified Modeling Language (UML) [55] Sequence Diagram that puts all of these messages in context (UML modeling concepts will be used heavily throughout this thesis [56]).

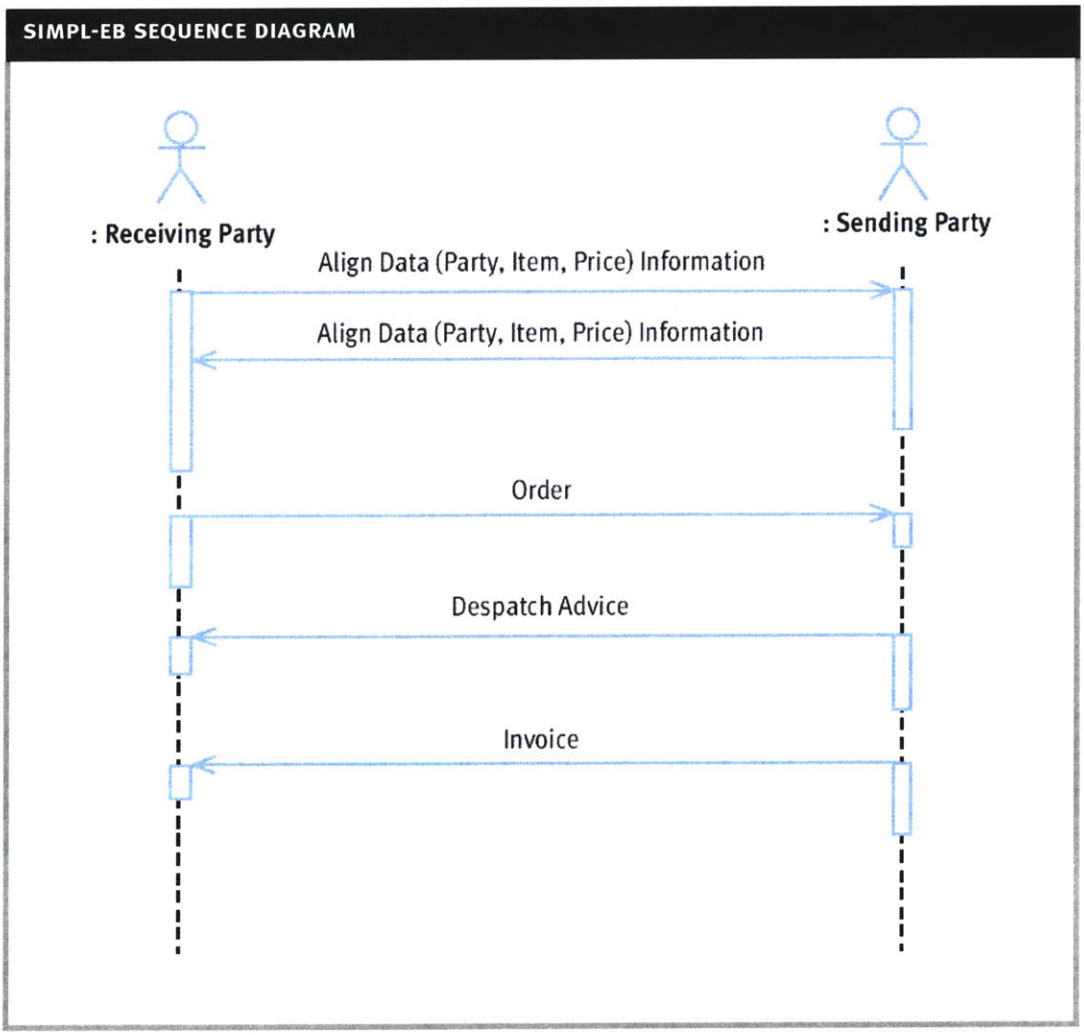


Figure 3 A Sequence Diagram of Simpl-eb Messages [54]

With the Simpl-eb framework defined, the UCC then set about gathering business requirements for the individual components. They captured these in Business UML (BUML) models. The BUML models are technology neutral, and don't specify or require any particular platform or technology. Once the BUML models are approved by the GSMP, they are passed to an Implementation UML (IUML) group, which is tasked with the technical details of implementing the business transaction. Figure 4 shows how the components of this development process fit together.

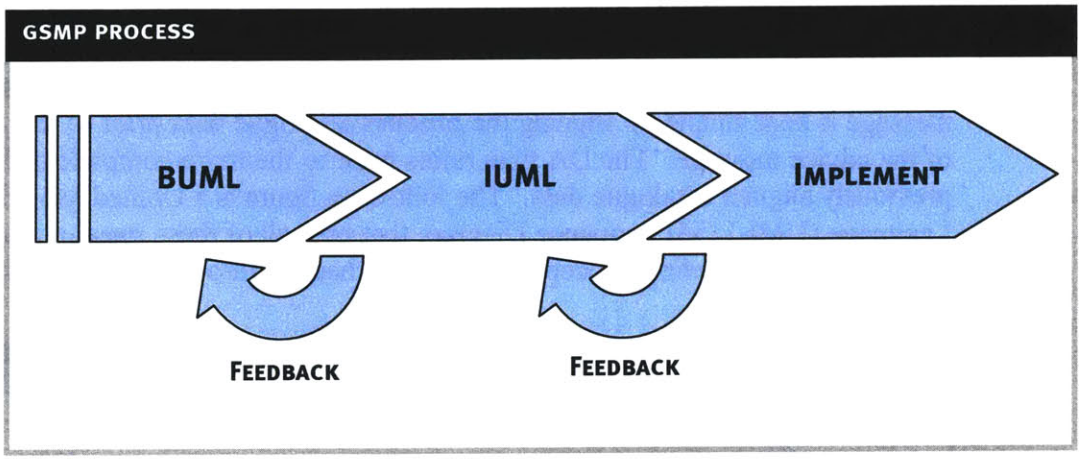


Figure 4 Overview of the GSMP Development Process [52]

The framework I present in this thesis will be based on the Simpl-eb model, and will rely on the fact that data alignment takes place prior to the placing of an order. I will also look at the Simpl-eb, XML-based messaging schemes and show how they work with the Auto-ID components [57].

2.2 Auto-ID Components

In the last section I gave a general overview of several Auto-ID related technologies, but in this section I will focus specifically on the technological contributions of the MIT Auto-ID Center itself.

Founded in 1999, the Auto-ID Center is a global partnership between over 100 companies and six of the world's leading research universities; the Massachusetts Institute of Technology in the US, the University of Cambridge in the UK, the University of Adelaide in Australia, Keio University in Japan, Fudan University in China, and the University of St. Gallen in Switzerland. Together they are designing, building, testing and deploying a global infrastructure that will make it possible for computers to identify any object anywhere in the world instantly. This network will provide the means to feed reliable, accurate, real-time information into existing business applications [22].

The critical elements of the new Auto-ID network include: the Electronic Product Code (EPC), a specification for cheap tags and cheap agile readers, the Object

Naming Service (ONS), the Physical Mark-up Language (PML) and Savant software technology [22].

I will describe these Auto-ID Center components in more detail in the following sections.

2.2.1 Tags

The tags are affixed to or manufactured into the product or its packaging and carry the unique EPC code for that object. The tags can fall under any of categorizations mentioned in Section 2.1.2 RFID. For a truly low cost, minimal solution, however, passive tags should be employed.

The following table shows the tag classifications.

TAG CLASSIFICATIONS		
Type	**Classification**	**Comments**
Passive	Class 0	Chips irreversibly pre-programmed with the EPC by the chip manufacturer
	Class 1	Chips programmable by the user in the field
	Class 2	Chips with full read/write capability, more memory, and more functionality
Semi Passive	Class 3	Battery enhanced for long range, more memory and more functionality
Active	Class 4	Battery enhanced with an active transmitter

Table 4 Tag Classifications

2.2.2 EPC

The Electronic Product Code (EPC) is the next generation UPC, and like the UPC, is divided into segments containing the version number, the domain manager, and the object class. But unlike the UPC, the EPC also has a field containing a unique serial number [27]. Standards for several EPC formats have been put forth ranging from 64 bits [58] to 256 bits [59]. Table 5 shows the bits assigned to the seven EPC versions available at the time of this thesis [60]. Each successive EPC representation is guaranteed to be backwards compatible [59]. (See the EPC Dot Notation I proposed in the appendix for more on the EPC code and its storage requirements).

EPC BIT ALLOCATIONS					
		Version Number	**Domain Manager**	**Object Class**	**Serial Number**
EPC-64	Type I	2	21	17	24
	Type II	2	15	13	34
	Type III	2	26	13	23
EPC-96	Type I	8	28	24	36

EPC BIT ALLOCATIONS					
EPC-256	Type I	8	32	56	192
	Type II	8	64	56	128
	Type III	8	128	56	64

Table 5 Bit Allocations for the Current EPC Versions [60]

2.2.3 ONS

The Object Name Service (ONS), based on the Internet's Domain Name Service (DNS), is a framework to locate networked services for tagged objects. Specifically, the networked services for an object can be identified based on the unique EPC tag on that object [61]. In other words, the ONS can take an EPC code and return a URI or URL for a server that can handle queries about that object.

Figure 5 shows the path of a typical ONS Query.

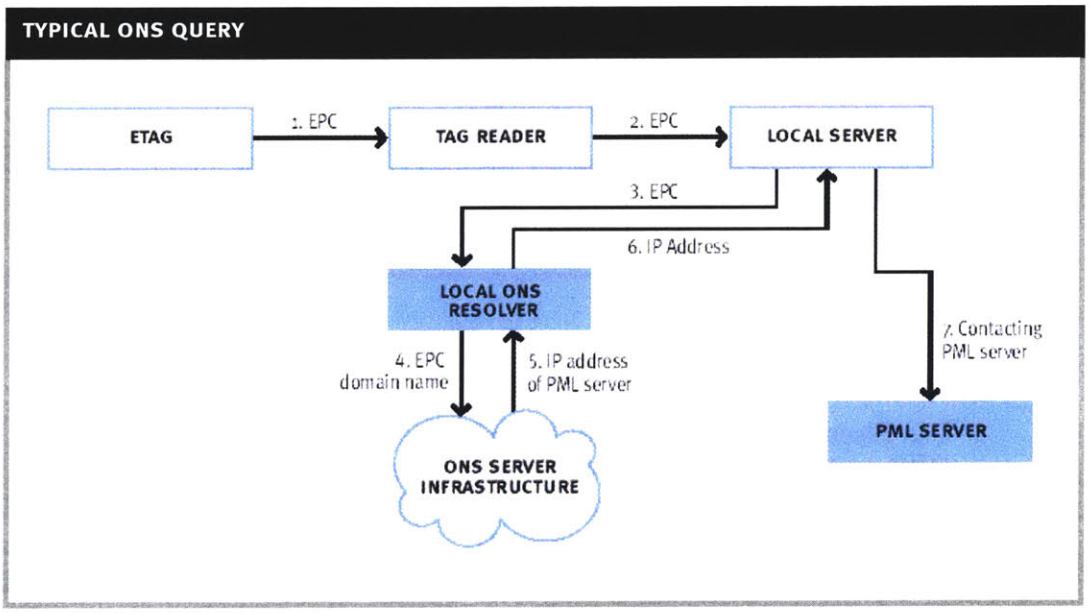


Figure 5 A Typical ONS Query [61]

The ONS is the key to removing the information from the tag and storing it somewhere else. It provides the bridge from the EPC to the services that provide the information associated to that EPC.

2.2.4 Savant

The Savant is a data router. It performs operations such as data capturing, data monitoring, data filtering, smoothing and transmission. The current reference implementation of the Savant consists of three modules [62]:

- Event Management System (EMS)
- Real-time In-memory Data structure (RIED)

- Task Management System (TMS)

The EMS connects readers to applications by managing the event flow generated by the reader. The RIED is an optimized, in-memory database that supports a subset of the Structured Query Language (SQL) used to query data from a database. The TMS coordinates processes initiated by higher level Savants [24]. Without the data smoothing, filtering and routing services of the Savant, the Internet backbone would be quickly flooded with reader-generated messages.

Figure 6 shows how the Savant can be deployed in a hierarchical fashion.

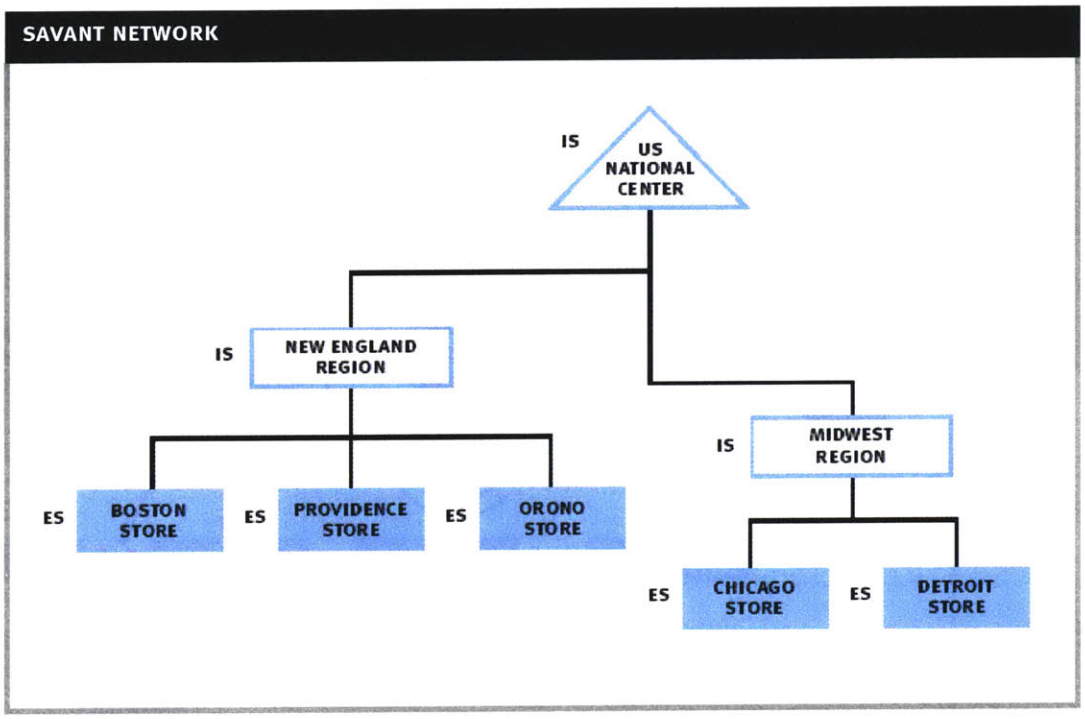


Figure 6 An Example Hierarchical Savant Network [62]

2.2.5 PML

The Physical Markup Language (PML) is the XML-based schema used to describe objects in the Auto-ID System [63]. PML will become the language of physical things. The aim of PML is to develop a standard representation that can be used by multiple industries and organizations throughout the life cycle of an individual EPC.

Recently, the development of PML was divided into two groups: PML Core, which handles the Auto-ID sensor events and represents only those messages generated by the Auto-ID tags, readers and Savant, and PML Extensions, which covers the business specific XML extensions [64].

PML Core includes time and physical measurements, as well as sensor observations and tags. (Please refer to the appendix for a UML Class diagram of the PML Core available at the time of this thesis).

2.2.6 PML Service

The PML Service, also referred to as PML Server, is responsible for serving up the PML data about an object, and is contacted after an ONS lookup on an EPC. The PML Service is currently a work in progress, and there are several custom but incomplete reference implementations available [65].

2.2.7 Overview

The following figure shows how the various components of the Auto-ID infrastructure fit together interact in the context of shipping and receiving.

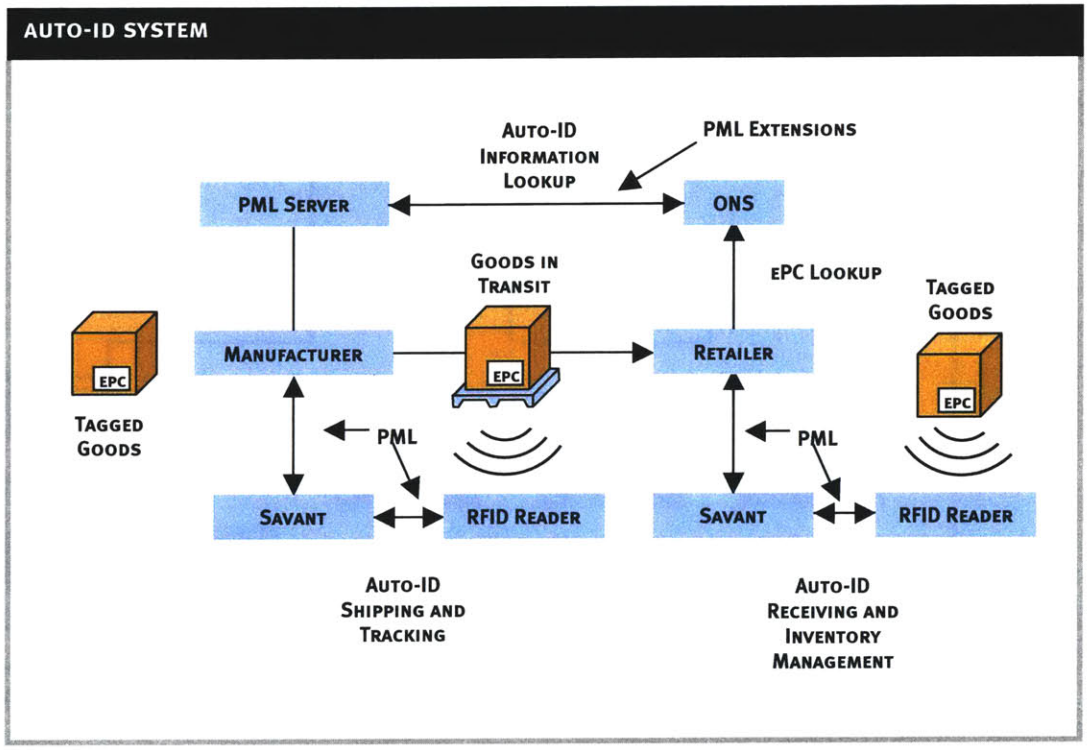


Figure 7 Auto-ID System Overview

2.3 Other Concepts

To close this chapter, I will briefly touch upon several topics related to the transmission of electronic information. There must be a message, a channel to convey the message, and a means of securing the transmission.

2.3.1 Content

There are currently many standards used for transmitting data between companies. EDI is a more established messaging standard, and it is compact by design. But problems abound, as previously mentioned, as there are many variations of the standard, and the standards are used in non-standard ways.

Recently, XML-based messaging schemes have come to the forefront including those proposed by the EAN.UCC and more recently with UBL. The XML tags add overhead to the payload, but they allow for the same information to be easily presented in many different ways using Cascading Style Sheets and XML Stylesheet Language (XSL) Transforms (XSLT). The ability for reuse should drive developers to carefully design schemas with unanticipated future uses in mind.

Table 6 shows a summary of some of the benefits and disadvantages of Simpl-eb, traditional EDI, and UBL. I've provided a more detailed discussion of the three schemes along with examples in the appendix.

MESSAGE SCHEME COMPARISON

Scheme	Benefits	Limitations
Simpl-eb	• Prior alignment of detail data results in minimal message content • Extensible and flexible • Same data easily represented in many different ways • Minor change to allow inclusion of EPC data	• No Receipt Advice specification • Large XML overhead • One of many specifications vying for acceptance • An error in the published schema with the omission of the quantity counts element
UBL	• Prior alignment of detail data results in minimal message content • Extensible and flexible • Same data easily represented in many different ways • Includes a Receipt Advice specification	• Large XML overhead • One of many specifications vying for acceptance • Changes necessary for EPC data • Overly restrictive about matching Despatch Advice and Receipt Advice line items
Traditional EDI	• Compact message (terse line qualifiers) • Venerable, and installed user base	• Includes some data that should be previously aligned • Expensive VAN and infrastructure requirements • Expensive to implement • Competing versions and standards • Inflexible, lengthy approval process • Changes necessary for EPC data • Limited future growth

Table 6 A Comparison of Messaging Schemes

A complete survey of current messaging standards is beyond the scope of this work. I will focus on the traditional EDI, and the XML-based EAN.UCC Simpl-eb and Sun's UBL. There are many good surveys of XML based standards, and the interested reader is referred to them [66, 67].

2.3.2 Channel

The EDI's VAN is a good example of a channel. As previously mentioned, setting up a dedicated VAN channel has the added benefit of built-in security, but that comes at a cost. The current industry charges a flat fee for the connection setup, and then per character transmitted thereafter. If this fee structure were to remain in place, transmitting serialized EPC data using a traditional EDI infrastructure would cost several orders of magnitude more than it does today.

This is one reason why many have looked to the Internet as a backbone channel to replace the dedicated connections. Abandoning the VAN is not without cost, however, as security over a public medium like the Internet must now be considered.

Many new Internet-based messaging channels are based on Java 2 Enterprise Edition (J2EE). J2EE technology and its component-based model simplify enterprise development and deployment. The J2EE platform manages the infrastructure and supports the Web services to enable development of secure, robust and interoperable business applications [68]. Figure 8 shows a high level view of the J2EE environment. The .NET technology from Microsoft addresses the same problems as J2EE, and the figure for a .NET-based infrastructure would be very similar to Figure 8 as well.

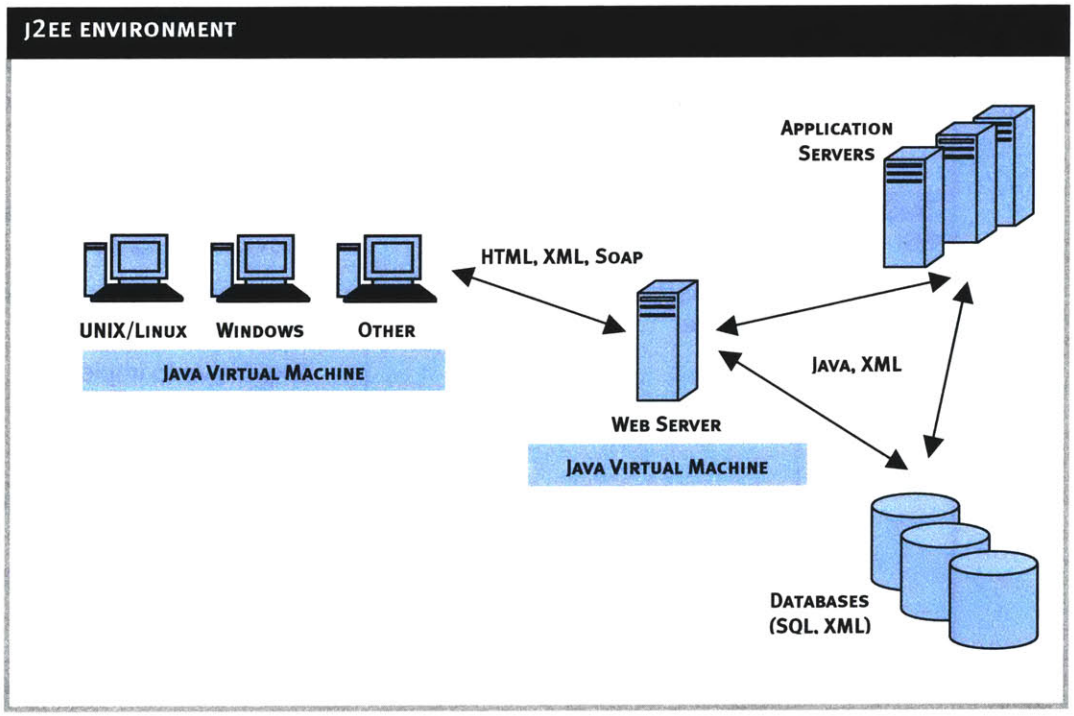


Figure 8 Overview of J2EE Environment (adapted from [49])

One component of the architecture shown in Figure 8 is an application server. There are many commercial application servers available today. Several of the leading application servers are listed in Table 7:

APPLICATION SERVERS	
Name	**Vendor**
WebSphere [69]	IBM
SunONE [70]	Sun Microsystems
WebLogic [71]	BEA
WebServer [72]	Borland
Oracle9i Application Server [73]	Oracle

Table 7 Some Major Application Servers

In addition to the ones listed in Table 7, the following also provide commercial application servers: ColdFusion, Delano, eXtend, Flash, JBoss, JRun, Orion, PowerTier, Pramati, Sybase, Tango, Total-e-server, Versata and others. An application server provides an environment that deploys compliant applications. For example, the J2EE based SunONE application server provides a platform to deploy J2EE compliant applications. Similarly, the .NET platform is appropriate for .NET based applications. An integration server works with an application server to provide a common API for legacy information systems, and is useful for integrating both new and legacy systems. The Auto-ID Center will not compete in the application/integration server market. An organization can leverage an appropriate and existing application or integration server.

The aim of these integration and application servers is to provide a reliable channel for communication. Some explicitly address security, while others provide security as an add-on. I'll return to the issue of security in the next section.

In addition to these application servers, there are a host of other software offerings like i2 [74], Manugistics [75], SAP [76] and others. These provide more than just the channel, however, and many are not currently compatible with serialized data schemes and processes.

