

Synchronized Exchange of Material and Information

by

Amit Goyal

S.B., Computer Science (2001)
Massachusetts Institute of Technology

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

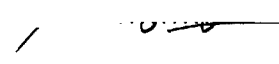
July 26, 2002

© Massachusetts Institute of Technology 2002. All Rights Reserved.

Author.....

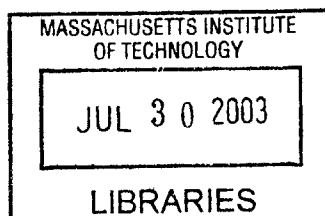
Department of Electrical Engineering and Computer Science
May 9, 2003

Certified By.....


Sanjay E. Sarma
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted By.....

Arthur C. Smith
Chairman, Department Committee on Graduate Theses



BARKER

Synchronized Exchange of Material and Information

by

Amit Goyal

Submitted to the
Department of Electrical Engineering and Computer Science
May 9, 2003

In partial fulfillment of the requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Commerce is all about the carefully managed exchange of material, money, and information. Traditionally, the connection between material and information has been tenuous, with humans acting as the intermediaries. This has made the supply chain inefficient and expensive. The Auto-ID Center has created a stronger, automatic link between inanimate objects and computers. This thesis completes the information exchange, or feedback loop, which makes commerce possible. Specifically, it identifies a framework for information exchange alongside material exchange using Savant-to-Savant communication. Messaging standards will need to support the Auto-ID Center's technology, and this thesis suggests how to augment existing and emerging communication standards to accomplish this feat. Finally, to address the issue of increasing information management, this thesis analyzes the aggregation database, an IT infrastructure component that might be of value to organizations. The outcome of this thesis is an understanding of the various issues necessary to develop a secure, efficient and robust system for tracking and automatically confirming material exchange.

Thesis Supervisor: Sanjay E. Sarma

Title: Associate Professor of Mechanical Engineering

Acknowledgements

I have never really had the opportunity to thank anyone formally as I am about to do now. Being my first opportunity, I have an urge to thank everyone for all the help it took to get me here, in all the different times and areas of my life. That would be extremely difficult to do though, so I'll settle with acknowledging only those people that had an immediate impact on this document.

First and foremost, I thank my friend and thesis advisor, Sanjay Sarma. I will miss our talks. I took to heart your advice and thoughts on life. They liberated me. I have grown in the two years under your guidance. The strong personal relationship you allowed me to have with you helped me discover myself. In the academic world, I never imagined I would have the freedom you afforded me. I learned so much. Thank you.

To Junius Ho, Nosh Petigara, and Arundhati Singh, my friendly labmates. It was not the same after you left. I enjoyed our uplifting conversations, but I enjoyed commiserating even more. They are some of my most fond memories over the last two years. I never anticipated enjoying the basement of building 1, but you guys made it happen.

To Robin Koh, Tom Scharfield, Dan Engles, and especially Kevin Ashton and Tim Milne. I've spoken with each of you a number of times individually - sometimes on a personal level and other times more professionally. You all have made my time at 400 Technology Square warm and comfortable. You all were always around for a joke, a serious discussion, or just to answer a question. Although Sanjay provided the blueprint, your daily level interaction was the bricks and mortar that made this thesis possible.

To Prasad Putta and the people at OAT Systems, thank you for always being there for me. Prasad, you patiently answered my initial questions. And when the details were holding me down, there was always an OAT Systems technologist I could call.

To Randy Smith, Owen Densmore, Chris Clauss, and Varun Gupta at Sun Microsystems, thank you for helping me develop the industry expertise I needed to feel confident about my work. Randy and Owen in particular helped me accomplish a tremendous amount. They claimed they did very little, but without them, I would not have had the vision and the technical ability to accomplish anything.

To Chetak Reshamwala and Dave Kong, my dear friends and roommates who were always there when I needed emotional support or a good laugh, I will miss seeing you everyday. You guys are brothers to me.

To Riaz Dhanani, Ashwin Krishnamurthy, Shrey Kumar, Vinod Natarajan, Sanjay Raman, Sailu Challapalli, Jeyun Choi, and Ruchi Shrivastava, I never would have made it this far without initially making it through undergrad. I miss you all dearly.

And finally, to my loving parents Dev P. Goyal and Madan L. Goyal, and my brothers Manish Goyal and Ashish Goyal. Any true success is ultimately credited to all of you. As always, my deepest love and appreciation.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION.....	7
CHAPTER 2: BACKGROUND	8
2.1 RADIO FREQUENCY IDENTIFICATION (RFID).....	8
2.1.1 <i>How Does an RFID System Work?</i>	8
2.2 THE AUTO-ID CENTER	9
2.3 AUTO-ID CENTER TECHNOLOGIES	10
2.3.1 <i>The Savant</i>	10
2.3.1.1 The Savant Event Management System	10
2.3.2 <i>Electronic Product Code (EPC)</i>	11
2.3.3 <i>Object Name Service (ONS)</i>	12
2.3.4 <i>Physical Markup Language (PML)</i>	12
CHAPTER 3: AUTOMATING BUSINESS PROCESSES	13
3.1 CURRENT BACKEND SYSTEMS	13
3.2 SAVANT-TO-SAVANT COMMUNICATION	14
3.2.1 <i>Messaging vs. Referencing</i>	14
3.2.2 <i>General Savant-to-Savant Infrastructure</i>	14
3.2.3 <i>Secure Message Communication</i>	16
3.3 WORKFLOW ENGINES	17
CHAPTER 4: CASE STUDY: SHIPPING AND RECEIVING VERIFICATION (SRV).....	20
4.1 OVERVIEW	20
4.2 ASN CHALLENGES	23
4.3 AUTOMATED SHIPPING AND RECEIVING VERIFICATION.....	24
4.3.1 <i>The Message Format</i>	24
4.3.1.1 EDI and the EPC.....	24
4.3.1.2 Simpl-eb and the EPC.....	26
4.3.2 <i>Communication Channel</i>	27
4.3.3 <i>Savant-to-Savant Sample Workflow</i>	27
4.3.4 <i>Workflow Exceptions</i>	29
4.3.4.1 Case 1: Receiving Party's Savant Malfunctions	29
4.3.4.2 Case 2: Shipping Party's Savant Malfunctions	30
4.3.4.3 Case 3: Shipping Party Does Not Support Auto-ID Center Technology..	31
4.4 IMPLEMENTATION	32
CHAPTER 5: THE AGGREGATION DATABASE - AN IT INFRASTRUCTURE IMPLICATION	33
5.1 DESIGN CONSIDERATIONS.....	33
5.2 DB DESIGN	34
5.3 DB ANALYSIS.....	34
5.4 AGGREGATION DB CONCLUSION.....	36