Message wrappers are also part of the channel. One of the most popular is the Simple Object Access Protocol (SOAP) [77]. SOAP, which is basically XML and HTTP, doesn't introduce any new technologies for distributed computing other that what is currently being used. In other words, SOAP adds some header information to HTTP, with no other changes. This is accomplished by defining a new MIME type: text/xml. Because SOAP is text based, it is not efficient for applications running on a single machine, but it provides more than adequate performance for distributed applications that involve machine-to-machine communication [49]. Please see the appendix for a couple of examples of SOAP messages.

Microsoft's .NET is an alternative web services platform that is currently gaining momentum [78]. There are also a slew of web service technologies based on .NET. Along with these services, there are ways to discover them using Universal Description, Discovery and Integration (UDDI) of Web Services [79] and the Web Services Description Language (WSDL) [80]. UDDI and WSD are also compatible with J2EE based applications.

The following table shows a comparison of some elements of the J2EE and .NET platforms [49].

J2EE AND .NET		
Property	**J2EE**	**.NET**
Implementation	Standard	Product
Product Vendor	Many (Sun, IBM, etc)	Microsoft
Web Pages	Java Server Pages (JSP)	Active Server Pages
Server Components	Enterprise JavaBeans	.NET Components
Database Access	JDBC and SQL/J	ADO.NET (on ODBC)
Middleware	SOAP, UDDI, WSDL	SOAP, UDDI, WSDL
Primary Language	Java	C++, C#, VB
Portability	Across many vendors	Microsoft only

Table 8 A Comparison of J2EE and .NET

Other than the differences mentioned in Table 8, the actual differences are relatively small. Some of the secondary differences include [49]:

1. ASP.NET is independent of client device, and the same user interfaces will work on PCs, handheld devices, etc. JSP requires platform specific changes.

2. .NET Components are simpler than Enterprise JavaBeans. If you don't need additional features, .NET is simpler. Otherwise you have to build the extra features yourself, making your project more complex (such as nested transactions, state machines).

4. J2EE tools come from many vendors and don't interoperate as smoothly as .NET, which is all Microsoft based.

5. J2EE is more mature. .NET is new and represents a major change for some companies.

I will mention in passing the existence of industry-sponsored communities that are developing XML based standards like Transora [81]. Again, a complete survey of the available B2B offerings is beyond the scope of this work, but the concept of J2EE based integration or application servers is an important part of the framework that will be introduced in the next chapter, and the reader is reminded that there are other available options.

2.3.3 Security

The movement to the Internet as a messaging channel has raised the importance of security. By security I may mean authentication, confidentiality, and integrity [24]. If you don't somehow secure the message and/or channel, then any interested third party can intercept the message when a public infrastructure like the Internet is used.

There has been limited research into many of the aspects of security as it relates to the Auto-ID infrastructure [82]. There are also many offerings that allow one to securely send messages over the Internet [83, 84].

In this thesis, I will address the issue of securing the message channel, but not necessarily the rest of the Auto-ID infrastructure.

2.4 Conclusion

In this chapter I introduced all of the major components and concepts that will be combined to form the Framework for Auto-ID Enabled Business. In the next chapter I will pull all of these pieces together and describe the framework in detail.

CHAPTER 3: The Framework

The Auto-ID system addresses the problem of coupling physical objects with their associated information models. I this chapter I present a framework for Auto-ID enabled business that makes use of this coupling. To frame the context of this discussion, I will focus on the flow of goods and information between two organizations. This parallel flow of goods and data could be within a company, e.g. from manufacturing to packaging to warehousing, or between two companies, e.g. from manufacturer to distributor to retailer. In either case, the fundamental concepts of synchronizing goods and data are the same.

3.1 Introduction

The framework begins with a modeling exercise that culminates in requirements for implementation. The modeling can be broken down into 4 distinct phases.

3.1.1 Overview of Modeling

The first step in the modeling process is to study the existing business processes and generate a prioritized list of opportunities for Auto-ID. The most compelling option should then be selected for more careful examination. The next step is to bring together a team of knowledgeable domain experts and technical experts to model the chosen process in detail. The Unified Modeling Language (UML) is an ideal choice for capturing the details of these discussions. The UML diagrams should remain technology neutral so they remain accessible to all members of the team.

Once the existing system has been modeled, the third step is to again use UML modeling concepts to redesign a new Auto-ID based system with the approval of all participating team members. These UML diagrams can then be turned over to the technical team to generate detailed requirements documents against which the technical implementation will be carried out, which is the last step.

There are many UML diagrams available, including sequence diagrams, collaboration models, use-case diagrams, and class diagrams. There are even software development tools that take UML class diagrams and automatically generate schemas and business engines.

I will present many particularly useful UML diagrams in this chapter.

3.1.2 Shipping and Receiving

The framework is best illustrated using a concrete example. Throughout the remainder of this chapter I will focus on the process by which the composition of a pallet is checked by the staff at the receiving party's warehouse to verify that the pallet has been assembled according to the product specification or order. This process currently falls somewhere in the range from the fully manual (breaking

down mixed pallets for a hand count) to semi-automatic (manually scanning the SSCC's and any product codes). The EAN.UCC business message item standard can be used to specify and electronically convey the configuration of the pallet in a DA to the receiving party, which can then use the Auto-ID infrastructure to automatically detect the actual configuration. This will move the currently manually intensive validation process towards full automation.

In the broader context, the goal of this framework is to demonstrate the benefit of combining the static information contained in the EAN.UCC business messages with the dynamic information gathered by the Auto-ID infrastructure. The EAN.UCC business messages are used to describe the required configuration, whereas the Auto-ID data are used to automatically verify that the actual configuration agrees with the required configuration. There are benefits for both the receiving and sending parties, particularly as the use of mixed pallets increases, which are currently even more labor intensive.

3.2 UML Diagrams

Due to space limitations, I will only present the UML diagrams for the redesigned focus transaction using Auto-ID components. The following models represent step 3 of the modeling phase described in the previous section.

3.2.1 Actors

The first modeling step is to define the actors involved in the use case. The following actors will be used in the UML diagrams in this section:

A.1. Sending Party - An entity, like a manufacturer, that provides goods and a PML Service with information about traded goods via EPC lookups. Equipped with Auto-ID enabled readers, Savant and PML Server.

A.2. Receiving Party - An entity, like a retailer, that receives goods from the Sending Party and can initiate queries to the Sending Party's PML service. Equipped with Auto-ID enabled readers, and Savants.

A.3. Transporting Party - An entity, like a third party logistics carrier or dedicated fleet that moves the physical goods between the Sending Party and the Receiving Party.

A.4. ONS Host - MIT or another organization that is equipped with an Object Name Service.

3.2.2 Collaboration Diagrams

Recall the concept of Simpl-eb introduced in the last chapter. In this model, the data is aligned prior to the placing and fulfilling of an order. When an order is being fulfilled, the sending party sends a Despatch Advice ahead of the shipment with details about the contents of the shipment. The receiving party can begin tentative allocations of the inbound goods even before they arrive.

Figure 9 shows a UML collaboration diagram that illustrates the parallel flow of goods and information in the shipping process.

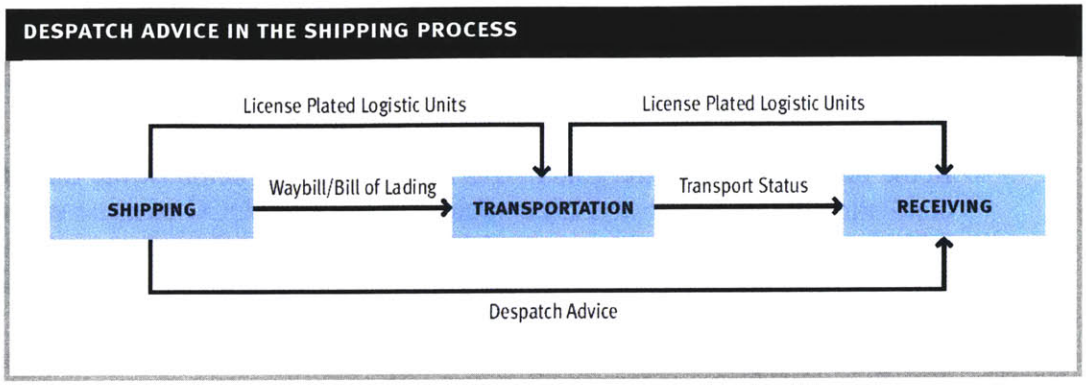


Figure 9 Despatch Advice Collaboration Diagram [26]

Whereas Figure 9 is a generalization of the shipping process, Figure 10 shows a particular instantiation of the direct process between a manufacturer and a retailer. Figure 11 shows how the same pattern repeats itself throughout the supply chain (Note that the distributor's upstream and downstream transactions are simply instances of the same general transaction shown in Figure 9).

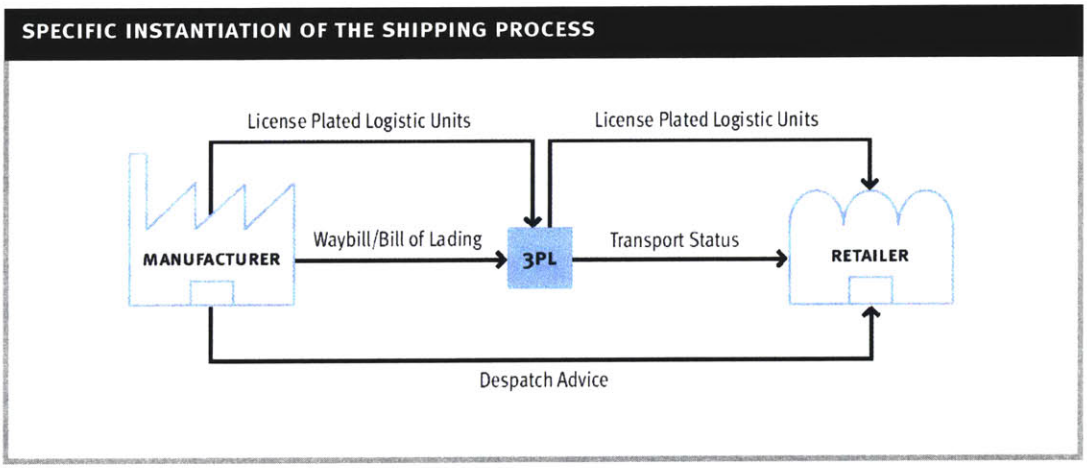


Figure 10 Instance of Despatch Advice Collaboration Model [26]

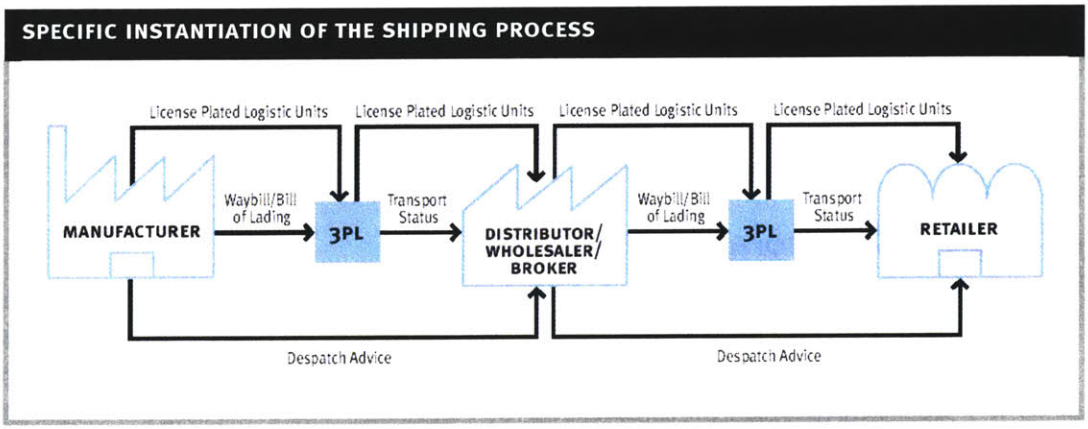


Figure 11 Collaboration Model Repeats in Supply Chain [26]

The challenges faced by the shipper are building and verifying a perfect order. Similarly, the challenge faced by the receiver is verifying that everything the shipper claims to have shipped was actually received.

3.2.3 Use-Case Diagram

Figure 12 contains a simplified Use-Case diagram, which shows how all of the actors involved in a shipping and receiving transaction can extend functionality from the Savant and Readers.

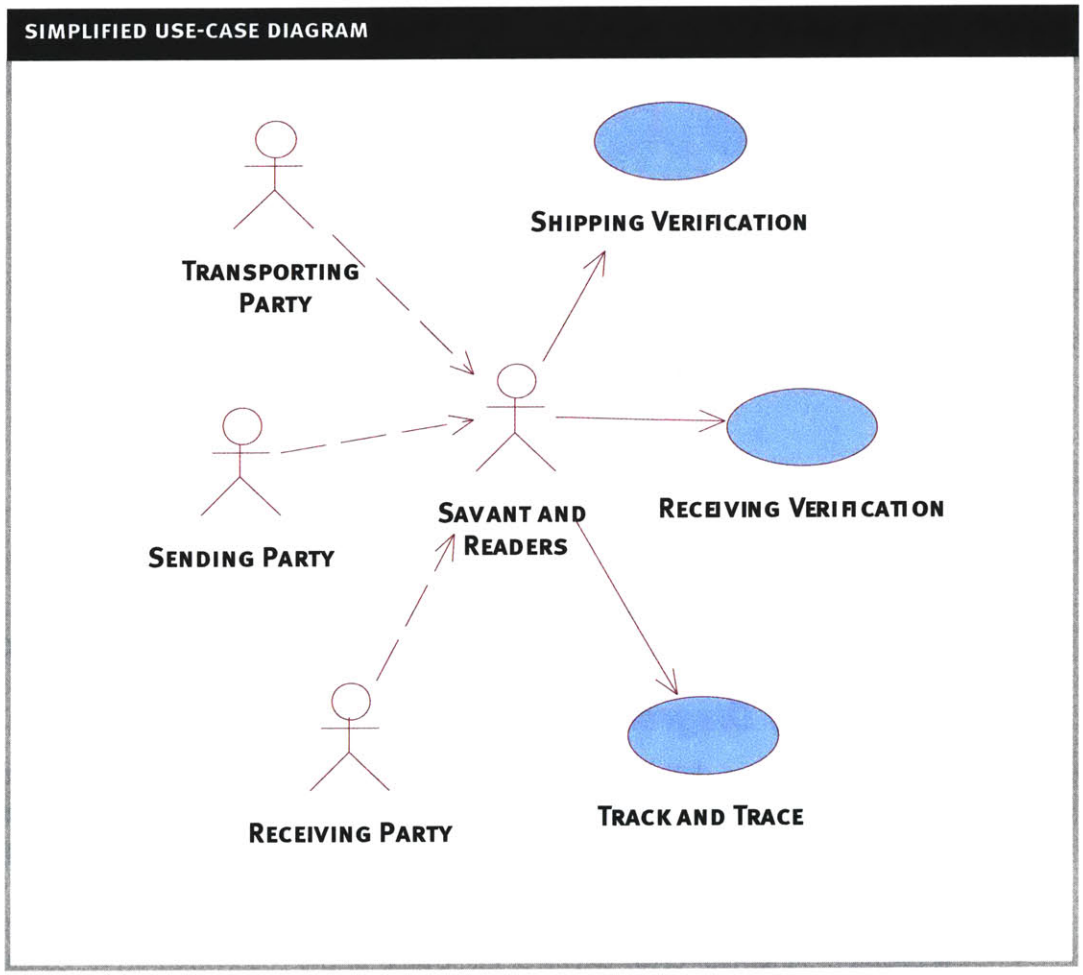


Figure 12 Simplified UML Use-Case Diagram with Savant and Readers

Figure 13 is a more detailed Use-Case diagram, which shows the activities for the sending party and the receiving party with the Auto-ID affected use cases in blue. Note that the error checking function has been abstracted and is included in the relevant Use Cases. The types of errors have also been generalized as shown.

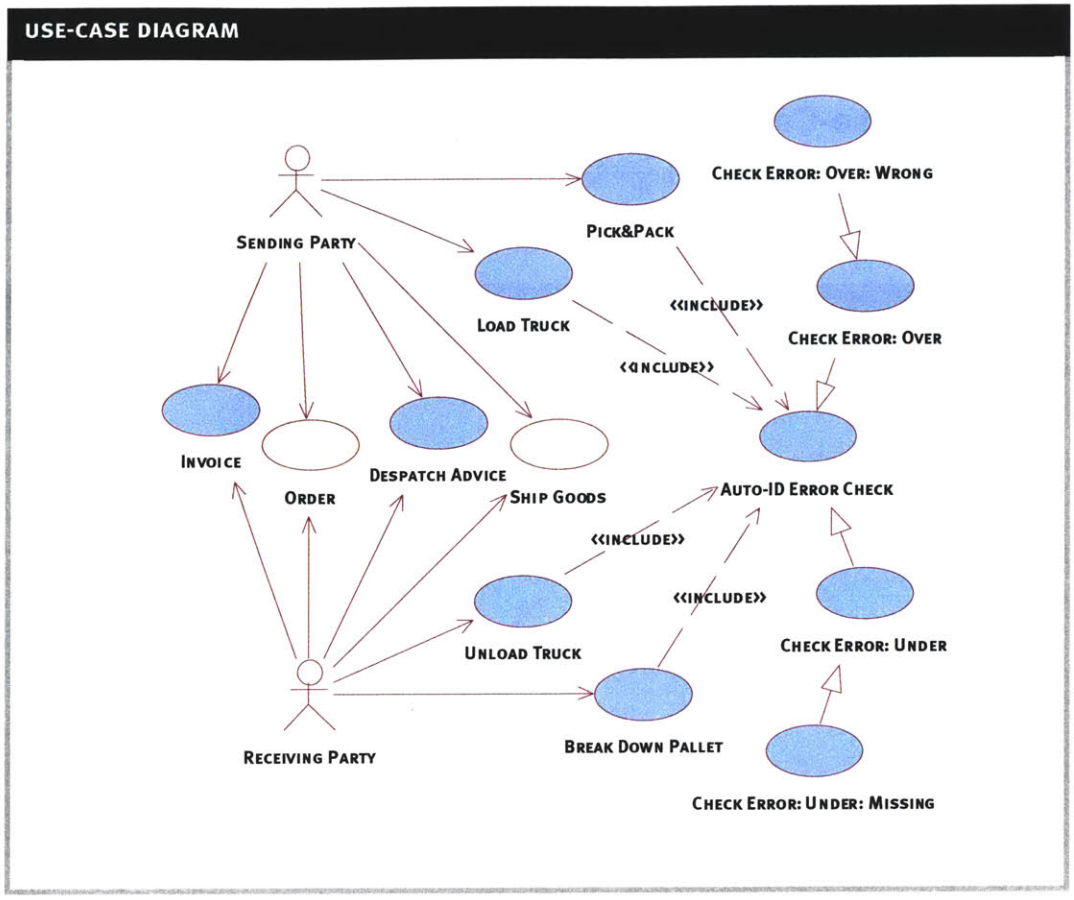


Figure 13 UML Use-Case Diagram with Error Checking

3.2.4 Activity Diagram

Figure 14 shows a detailed activity diagram for the entire order process with the Auto-ID affected activities highlighted in blue. The alignment operations are not represented, but the movement of goods parallel to the information messages is. The figure highlights the feedback loop in the Pick&Pack operation at the sending party. This Auto-ID enabled error check continues until the Pick&Pack operation is correct. A similar error check is illustrated at the receiving party. Of particular interest is how the receiving party's error checking will interface with their invoicing system. A confirmation message or Receipt Advice (RA) is sent immediately after the shipment is received and checked so the sending party can begin investigating any problems. The RA is sent in addition to the invoice, but in a timelier manner.

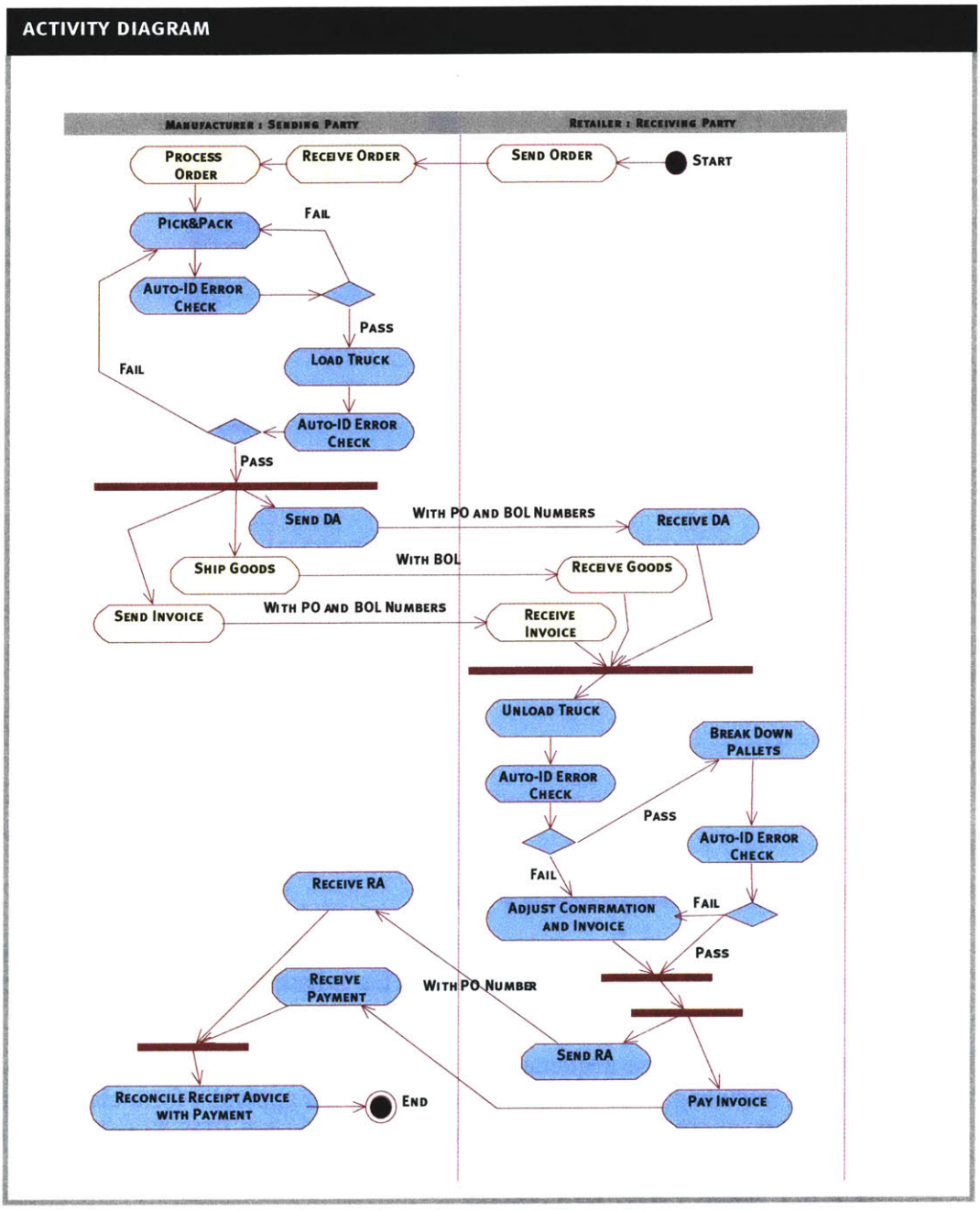


Figure 14 UML Activity Diagram with Error Checking

3.2.5 Deployment Diagram

Figure 15 shows the deployment diagram for the framework. Note that the ONS is first introduced here. The detailed sequence flows in the next section will show how all of these components come into play. The deployment diagram shows how the Readers and Savants can feed data into the BIS, which can automatically convert this data into the Despatch Advice or Receipt Advice.

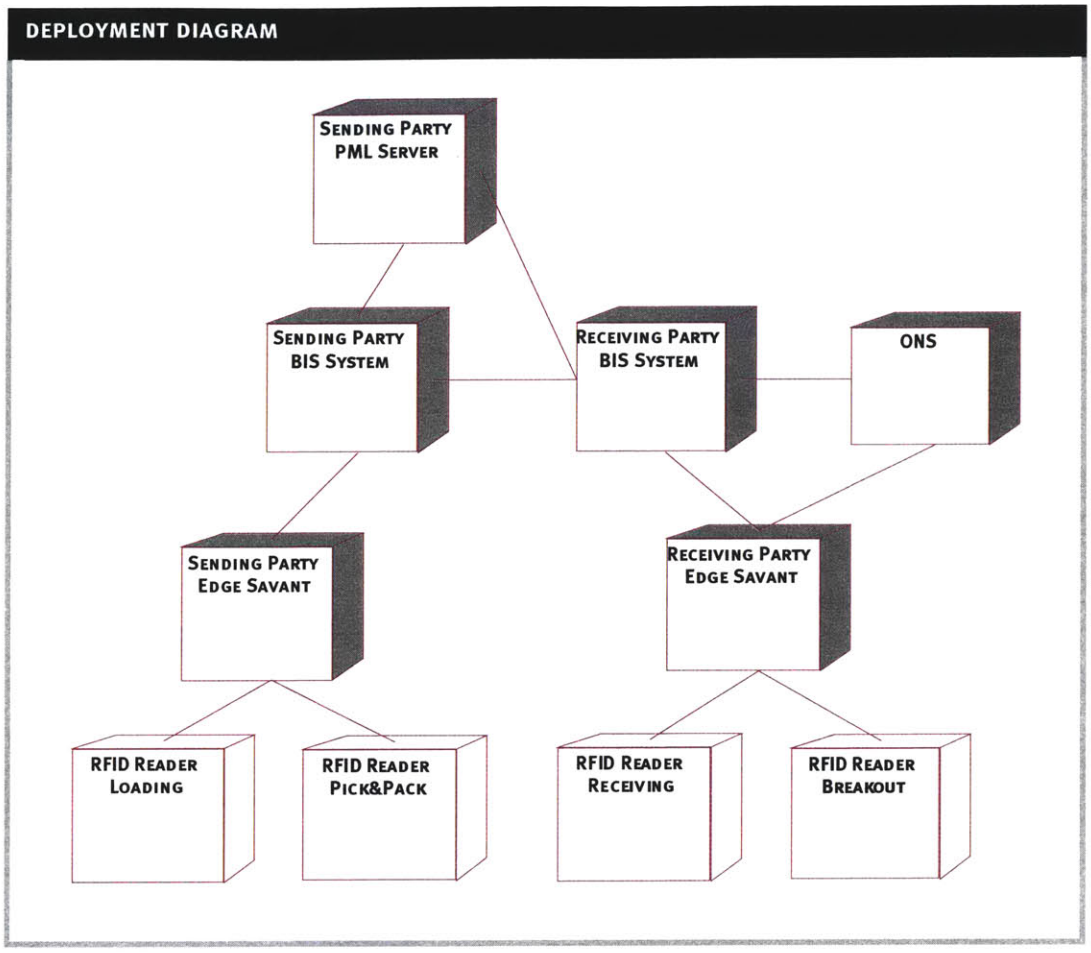


Figure 15 UML Deployment Diagram with ONS

3.2.6 Detailed Sequence Flows

In this section I will present detailed sequence flows that include the DA and RA for shipping and receiving verification. This is based on a format I developed for use in the Auto-ID Center for generating use cases [85, 86].

I have designed these sequences to exercise all major components of the Auto-ID system, including Core PML. Please refer to the appendix for a class diagram of Core PML as well as example message payloads using EDI, Simpl-eb and UBL.

3.2.6.1 Main Flow of Events

The main flow does not rely on pre-positioning EPC data using the traditional Despatch Advice (DA). This will mitigate unnecessary bandwidth utilization and data processing and storage requirements for the majority of transactions. However, the main flow does illustrate how Auto-ID will facilitate the preparation of perfect orders that exactly match the data contained in the DA.

M.1. Receiving Party

M.1.1. Send order to Sending Party

M.2. Sending Party

M.2.1. Receive order from Receiving Party

M.2.2. Process order

M.2.3. Fulfill order (build pallet/tote)

M.2.3.1. Scan goods while building pallet/tote

M.2.3.2. Verify when order complete or note missing items for DA

M.2.4. Scan goods before shipping

M.2.4.1. Verify order or note missing items for DA and BOL

M.2.4.2. Generate DA and BOL

M.2.4.2.1. Traditional EDI 856

M.2.4.2.1.1. Convert scanned EPCs to appropriate GTIN's and Line item numbers in EDI document to improve accuracy

M.2.4.2.2. Alternately use the Simpl-eb Despatch Advice

M.2.4.2.2.1. Using EPCs to improve accuracy

M.2.4.2.3. Alternately use the UBL Despatch Advice

M.2.4.2.3.1. Using EPCs to improve accuracy

M.2.4.3. Save the scanned EPC and PO data referenced by BOL Number

M.2.5. Send DA to Receiving Party including BOL Number

M.2.6. Ship pallet/tote to Receiving Party

M.3. Receiving Party

M.3.1. Receive DA from Sending Party

M.3.2. Process/parse DA data

M.3.3. Receive pallet/tote from Sending Party

M.3.4. Scan received pallet/tote

M.3.4.1. Aggregate carton counts

M.3.5. Reconcile received goods with DA

M.3.5.1. If ok

M.3.5.1.1. Flow goods into facility

M.3.5.1.2. Send Confirmation to Sending Party

M.3.5.2. If not ok

M.3.5.2.1. Handle one of two exceptions outlined below

M.4. Sending Party

M.4.1. Receive confirmation from Receiving Party

M.4.2. Close the transaction

3.2.6.2 Alternate Flow One: Something Missing (Under)

In this alternate flow, something comes up missing from the shipment when the Receiving Party receives it. This could be the result of shrinkage, damage, or an error in the DA data. This error condition is illustrated as an "under" in Figure 13. (Note: steps shown in *italics* are identical to the main flow.)

A1.1. *Receiving Party*

A1.1.1.*Send order to Sending Party*

A1.2. *Sending Party*

A1.2.1.*Receive order from Receiving Party*

A1.2.2.*Process order*

A1.2.3.*Fulfill order (build pallet/tote)*

A1.2.3.1. *Scan goods while building pallet/tote*

A1.2.3.2. *Verify when order complete or note missing items for DA*

A1.2.4.*Scan goods before shipping*

A1.2.4.1. *Verify order or note missing items for DA and BOL*

A1.2.4.2. *Generate DA and BOL*

A1.2.4.2.1.*Traditional EDI 856*

A1.2.4.2.1.1. *Convert scanned EPCs to appropriate GTIN's and Line item numbers in EDI document to improve accuracy*

A1.2.4.2.2.*Alternately use the Simpl-eb Despatch Advice*

A1.2.4.2.2.1. *Using EPCs to improve accuracy*

A1.2.4.2.3.*Alternately use UBL Despatch Advice*

A1.2.4.2.3.1. *Using EPCs to improve accuracy*

A1.2.4.3. *Save the scanned EPC and PO data referenced by BOL Number*

A1.2.5.*Send DA to Receiving Party including BOL Number*

A1.2.6. Remove one item from the pallet/tote (Simulate loss)

A1.2.7.*Ship pallet/tote to Receiving Party* (with removed item)

A1.3. *Receiving Party*

A1.3.1. *Receive DA from Sending Party*

A1.3.2. *Process/parse DA data*

A1.3.3. *Receive pallet/tote from Sending Party*

A1.3.4. *Scan received pallet/tote*

A1.3.4.1. *Aggregate carton counts*

A1.3.5. *Reconcile received goods with DA*

A1.3.5.1. Detect missing item

A1.3.5.2. Build Augmented DA/RA from scanned data

A1.3.5.3. Send Augmented DA/RA[1] to Sending Party

A1.3.5.3.1. Include BOL Number or original DA number

A1.4. Sending Party

A1.4.1. Receive the Augmented DA/RA from Receiving Party

A1.4.2. Process/parse Augmented DA/RA

A1.4.3. Retrieve the scanned EPC data based on the BOL Number sent from Receiving Party

A1.4.4. Reconcile previously scanned items with EPCs Receiving Party claims to have received

A1.4.4.1. Determine exactly what (if anything) is missing[2]

A1.4.5. Send an updated DA to Receiving Party

A1.4.6. Update internal billing systems (beyond scope of thesis)

A1.5. Receiving Party

A1.5.1. Receive the updated DA from Sending Party

A1.5.2. Reconcile with actual shipment

A1.5.3. Update home office systems (beyond scope of thesis)

Alternately, if this flow allowed the Receiving Party to request a DA from the Sending Party using the BOL Number, then the Receiving Party could begin polling for Augmented DA's as soon as they receive the shipment status message (EDI 214) from the carrier indicating the shipment is ready. This is possible because the Sending Party has already scanned and stored the EPC data for the PO before the goods are released to the Transporting Party, so the Receiving Party will know that the data is available. This would enable pre-positioning of the data in those situations where the Receiving Party might deem it necessary to enable

[1] Augmented DA/RA includes a detailed list of EPC's that Receiving Party has received

[2] Minutes after the exception occurs, not days or months

real-time decision-making systems[3]. I will not implement pull messaging in this thesis, but the reader is hereby notified that it is an available option.

3.2.6.3 Alternate Flow Two: Something Extra (Over)

In this alternate flow, something extra or unexpected shows up at the Receiving Party. This could be the result of erroneous case or item counts, promotional items, a substitution, an error in the DA data[4], or something altogether unexpected (no DA data at all). Auto-ID will enable the Receiving Party's receiving systems, with the aid of the dockworker, to quickly reconcile this discrepancy and flow the goods into the facility. This error condition is illustrated as an "over" in Figure 13. (Note: steps shown in italics are identical to the main flow.)

A2.1. *Receiving Party*

A2.1.1.*Send order to Sending Party*

A2.2. *Sending Party*

A2.2.1.*Receive order from Receiving Party*

A2.2.2.*Process order*

A2.2.3.*Fulfill order (build pallet/tote)*

A2.2.3.1. *Scan goods while building pallet/tote*

A2.2.3.2. *Verify when order complete or note missing items for DA*

A2.2.4.*Scan goods before shipping*

A2.2.4.1. *Verify order or note missing items for DA and BOL*

A2.2.4.2. *Generate DA and BOL*

A2.2.4.2.1.*Traditional EDI 856*

A2.2.4.2.1.1. *Convert scanned EPCs to appropriate GTIN's and Line item numbers in EDI document to improve accuracy*

A2.2.4.2.2.*Alternately use the Simpl-eb Despatch Advice*

A2.2.4.2.2.1. *Using EPCs to improve accuracy*

A2.2.4.2.3.*Alternately use UBL Despatch Advice*

A2.2.4.2.3.1. *Using EPCs to improve accuracy*

A2.2.4.3. *Save the scanned EPC and PO data referenced by BOL Number*

---

[3] Sending Party and Receiving Party will have to work out how long Sending Party should keep this data, and if Receiving Party wants to guarantee that the data be available for future use, they should pull it over to their internal systems while it is still available.

[4] Please note that following the steps outlined in the Main Flow will reduce the errors in the DA.

A2.2.5. *Send DA to Receiving Party including BOL Number*

A2.2.6. Add one item to the pallet/tote (Simulate something extra)

A2.2.7. *Ship pallet/tote to Receiving Party* (with additional item)

A2.3. *Receiving Party*

A2.3.1. *Receive DA from Sending Party*

A2.3.2. *Process/parse DA data*

A2.3.3. *Receive pallet/tote from Sending Party*

A2.3.4. *Scan received pallet/tote*

A2.3.4.1. *Aggregate carton counts*

A2.3.5. *Reconcile received goods with DA*[5]

A2.3.5.1. Detect additional item's EPC

A2.3.5.2. Perform an ONS lookup

A2.3.5.2.1. Use unexpected EPC

A2.3.5.2.2. Receive Sending Party's PML Server from ONS

A2.3.5.3. Send PML Server request to Sending Party

A2.3.5.3.1. Use unexpected EPC

A2.3.5.3.2. Request product information

A2.4. Sending Party

A2.4.1. Receive PML Server request from Receiving Party

A2.4.2. Retrieve the catalogue data based on EPC sent from Receiving Party

A2.4.3. Format a PML message

A2.4.3.1. Include product description

A2.4.3.2. Include product EPC

A2.4.3.3. Include product GTIN

A2.4.3.4. Include associated PO and BOL Number (if available)

A2.4.4. Send PML reply to Receiving Party

A2.5. Receiving Party

A2.5.1. Receive PML product data from Sending Party

---

[5] Note: This flow can also be used if the DA has not yet arrived, and there is no information about the inbound shipment. In this case *everything* is considered extra or over. Alternately steps A2.3.5.2-A2.4.3 can be replaced by the Receiving Party requesting an Augmented DA as mentioned at the end of the first Alternate Flow. The Augmented DA would also contain all of the (additional) EPCs for reconciliation purposes.

A2.5.2. Process/parse the PML data

A2.5.3. Display product data to receiving personnel

A2.5.3.1. If there is a PO and BOL Number

A2.5.3.1.1. Request an Augmented DA (beyond scope of thesis)

A2.5.3.1.2. Accept it into facility and information systems[6]

A2.5.3.2. If there is no PO or BOL Number

A2.5.3.2.1. Manually enter returned GTIN into PRD

A2.5.3.2.2. Flow into facility[7]

3.2.7 Summary

In this section I have presented UML models for a redesigned Shipping and Receiving process that makes use of Auto-ID components for automatic verification. The following table shows a summary of the capabilities for this redesigned system.

ENABLED CAPABILITIES	
Benefit	**Supporting Features**
Reduced labor costs at receiving party	Auto-ID infrastructure at the receiving party **automatically** identifies cases and pallet and checks pallet configuration against required configuration, no breakdown necessary
Reduced labor costs at sending party	Auto-ID infrastructure at the sending party **automatically** identifies cases and pallet and checks pallet configuration against required configuration as pallet is being built
Decrease number of "incorrect" shipments	Auto-ID infrastructure validates pallet configuration before a shipment leaves a location
Reduced gap between physical and system inventory levels	**Automatic** validation is less error-prone than manual checking, and much faster
Cost-efficient solution because of reliance on existing standards work	Using the EAN.UCC item business message specification guarantees interoperability between the two parties

Table 9 Summary of the Framework Enabled System's Capabilities

[6] Minutes after the exception occurs, not hours

[7] Minutes after the exception occurs, not hours or days

3.3 Framework Notes

By application of the framework, I was able to redesign a new system for Shipping and Receiving. Note that the system proposed in this chapter can use either an existing channel and VAN network with an existing or modified payload message, or it could short circuit the existing channel in lieu of an alternate channel, like the Internet. It is technology neutral, flexible, and can adapt as conditions and circumstances dictate.

Another important feature of the redesigned system, and indeed for any Auto-ID deployment, is the minimization of transferred data. This is achieved first by prior alignment of data as per Simpl-eb, which allows the DA and RA messages sent back and forth to be minimally terse. Secondly, and more importantly, the system's sequence has been explicitly designed to send minimal messages with entire lists of EPCs only sent to handle increasingly rare exceptions.

These are important considerations that should be kept in mind when using the framework.

3.4 Conclusion

In this chapter I presented a basic framework for analyzing Auto-ID enabled business with the transfer of goods and data. In the next chapter I will detail a reference implementation based on the Despatch Advice from Simpl-eb that illustrates how Auto-ID can be effectively used to solve the fundamental problem I've posed in this thesis, namely, the coupling of information with the physical world.

CHAPTER 4: Reference Implementation: Shipping and Receiving Verification

In this chapter I will cover the details of a reference implementation of the Shipping and Receiving Verification use case described in the previous chapter. While the models in the last chapter were technology neutral, this chapter shows an actual implementation and draws upon many specific technologies introduced in Chapter 2.

The implementation details presented in this chapter are state of the art, and among the best of breed for e-commerce systems, however, the reader should bear in mind that the Auto-ID Center neither recommends nor makes any claim of usability for any of the technologies and methods used in the reference implementation. The system is merely a proof of concept for which other, similar technologies could easily be substituted.

Roemer and Schoch's paper presenting a framework for the development of Auto-ID enabled application development is another good resource [87]

4.1 Introduction

During my research at the Auto-ID Center, I conceived and directed the a-Biz Phase Joint Project to exercise dataflow elements of the Shipping and Receiving Verification use case covered in Chapter 3. This chapter will cover a-Biz in detail, but before presenting a-Biz, let's reconsider the framework.

4.1.1 The Framework

What is the framework? It includes the steps and technologies that I've outlined thus far in this thesis, and culminates in a plan for execution. This chapter details how the a-Biz Joint Project implemented the process flows modeled in the Shipping and Receiving use case, and serves as a roadmap for similar future projects. In other words, it is the natural culmination and implementation of the framework. So before beginning, a quick review of the checklist used for a-Biz is in order.

4.1.2 Framework Checklist

The following summarizes the steps I took for the a-Biz Joint Project and can be used as a checklist for project execution against the framework presented in this thesis:

- Begin with a charter
    - An example is the charter from the Business Information and Industrial Control Action Group within the Auto-ID Center:

Charter

1. Focus on evolutionary applications, not revolutionary ones
2. Identify how Auto-ID fits into existing processes and systems
3. Identify use cases and provide rationale

- Decide on a broad category for study
- Research the available and emerging technologies used within the category
- Put together a project timeline
- Pull together a select few experts from category companies
    - Make sure the team includes appropriate representatives
        - For example, this research drew from:
            - Manufacturing
            - Logistics
            - Finance
            - Sales and Marketing
            - Others
- Have a team kickoff meeting
    - Review the charter
    - Brief the attendees on Auto-ID technology
    - Generate a list of potential applications with justifications
    - Prioritize the list
    - Create Use Case focus groups to look at the top 1 or 2
    - Determine what organizations should be represented in the use case focus groups
        - Companies affected
        - Relevant standards bodies or institutions that may have done work in this area
    - Recommend
        - The proposed use case focus group's composition
        - An action plan for moving forward
- After the kickoff meeting
    - Contact the companies identified for the use case focus group

    - Invite the targeted companies to meet with the use case focus group
        - Provide in advance a high level overview of the use case focus group's aims
        - Provide in advance the meeting notes from the first meeting
        - Follow up to verify attendance
        - Verify the skill set/domain expertise of attendees
    - At least one group member should possess UML modeling capability
    - At least one group member should be familiar with the technology to be used (e.g. XML and XML Schemas)
    - At least one group member should be intimately familiar with the Auto-ID technology
- Ongoing meetings
    - Review the charter
    - Conduct a detailed analysis of the use case assigned to the focus group
    - Evaluate which Auto-ID components come to bear
    - Develop a use case using Auto-ID
    - Capture all details with UML models [55]
        - Follow the format demonstrated by the a-Biz Framework [26]
        - The following UML diagrams are recommended
            - UML Sequence diagrams
            - UML Activity diagrams
            - UML Use Case diagrams
            - UML Deployment diagrams
            - UML Class diagrams
            - Any other appropriate UML diagrams
    - Gain consensus (GSMP model [52])
    - Should any XML work be necessary
        - Use the Core PML schemas from the Auto-ID Center [93]
        - Use the ebXML Core Components [88]
- Execute a pilot

- To determine the feasibility of the Auto-ID enabled solution
- Base it upon the models developed
- Deploy the necessary Auto-ID components
- Use a spiral approach to development [89]
    - The first implementation doesn't need to be gold-plated or robust, just proof of concept
- Document the findings
    - Something along the lines of this thesis

This process flow, which was followed by the a-Biz Joint Project, is based on an unpublished work I wrote entitled *Business and Industrial Control Action Group: Sub Group and Use Case Focus Group Methodology* [90]. This work led to the more detailed analysis recently published by the Auto-ID Center entitled *The development of Use Cases as an alternative approach to identify the impact of Auto-ID implementations on Business* [85]. I would refer anyone interested in pursuing similar projects to these documents as reference guides to get started.

4.2 a-Biz Demo Charter

The second phase of the a-Biz Joint Project involved setting up a desktop demo that would include all the major software and hardware components of the use case. This implementation focuses upon:

- Scanning goods upon shipment to verify order and bill of lading
- Scanning goods upon receipt to verify order and bill of lading.
- Reconciling shipped goods with orders.
- Interacting with a PML service to investigate goods received without documentation.
- Shipping an Augmented Despatch Advice to enumerate the EPCs of products shipped.
- Demonstrate the main flow and two alternate flows outlined in the Chapter 3.

This chapter outlines the details of this reference implementation demo.

4.3 Project Team

With a focus of shipping and receiving in mind, I recruited a manufacturer, retailer, a technology provider and a standards body from the Auto-ID community to provide expert domain knowledge. The following table shows the project participants and the role they fulfilled:

A-BIZ JOINT PROJECT TEAM	
Role	**Organization**
Manufacturer	Procter & Gamble Corporation
Retailer	Target Corporation
Technology Provider	Sun Microsystems
Standards Body	Uniform Code Council (UCC)
Project Management	MIT Auto-ID Center

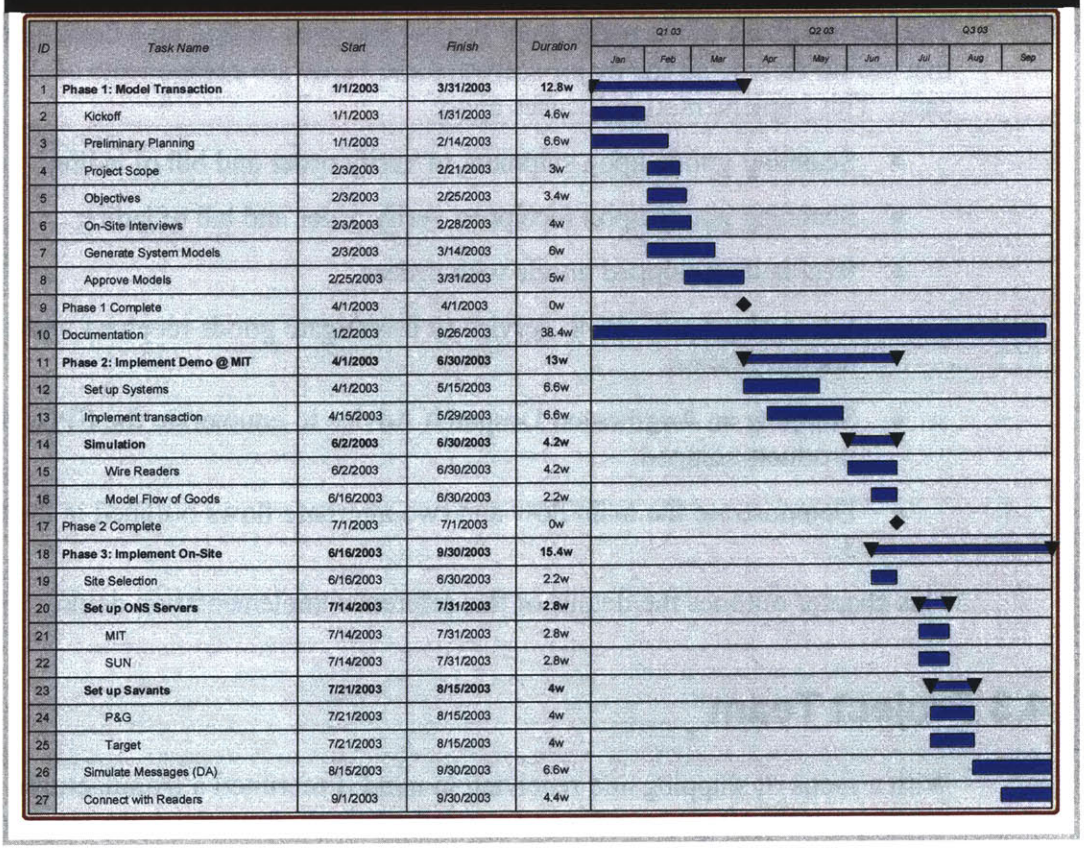
Table 10 a-Biz Joint Project Participating Sponsors

I contacted and brought together personnel from these companies to form the core modeling and execution team that met in bi-weekly meetings for the duration of the a-Biz Joint Project.

4.4 Project Timeline

The following Gantt chart shows the major and minor milestones and deliverables of the a-Biz Joint Project.

A-BIZ PROJECT TIMELINE



ID	Task Name	Start	Finish	Duration
1	**Phase 1: Model Transaction**	**1/1/2003**	**3/31/2003**	**12.8w**
2	Kickoff	1/1/2003	1/31/2003	4.6w
3	Preliminary Planning	1/1/2003	2/14/2003	6.6w
4	Project Scope	2/3/2003	2/21/2003	3w
5	Objectives	2/3/2003	2/25/2003	3.4w
6	On-Site Interviews	2/3/2003	2/28/2003	4w
7	Generate System Models	2/3/2003	3/14/2003	6w
8	Approve Models	2/25/2003	3/31/2003	5w
9	Phase 1 Complete	4/1/2003	4/1/2003	0w
10	Documentation	1/2/2003	9/26/2003	38.4w
11	**Phase 2: Implement Demo @ MIT**	**4/1/2003**	**6/30/2003**	**13w**
12	Set up Systems	4/1/2003	5/15/2003	6.6w
13	Implement transaction	4/15/2003	5/29/2003	6.6w
14	**Simulation**	**6/2/2003**	**6/30/2003**	**4.2w**
15	Wire Readers	6/2/2003	6/30/2003	4.2w
16	Model Flow of Goods	6/16/2003	6/30/2003	2.2w
17	Phase 2 Complete	7/1/2003	7/1/2003	0w
18	**Phase 3: Implement On-Site**	**6/16/2003**	**9/30/2003**	**15.4w**
19	Site Selection	6/16/2003	6/30/2003	2.2w
20	**Set up ONS Servers**	**7/14/2003**	**7/31/2003**	**2.8w**
21	MIT	7/14/2003	7/31/2003	2.8w
22	SUN	7/14/2003	7/31/2003	2.8w
23	**Set up Savants**	**7/21/2003**	**8/15/2003**	**4w**
24	P&G	7/21/2003	8/15/2003	4w
25	Target	7/21/2003	8/15/2003	4w
26	Simulate Messages (DA)	8/15/2003	9/30/2003	6.6w
27	Connect with Readers	9/1/2003	9/30/2003	4.4w

Figure 16 Major and Minor Milestones of the a-Biz Project

4.5 Implementation Architecture

The intention of the implementation was to explore all the core interactions of each of the software components within a controlled, but accurate environment. The a-Biz team employed a representative distribution of hardware and software components to demonstrate the architecture.

4.5.1 Software

4.5.1.1 Savant

We used Savant version 1.1 with a customized event management system processing units. This Savant is deployed into the SunONE Application Server 7 on Solaris 9.

SunONE Application Server is a J2EE 1.3 compatible and certified technology. The Application Server 7 Platform Edition is offered free of license costs for development and deployment, with support available at an additional charge. It is built for high performance, leveraging proven HTTP and JMS engines.

4.5.1.1.1 Reader Adapter

We also used ThingMagic readers manufactured by Markem. The MIT Auto-ID Center provided an appropriate reader adapter from Oat Systems for the ThingMagic readers. The reader adapter connects to the readers via a TCP/IP interface, and has three tunable parameters – poll_time, monitor_poll_time, and timeout.

4.5.1.1.2 Loggers

SoapLogger

A Core PML message contains a reader identity, an observation, and a set of tag identifiers. The logger's interface from the Savant system, however, reports tags one at a time. The transaction engine expected groups of tags in a Core PML message, so the individual readings needed to be gathered together and batched. A custom logger called "SoapLogger" was written and added to the Savant to package together and batch the observed tags. The logger gathers all tags observed since the last pass and reformats them into a CorePML message, which is sent to the PML Service.

The SoapLogger has one tunable parameter, endpoint, which defines the SOAP/HTTP endpoint to receive the CorePML messages. The source code is reproduced in the appendix.

4.5.1.1.3 Filters

EnterOnlyFilter

Since products may be sitting on the reader for an extended period of time during the scanning operations, an EnterOnly filter was required. This filter only reports the EPC the fist time it is seen in the reader field. For the EPC to be reported

again, it must be absent from the field for a given number of passes, configurable by the number_of_passes parameter. The source code is reproduced in the appendix.

PassSmoothingFilter

Since some readers will constantly report each tag seen and sometime miss a tag during a pass, the PassSmoothingFilter was created. It's purpose is to take a number of passes, tunable by the number_of_passes parameter, and normalize all the tags reported during that period into one pass. The source code is reproduced in the appendix.

4.5.1.1.4 EMS.conf

The Event Management System configuration file is used by the Savant to set up the data routing and logging services of the Savant at startup. The source code for all of the filters and loggers is reproduced in the appendix as well as the Savant EMS.conf that initializes the data flows through the Savant.

4.5.1.1.5 PostgreSQL Database

The Savants were equipped with a PostgreSQL database, version 7.2.3 [91]. This is an open source SQL database.

4.5.1.2 ONS

ONS is the Object Name Service used to resolve a PML Service address from an EPC. The ONS is used when an "unknown items received" event occurs during alternate flow of the use case. In this instance, the receiving party receives something that does not have an order or bill of lading associated with it, so a query must be made to an ONS to determine the appropriate PML Service with data about the item.

Because I was not able to secure properly encoded tags, we did not use the ONS in the current implementation. Instead, the EPCs were related to products using a hash table.

4.5.1.3 PML Service

The PML Service accepts Core PML messages from Savant about the sensor reads. It accepted the Core PML messages and persisted them for later use.

As there is no formal PML Service Specification from the Auto-ID Center at the time of this writing, a tool from GorillaLogic [92] was used for the PML Service. The GorillaLogic platform is deployed in the backend systems inside the SunONE Application Server 7 as a J2EE application.

4.5.1.4 Order Processing System

To simulate the order processing system, we used another tool from GorillaLogic that automatically generates a working system from a UML model. Using the interfaces generated from the GorillaLogic platform, we are able to enter orders

via a web browser. The orders can contain line items with products and product counts.

4.5.1.5 Business Information System

The BIS serializes the data into an XML format designed to be compatible with the Simpl-eb message format. Thus, between the customer and supplier, only Simpl-eb messages are sent for the Order and Despatch Advice. This uses HTTP/Soap at the moment. By integrating with the SunONE Integration Server B2B platform including the SunONE Secure Trading Agent, the transport system can be easily secured. Similarly, the messaging format can be easily changed between Simpl-eb, UBL, EDI and other XML-based schemas.

4.5.2 Hardware

4.5.2.1 Savant Servers

Simple, low cost SunFire X1 servers are deployed in the warehouse location to act as Auto-ID Savant hosts. The throughput in this instance is fairly small, so any low-end server would be adequate.

The X1 has two Ethernet cards, and one was configured on the front end to be connected to the Internet. Routing was turned off, and the second Ethernet card was configured on the back end to connect directly to the reader with a crossover cable. The IP address for the reader on the backend was 10.0.0.101 (the default), and the X1 was 10.0.0.100.

Easier administration and recovery is provided through Sun's Lights-Out Management (LOM) and a System Configuration Card (SCC). LOM provides monitoring of the system and diagnosis, even when Solaris is not running and the system is in stand-by mode. The SCC enables quick replacement of a faulty system by transferring the system identity to ensure rapid recovery.

LOM lets system administrators remotely monitor and manage the Sun Fire V120 server. Automatic Server Restart provides system and component monitoring and initiates an automatic restart without intervention if the system or application freezes. Together, LOM and Automatic Server Restart software maximizes system availability and manageability.

4.5.2.2 Back End Servers

For the Back End portion of the architecture, which would be located at the home office, slightly larger machines were chosen. We used Netra 20's from Sun, which are part of Sun's entry-level server line.

4.5.2.3 Markem/ThingMagic Readers

We used Markem 6100 readers from ThingMagic. These readers have both a serial and an Ethernet interface. The readers are not dual frequency, and operate only in the UHF band (915 MHz). These are the same readers available in the Auto-ID evaluation kits [22]. For worldwide demos, multi-frequency agile

readers would be preferable. Note: Tyco/ThingMagic readers could also be used (with the same reader adapter) to demonstrate interoperability.

4.5.2.4 Rafsec S Tags

We used Class 0 Rafsec S tags in the demo. I chose this tag because the S configuration of the antenna makes the tag less sensitive to orientation issues. Unfortunately, however, the Class 0 nature the tags meant they were "locked", and the EPC number was already set and could not be changed. Unlocked tags are desirable so that the domain manager and object class can be properly encoded. This will allow us to test the algorithmic conversion of EPC to GTIN, which is currently being handled by a hash table. Note: iCode tags could also be used to demonstrate interoperability.

4.5.2.5 Cushcraft Antennas

We chose right hand circular polarized Cushcraft Antennas (model S9028PC12NF) for the UHF band (902-928 MHz). These have nice range capabilities, but are not a viable option for worldwide deployment (see RFID in Chapter 2).

4.5.2.6 RPCs

I acquired Reusable Pallet Containers from CHEP to hold the products during the inbound and outbound scans.

4.6 Messages

This section gives a brief overview of the messages I used, and how I modified them to include lists of EPCs.

4.6.1 CorePML

The Core PML messages are used to tell the PML Service when an EPC is observed. The message contains a reader EPC, a date time stamp, and a list of observed EPCs [93].

Here is an example CorePML message:

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlc:Sensor
xmlns:pmlc="urn:autoidcenter:pml:Core:1.0:0.50"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:autoidcenter:pml:Core:1.0:0.50
 PML_Library_Core.xsd">
   <ID>01.0000A89.035F23.000169DC1</ID>
   <Observation>
      <ID>000345869</ID>
      <DateTime>2003-05-19T13:04:34-06:00</DateTime>
      <Tag>
         <ID>01.0000A89.00016F.000169DC1</ID>
      </Tag>
```

```
            <Tag>
               <ID>01.0000A89.00016F.000169DC3</ID>
            </Tag>
            <Tag>
               <ID>01.0000A89.00016F.000169DC4</ID>
            </Tag>
         </Observation>
      </pmlc:Sensor>
```

4.6.2 Order

An EAN.UCC Order message initiates an order between a customer and a supplier. The EAN.UCC schemas for Order and all other messages are available from the UCC Solution Center [94].

Within the a-Biz architecture, the order contains GTIN numbers and counts for the ordered items. The following is an XML fragment that illustrates how I used the "requestedQuantity" element to indicate desired order quantities (the complete example can be found in the Sample Messages section of the Appendix under Simpl-eb: Order):

```
<order>
   ...
   <!-- Pampers Bibsters-->
   <lineItem>
      <number>0</number>
      <requestedQuantity>6</requestedQuantity>
      <itemIdentification>
         <gtin>037000401803</gtin>
      </itemIdentification>
   </lineItem>
   <!-- Bounce -->
   <lineItem>
      <number>0</number>
      <requestedQuantity>6</requestedQuantity>
      <itemIdentification>
         <gtin>037000801689</gtin>
      </itemIdentification>
   </lineItem>
   ...
</order>
```

4.6.3 Despatch Advice - Standard

The EAN.UCC Despatch Advice message is sent from the supplier to the customer to indicate the goods that are being shipped for a given order [94]. The following is an XML fragment which shows how I used the "quantityContained" element to indicate scanned quantities (the complete example can be found in the Sample Messages section of the Appendix under Simpl-eb Despatch Advice - Standard):

```
<despatchItem>
   <!-- Pampers Bibsters-->
   <itemsContained>
      <containedItemID>037000401803</containedItemID>
```

```
            <quantityContained>2</quantityContained>
        </itemsContained>
        <!-- Bounce -->
        <itemsContained>
            <containedItemID>037000801689</containedItemID>
            <quantityContained>1</quantityContained>
        </itemsContained>
    </despatchItem>
```

4.6.4 Despatch Advice - Augmented

The following is an XML fragment from an EAN.UCC Despatch Advice. I had to slightly modify the schema for the Despatch Advice to facilitate EPC data. The example shows how the modified "serial" element is used to transmit a list of scanned EPCs (the complete example can be found in the Sample Messages section of the Appendix under Simpl-eb Despatch Advice - Augmented):

```
<despatchItem xsi:type="eanucc:LogisticUnitsType"
 number="1" id="100370000014190998">
   <!-- Pampers Bibsters-->
   <itemsContained>
      <containedItemID>
          037000401803
      </containedItemID>
      <listForEachItem>
         <serial>
            01.0000A89.00016F.000169DC1
         </serial>
      </listForEachItem>
      <listForEachItem>
         <serial>
            01.0000A89.00016F.000169DC2
         </serial>
      </listForEachItem>
   </itemsContained>
   <!-- Bounce -->
   <itemsContained>
      <containedItemID>
         037000801689
      </containedItemID>
      <listForEachItem>
         <serial>
            01.0000A89.003014.000169AB1
         </serial>
      </listForEachItem>
   </itemsContained>
</despatchItem>
```

4.6.5 Receipt Advice

Unfortunately, the EAN.UCC does not have a Receipt Advice message. An Augmented Despatch Advice message will be used for this purpose. With slight modification to the DA, the Augmented DA can also include lists of EPCs for reconciliation purposes.

4.7 a-Biz Implementation

4.7.1 Highlights

The following few figures show the completed a-Biz Joint Project Demo. The various systems representing the Savants and home office Business Information Systems can be seen along with the readers and tagged products.

All of the a-Biz systems are shown in the same room in Figure 17, but I exhibited the demo in a distributed fashion during the June 2003 Board Meetings of the Auto-ID Center. To do this I left the BIS hosts in Cambridge, MA, and carried the readers and Savants with me to Zurich, Switzerland. I reconfigured the system so the Savants would log PML Core messages via the Internet back across the Atlantic Ocean. The BIS servers then handled and rerouted the messages between companies. This deployment is very similar to an actual field deployment of these systems, and the distributed framework worked exceptionally well.

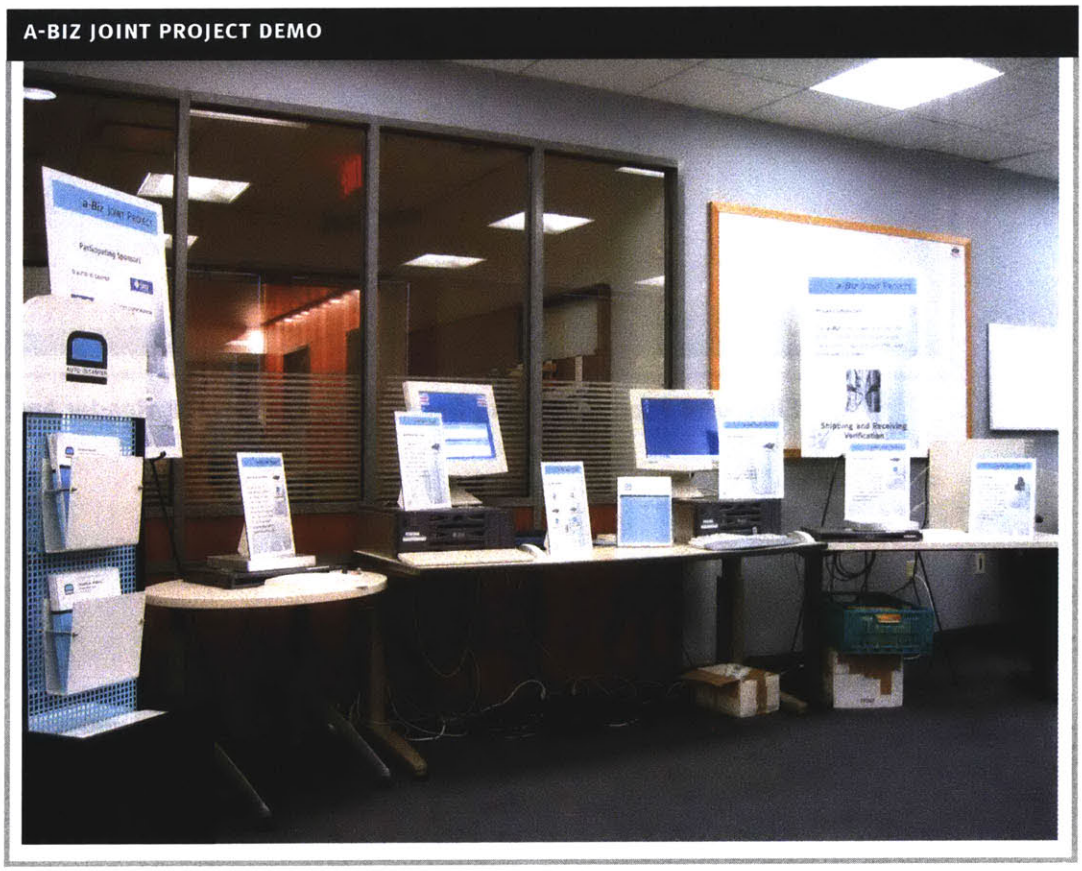


Figure 17 The a-Biz Joint Project Demo

Figure 18 shows several tagged products in the CHEP pallet container, which is sitting on top of the reader and antenna during a scan. I set up and used the systems and products shown here in Zurich.

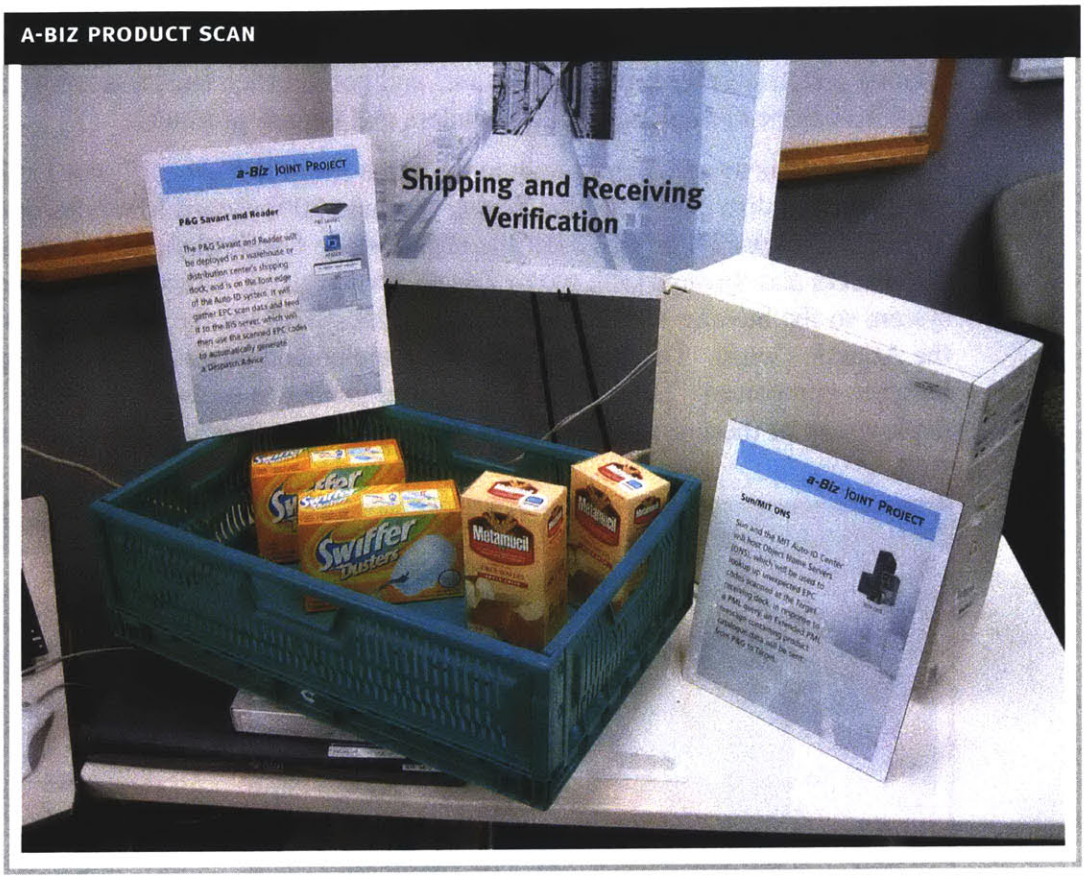


Figure 18 Products Being Scanned by the Savant, Reader and Antenna

Figure 19 shows the BIS Server representing P&G. A user can point a web browser to this server and enter orders and process Despatch Advice and Receipt Advice messages that are automatically generated from the Savant data. A similar setup exists for the Target BIS. I configured and left these in Cambridge, Massachusetts.

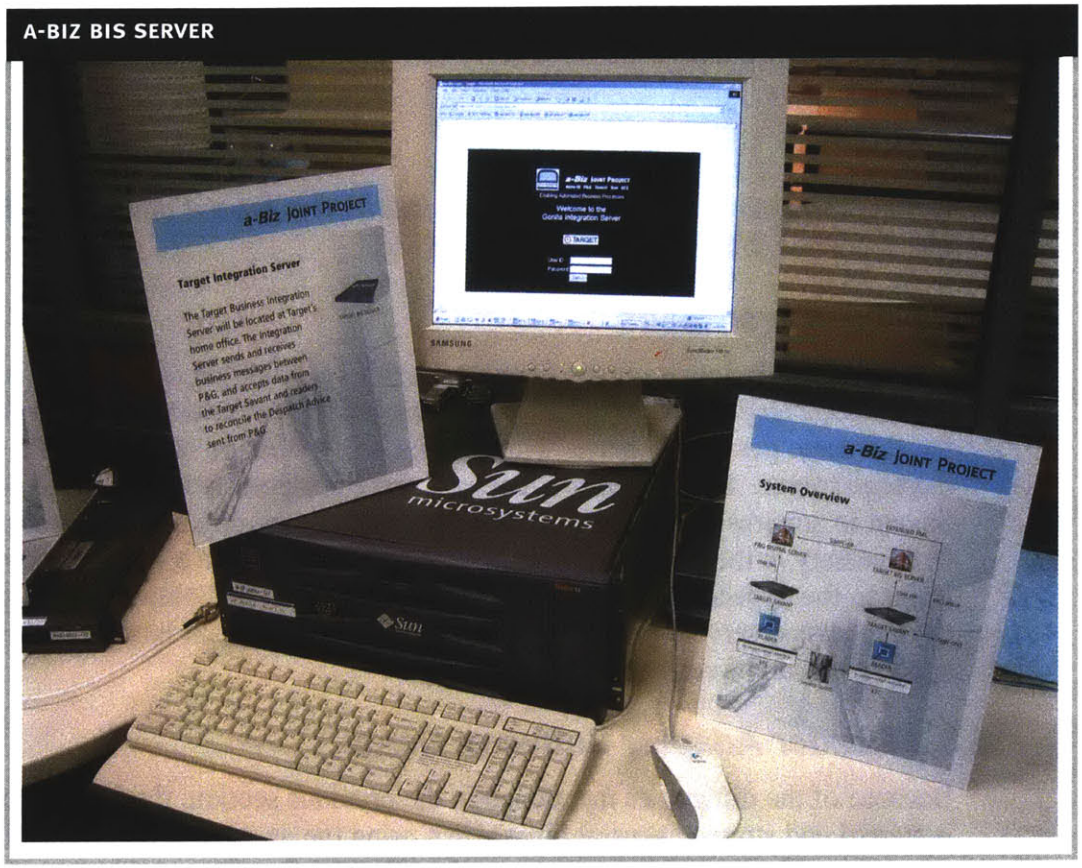


Figure 19 P&G BIS Server and the Web Interface for Order Shipping Billing

4.7.2 Filter Settings

The following table shows the parameter settings for the Savant filters that I used in the a-Biz demo:

FILTER SETTINGS		
Filter	**Parameter**	**Setting**
EnterOnlyFilter	number_of_passes	5
PassSmoothingFilter	number_of_passes	3

Table 11 Parameter Settings for the a-Biz Savant Filters

The EnterOnlyFilter's number_of_passes setting indicates a tag must be missing from the field for 5 consecutive passes before it will be picked up again. Otherwise, it will be suppressed. This cuts down on the chatter of continually reading tags that have not moved into or out of the field.

The PassSmoothingFilter's number_of_passes setting allows 3 passes to be normalized into one read. This allows three passes, which that may include random dropouts, to be consolidated into a single message sent to the BIS Server.

The combination of these two filters provided good results, with normalized and complete messages being sent via the SoapLogger to the BIS Server in every instance.

4.8 Problems

During the deployment of the demo, we encountered several problems. They are detailed in the following sections for reference.

4.8.1 Spurious Reads

At first, the BIS Server couldn't handle spurious or multiple reads being sent to the message engine, so we quickly fixed this problem. However, even with this fix, the demo performed better without a lot of extra reads sent to the BIS Server's message router. For this reason, controls were implemented to start and stop the Savant to suppress the logging of messages. Similar controls will almost certainly be necessary in the field, with a technician controlling when a reader is active. Refer to the appendix for the source listing for my SavantController that controls the application server and the Savant software on the Savant host. This java application can be run remotely.

4.8.2 Reader Adapter Software Reset

Because of the difficulties mentioned in the previous section, the Savant software is started and stopped several times when using the demo, but the supplied reader adapter could not reconnect to the reader after the first Savant shutdown. The reader adapter was not closing the socket to the reader during shutdown, so subsequent connection attempts were prohibited because the socket was still open (Note: By design, a reader can only communicate with one device at a time). The solution to this problem was to shutdown the process that was running the reader adapter, which forced the socket to be closed.

4.8.3 Savant Shutdown/Startup

While scripts and a SOAP interface exist to start and stop the Task Management System (TMS) and Event Management System (EMS) portions of the Savant software, the software is not durable enough to accommodate a shutdown and subsequent startup. In particular the TMS system does not reopen the PostGreSQL database connection during startup, but it closes it during shutdown, resulting in a broken TMS being activated. The short-term solution to this problem is to always bring down the application server instance when stopping the Savant software, this will cause all the components to be initialized properly when the instance is started.

4.9 Conclusion

In this chapter I covered the details of a reference implementation of the Auto-ID enabled Business Framework. More technical details can be found in the appendix. In the next chapter I will conclude with a summary of this work and provide future recommendations.

CHAPTER 5: Conclusions and Recommendations

In this thesis I have presented a framework for incorporating Auto-ID technologies to automate business transactions. In this final chapter I will discuss some of the findings of my research, and make some recommendations for future study.

5.1 Conclusions

5.1.1 Fundamental Problem

Recall that the fundamental problem I posed in this thesis is the synchronization of the physical and informational worlds. When this synchronization breaks down during a business transaction, there are penalties to be paid by both transacting parties resulting in lost time and money. In this thesis I have presented a framework for applying Auto-ID to a business transaction that can bridge the gap between physical objects and their related information. This eases and automates the flow of goods and information through the supply chain. The a-Biz Joint Project, a result of my research, has shown that this framework can be successfully employed to resolve a real business need.

5.1.2 Usability of Core PML

One key objective of my research was to test whether Core PML was capable of supporting a business transaction such as shipping and receiving. With the a-Biz Joint Project, I've shown in detail how Core PML succinctly automates an existing process, and facilitates a redesign of the process that expedites the verification of goods shipped using an automatically generated Despatch Advice and Receipt Advice.

Both Core PML and the Savant network easily facilitate counting operations, but the ability to uniquely identify products also aids in tracking. If the tracking data is persisted, a PML Service providing access this data enables tracing operations as well. In addition to tracking and tracing, this same framework can be extended to enable other mission critical operations including authentication [95], anti-theft [96], and dynamic expiration dates among others.

5.2 Recommendations

5.2.1 Changes to Existing Standards

In this section I will outline some proposed changes to the messaging standards used in my thesis: EDI, UBL, and Simpl-eb.

5.2.1.1 EDI

If Auto-ID is to be used with Traditional EDI, then line qualifiers will need to be added to the EDI standards to support Electronic Product Codes. This approval process is a lengthy one, however, and by the time it is ready for use, traditional EDI may no longer be in wide spread use. Efforts should instead be focused on emerging XML based schemas to make sure they include elements capable of supporting the EPC.

5.2.1.3 UBL

The Universal Business Language includes a Despatch Advice and a Receipt Advice, both useful for automating electronic commerce and closing the information loop. However, there is an unnecessary restriction forcing each line in the Receipt Advice to correspond to a line in the Despatch Advice. If a receiving company wishes to transmit a list of EPCs in the Receipt Advice so the sending party can reconcile any errors or exceptions, then the sending party is forced to send the same EPCs in the original Despatch Advice. This wastes bandwidth and is extremely inefficient. The UBL framework needs to ease this restriction to facilitate the shipping and receiving verification process proposed in this thesis, which provides a Receipt Advice to handle minimal exceptions.

5.2.1.2 Simpl-eb

I used the EAN.UCC business schemas as a basis for the messages sent between the supplier and customer in the reference implementation outlined in Chapter 4. To include the EPC data, I needed to slightly modify the schemas for the Despatch Advice. This change should be officially adopted by the UCC schema, which would make it more flexible for future use.

In addition to a Despatch Advice, a Receipt Advice should be added to the messaging standard. The Receipt Advice should not place limitations on the form and content of the message body as you find with UBL. This will allow minimal messages to be sent from the sending party to the receiving party with only quantity counts, but more detailed messages to be returned when exceptions occur. The more detailed Receipt Advice can then contain detailed lists of EPCs.

While there isn't anything wrong with the current schemas, the following table summarizes some things that go against recommended practice:

EAN.UCC BUSINESS SCHEMA ISSUES	
Problem	**Description**
Attributes hold data	Generally, elements should be used for data, and attributes for metadata. The reason for this is attributes can't be extended by the addition of children. Complex types are therefore preferred.
Type substitution	Type substitution is the use of derived types in place of a base type. This is sensible from an Object Oriented perspective, but often adds (needless) complexity to the XML document exchange.
Abstract types	Abstract types can be difficult to handle for processing applications, which must often be written to handle some, but not

EAN.UCC BUSINESS SCHEMA ISSUES	
	applications, which must often be written to handle some, but not all, concrete derived types. Again, this practice seems to make sense from an Object Oriented design approach, but it doesn't lend itself well to document exchange.
Deep element nesting	There are many nesting levels with some elements buried within many layers. This can be awkward in the same way that deeply nested classes can be.
Mixed content	The text content in mixed elements can't be validated or constrained. This is also problematic because it won't map to all specifications as Java objects.

Table 12 Summary of Issues encountered with the EAN.UCC Schemas

5.2.2 Future Work

5.2.2.1 ONS and PML

There is currently much ongoing work to standardize the PML Service. There are several reference implementations for specific use cases, but no general consensus has emerged to date. In addition, the ONS working group has just been formed, and the ONS itself is undergoing some changes. As work progresses in these fields, ongoing research and development should take place with PML extensions that will be suitable for specific industries.

When the new ONS specification is published, it should be added to the reference implementation and tested with tags and readers that support all of the newly proposed EPC recommendations. In addition, as the PML Service matures, the ONS lookup should be used for a product catalogue data lookup on unknown EPCs. This will involve coordination with existing product data catalogue services like Transora and UCCNet.

5.2.2.2 Legacy Numbering Schemes

An important area of research is the exploration of algorithmic methods to automatically convert the EPC into legacy numbering schemes, including the Globe Trade Item Number (GTIN), the Vehicle Identification Number (VIN), the National Drug Code (NDC) and others. In this work we've used a hash table to avoid a complete ONS and PML Service lookup.

5.2.2.3 Other Applications

The a-Biz reference implementation should be expanded using the framework to become a test bed for track and trace, authentication, theft and other Auto-ID enabled applications.

5.2.2.4 Data Storage

The data storage and transmission requirements of Auto-ID enabled business will strain current systems. In the appendix I've included a brief analysis of the

projected data loads. This is not an insurmountable obstacle, but the topic does require further work.

One idea is the use of patterns analysis to detect and label patterns in the accumulated data, thus reducing the data storage requirements for each individual item by referencing the associated patterns.

5.2.2.5 Logistics

Finally, many of the models used in logistics today will need to be revised or changed based on the ability to capture real-time information about the flow of goods through the supply chain. This will dramatically reduce inventories in the supply chain and simultaneously increase service levels. Some work has already been done (e.g. Yun Kang's and Stephen Ho's work in the Auto-ID Center, soon to be published), but there is much more to be done. Some of the methods and tools from the field of Fluid Dynamics could be used to build some of the new models.

5.3 Conclusion

In conclusion, there is much to be done. This field is ripe with possibilities that will yield many important advances in the coming century. The applications are endless, and the framework I've presented in this paper will serve as a roadmap to ongoing development and adoption.

APPENDIX

The appendix contains detailed technical discussions of some aspects of the research documented in my thesis. Each major heading in this appendix can be taken as a self-contained mini-chapter, and is provided for the interested reader.

EPC Dot Notation

Introduction

As we begin looking into inserting EPC data into third party systems, it is important to take a quick look at the storage requirements this will impose, based on the different representations that may be available. This short brief introduces a dot notation for the EPC, and shows the representation field lengths for the EPC using this notation in binary, hexadecimal and decimal formats in comparison with the field lengths for the EPC in each of these native formats. The reader should bear in mind that the dot notation is NOT how an EPC will be transmitted via Core PML messages within the Auto-ID system. This notation is simply meant to show one way that an outside application can reformat and/or store the EPC data for quick parsing and analysis.

This dot notation is used with the 64 bit EPCs represented in some of the other XML examples found later in the appendix.

Representations

The following table shows the break down of the 96 Bit Type I EPC.

96 BIT TYPE I EPC			
Version	**Domain Manager**	**Object Class**	**Serial Number**
8 bits	28 bits	24 bits	36 bits

Table 13 Breakdown of the 96 Bit Type 1 EPC

The 96 Bit Type I EPC's header is 00100001. The following table shows different representations of this number.

MAXIMUM HEADER NUMBER		
Binary	**Hexadecimal**	**Decimal**
0010 0001	33	21

Table 14 Maximum Header Number for the 96 Bit Type 1 EPC

The following table shows different representations of the maximum domain manager number.

MAXIMUM DOMAIN MANAGER NUMBER		
Binary	**Hexadecimal**	**Decimal**
11111111111111111111111111	FFFFFFF	268435455

Table 15 Maximum Domain Manager Number for the 96 Bit Type 1 EPC

The following table shows different representations of the maximum object class number.

MAXIMUM OBJECT CLASS NUMBER		
Binary	**Hexadecimal**	**Decimal**
111111111111111111111111	FFFFFF	16777215

Table 16 Maximum Object Class Number of the 96 Bit Type 1 EPC

The following table shows different representations of the maximum serial number.

MAXIMUM SERIAL NUMBER		
Binary	**Hexadecimal**	**Decimal**
111111111111111111111111111111111111	FFFFFFFFF	68719476735

Table 17 Maximum Serial Number of the 96 Bit Type 1 EPC

The following table shows three representations of a dot notation that may be useful for parsing and processing the 96 bit EPC (on computers that can't handle true 96 bit representations).

MAXIMUM COMBINED NUMBER (DOT NOTATION)	
Bin	00100001.11111111111111111111111111.11111111111111111111111111.11111111111111111111111111111111
Hex	33.FFFFFFF.FFFFFF.FFFFFFFFF
Dec	21.268435455.16777215.68719476735

Table 18 Maximum Combined Number of the 96 Bit Type 1 EPC Using Dot Notation

Because the EPC data may be inserted in a flat text file for use in a company's systems, the following table shows a side-by-side comparison of the representation field lengths of an EPC both with and without dot notation.

REPRESENTATION FIELD LENGTHS		
	Hexadecimal Fields	**Decimal Fields**
Max Binary Number (~2^{94})	~23	~29
Max Dot Notation (Previous table)	27	33

Table 19 Maximum Field Lengths of the Various Representations

All of the representations shown in the tables above are for the maximum EPC number that can be represented. Recently, the ONS Working Group of the Auto-ID Center has taken up the issue of representations of the EPC patterned after the International Standard Book Number (ISBN) and Uniform Resource Name (URN) [97] standards, among others. The URN is encompassed by the Uniform Resource Identifier (URI) [98, 99], and the ISBN may eventually be as well [100]. The ONS Working Group is debating the elimination of leading zeroes, and using a decimal format. The following table shows a recommended dot notation using a URI or URN naming scheme.

POSSIBLE UNIFORM NAMING SCHEMES	
Scheme	**Dot Notation**
URI	uri:epc:1.24564.21346.21678856
URN	urn:epc:1.24564.21346.21678856

Table 20 Possible Uniform Naming Schemes for the EPC

The interested reader should follow the work of the Auto-ID ONS Working Group for a final recommendation and representation.

Conclusion

In summary, Table 19 shows the maximum field lengths for the different representations. In native form, the EPC will require 23 to 29 fields in a flat text file, or 27 to 33 fields using the dot notation, depending on the representation chosen.

Types of Queries

There are basically three classes of queries that need to be handled to properly address the needs of sequence flows presented in this thesis. The following questions exemplify the three classes of queries:

- Where is this item? Alternately, how many of these SKUs are at location X?
- Which readers saw this item in this time frame?
- Did any items cross path with any other items?

The first two types of queries are similar to the Eulerian and Lagrangian methods for tracking particles in a fluid flow, one follows the path of an item and can determine it's properties at any given time, the other watches a system within a reference frame and captures information about all items that flow through the reference frame boundaries during the capture interval. The third group of queries can be categorized as data mining for patterns.

The different types of queries will result in different data load requirements. For example, the simple query "Where is this item?" can be answered with minimal

data. The data requirements of the remaining queries outlined in this section depend on the number of item EPCs involved with the request and/or the length of the time frame. Refer to the next section entitled Estimated Data Load Requirements for some representative numbers useful for these types of calculations.

Estimated Data Load Requirements

The following estimates should be kept in mind when designing Auto-ID systems for Shipping and Receiving, and Track and Trace systems:

- The Auto-ID sponsor community is responsible for about 1 trillion items in the supply chain annually.
- There are almost 300 million people in the US, and there are currently 100,000 retail outlets in the US (1 for every 3000 shoppers).
- A typical large retailer may have 2,000 to 3,000 domestic outlets and up to 4,500 worldwide. Growth varies from several dozen new stores to 350 new stores annually.
- A typical large retailer may handle between 20 and 40 billion individual items system wide each year.
- A typical retailer/outlet may carry anywhere from 20,000 to 120,000 SKU's, depending on the size and type of the store (e.g. Retail, Discount, Warehouse).

Choosing some representative numbers, if a company handles 40 billion items spread among 4,000 retail stores, each store handles roughly 10 million items a year, 833,000 items a month, and 28,000 items a day. If a daily replenishment cycle is assumed, that means 28,000 items move in and out of a store every day.

Using the calculated number of 28,000 items being received by a store each day, if a receiving system is designed where only the 96 bit EPC is captured and used, 328 KB of data would be generated with each complete scan *for the EPC data alone*. (Using the 64-bit EPC would require 218 KB.)

((28,000 items) * (96 bits/item)) / ((8 bits/byte) * (1024 bytes/KB)) ~= 328 KB

For Shipping and Receiving Verification on inbound shipments for any given store, the list of EPCs sent in the DA would require an additional 328/218 KB of data above and beyond the normal data shipped in the DA. The Auto-ID system will also have to quickly scan and process this information on the receiving end to quickly reconcile the EPCs.

Assuming there is about 5 KB of static product data for each item, if an EPC lookup is performed to request and receive this data for each of the 28,000 items, about 137 MB of memory would be required.

((28,000 items) * (5 KB/item)) / (1024 KB/MB) ~= 137 MB

The 5 KB per item of static product data is not necessary for Shipping and Receiving Verification, as this can be accomplished by comparing only the list of EPCs. More detailed product data approaching the 5 KB per item limit may be necessary for tracking operations, but a lot of the static data can be cached at the SKU level. To properly support tracing operations, however, you must account for the persistent storage requirements of the tracking data, which could grow well beyond the 5 KB of static data, and depends on a combination of factors including the number scans performed, the amount of data gathered, and the method used to persist the data to a data store for later mining.

If 1 trillion items are traded in a year, and each has about 5 KB of static product data associated with it, the data storage requirement for these items is about 960 GB.

((1 trillion items) * (5 KB/item)) / ((1024 KB/MB) * (1024 MB/GB)) ~= 960 GB

If over it's lifetime each product amasses an additional 20 KB of tracking data, then we are looking at 3.84 TB of historical data *each year*:

((1 trillion items) * (20 KB/item)) / ((1024 KB/MB) * (1024 MB/GB) * (1024 GB/TB)) ~= 3.84 TB

Note that all of the calculations just presented assume that each item is treated individually. If instead you treated the items as the SKU level (no item serialization), then the storage requirements are dropped by several orders of magnitude, and are in fact more in line with current storage requirements. The numbers presented herein are subject to interpretation, and are meant only to provide a rough order of magnitude analysis.

Also note that some retailer's data requirements currently approach 250 TB of data each year, and that is without the granularity imposed by serialized EPC product data. These numbers and formulas are meant as a reference, and should be adjusted as situations dictate.

Sample Messages

Introduction

This section presents complete examples of the various electronic messages formats considered. It concludes with a summary of the different formats.

PML Core

The following is a UML class diagram representing the PML Core [93].

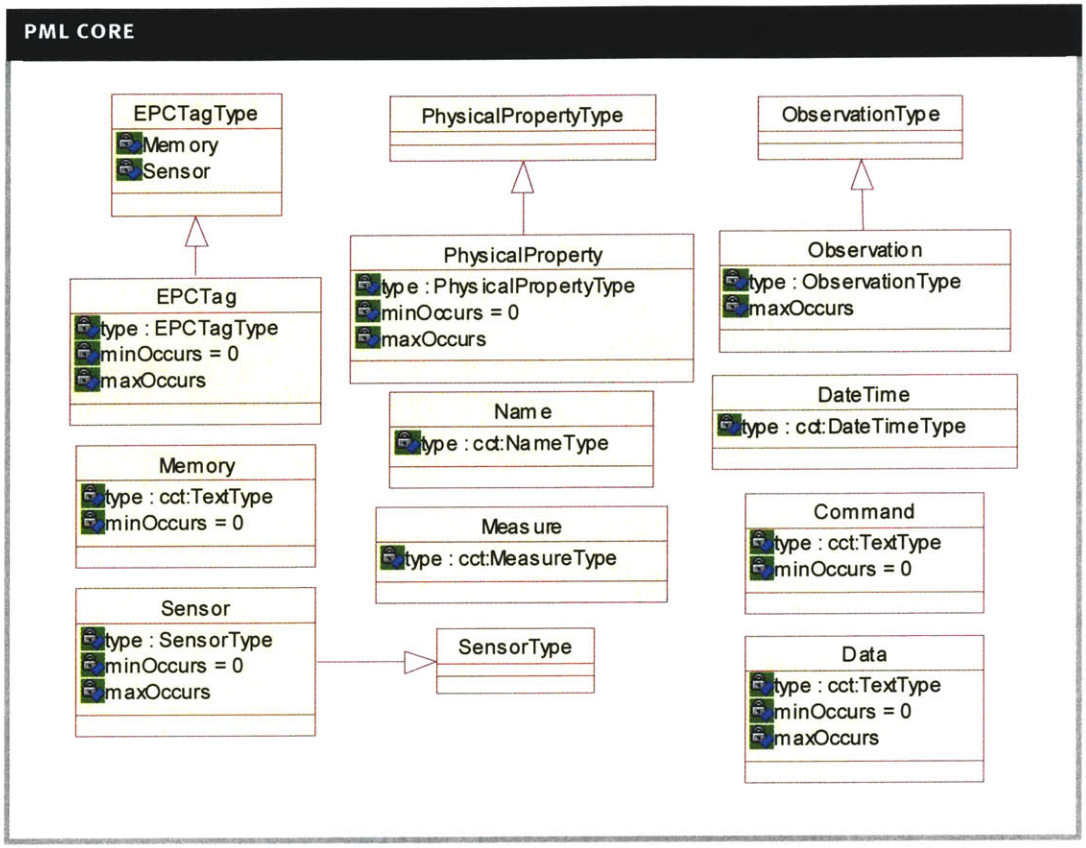


Figure 20 UML Class Diagram of PML Core [93]

Here is a sample Core PML message that provides a list of EPCs:

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlc:Sensor
xmlns:pmlc="urn:autoidcenter:pml:Core:1.0:0.50"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:autoidcenter:pml:Core:1.0:0.50
 PML_Library_Core.xsd">
   <ID>01.0000A89.035F23.000169DC1</ID>
   <Observation>
      <ID>000345869</ID>
      <DateTime>2003-05-19T13:04:34-06:00</DateTime>
      <Tag>
         <ID>01.0000A89.00016F.000169DC1</ID>
      </Tag>
      <Tag>
         <ID>01.0000A89.00016F.000169DC3</ID>
      </Tag>
      <Tag>
         <ID>01.0000A89.00016F.000169DC4</ID>
      </Tag>
   </Observation>
</pmlc:Sensor>
```

SOAP

SOAP can be used to wrap a message, and it can also be used for Remote Procedure Calls (RPC). The following is an example of a simple SOAP wrapper for a remote request that uses a method called ONSLookup using an EPC code:

```
POST /ONS HTTP/1.1
Host: www.autoidcenter.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
   <soap:Body xmlns:m="http://www.autoidcenter.org/ons">
      <m:ONSLookup>
         <m:EPC>
           01101010011110010110100111001011110100011010110001101101011100100
         </m:EPC>
      </m: ONS_Lookup>
   </soap:Body>
</soap:Envelope>
```

A sample SOAP response may take the form:

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: nnn<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
   <soap:Body xmlns:m="http://www.autoidcenter.org/ons">
      <m:ONSLookupResponse>

<m:URL>1.D.69CB.1.0.0.7.1.5.006.epc.objid.net</m:URL>
      </m: ONSLookupResponse>
   </soap:Body>
</soap:Envelope>
```

One final example shows a SOAP message sent from the Savant to the BIS server containing a Core PML Observation:

```
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
   <s:Body>
      <Sensor>
         <ID>SAMPLEUHF</ID>
         <Observation>
            <DateTime>2003-06-17T18:00:00-0700</DateTime>
               <Tag>
                  <ID>8000800424086080</ID>
               </Tag>
```

```
            </Observation>
         </Sensor>
      </s:Body>
   </s:Envelope>
```

Simpl-eb

This section contains sample XML files showing the format of the EAN.UCC's Simpl-eb messaging. For the Despatch Advice, both a standard message, which is much like you would see today without EPC data, and an augmented message, which includes serialized lists of EPCs, are presented.

The Despatch Advice as described by the UCC in DespatchAdvice.xsd was insufficient, so I had to make two small changes to the schema. In particular, the following problems were encountered when trying to use the default Simpl-eb schema definition:

- the "serial" number in the "ItemContainmentType" element was ideal to transmit EPCs, but it is only 20 characters long
    - I changed the type of the "serial" element from "String1to20Type" to "xsd:string"
        - The string type has the disadvantage of allowing spaces
- the SSCC (serial shipping container code) is only 18 digits, but the EDI definition for our example ASN uses 20 digits
- the original DA schema didn't allow you to specify the quantity of items shipped
    - I modified the schema by adding a "quantityContained" element to the " ItemContainmentType" element
    - The original BRD from the UCC includes this element, but it is missing from the schema (see below)

I noted the following deficiency with the Simpl-eb schema:

- No carrier details

One final note:

Simpl-eb uses identifier (GLN) to represent parties whereas EDI uses Name and ID code

Simpl-eb Order

The following is a sample Order using the EAN.UCC business-messaging format:

```
<order>
   <creationDate>06/20/03</creationDate>
   <documentStatus>ORIGINAL</documentStatus>
   <contentVersion>1.1</contentVersion>
   <documentStructureVersion>1.1</documentStructureVersion>
   <lastUpdateDate>06/20/03</lastUpdateDate>
   <movementDate>06/21/03</movementDate>
```

```
      <movementDateType>REQUESTED_PICKUP</movementDateType>
      <!-- Pampers Bibsters-->
      <lineItem>
         <number>0</number>
         <requestedQuantity>2</requestedQuantity>
         <itemIdentification>
            <gtin>037000401803</gtin>
         </itemIdentification>
      </lineItem>
      <!-- Bounce -->
      <lineItem>
         <number>0</number>
         <requestedQuantity>1</requestedQuantity>
         <itemIdentification>
            <gtin>037000801689</gtin>
         </itemIdentification>
      </lineItem>
      <buyer>
         <gln>!146@@760?7@@718</gln>
      </buyer>
      <seller>
         <gln>!146@@760?9@@719</gln>
      </seller>
      <typedEntityIdentification>
         <entityIdentification>
            <uniqueCreatorIdentification>Order 1
            </uniqueCreatorIdentification>
         </entityIdentification>
      </typedEntityIdentification>
   </order>
```

Simpl-eb Despatch Advice Quantity Counts

This section addresses how to include count information using Simpl-eb. Referring to the UML class diagram from the Business Requirements Document (BRD) for Despatch Advice [101], consider the situation where you want to describe a pallet containing 3 cases with each case containing 10 units.

From root class DespatchAdvice there are one or more DespatchItems (1). The DespatchItem can be represented as either a TradeItemUnit (2) or a LogisticsUnit (2). Once the selection has been made, there is an ItemContainment (3), which then allows for quantityContained (4).

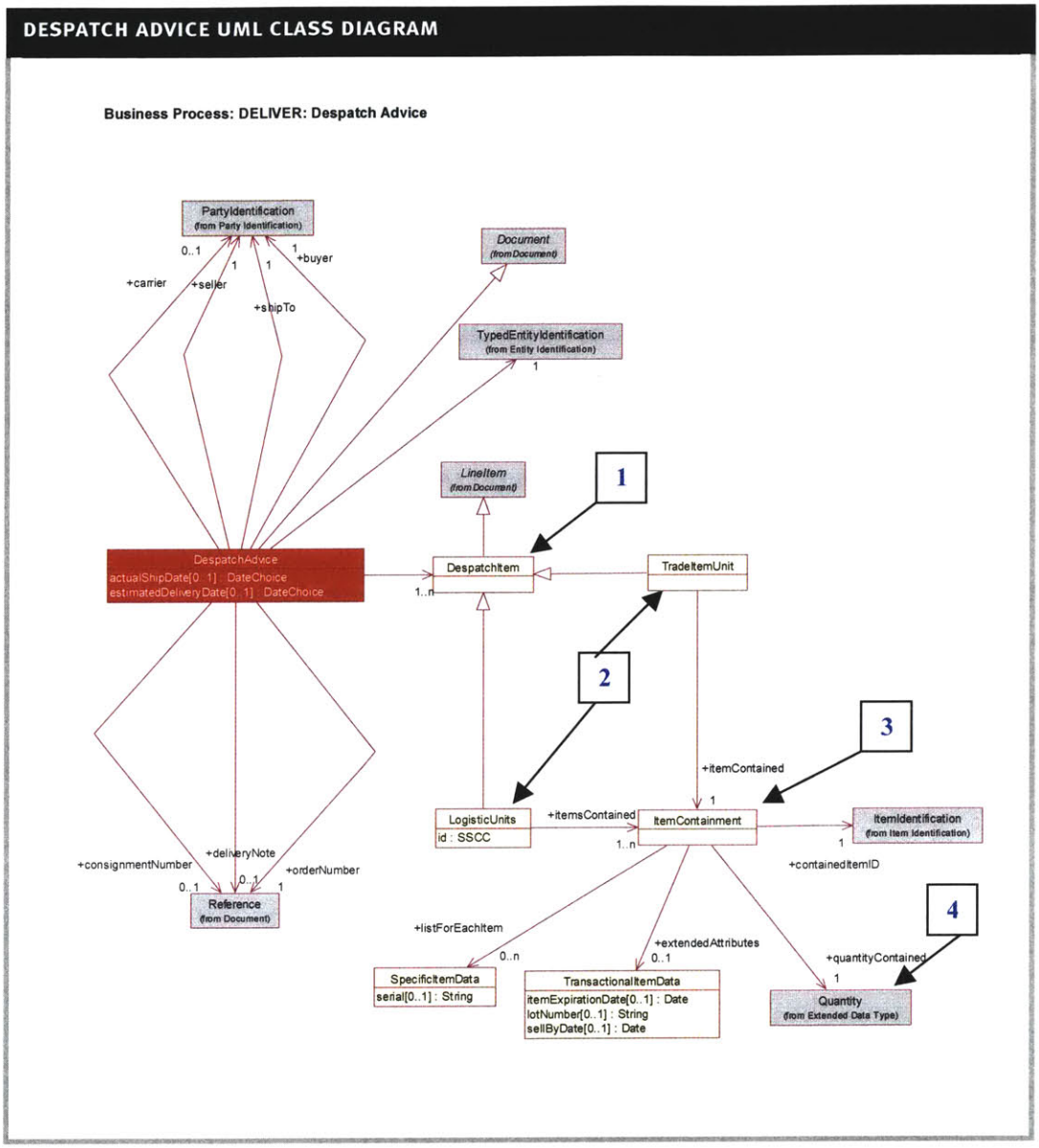


Figure 21 UML Class Diagram for Despatch Advice [101]

So using the example mentioned above:

- 1 Pallet with
- 3 Cases
- 10 Units per case

There will be one DespatchItem for the 1 pallet represented as a LogisticsUnits. From there ItemContainment will be used twice:

- Once for the Cases with the appropriate GTIN and quantity
- Second for the Each with the appropriate GTIN and quantity

For the purposes of this thesis, I skipped the Case GTINs and used only the Package GTINs, as I used individual packages. See Simpl-eb DA Standard and Simpl-eb DA Augmented later in the appendix for XML instance examples.

Simpl-eb Despatch Advice - Standard

The following is a complete sample Despatch Advice using the EAN.UCC business-messaging format. Note how the "quantityContained" element is used to indicate scanned quantities:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by
Timothy P. Milne (MIT Auto-ID Center) -->
<!--
<eanucc:envelope
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:eanucc="http://www.uc-
council.org/smp/schemas/eanucc"
 xsi:noNamespaceSchemaLocation="EanUccProxy.xsd"
 communicationVersion="1.1">
-->
<!-- prolog for xml spy & xsv-->
<eanucc:envelope
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:eanucc="http://www.uc-
council.org/smp/schemas/eanucc"
 xsi:schemaLocation="http://www.uc-
council.org/smp/schemas/eanucc
 As2Envelope.xsd http://www.uc-
council.org/smp/schemas/eanucc
 Transaction.xsd http://www.uc-
council.org/smp/schemas/eanucc
 DocumentCommand.xsd  http://www.uc-
council.org/smp/schemas/eanucc
 DespatchAdviceAutoID.xsd" communicationVersion="1.1">
   <messageHeader creationDate="2001-08-02T12:00:00">
      <userId>ProctorAndGamble</userId>
      <password>SECRET</password>
      <messageIdentifier>938890001</messageIdentifier>
      <to>
         <gln>0011223344556</gln>
      </to>
      <from>
         <gln>9988776655443</gln>
      </from>
      <representingParty>
         <gln>0011223344556</gln>
      </representingParty>
   </messageHeader>
   <body>
      <eanucc:transaction>
         <entityIdentification>
            <uniqueCreatorIdentification>A-BIZ-TRANS-12345
            </uniqueCreatorIdentification>
            <contentOwner>
               <gln>9988776655443</gln>
```

```
   </contentOwner>
</entityIdentification>
<command>
   <eanucc:documentCommand>
      <documentCommandHeader type="ADD">
         <entityIdentification>
            <uniqueCreatorIdentification>
               A-BIZ-ITEM-12345
            </uniqueCreatorIdentification>
            <contentOwner>
               <gln>9988776655443</gln>
            </contentOwner>
         </entityIdentification>
      </documentCommandHeader>
      <documentCommandOperand>
         <eanucc:despatchAdvice
          creationDate="2003-05-28T12:13:14"
          documentStatus="ORIGINAL"
          contentVersion="1.1.1"
          documentStructureVersion="1.1.1"
          lastUpdateDate="2003-05-28">
            <estimatedDeliveryDate>
               <date>2003-08-28</date>
            </estimatedDeliveryDate>
            <typedEntityIdentification
             entityType="DESPATCH_ADVICE">
               <entityIdentification>
                  <uniqueCreatorIdentification>
                     A-BIZ-TRANS12345
                  </uniqueCreatorIdentification>
                  <contentOwner>
                     <gln>9988776655443</gln>
                  </contentOwner>
               </entityIdentification>
            </typedEntityIdentification>
            <buyer>
               <gln>0011223344556</gln>
            </buyer>
            <seller>
               <gln>9988776655443</gln>
            </seller>
            <shipTo>
               <gln>0011223344556</gln>
            </shipTo>
            <carrier>
               <gln>9988776655443</gln>
            </carrier>
            <deliveryNote number="12">
               <date>2003-06-28</date>
            </deliveryNote>
            <orderNumber number=
               "0094-4316399-0577">
               <dateTime>
                  2003-04-23T12:13:14
               </dateTime>
            </orderNumber>
            <consignmentNumber number="06716420">
```

```
                        <date>2003-04-29</date>
                    </consignmentNumber>
                    <actualShipDate>
                        <date>2003-04-27</date>
                    </actualShipDate>
                    <despatchItem
                     xsi:type="eanucc:LogisticUnitsType"
                     number="1" id="100370000014190998">
                        <!-- Pampers Bibsters-->
                        <itemsContained>
                            <containedItemID>
                                037000401803
                            </containedItemID>
                            <quantityContained>
                                2
                            </quantityContained>
                        </itemsContained>
                        <!-- Bounce -->
                        <itemsContained>
                            <containedItemID>
                                037000801689
                            </containedItemID>
                            <quantityContained>
                                1
                            </quantityContained>
                        </itemsContained>
                    </despatchItem>
                </eanucc:despatchAdvice>
            </documentCommandOperand>
        </eanucc:documentCommand>
    </command>
</eanucc:transaction>
</body>
</eanucc:envelope>
```

Simpl-eb Despatch Advice - Augmented

The following is a complete sample Despatch Advice using the EAN.UCC business-messaging format that has been augmented to allow lists of EPCs. Note how the modified "serial" element is used to transmit the list of scanned EPCs:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by
Timothy P. Milne (MIT Auto-ID Center) -->
<!--
<eanucc:envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:eanucc="http://www.uc-council.org/smp/schemas/eanucc"
xsi:noNamespaceSchemaLocation="EanUccProxy.xsd"
communicationVersion="1.1">
-->
<!-- prolog for xml spy & xsv-->
<eanucc:envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:eanucc="http://www.uc-council.org/smp/schemas/eanucc"
xsi:schemaLocation="http://www.uc-
council.org/smp/schemas/eanucc As2Envelope.xsd
```

```
http://www.uc-council.org/smp/schemas/eanucc
Transaction.xsd http://www.uc-
council.org/smp/schemas/eanucc DocumentCommand.xsd
http://www.uc-council.org/smp/schemas/eanucc
DespatchAdviceAutoID.xsd" communicationVersion="1.1">
   <messageHeader creationDate="2001-08-02T12:00:00">
      <userId>ProctorAndGamble</userId>
      <password>SECRET</password>
      <messageIdentifier>938890001</messageIdentifier>
      <to>
         <gln>0011223344556</gln>
      </to>
      <from>
         <gln>9988776655443</gln>
      </from>
      <representingParty>
         <gln>0011223344556</gln>
      </representingParty>
   </messageHeader>
   <body>
      <eanucc:transaction>
         <entityIdentification>
            <uniqueCreatorIdentification>
               A-BIZ-TRANS-12345
            </uniqueCreatorIdentification>
            <contentOwner>
               <gln>9988776655443</gln>
            </contentOwner>
         </entityIdentification>
         <command>
            <eanucc:documentCommand>
               <documentCommandHeader type="ADD">
                  <entityIdentification>
                     <uniqueCreatorIdentification>
                        A-BIZ-ITEM-12345
                     </uniqueCreatorIdentification>
                     <contentOwner>
                        <gln>9988776655443</gln>
                     </contentOwner>
                  </entityIdentification>
               </documentCommandHeader>
               <documentCommandOperand>
                  <eanucc:despatchAdvice
                   creationDate="2003-05-28T12:13:14"
                   documentStatus="ORIGINAL"
                   contentVersion="1.1.1"
                   documentStructureVersion="1.1.1"
                   lastUpdateDate="2003-05-28">
                     <estimatedDeliveryDate>
                        <date>2003-08-28</date>
                     </estimatedDeliveryDate>
                     <typedEntityIdentification
                      entityType="DESPATCH_ADVICE">
                        <entityIdentification>
                           <uniqueCreatorIdentification>
                              A-BIZ-TRANS12345
                           </uniqueCreatorIdentification>
```

```
      <contentOwner>
         <gln>9988776655443</gln>
      </contentOwner>
   </entityIdentification>
</typedEntityIdentification>
<buyer>
   <gln>0011223344556</gln>
</buyer>
<seller>
   <gln>9988776655443</gln>
</seller>
<shipTo>
   <gln>0011223344556</gln>
</shipTo>
<carrier>
   <gln>9988776655443</gln>
</carrier>
<deliveryNote number="12">
   <date>2003-06-28</date>
</deliveryNote>
<orderNumber number=
   "0094-4316399-0577">
   <dateTime>
      2003-04-23T12:13:14
   </dateTime>
</orderNumber>
<consignmentNumber number="06716420">
   <date>2003-04-29</date>
</consignmentNumber>
<actualShipDate>
   <date>2003-04-27</date>
</actualShipDate>
<despatchItem
 xsi:type="eanucc:LogisticUnitsType"
 number="1" id="100370000014190998">
   <!-- Pampers Bibsters-->
   <itemsContained>
      <containedItemID>
         037000401803
      </containedItemID>
      <listForEachItem>
         <serial>
            1.0000A89.00016F.000169DC1
         </serial>
      </listForEachItem>
      <listForEachItem>
         <serial>
            1.0000A89.00016F.000169DC2
         </serial>
      </listForEachItem>
   </itemsContained>
   <!-- Bounce -->
   <itemsContained>
      <containedItemID>
         037000801689
      </containedItemID>
      <listForEachItem>
```

```
                                    <serial>
                                      01.0000A89.003014.000169AB1
                                    </serial>
                                  </listForEachItem>
                                </itemsContained>
                              </despatchItem>
                            </eanucc:despatchAdvice>
                          </documentCommandOperand>
                        </eanucc:documentCommand>
                      </command>
                    </eanucc:transaction>
                  </body>
                </eanucc:envelope
```

UBL

This section contains sample XML files showing the format of the ebXML based UBL messages. Like Simpl-eb, this section contains both standard and augmented versions of the messages, without and with EPCs respectively.

It should be noted that unlike Simpl-eb, UBL does provide a Receipt Advice (RA). Unfortunately, the format of the RA must exactly match the corresponding DA (meaning there must be a 1:1 correlation between the DespatchLines in the DA and the ReceiptLines in the RA). If you want to transmit detailed EPC information in the RA, it will have to be broken down into separate Logistical Units for each EPC, and this will force the original DA to be similarly formatted (Augmented with EPCs). This means that the sending party will be forced to transmit all of the EPC scan data in the original message, a situation we are clearly trying to avoid. In other words, if you use the RA described in UBL Receipt Advice Augmented, then the original DA should be as outlined in UBL Despatch Advice Augmented. Bandwidth limitations alone make this an unattractive option.

UBL Order

UBL provides an order that could have been used. But since Simpl-eb, was used, the UBL order will not be included here.

UBL Despatch Advice - Standard

The following is a complete sample Despatch Advice using a UBL business-messaging format. Note how the UPC code is used for the LineID and the DeliveredQuantity shows the count:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by
Timothy P. Milne (MIT Auto-ID Center) -->
<DespatchAdvice
xmlns="urn:oasis:names:tc:ubl:DespatchAdvice:1.0:0.70"
xmlns:cat="urn:oasis:names:tc:ubl:CommonAggregateTypes:1.0:
0.70"
xmlns:cct="urn:oasis:names:tc:ubl:CoreComponentTypes:1.0:0.
70"
xmlns:ccts="urn:oasis:names:tc:ubl:CoreComponentParameters:
```

```
1.0:0.70" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="urn:oasis:names:tc:ubl:DespatchAdvice:1
.0:0.70
UBL_Library_0p70_DespatchAdvice.xsd">
   <cat:ID>99-99-000</cat:ID>
   <cat:IssueDate>2003-05-14</cat:IssueDate>
   <DeliveryDate>2003-05-19</DeliveryDate>
   <LanguageCode>US-ENG</LanguageCode>
   <cat:ReferencedOrder>
      <cat:BuyersOrderID>
        0094-4316399-0577
      </cat:BuyersOrderID>
      <cat:SellersOrderID>06716420</cat:SellersOrderID>
      <cat:IssueDate>2003-02-14</cat:IssueDate>
   </cat:ReferencedOrder>
   <cat:BuyerParty>
      <cat:ID>T0001</cat:ID>
      <cat:PartyName>
         <cat:Name>Target DC 557</cat:Name>
      </cat:PartyName>
      <cat:Address>
         <cat:ID/>
         <cat:Street>413 Spring St.</cat:Street>
         <cat:CityName>OCONOMOWOC</cat:CityName>
         <cat:PostalZone>53066</cat:PostalZone>
         <cat:CountrySub-Entity>USA</cat:CountrySub-Entity>
      </cat:Address>
   </cat:BuyerParty>
   <cat:SellerParty>
      <cat:ID>PG0001</cat:ID>
      <cat:PartyName>
         <cat:Name>
            Procter and Gamble Distribution Co.
         </cat:Name>
      </cat:PartyName>
      <cat:Address>
         <cat:ID/>
         <cat:Street>1 Main Street</cat:Street>
         <cat:CityName>OCONOMOWOC</cat:CityName>
         <cat:PostalZone>52358</cat:PostalZone>
         <cat:CountrySub-Entity>USA</cat:CountrySub-Entity>
      </cat:Address>
      <cat:ShippingContact>
         <cat:ID/>
         <cat:Name>Door Mat</cat:Name>
         <cat:Phone>(763) 865-2194</cat:Phone>
      </cat:ShippingContact>
   </cat:SellerParty>
   <cat:DeliveryRequirement>
      <cat:ID/>
      <cat:DeliverToAddress>
         <cat:ID/>
         <cat:Street>413 N Spring St. </cat:Street>
         <cat:CityName>OCONOMOWOC</cat:CityName>
         <cat:PostalZone>53066</cat:PostalZone>
         <cat:CountrySub-Entity>USA</cat:CountrySub-Entity>
```

```
      </cat:DeliverToAddress>
   </cat:DeliveryRequirement>
   <!-- Auto-ID: Pampers Bibsters -->
   <cat:DespatchLine>
      <!-- Auto-ID: UPC Code -->
      <cat:LineID>037000401803</cat:LineID>
      <cat:OrderLineID>1</cat:OrderLineID>
      <cat:DeliveredQuantity unitCode="PKG">
         2
      </cat:DeliveredQuantity>
      <cat:DeliverySchedule>
         <cat:ID/>
         <cat:DeliveryRequirement>
            <cat:ID/>
         </cat:DeliveryRequirement>
      </cat:DeliverySchedule>
      <cat:ReferencedTransportHandlingUnit>
         <cat:ID>OCLU1234556</cat:ID>
      </cat:ReferencedTransportHandlingUnit>
   </cat:DespatchLine>
   <!-- Auto-ID: Bounce -->
   <cat:DespatchLine>
      <cat:LineID>037000801689</cat:LineID>
      <cat:OrderLineID>2</cat:OrderLineID>
      <cat:DeliveredQuantity unitCode="PKG">
         1
      </cat:DeliveredQuantity>
      <cat:DeliverySchedule>
         <cat:ID/>
         <cat:DeliveryRequirement>
            <cat:ID/>
         </cat:DeliveryRequirement>
      </cat:DeliverySchedule>
      <cat:ReferencedTransportHandlingUnit>
         <cat:ID>OCLU1234556</cat:ID>
      </cat:ReferencedTransportHandlingUnit>
   </cat:DespatchLine>
   <cat:ActualShipment>
      <cat:ID/>
   </cat:ActualShipment>
</DespatchAdvice
```

UBL Despatch Advice - Augmented

The following is a complete sample Despatch Advice using a UBL business-messaging format that has been augmented to allow lists of EPCs. Note how the Logistics Unit can be separated into units of 1 each, with the LineID being used for the individual EPC codes:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by
Timothy P. Milne (MIT Auto-ID Center) -->
<DespatchAdvice
xmlns="urn:oasis:names:tc:ubl:DespatchAdvice:1.0:0.70"
xmlns:cat="urn:oasis:names:tc:ubl:CommonAggregateTypes:1.0:
0.70"
xmlns:cct="urn:oasis:names:tc:ubl:CoreComponentTypes:1.0:0.
```

```
70"
xmlns:ccts="urn:oasis:names:tc:ubl:CoreComponentParameters:
1.0:0.70" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="urn:oasis:names:tc:ubl:DespatchAdvice:1
.0:0.70
UBL_Library_0p70_DespatchAdvice.xsd">
   <cat:ID>99-99-000</cat:ID>
   <cat:IssueDate>2003-02-14</cat:IssueDate>
   <DeliveryDate>2003-05-19</DeliveryDate>
   <LanguageCode>US-ENG</LanguageCode>
   <cat:ReferencedOrder>
      <cat:BuyersOrderID>0094-4316399-0577
      </cat:BuyersOrderID>
      <cat:SellersOrderID>06716420</cat:SellersOrderID>
      <cat:IssueDate>2003-02-14</cat:IssueDate>
   </cat:ReferencedOrder>
   <cat:BuyerParty>
      <cat:ID>T0001</cat:ID>
      <cat:PartyName>
         <cat:Name>Target DC 557</cat:Name>
      </cat:PartyName>
      <cat:Address>
         <cat:ID/>
         <cat:Street>413 Spring St.</cat:Street>
         <cat:CityName>OCONOMOWOC</cat:CityName>
         <cat:PostalZone>53066</cat:PostalZone>
         <cat:CountrySub-Entity>USA</cat:CountrySub-Entity>
      </cat:Address>
   </cat:BuyerParty>
   <cat:SellerParty>
      <cat:ID>PG0001</cat:ID>
      <cat:PartyName>
         <cat:Name>Procter and Gamble Distribution Co.
         </cat:Name>
      </cat:PartyName>
      <cat:Address>
         <cat:ID/>
         <cat:Street>1 Main Street</cat:Street>
         <cat:CityName>OCONOMOWOC</cat:CityName>
         <cat:PostalZone>52358</cat:PostalZone>
         <cat:CountrySub-Entity>USA</cat:CountrySub-Entity>
      </cat:Address>
      <cat:ShippingContact>
         <cat:ID/>
         <cat:Name>Door Mat</cat:Name>
         <cat:Phone>(763) 865-2194</cat:Phone>
      </cat:ShippingContact>
   </cat:SellerParty>
   <cat:DeliveryRequirement>
      <cat:ID/>
      <cat:DeliverToAddress>
         <cat:ID/>
         <cat:Street>413 N Spring St. </cat:Street>
         <cat:CityName>OCONOMOWOC</cat:CityName>
         <cat:PostalZone>53066</cat:PostalZone>
         <cat:CountrySub-Entity>USA</cat:CountrySub-Entity>
```

```
      </cat:DeliverToAddress>
   </cat:DeliveryRequirement>
   <!-- Auto-ID: Pampers Bibsters -->
   <cat:DespatchLine>
      <!-- Auto-ID: EPC Code -->
      <cat:LineID>01.0000A89.00016F.000169DC1</cat:LineID>
      <cat:OrderLineID>1</cat:OrderLineID>
      <cat:DeliveredQuantity
       unitCode="PKG">1</cat:DeliveredQuantity>
      <cat:DeliverySchedule>
         <cat:ID/>
         <cat:DeliveryRequirement>
            <cat:ID/>
         </cat:DeliveryRequirement>
      </cat:DeliverySchedule>
      <cat:ReferencedTransportHandlingUnit>
         <cat:ID/>
      </cat:ReferencedTransportHandlingUnit>
   </cat:DespatchLine>
   <cat:DespatchLine>
      <!-- Auto-ID: EPC Code -->
      <cat:LineID>01.0000A89.00016F.000169DC2</cat:LineID>
      <cat:OrderLineID>1</cat:OrderLineID>
      <cat:DeliveredQuantity
       unitCode="PKG">1</cat:DeliveredQuantity>
      <cat:DeliverySchedule>
         <cat:ID/>
         <cat:DeliveryRequirement>
            <cat:ID/>
         </cat:DeliveryRequirement>
      </cat:DeliverySchedule>
      <cat:ReferencedTransportHandlingUnit>
         <cat:ID/>
      </cat:ReferencedTransportHandlingUnit>
   </cat:DespatchLine>
   <!-- Auto-ID: Bounce -->
   <cat:DespatchLine>
      <!-- Auto-ID: EPC Code -->
      <cat:LineID>01.0000A89.003014.000169AB1</cat:LineID>
      <cat:OrderLineID>1</cat:OrderLineID>
      <cat:DeliveredQuantity
       unitCode="PKG">1</cat:DeliveredQuantity>
      <cat:DeliverySchedule>
         <cat:ID/>
         <cat:DeliveryRequirement>
            <cat:ID/>
         </cat:DeliveryRequirement>
      </cat:DeliverySchedule>
      <cat:ReferencedTransportHandlingUnit>
         <cat:ID/>
      </cat:ReferencedTransportHandlingUnit>
   </cat:DespatchLine>
   <cat:ActualShipment>
      <cat:ID/>
   </cat:ActualShipment>
</DespatchAdvice
```

UBL Receipt Advice - Augmented

The following is a complete sample Receipt Advice using a UBL business-messaging format that has been augmented to allow lists of EPCs. It shows how the Logistics Unit can be separated into units of 1 each, with the ID being used for the individual EPC codes (Note how the first Pampers Bibster is missing, and specified as a short):

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by
Timothy P. Milne (MIT Auto-ID Center) -->
<ReceiptAdvice
xmlns="urn:oasis:names:tc:ubl:ReceiptAdvice:1.0:0.70"
xmlns:cat="urn:oasis:names:tc:ubl:CommonAggregateTypes:1.0:
0.70"
xmlns:cct="urn:oasis:names:tc:ubl:CoreComponentTypes:1.0:0.
70"
xmlns:ccts="urn:oasis:names:tc:ubl:CoreComponentParameters:
1.0:0.70" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="urn:oasis:names:tc:ubl:ReceiptAdvice:1.
0:0.70
UBL_Library_0p70_ReceiptAdvice.xsd">
   <cat:ID>99-99-000</cat:ID>
   <cat:IssueDate>2003-02-14</cat:IssueDate>
   <cat:ReferencedDespatchAdvice>
      <cat:ID>99-99-000</cat:ID>
      <cat:IssueDate>2003-05-14</cat:IssueDate>
   </cat:ReferencedDespatchAdvice>
   <cat:BuyerParty>
      <cat:ID>T0001</cat:ID>
      <cat:PartyName>
         <cat:Name>Target DC 557</cat:Name>
      </cat:PartyName>
      <cat:Address>
         <cat:ID/>
         <cat:Street>413 Spring St.</cat:Street>
         <cat:CityName>OCONOMOWOC</cat:CityName>
         <cat:PostalZone>53066</cat:PostalZone>
         <cat:CountrySub-Entity>USA</cat:CountrySub-Entity>
      </cat:Address>
   </cat:BuyerParty>
   <cat:SellerParty>
      <cat:ID>PG0001</cat:ID>
      <cat:PartyName>
         <cat:Name>Procter and Gamble Distribution Co.
         </cat:Name>
      </cat:PartyName>
      <cat:Address>
         <cat:ID/>
         <cat:Street>1 Main Street</cat:Street>
         <cat:CityName>OCONOMOWOC</cat:CityName>
         <cat:PostalZone>52358</cat:PostalZone>
         <cat:CountrySub-Entity>USA</cat:CountrySub-Entity>
      </cat:Address>
      <cat:ShippingContact>
```

```
      <cat:ID/>
      <cat:Name>Door Mat</cat:Name>
      <cat:Phone>(763) 865-2194</cat:Phone>
   </cat:ShippingContact>
</cat:SellerParty>
<cat:DeliveryRequirement>
   <cat:ID/>
   <cat:DeliverToAddress>
      <cat:ID/>
      <cat:Street>413 N Spring St. </cat:Street>
      <cat:CityName>OCONOMOWOC</cat:CityName>
      <cat:PostalZone>53066</cat:PostalZone>
      <cat:CountrySub-Entity>USA</cat:CountrySub-Entity>
   </cat:DeliverToAddress>
</cat:DeliveryRequirement>
<!-- Auto-ID: Pampers Bibsters -->
<cat:ReceiptLine>
   <!-- Auto-ID: EPC Code -->
   <cat:ID>01.0000A89.00016F.000169DC1</cat:ID>
   <!-- Auto-ID: The next two lines indicate a short -->
   <cat:ReceivedQuantity
    unitCode="PKG">0</cat:ReceivedQuantity>
   <cat:ShortQuantity unitCode="PKG">
      1
   </cat:ShortQuantity>
   <cat:ReceivedDate>2003-05-19</cat:ReceivedDate>
   <cat:DeliverySchedule>
      <cat:ID/>
      <cat:DeliveryRequirement>
         <cat:ID/>
      </cat:DeliveryRequirement>
   </cat:DeliverySchedule>
   <cat:ReferencedTransportHandlingUnit>
      <cat:ID/>
   </cat:ReferencedTransportHandlingUnit>
</cat:ReceiptLine>
<cat:ReceiptLine>
   <!-- Auto-ID: EPC Code -->
   <cat:ID>01.0000A89.00016F.000169DC2</cat:ID>
   <cat:ReceivedQuantity
    unitCode="PKG">1</cat:ReceivedQuantity>
   <cat:ReceivedDate>2003-05-19</cat:ReceivedDate>
   <cat:DeliverySchedule>
      <cat:ID/>
      <cat:DeliveryRequirement>
         <cat:ID/>
      </cat:DeliveryRequirement>
   </cat:DeliverySchedule>
   <cat:ReferencedTransportHandlingUnit>
      <cat:ID/>
   </cat:ReferencedTransportHandlingUnit>
</cat:ReceiptLine>
<!-- Auto-ID: Bounce -->
<cat:ReceiptLine>
   <!-- Auto-ID: EPC Code -->
   <cat:ID>01.0000A89.003014.000169AB1</cat:ID>
   <cat:ReceivedQuantity
```

```
        unitCode="PKG">1</cat:ReceivedQuantity>
      <cat:ReceivedDate>2003-05-19</cat:ReceivedDate>
      <cat:DeliverySchedule>
         <cat:ID/>
         <cat:DeliveryRequirement>
            <cat:ID/>
         </cat:DeliveryRequirement>
      </cat:DeliverySchedule>
      <cat:ReferencedTransportHandlingUnit>
         <cat:ID/>
      </cat:ReferencedTransportHandlingUnit>
   </cat:ReceiptLine>
</ReceiptAdvice
```

EDI

This section contains sample EDI documents showing the EDI format of the business messages. No augmented version of the EDI will be presented.

EDI 856 Advanced Shipping Notification - Traditional

This section contains a traditional EDI 856, or Advanced Shipping Notification, with annotations describing the qualifiers. Note how minimal the line qualifiers are, resulting in compact message [102].

EXAMPLE EDI 856

EDI	Notes
ISA*00* *00* *08*9251590000 *08*6111470100 *030428*0046*U*00401*000093900*0*P*>	
GS*SH*9251590000*6111470100*20030428*0046*93889*X*004010	
ST*856*938890001	Message ID
BSN*00*06716420*20030428*0034*0001	Shipment Number, Date & Time Ship Notice Created
HL*1**S	
PO4********16*CF	
TD1*CTN25*10****G*203*LB	
TD5*B*2*P&G*M*P&G	Carrier Details
TD3*TL**8520	
REF*BM*06716420	Bill of Lading (same as Shipment Number)
REF*MB*00370002072118207	Master Bill of Lading
DTM*011*20030427	Ship Date
DTM*067*20030429	Arrive Date
FOB*PP*DE	
N1*SF*PROCTER & GAMBLE DISTRIBUTING CO.*9*001902212WY00	Ship from Organization
N4*WEST BRANCH*IA*52358	Ship from Location (City, State, Zip)
N1*ST*TARGET DC 557*92*0557	Ship to Organization
N4*OCONOMOWOC*WI*53066	Ship to Location (City, State, Zip)
HL*2*1*O	Hierarchical Level (Order)

EXAMPLE EDI 856	
PRF*0094-4316399-0557***20030423	Customer Purchase Order Number and Date
TD1*CTN25*10****G*203*LB	
REF*IA*3872918	Internal Vendor Number
N1*BY*TARGET DC 557*92*0557	
HL*3*2*T	Hierarchical Level (Tare)
MAN*GM*00100370000014190998	SSCC (Serial Shipping Container Code)
HL*5*4*P	Hierarchical Level (Pack)
LIN**UA*00370004018036*LT*3025BL01XX	UPC Case Code (GTIN) and Product and Lot ID
SN1**1*CA	Number of Cases shipped
MAN*UC*10370004018036	UCC 128 SSCC
HL*7*6*I	Hierarchical Level (Item)
LIN**UP*037000401803	UPC code (Bibsters)
SN1**6*EA	Number of Eaches shipped
PO4*1	Case Pack count or number of cartons per case, if there are cartons in the case
PID*F****35791 PAMP BIBSTERS 6/20 SMALL	
HL*9*8*P	
LIN**UA*003700080168*LT*3025BL03XX	
SN1**1*CA	
MAN*UC*10037000801686	
HL*11*10*I	
LIN**UP*037000801689	UPC code (Bounce)
SN1**6*EA	
PO4*1	
PID*F****80168 BOUNCE SHT SNGL SC 6/160 CT	
HL*13*12*P	
LIN**UA*003700040509*LT*3025BL05XX	
SN1**1*CA	
MAN*UC*10037000405099	
HL*15*14*I	
LIN**UP*037000405092	UPC code (Swiffer)
SN1**6*EA	
PO4*1	
PID*F****40509 SWIFFER DUSTER 9/5CT KIT	
HL*17*16*P	
LIN**UA*00370000340394*LT*3025BL07XX	
SN1**1*CA	
MAN*UC*10037000340393	
HL*19*18*I	
LIN**UP*037000340393	UPC code (Cascade)

EXAMPLE EDI 856	
SN1**6*EA	
PO4*1	
PID*F****37197 CASC LEMON CITRON	
HL*21*20*P	
LIN**UA*003700740920*LT*3025BL09XX	
SN1**1*CA	
MAN*UC*1003700740919	
DTM*036*20041130	
HL*23*22*I	
LIN**UP*03700740919	UPC code (Metamucil)
SN1**6*EA	
PO4*1	
PID*F****40514 METAMUCIL CAPS 12/100 CT BOTTLE	
CTT*23	Number of Hierarchical Level Segments
SE*55*938890001	
GE*10*93890	
IEA*1*000093901	

Table 21 Example EDI 856 with Qualifier Annotations

Conclusion

It is not really an option, because Core PML is not an established business message, but the format of Core PML for transmitting lists of scanned EPCs is the most succinct of the options herein presented. A close second is the Augmented Simpl-eb, with the Augmented UBL coming in a distant third. I have not made the modifications necessary to the EDI format to include EPC codes, but the standard format of EDI tags carry much less overhead than the XML based markup. To add EPCs to an EDI document, however, would require a new line qualifier, and would result in a lot of characters being added to the EDI document to handle the EPC data. The current EDI user community is hesitant to do this, as they pay per character of data transmitted.

Source Code

This section contains all of the source code for the loggers and filters used in this thesis. These code listings provide instructive examples for development projects using Savant 1.1. It should be noted that the Auto-ID SAG Working Group for the Savant is currently working a new Savant specification that may alter or eliminate the fundamental Savant framework and APIs used in this thesis. The author would like to thank Ben Griffin from Sun Microsystems for his work on many of these routines.

SoapLogger

The following is a code listing in java of the SoapLogger:

```
/*
```

```
 * SoapLogger.java
 *
 * Created on May 19, 2003, 2:37 PM
 */
package com.sun.autoid.savant;

import java.util.Properties;
import java.net.URL;
import java.net.MalformedURLException;

import org.w3c.dom.Document;

import org.autoidcenter.ems.ReaderInterface;
import org.autoidcenter.ems.Event;
import org.autoidcenter.exception.EPMFException;
import org.autoidcenter.exception.EPMFErrorCodes;
import org.autoidcenter.util.EPMFLog;
import com.sun.sjc.jms.ArgParser;

import org.apache.soap.Envelope;
import org.apache.soap.messaging.Message;
import org.apache.soap.util.xml.DOMWriter;

/** A Savant Event Management System logger that forwards
the events as
 * SOAP messages encoded in CorePML.
 * @author BenGriffin
 * @see http://www.inf.ethz.ch/~floerkem/pml/
 */
public class SoapLogger extends BatchEventLogger implements
ReaderInterface {
   private URL _endpointURL = null;
   private PMLUtilities pmlUtilities = new PMLUtilities();

   /** Creates a new instance of SoapLogger
    * @param initialization The initialization string
    * passed in from the .conf file.
    *
    * @throws EPMFException if the initialization
    * parameters are incorrect.
    */
   public SoapLogger(String initialization)
      throws EPMFException {
      super("SoapLogger");
      Properties properties =
         ArgParser.processArgs(initialization);
      String endpoint = properties.getProperty("endpoint");

      if ( endpoint == null ) {
         throw new PMFException(
            EPMFErrorCodes.BAD_LOGGER_INIT_STRING,
            "Missing endpoint name argument");
      }
      try {
         _endpointURL = new URL(endpoint);
      } catch (MalformedURLException ex) {
         throw new EPMFException(
```

```
                EPMFErrorCodes.BAD_LOGGER_INIT_STRING,
                "Malformed endpoint " + ex.getMessage());
        }
    }

    /** Shuts down this the logger */
    public void shutdown() {
    }

    /** Create a CorePML document from all the queued
     * events, and send them to the <i>endpoint</i> via a
     * soap message.
     **/
    protected void packageAndSendEvents() {
        try {
            Document doc = pmlUtilities.createCorePMLDocument(
                _recentEvents);
            if (doc != null) {
                Envelope msgEnv =
                    SOAPUtils.toSOAPEnvelope(
                        doc.getDocumentElement());
                // send the message
                Message msg = new Message();

                EPMFLog.debug("About to send xml: " +
                    DOMWriter.nodeToString(
                        doc.getDocumentElement()));

                EPMFLog.debug("Sending to endpoint: " +
                    _endpointURL.toString());
                msg.send(_endpointURL,
                    "urn:this-is-the-action-uri", msgEnv);
            }
        } catch ( Exception ex ) {
            EPMFLog.error(EPMFErrorCodes.MOD_EMS,
                "SoapLogger received an Exception while " +
                "logging events",
                ex);
        }
    }

    /** Test routine to populate the queue with a few events
     * then send them out via a soap message **/
    public static void main(String args[]) {
        try {
            SoapLogger soaplogger = new
                SoapLogger(
                    "endpoint=http://localhost:8080/
quiet_time=5000");
            soaplogger.addEvent(
                new Event(
                    Event.EPC_EVENT,System.currentTimeMillis(),
                "TAG-EPC",
                "READEREPC"));
            soaplogger.addEvent(
                new Event(
                    Event.EPC_EVENT,System.currentTimeMillis(),
```

```
            "TAG-EPC2",
            "READEREPC"));
         soaplogger.addEvent(
            new Event(
               Event.EPC_EVENT,System.currentTimeMillis(),
            "TAG-EPC3",
            "READEREPC"));

         soaplogger.packageAndSendEvents();
      } catch ( Exception ex ) {
         System.out.println(
            "Error" + ex + ex.getMessage());
         ex.printStackTrace();
      }
   }
```

EnterOnlyFilter

The following is a code listing in java of the EnterOnlyFilter:

```
/*
 * EnterOnlyFilter.java
 *
 * Created on June 12, 2003 1:33 PM
 */

package com.sun.autoid.savant;

import java.util.Timer;
import java.util.HashMap;
import java.util.TimerTask;
import java.util.Properties;
import java.util.Iterator;
import com.sun.sjc.jms.ArgParser;

import org.autoidcenter.ems.EventFilterInterface;
import org.autoidcenter.ems.ReaderInterface;
import org.autoidcenter.exception.EPMFException;
import org.autoidcenter.exception.EPMFErrorCodes;
import org.autoidcenter.util.EPMFLog;
import org.autoidcenter.ems.Event;
import com.sun.sjc.jms.ArgParser;

/** A Savant filter
 *
 * @author BenGriffin
 */
public class EnterOnlyFilter implements
EventFilterInterface {
   private ReaderInterface loggers[] = null;
   private HashMap currentField =new HashMap();
   private HashMap newField =new HashMap();
   private int _numPasses = 5;
   private int _passNumber = _numPasses;

   /** Creates a new instance of EnterOnlyFilter.
    *
```

```
 */
public EnterOnlyFilter(String initialization)
   throws EPMFException {
   Properties properties =
      ArgParser.processArgs(initialization);
   _numPasses =
      Integer.parseInt(
         properties.getProperty("number_of_passes"));

   if ( _numPasses  <= 0 ) {
      throw new EPMFException(
         EPMFErrorCodes.BAD_LOGGER_INIT_STRING,
         "invalid number_of_passes argument");
   }
}

/** Is called when an epc is read by the reader.
 * @param timestamp The time the epc was read.
 * @param epc String containing the epc.
 * @param readerEPC The identifier of the reader that
 * saw the tag.
 */
public void logEpcEvent(long timestamp,
   String epc, String readerEPC) {
   try {
      Event event =
         new Event(
            Event.EPC_EVENT,timestamp,epc,readerEPC);
      newField.put(event.getEpc(),event);
      if ( !currentField.containsKey(event.getEpc()) ) {
         currentField.put(event.getEpc(),event);
         if ( loggers != null ) {
            for ( int i = 0; i < loggers.length; i++ ) {
               loggers[i].logEpcEvent(timestamp, epc,
                  readerEPC);
            }
         }
      }
   } catch ( Exception ex ) {
      EPMFLog.error(EPMFErrorCodes.MOD_EMS,
      "EnterOnlyFilter received an Exception while " +
      "logging event at  "
      + timestamp,
      ex);
   }
}

public void logNonEpcEvent(long timestamp,
   String readingType,
   String value, String readerEPC) {
   if ( loggers != null ) {
      for ( int i = 0; i < loggers.length; i++ ) {
         loggers[i].logNonEpcEvent( timestamp,
            readingType, value,
            readerEPC );
      }
   }
```

```
    }

    public void logStatusEvent(long timestamp,
       String statusMessage) {
       if ( statusMessage.startsWith("PASS_DONE") ) {
          if (_passNumber-- <= 0) {
             _passNumber = _numPasses;
             cleanUpCurrentField();
          }
       }
       if ( loggers != null ) {
          for ( int i = 0; i < loggers.length; i++ ) {
             loggers[i].logStatusEvent( timestamp,
                statusMessage );
          }
       }
    }

    public void setListeners(
       ReaderInterface[] readerInterface)
       throws org.autoidcenter.exception.EPMFException {
       loggers = readerInterface;
    }

    protected void cleanUpCurrentField() {
       // remove all the events that haven't been seen since
       // the last checkpoint find all the events that are
       // not in newField but are in currentField

       Iterator currentKeys =
          currentField.keySet().iterator();
       while (currentKeys.hasNext()) {
          String key = (String)currentKeys.next();
          if ( !newField.containsKey(key) ) {
             currentKeys.remove();
          }
       }
       newField.clear();
    }

    public void shutdown() {
    }
}
```

PassSmoothingFilter

The following is a code listing in java of the PassSmoothingFilter:

```
/*
 * PassSmoothingFilter.java
 *
 * Created on May 27, 2003, 10:09 AM
 */

package com.sun.autoid.savant;

import java.util.HashMap;
```

```
import java.util.Properties;

import java.util.Iterator;
import com.sun.sjc.jms.ArgParser;

import org.autoidcenter.ems.EventFilterInterface;
import org.autoidcenter.ems.ReaderInterface;
import org.autoidcenter.exception.EPMFException;
import org.autoidcenter.exception.EPMFErrorCodes;
import org.autoidcenter.util.EPMFLog;
import org.autoidcenter.ems.Event;
/**
 *
 * @author  BenGriffin
 */
public class PassSmoothingFilter implements
EventFilterInterface {

   private HashMap events = new HashMap();
   private ReaderInterface loggers[] = null;
   private int _numPasses = 3;
   private int _passNumber = 0;

   /** Creates a new instance of PassSmoothingFilter */
   public PassSmoothingFilter(String initialization)
      throws EPMFException {
      Properties properties =
         ArgParser.processArgs(initialization);
      _numPasses =
         Integer.parseInt(
            properties.getProperty("number_of_passes"));

      if ( _numPasses  <= 0 ) {
         throw new
            EPMFException(
               EPMFErrorCodes.BAD_LOGGER_INIT_STRING,
               "invalid number_of_passes argument");
      }
   }

   public void logEpcEvent(long timestamp,
      String epc, String
      readerEPC) {
      try {
         if ( !events.containsKey(epc) ) {
            _passNumber = 0;
         }
         events.put(epc,
            new Event(
               Event.EPC_EVENT,timestamp,epc,readerEPC));
      } catch ( EPMFException ex ) {
         EPMFLog.error(EPMFErrorCodes.MOD_EMS,
         "PassSmoothingFiler received an Exception while "
         + "logging event at  "
         + timestamp,
         ex);
      }
```

```
    }

    public void logNonEpcEvent(long timestamp,
       String readingType,
       String value, String readerEPC) {
       if ( loggers != null ) {
          for ( int i = 0; i < loggers.length; i++ ) {
             loggers[i].logNonEpcEvent( timestamp,
                readingType, value, readerEPC );
          }
       }
    }

    public void logStatusEvent(long timestamp,
       String status) {
       if ( loggers != null ) {
          if (status.startsWith("PASS_DONE") &&
              (_passNumber++ > _numPasses)){
             _passNumber = 0;
             // send all the tags in the hashmap & the pass
             // event
             for ( int i = 0; i < loggers.length; i++ ) {
                Iterator iter = events.values().iterator();
                while (iter.hasNext() ) {
                   Event event = (Event)iter.next();
                   loggers[i].logEpcEvent(
                      event.getTimestamp(), event.getEpc(),
                      event.getReaderEpc() );
                }
                loggers[i].logStatusEvent( timestamp,
                   status);
             }
             events.clear();
          } else {
             for ( int i = 0; i < loggers.length; i++ ) {
                loggers[i].logStatusEvent( timestamp,
                   status);
             }
          }
       }
    }

    public void setListeners(
       ReaderInterface[] readerInterface)
    throws org.autoidcenter.exception.EPMFException {
       loggers = readerInterface;
    }

    public void shutdown() {
    }
}
```

EMS.conf

The following shows the contents of the EMS.conf file, and illustrates how the filters and loggers were assembled in the Savant:

```
config database "jdbc:postgresql://localhost/savant"
  user "postgres" password "postgres";

logger file_logger is org.autoidcenter.ems.logger.ConsoleLogger
   startup "name=file_logger log_type=all
   file_name=//var/opt/SUNWsavant/pgsql/log/events.log";

logger soap_logger is com.sun.autoid.savant.SoapLogger
   startup

"endpoint=http://192.168.2.114/supplier/servlet/messagerouter";

filter smoother is com.sun.autoid.savant.PassSmoothingFilter
   startup "number_of_passes=3" output (file_logger soap_logger);

filter enter_only is com.sun.autoid.savant.EnterOnlyFilter
   startup "number_of_passes=5" output (smoother);

public queue main_queue size 1000 output (enter_only);

adapter adapter_name is
com.oatsystems.ems.adapter.thingmagic.ThingMagicAdapter
   startup "poll_time=2000 monitor_poll_time=100000 timeout=5000
   reader_locations=10.0.0.1/UHF1" for main_queue
```

SavantController

The following is a code listing in java of my SavantController:

```
/*
 * SavantController.java
 *
 * Created on June 18, 2003, 11:05 PM
 */

package org.autoidcenter.utils;

/**
 *
 * @author  tmilne
 */

//Import the java lang package
import java.lang.Runtime;
import java.io.IOException;

//Import the java swing package.
import javax.swing.*;

//Import the java Color package.
import java.awt.Color;

public class SavantController extends javax.swing.JFrame {

   /** Creates new form SavantController */
   public SavantController() {
      initComponents();
   }
```

```
/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this
 * method is always regenerated by the Form Editor.
 */
private void initComponents() {
//GEN-BEGIN:initComponents
   jbExit = new javax.swing.JButton();
   jpSavant = new javax.swing.JPanel();
   jtbSavant = new javax.swing.JToggleButton();
   jtfSavant = new javax.swing.JTextField();
   jpAppServer = new javax.swing.JPanel();
   jtbAppServer = new javax.swing.JToggleButton();
   jtfAppServer = new javax.swing.JTextField();

   addWindowListener(
      new java.awt.event.WindowAdapter() {
      public void windowClosing(
         java.awt.event.WindowEvent evt)
      {
         exitForm(evt);
      }
   });

   jbExit.setText("Exit");
   jbExit.addActionListener(
      new java.awt.event.ActionListener(
      {
      public void actionPerformed(
         java.awt.event.ActionEvent evt) {
         jbExitActionPerformed(evt);
      }
   });

   getContentPane().add(jbExit,
      java.awt.BorderLayout.SOUTH);

   jpSavant.setLayout(new java.awt.BorderLayout());

   jtbSavant.setText("  Start Savant  ");
   jtbAppServer.setText(sStartAppServer);
   jtbSavant.addActionListener(
      new java.awt.event.ActionListener() {
         public void actionPerformed(
            java.awt.event.ActionEvent evt) {
               jtbSavantActionPerformed(evt);
      }
   });

   jpSavant.add(jtbSavant,
      java.awt.BorderLayout.CENTER);

   jtfSavant.setBackground(java.awt.Color.red);
   jtfSavant.setEditable(false);
   jtfSavant.setHorizontalAlignment(
      javax.swing.JTextField.CENTER);
   jtfSavant.setText("Off");
```

```
        jtfSavant.setText(sOff);
        jtfSavant.setPreferredSize(new java.awt.Dimension(49,
            20));
        jtfSavant.setAutoscrolls(false);
        jpSavant.add(jtfSavant, java.awt.BorderLayout.SOUTH);

        getContentPane().add(jpSavant,
            java.awt.BorderLayout.CENTER);

        jpAppServer.setLayout(new java.awt.BorderLayout());

        jtbAppServer.setText("Start App Server");
        jtbAppServer.setText(sStartAppServer);
        jtbAppServer.addActionListener(
            new java.awt.event.ActionListener() {
                public void actionPerformed(
                    java.awt.event.ActionEvent evt) {
                        jtbAppServerActionPerformed(evt);
            }
        });

        jpAppServer.add(jtbAppServer,
            java.awt.BorderLayout.CENTER);

        jtfAppServer.setBackground(java.awt.Color.red);
        jtfAppServer.setEditable(false);
        jtfAppServer.setHorizontalAlignment(
            javax.swing.JTextField.CENTER);
        jtfAppServer.setText("Off");
        jtfAppServer.setText(sOff);
        jtfAppServer.setPreferredSize(
            new java.awt.Dimension(49, 20));
        jtfAppServer.setAutoscrolls(false);
        jpAppServer.add(jtfAppServer,
            java.awt.BorderLayout.SOUTH);

        getContentPane().add(jpAppServer,
            java.awt.BorderLayout.NORTH);

        pack();
    }//GEN-END:initComponents

    private void jbExitActionPerformed(
        java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_jbExitActionPerformed
        System.exit(0);
    }//GEN-LAST:event_jbExitActionPerformed

    private void jtbSavantActionPerformed(
        java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_jtbSavantActionPerformed
        // Check the button state:
        if (evt.getActionCommand().equals(sStartSavant))
        {
            //Start the Savant.
            try{
                Runtime.getRuntime().exec(
```

```
            "/opt/SUNWsavant/bin/savant_s1a7 start");

            //Wait for the Savant to start up.
            sleep(38);

            //Change the state.
            jtbSavant.setText      (sStopSavant);
            jtfSavant.setText      (sOn);
            jtfSavant.setBackground(cOn);

         } catch (IOException ex){
            JOptionPane.showMessageDialog(this,
                     "Error: Starting Savant",
                     "Error", JOptionPane.ERROR_MESSAGE);

            //Change the state.
            jtbSavant.setSelected  (false);
         }
      }
      else
      {
         //Stop the Savant
         try{
            Runtime.getRuntime().exec(
               "/opt/SUNWsavant/bin/savant_s1a7 stop");

            //Wait for the Savant to stop.
            sleep(10);

            //Change the state.
            jtbSavant.setText      (sStartSavant);
            jtfSavant.setText      (sOff);
            jtfSavant.setBackground(cOff);

         } catch (IOException ex){
            JOptionPane.showMessageDialog(this,
                     "Error: Stopping Savant",
                     "Error", JOptionPane.ERROR_MESSAGE);

            //Change the state.
            jtbSavant.setSelected  (true);
         }
      }
   }//GEN-LAST:event_jtbSavantActionPerformed

   private void jtbAppServerActionPerformed(
       java.awt.event.ActionEvent evt)
   {//GEN-FIRST:event_jtbAppServerActionPerformed
      // Check the button state:
      if (evt.getActionCommand().equals(sStartAppServer))
      {
         //Start the App Server.
         try{
            Runtime.getRuntime().exec(
               "asadmin start-appserv");

            //Wait for the App Server to start up.
```

```
            sleep(90);

            //Change the state.
            jtbAppServer.setText      (sStopAppServer);
            jtfAppServer.setText      (sOn);
            jtfAppServer.setBackground(cOn);

         } catch (IOException ex){
            JOptionPane.showMessageDialog(this,
                     "Error: Starting App Server",
                     "Error", JOptionPane.ERROR_MESSAGE);

            //Change the state.
            jtbAppServer.setSelected  (false);
         }
      }
      else
      {
         //Stop the App Server
         try{
            Runtime.getRuntime().exec(
               "asadmin stop-appserv");

            //Wait for the App Server to stop.
            sleep(15);

            //Change the state.
            jtbAppServer.setText      (sStartAppServer);
            jtfAppServer.setText      (sOff);
            jtfAppServer.setBackground(cOff);

         } catch (IOException ex){
            JOptionPane.showMessageDialog(this,
                     "Error: Stopping App Server",
                     "Error", JOptionPane.ERROR_MESSAGE);

            //Change the state.
            jtbAppServer.setSelected  (true);
         }
      }
   }//GEN-LAST:event_jtbAppServerActionPerformed

   /** Exit the Application */
   private void exitForm(java.awt.event.WindowEvent evt)
   {//GEN-FIRST:event_exitForm
      System.exit(0);
   }//GEN-LAST:event_exitForm

   private void sleep(int seconds){
      //Calculate milliseconds
      int milliseconds = seconds*1000;

      //Sleep for the number of seconds passed in
      try {
         Thread.sleep(milliseconds);
      } catch (InterruptedException e) {
         JOptionPane.showMessageDialog(this,
```

```
                    "Error: Sleeper interrupted",
                    "Error", JOptionPane.ERROR_MESSAGE);
            }
        }

        /**
         * @param args the command line arguments
         */
        public static void main(String args[]) {
            new SavantController().show();
        }


        // Variables declaration - do not modify//GEN-
    BEGIN:variables
        private javax.swing.JTextField jtfSavant;
        private javax.swing.JToggleButton jtbSavant;
        private javax.swing.JButton jbExit;
        private javax.swing.JPanel jpSavant;
        private javax.swing.JPanel jpAppServer;
        private javax.swing.JToggleButton jtbAppServer;
        private javax.swing.JTextField jtfAppServer;
        // End of variables declaration//GEN-END:variables

        private String sStartAppServer = "Start App Server";
        private String sStopAppServer  = "Stop  App Server";
        private String sStartSavant    = "  Start Savant  ";
        private String sStopSavant     = "  Stop  Savant  ";
        private String sOn             = "On";
        private String sOff            = "Off";
        private Color  cOn             = Color.green;
        private Color  cOff            = Color.red;
    }
```

Glossary

a-Biz - Auto-ID Enabled Business

a-Biz is the Auto-ID Joint Project that will eventually consider the integration of Auto-ID technology with many real world business use-cases, thus enabling "Automated Business", or a-Biz.

ANSI - American National Standards Institute

The American National Standards Institute is a body that recommends a wide variety of standards used in North America ranging from weights and measures to EDI standards.

ASN - Advanced Shipping Notification

The ASN is a document that is sent ahead of the shipped goods to give notice that they are in transit and to convey the composition of the shipment. Also referred to as Despatch Advice (DA).

ASP - Active Server Pages

Active Server Pages or ASP, as it is more commonly known, is a Microsoft technology that enables you to make dynamic and interactive web pages. ASP uses server-side scripting to dynamically produce web pages that are not affected by the type of browser the web site visitor is using. The default scripting language used for writing ASP is VBScript, although you can use other scripting languages like JScript (Microsoft's version of JavaScript).

BIS – Business Information System

Business Information System, or BIS, is the system used to handle information about commerce transactions.

BOL - Bill of Lading

The BOL is an inventory sent along with a shipment that lists the items in the shipment and the associated PO(s).

Chargeback

A chargeback is an adjustment made to an invoice to reconcile an error in the shipment. This results in a credit with the Sending Party. See also Deduction.

Data Channel

A data channel is a pipeline for transmitting data and can include a phone call, fax, or VAN.

Deduction

A deduction is an adjustment made to an invoice to reconcile an error in a shipment. This reduces the amount paid to the Sending Party. See also Chargeback.

Despatch Advice

The DA is a document that is sent ahead of the shipped goods to give notice that they are in transit and to convey the composition of the shipment. Also referred to as an Advanced Shipping Notification (ASN).

DNS - Domain Name Service

DNS is an abbreviation for Domain Name Service (or System), a system for naming computers and network services that is organized into a hierarchy of domains. DNS naming is used in TCP/IP networks, such as the Internet, to locate computers and services through user-friendly names.

EAN - European Article Numbering

The EAN system, administered by the EAN International and similar to the UCC system, is used in Europe.

EAN.UCC

The EAN International and the UCC have joined together under the EAN.UCC to create open, global, multisectoral standards based on best business practices, and

drive their implementation, play a leading role in supply & demand chain management improvement worldwide.

EDI - Electronic Data Interchange

Developed to allow different programs using proprietary data formats to talk to each other using an *EDI Standard Format*.

EDIFACT - Electronic Data Interchange For Administration, Commerce, and Transportation

EDIFACT is a form of EDI sponsored by the United Nations and is the basis of international trade and trade within countries that have adopted EDI as their trading standard.

EMS - Event Management System

Part of the Savant architecture, the EMS connects readers to applications by managing the event flow generated by the reader. You can write loggers, and filters to control the destination and flow of the events through the EMS.

EPC - Electronic Product Code

The EPC is the unique code used to identify an object in the Auto-ID infrastructure. It is similar to other numbering schemes (GTIN, UPC, VIN and others).

Filter

A filter is used to suppress the amount of data being passed through the Savant's EMS.

GSMP - Global Standards Management Process

The Global Standards Management Process (GSMP) develops and maintains standards based solutions for global trade using the enabling EAN.UCC System Technologies.

GTIN - Global Trade Item Number

The GTIN, administered by the EAN.UCC, assigns a unique number to each product class, and is globally unique. Ideal for international trade, it does not include item level serialization.

HTML - Hypertext Markup Language

HTML is based on SGML, and is used for marking up documents suitable for display in a web browser.

HTTP - Hypertext Transfer Protocol

The hypertext transfer protocol is the protocol used to transmit HTML and related documents types over the Internet.

ISO - International Standards Organization

A network of national standards institutes from 147 countries working in partnership with international organizations, governments, industry, business and consumer representatives.

J2EE - Java 2 Enterprise Edition

J2EE is a set of Java based standards for electronic commerce. The J2EE platform manages the infrastructure and supports the Web services to enable development of secure, robust and interoperable business applications.

JSP - Java Server Pages

JSP, which stands for Java Server Pages, is a server side technology that controls the content of web pages through the use of servlets, small programs that are defined in the web page and run on the server to modify the page before it's downloaded to the user who requested it. JSP technology is built on top of servlets, a portable Java program that provides server side processing.

Logger

A logger is used to define the endpoint or destination of an information stream in the Savant. It can be a database, a flat file, or a port on a remote server that is waiting for the incoming data.

Message Format

The message format may be an augmented EDI format, an instantiation of ebXML, UBL or many other payloads that are available.

Mixed Pallet

The procedure where multiple, different items are stacked or "mixed" on the same pallet. This occurs often with direct store delivery or cross-docking operations, where pallets are assembled at the manufacturer's site for a specific store to minimize the handling and reconfiguration in the distribution network that would otherwise be necessary.

.NET

Microsoft .NET is a set of software technologies for connecting information, people, systems, and devices. This new generation of technology is based on Web services—small building-block applications that can connect to each other as well as to other, larger applications over the Internet.

ONS – Object Naming Service

The Object Naming Service, or ONS, is a component of the Auto-ID framework, and performs a name resolving function similar to the Domain Naming Service employed by the Internet today.

OSB - Order Shipping Billing

The OSB is a system or suite of processes and tools used to handle all of the operations involved in the order process from order to payment. It is as much concept as much as it is an actual system.

PML – Physical Markup Language

The Physical Markup Language is used by the Auto-ID infrastructure to communicate information about physical objects.

PO - Purchase Order

The PO is sent from the receiving party to the sending party to initiate an order.

RA - Receipt Advice

The RA is a document that is sent by the Receiving Party back to the Sending Party after the physical goods have been received. The RA is used to signify that the shipment was ok, or to convey problems about the shipment (over/under/damage). Also referred to as a Confirmation Receipt.

Receiving Party

Any party receiving a shipment of goods (i.e. retailer, warehouse, etc.).

RFID - Radio Frequency Identification

RFID refers to the transmission of an identification code over an air interface using radio waves.

Savant

The Savant is a part of the Auto-ID framework. It is a globally distributed server that serves as a data router performing the functions of data capturing, data monitoring, and data transmission.

Sending Party

Any party sending goods (i.e. manufacturer, packager etc.).

Shrinkage

A logistics term used for goods that are lost, stolen or damaged in the supply chain delivery process.

Simpl-eb - Simple Electronic Business

Simpl-eb is a framework or set of business messaging standards being developed by the UCC for conducting electronic business.

SGML - Standard Generalized Markup Language

SGML is the predecessor of many of the other markup languages in current use, including HTML and XML. It can be used as a direct markup language, or used to define other markup languages. In fact, XML and HTML have been defined with SGML. Due to its unwieldy size, it does not enjoy wide spread use.

SQL - Structured Query Language

SQL is a structured language used to query and retrieve information from standard data stores.

SOAP - Simple Object Access Protocol

SOAP is an XML-based protocol (originally written by Microsoft, IBM, and others) for the message-based exchange of distributed application components and information. [103]

SRV - Shipping and Receiving Verification

Shipping and Receiving Verification is the process by which perfect orders and prepared, shipped and received using an Auto-ID enabled mechanism for verification.

SSCC - Serialized Shipping Container Code

The SSCC is set forth by the EAN.UCC and is used to mark and identify logistics units during transport. It is unique for a particular aggregate logistics unit, and if any changes are made it's associated item or bundle of items, it is invalidated and set aside.

TMS - Task Management System

The TMS is part of the Savant, and coordinates processes initiated by higher level Savants.

Transporting Party

Any party that transports the goods (i.e. dedicated fleet or third party carrier).

UBL - Universal Business Language

UBL is an ebXML-based business-messaging framework that includes the notion of Despatch Advice and Receipt Advice, along with other useful business messages.

UCC - Uniform Code Council

The UCC is a family of wholly-owned subsidiaries, divisions, and partnerships that connects companies in the supply chain with standards-based solutions that are universally open, industry-driven, and globally endorsed.

UDDI - Universal Description, Discovery, and Integration

UDDI is a specification (driven initially by IBM, Microsoft, and Ariba) composed of SOAP application programming interfaces required to enable a service broker and ultimately simplify inter-enterprise integration. [103]

URI - Uniform Resource Identifier

The generic set of all names/addresses that are short strings that refer to resources.

URL - Uniform Resource Locator

An informal term (no longer used in technical specifications) associated with popular URI schemes: http, ftp, mailto, etc.

URN - Uniform Resource Name

A URI that has an institutional commitment to persistence, availability, etc. Note that this sort of URI may also be a URL.

Alternately, a particular scheme, urn:, intended to serve as persistent, location-independent, resource identifiers.

UML – Unified Modeling Language

The Unified Modeling Language, or UML, is a descriptive modeling framework for modeling the requirements and business processes via Use-Cases, Activity Diagrams, etc.

UPC - Universal Product Code

A product code published by the EAN.UCC that embeds manufacturer and product class information.

VAN - Value Added Network

A Value Added Network refers to the dedicated network of telecommunications equipment, links and services with which companies conduct business using EDI.

Valid XML Document

Follows all of the rules of a well-formed document, but in addition it follows strictly all the rules set forth in its schema or DTD.

VIN - Vehicle Identification Number

The VIN is used in the automotive industry to uniquely identify individual vehicles.

Well Formed XML Document

A well-formed XML document follows all of the XML syntax rules, but may not be valid. To be valid, it must be checked against the DTD or schema. However, a document need only be well formed for browsers to accept the XML that has already been validated by the server, so there is no need to download the DTD or schema to the client browser and revalidate.

WSDL - Web Services Description Language

WSDL is an XML vocabulary (driven by IBM, Microsoft, and Ariba) for describing Web services interfaces, defining the published service operations, and defining definition the location and binding details of the service. [103]

REFERENCES

[1] Netscape Corporation. *ECXpert Site Administrator's Handbook The Netscape ECXpert System Version 1.1.* Published September, 1998. http://developer.netscape.com/docs/manuals/ecxpert/admin/sadmb.htm, July, 2003.

[2] http://www.x12.org, July, 2003.

[3] http://www.ansi.org, July, 2003.

[4] http://www.unece.org/trade/untdid/welcome.htm, July, 2003.

[5] http://www.un.org, July, 2003.

[6] Research on the Slow Adoption of Traditional EDI. Published May, 1996. http://www.icaris.net/icaris/tradedi.html, July, 2003.

[7] Fu, S. et al, *A Practical Approach to Web-Based Internet EDI*, IBM IAC, T. J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598. Published 1999. http://www.research.ibm.com/iac/papers/icdcsws99.pdf

[8] Steel, K. *The Case for Next Generation EDI*, The University of Melbourne, Department of Computer Science. Published 21 June, 1995.

[9] Hogan, M., *XML versus EDI*, Published July, 2001. http://www.esuppliersolutions.com/eu/articles/ContentWire_070401.htm

[10] Ericson, J., *EDI and the Internet*, Published Jan, 2002. http://www.line56.com/articles/default.asp?NewsID=3287

[11] Electronics Industry Data Exchange Association. *Traditional EDI via a VAN.* Published March, 2003. http://www.eidx.org/publications/techinfo/techopt1.html, July, 2003.

[12] ASCX12 Press Release: *ASC X12 Defines 2003 Strategic Direction, Develops New EDI Messages in X12 & XML Formats.* http://www.x12.org/x12org/prdoc.cfm?Name=780, July 2003.

[13] UN-XML EEMA EDI/EC Work Group http://www.edi-tie.nl/edifact/xml-edi.htm, July, 2003.

[14] Oracle Corporation. *Traditional EDI and XML/EDI: A Reality Check* http://www.oracle.com/appsnet/technology/integration/collateral/williams_2.pdf

[15] http://xml.coverpages.org/ni2001-08-22-c.html, July, 2003.

[16] http://www.globalcommerceinitiative.org/oas/gci/gci.home, July, 2003.

[17] http://www.uc-council.org, July, 2003.

[18] http://www.rosettanet.org, July, 2003.

[19] http://www.ebxml.org, July, 2003.

[20] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl, July, 2003.

[21] *The Case for Global Standards: Creating the Business Case for Global Data Synchronisation in Your Company.* Oct 2002. Cap Gemini Ernst & Young/Global Commerce Initiative.

[22] http://www.autoidcenter.org, July, 2003

[23] Kundapur, N. *On Integrating Physical Objects with the Internet*, Masters Thesis, MIT, Cambridge, Massachusetts. Published June, 2000.

[24] Goyal, A., *Synchronized Exchange of Material and Information*, Masters Thesis, MIT, Cambridge, Massachusetts. Published May, 2003.

[25] Milne, T.P. *Auto-ID Business Use-Case Framework (A-Biz) - Background.* Technical Report MIT-AUTOID-TM-009. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Nov 1, 2002. http://www.autoidcenter.org/research.asp

[26] Milne, T.P. *Auto-ID Business Use-Case Framework (A-Biz) - Despatch Advice Use-Case.* Technical Report MIT-AUTOID-TM-010. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Nov 1, 2002. http://www.autoidcenter.org/research.asp

[27] Brock, D. L., *The Electronic Product Code (EPC) A Naming Scheme for Physical Objects*, White Paper MIT-AUTOID-WH-002. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Jan 1, 2001. http://www.autoidcenter.org/research.asp

[28] http://www.sae.org/technicalcommittees/vin.htm, July, 2003

[29] Saar, S., Thomas, V., *Toward Trash That Thinks Product Tags for Environment Management*, Journal of Industrial Ecology, Volume 6, Number 2 P. 133. MIT Press. http://mitpress.mit.edu/catalog/item/default.asp?sid=4418CA7E-1673-4ED0-B857-F8C5AB986473&ttype=6&tid=9887

[30] UCC Inc., *Serialized Shipping Container Code (SSCC) Implementation Guide.* Published May, 2002 http://www.uc-council.org/ean_ucc_system/pdf/SSCC.pdf

[31] http://www.eccpage.com, July, 2003.

[32] http://web.usna.navy.mil/~wdj/reed-sol.htm, July, 2003.

[33] Adams, R. *Bar Code 1.* http://www.adams1.com, July, 2003.

[34] Adams, R. *2-Dimensional Bar Code Page.* http://www.adams1.com/pub/russadam/stack.html, July, 2003.

[35] A Historical Account of Radio Frequency Identification, July, 2003. http://web.ics.purdue.edu/~pobanz/Final/RFID%20history.htm

[36] Eagle, J., *RFID: the Early Years 1980-1991.* Published Sept, 2002 http://members.surfbest.net/eaglesnest/rfidhist.htm, July, 2003.

[37] Scharfield, T. *An Analysis of the Fundamental Constraints On Low Cost Passive Radio-Frequency Identification System Design.* The Auto-ID Center. Cambridge, Massachusetts. Published Aug 1, 2001.

[38] http://www.rafsec.com/products.htm, July, 2003.

[39] Scharfeld, T., *Compliance and Certification: Ensuring RFID Interoperability,* Technical Report MIT-AUTOID-TR017. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published June 1, 2003. http://www.autoidcenter.org/research.asp

[40] Overby, C., *RFID: The Smart Product (R)evolution.* Forrester Research, Cambridge, MA. Published August, 2002.

[41] Finkenzeller, K., *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification,* 2nd edition, Wiley & Sons LTD, ISBN: 0-470-84402-7, Published May, 2003.

[42] http://www.rfidjournal.com/article/articleview/477/1/1/, July, 2003.

[43] http://www.w3.org/XML/, July, 2003.

[44] w3c.org, *Extensible Markup Language (XML) 1.0 (Second Edition).* Published Oct, 2000. http://www.w3.org/TR/2000/REC-xml-20001006, July, 2003

[45] http://www.w3.org

[46] http://www.w3.org/MarkUp

[47] http://www.w3.org/XML/1998/06/xmlspec-report.htm, July, 2003.

[48] http://www.w3.org/XML/Schema, July, 2003.

[49] Kocur, G., *Database, Internet and Systems Integration Technologies.* Course Notes, 1.264, MIT. Cambridge, Massachusetts. Fall, 2002

[50] International Standards Organization, *Information processing -- Text and office systems -- Standard Generalized Markup Language (SGML).* http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16387, July, 2003.

[51] http://www.iso.org, July, 2003.

[52] http://www.ean-ucc.org/global_smp/gsmp_smp.htm, July, 2003.

[53] http://usnet03.uc-council.org/smp/xml_schemas_and_business_messa.html, July, 2003.

[54] EAN.UCC, *Business Modeling Group (BMG) Business Requirements Analysis for Simple Despatch Advice.* Published Jan, 2001. http://www.uc-council.org

[55] http://www.uml.org, July, 2003.

[56] Fowler, M., Scott, K., *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 2nd Edition, Addison-Wesley Publishing Co., ISBN: 020165783X, Published Aug, 1999.

[57] http://usnet03.uc-council.org/smp/schemas.html, July, 2003.

[58] Brock, D. L., *The Compact Electronic Product Code, a 64-bit Representation of the Electronic Product Code*, White Paper MIT-AUTOID-WH-008. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Nov 1, 2001. http://www.autoidcenter.org/research.asp

[59] Engels, D., *EPC-256: The 256-bit Electronic Product Code Representation.* Technical Report MIT-AUTOID-TR-010. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published May 1, 2003. http://www.autoidcenter.org/research.asp

[60] Engels, D., *The Use of the Electronic Product Code.* Technical Report MIT-AUTOID-TR-009. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published May 1, 2003. http://www.autoidcenter.org/research.asp

[61] Oat Systems and MIT Auto-ID Center. *The Object Name Service - Version 0.5 (Beta).* Technical Manual MIT-AUTOID-TM-004. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Feb 1, 2002. http://www.autoidcenter.org/research.asp

[62] Oat Systems and MIT Auto-ID Center. *The Savant - Version 0.1 (Alpha),* Technical Manual MIT-AUTOID-TM-003. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Feb 1, 2002.http://www.autoidcenter.org/research.asp

[63] Brock, D., Milne, T. P., Kang, Y., Lewis, B. *The Physical Markup Language, Core Components: Time and Place.* White Paper MIT-AUTOID-WH-005. The Auto-ID

Center, MIT. Cambridge, Massachusetts. Published Jun 1, 2001. http://www.autoidcenter.org/research.asp

[64] Floerkemeier, C., Koh, R.. *Physical Mark-Up Language Update.* Technical Memo MIT-AUTOID-TM-006. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published September 1, 2002. http://www.autoidcenter.org/research.asp

[65] Harrison, M., Moran, H., Brusey, J., McFarlane, D., *PML Server Developments*, White Paper CAM-AUTOID-WH-015. The Auto-ID Centre, UK. Cambridge, England. Published Jun 1, 2003. http://www.autoidcenter.org/research.asp

[66] Janakiraman, M., *Web Service Standards - A Survey.* http://www.gca.org/papers/xmleurope2001/papers/html/s30-1.html, July, 2003.

[67] Willaert, F., *XML-Based Frameworks and Standards for B2B ECommerce.* http://www.ebxml.org/documents/ebxml-thesis.pdf, July, 2003.

[68] http://java.sun.com/j2ee, July, 2003.

[69] http://www-3.ibm.com/software/info1/websphere/index.jsp?tab=products/appserv, July, 2003.

[70] http://wwws.sun.com/software/products/integration_srvr_eai/home_int_eai.html, July, 2003.

[71] http://e-docs.bea.com/wls/docs81/index.html, July, 2003.

[72] http://www.borland.com/besappserver/index.html, July, 2003.

[73] http://www.oracle.com/ip/deploy/ias, July, 2003.

[74] http://www.i2.com, July, 2003.

[75] http:// www.manugistics.com, July, 2003.

[76] http://www.sap.com, July, 2003.

[77] W3C.org, *SOAP Version 1.2 Part 1: Messaging Framework*, published June, 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624, July, 2003

[78] http://www.microsoft.com/net, July, 2003.

[79] http://www.uddi.org, July, 2003.

[80] W3C.org, *Web Services Description Language (WSDL) Version 1.2 Part 1: Core Language*, Working Draft, June, 2003, http://www.w3.org/TR/2003/WD-wsdl12-20030611, July, 2003.

[81] http://www.transora.com, July, 2003.

[82] Sarma S., Weis S.A., Engels D.W. *RFID Systems, Security, & Privacy Implications.* White Paper MIT-AUTOID-WH-014. The Auto-ID Center. Cambridge, Massachusetts. Published Nov 1, 2002.

[83] Secure Processing. http://www.accpac.com/products/exchange/SecureProcessing.asp, July, 2003.

[84] Secure Trading Agent. http://www.sun.com/software/download/products/Sun_tm__ONE_Integration_Server,_Secure_Trading_Agent.html, July, 2003.

[85] Moran, H., McFarlane, D., Milne, T., *The development of Use Cases as an alternative approach to identify the impact of Auto-ID implementations on Business*, White Paper AUTOID-WH-016. The Auto-ID Centre, UK. Cambridge, England. Published Jun 1, 2003. http://www.autoidcenter.org/research.asp

[86] Milne, T., *SAG - Shipping and Receiving Use Case.* Last Call Working Draft. The Auto-ID Center. Cambridge, Massachusetts. Published June, 2003.

[87] Roemer, K., Schoch, T., *Infrastructure Concepts for Tag-Based Ubiquitous Computing Applications*, Department of Computer Science, ETH Zurich. Published 2002

[88] UN/CEFACT, *UN/CEFACT ebXML Core Components Technical Specification, Part 1*, http://www.unece.org/cefact/ebxml/ebXML_CCTS_Part1_V1-8.pdf, July, 2003.

[89] McConnell, S., *Rapid Development: Taming Wild Software Schedules*, 1st edition, Microsoft Press, ISBN: 1556159005, Published July, 1996.

[90] Milne, T. P., *Business Information and Industrial Control Action Group: Sub Group and Use Case Focus Group Methodology*, dated 20 November, 2002, unpublished.

[91] http://www.postgresql.org, July, 2003.

[92] http://www.gorillalogic.com, July, 2003.

[93] Core PML Specification. http://www.inf.ethz.ch/~floerkem/pml, July, 2003.

[94] EAN.UCC Inc. https://solutionscenter.uc-council.org/index.cfm, July, 2003.

[95] Koh, R., Schuster, E., Chackrabarti, I., Bellman, A., *Securing the Pharmaceutical Supply Chain*, White Paper MIT-AUTOID-WH021. The Auto-ID Center. Cambridge, Massachusetts. Published Jun 1, 2003.

[96] Koh, R., Schuster, E., Lam, N., *Prediction, Detection, and Proof: An Integrated Auto-ID Solution to Retail Theft*, White Paper MIT-AUTOID-WH022. The Auto-ID Center. Cambridge, Massachusetts. Published Jun 1, 2003.

[97] http://uri.net and http://uri.net/urn-nid-status.html, July, 2003

[98] http://www.w3.org/Addressing/, July, 2003

[99] http://www.w3.org/TR/2001/NOTE-uri-clarification-20010921, July, 2003.

[100] http://asg.web.cmu.edu/rfc/rfc3305.html, July, 2003.

[101] EAN.UCC, *Business Requirements Document (BRD) for Despatch Advice.* Published 2002.

[102] Used with permission of Procter & Gamble Inc. and Target Corporation.

[103] Sholler, D. *SOAP, UDDI, and WSDL Defined.* Meta Group. Published April 9, 2002. http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2859619-6,00.html, July, 2003.