CHAPTER 6: CONCLUSION 37
 6.1 FUTURE WORK 37
REFERENCES..... 39
APPENDIX..... 42

LIST OF FIGURES

Figure 1: Basic RFID Components.....	8
Figure 2: Auto-ID System.....	9
Figure 3: The Savant’s Event Management System [21]	11
Figure 4: Enterprise Terminology [9].....	13
Figure 5: Messaging Service Features/Differentiators [10].....	16
Figure 6: Security Levels.....	16
Figure 7: Workflow Diagram [15].....	18
Figure 8: Sample Workflow GUI [27].....	18
Figure 9: Basic Shipping and Receiving Verification Process [16].	21
Figure 10: Excerpt from LIN Segment Definition [18].....	24
Figure 11: Excerpt from MAN Segment Definition [18]	25
Figure 12: Excerpt from a Sample 214 Document [19].....	25
Figure 13: Excerpt from Simpl-eb Identification.xsd [22]	27
Figure 14: Savant-to-Savant Automated SRV	28

CHAPTER 1: Introduction

RFID adoption in industry has only taken place thus far in a select few applications. It failed to make greater impact in industry because of the lack of standards and its high cost. The Auto-ID Center has introduced standards in the RFID industry, and it has proposed a way to reduce RFID tag cost when manufactured in high volumes [1]. Because of the interoperability standards make possible and the technology's lower cost, industry now has an incentive for utilizing this technology on a large scale. RFID/Auto-ID technology will most likely be applied first in applications which improve efficiency in the supply chain. This will help overcome the initial investment of installing the system [2].

RFID/Auto-ID technology can automate many business processes and reduce their cost. Automating the shipping and receiving verification process will provide significant return on investment because it will expedite the receiving process and eliminate the human data entry error component [2,3]. One market research firm claims more than 85% reduction in receiving time, from 2 hours to 15 minutes, owing to the elimination of manual scanning and data entry [2]. Such automation will also enable inventory tracking at key points in the supply chain and facilitate supply chain visibility [4]. Visibility helps manufacturers determine current demand as well as the amount of inventory present in the supply chain. This information allows them to appropriately modify their output, thereby reducing the bullwhip effect and high stock-out situations [3, 4]. In addition to all these benefits, tests can be conducted in closed environments which are invisible to consumers. Given all these favorable conditions, automated shipping and receiving has been identified as the entry point for the Auto-ID Center's technology into organizations' infrastructures.

This thesis explores the various components of an automated shipping and receiving system. It suggests ways to augment existing communication standards with Auto-ID Center technology, and it proposes a new communication infrastructure using the Auto-ID Center core infrastructure components.

The thesis focuses on several scenarios in which the Savant, an Auto-ID Center infrastructure component whose role includes inter-organization communication, fails to operate properly. The motivation for developing these scenarios is to determine whether the Auto-ID system is resilient to such system failures. Through the analysis, this thesis also identifies certain preventive measures the system can take to minimize overall supply chain disruption.

Finally, this thesis discusses the aggregation database, an optimized component for storing hierarchy information that could be added to the Auto-ID Center's technology offering to assist organizations with increased information management. This component is just one of the many data structures that will need to be developed to capture physical realities associated with item level tracking.

CHAPTER 2: Background

This chapter focuses on the Auto-ID Center (AIDC) and the information technology infrastructure AIDC proposes. It then supplements that discussion with a discussion of RFID. Please note that today, RFID is the enabling technology for the Auto-ID IT infrastructure. Tomorrow, the enabling technology could change - but the IT infrastructure would stay the same.

2.1 Radio Frequency Identification (RFID)

RFID systems consist of readers and tags. Readers can read tags from distances ranging from a few to tens of feet. They are particularly attractive because they do not require contact, and therefore do not suffer from direct line-of-sight limitations. Tags can also be read *through* some materials.

RFID tags come in a variety of shapes and sizes and are categorized as either active, semi-passive, or passive. Active tags use an on-board battery to power tag functionality, and they have a transmitter for communication. Passive tags harvest energy from the field emitted by the reader, and they lack a transmitter, operating instead by scattering back incident radiation. Semi-passive tags communicate by backscatter, but use battery-power for on-board functionality such as digital signal processing, noise filtering, *etc.* [24].

2.1.1 How Does an RFID System Work?

A basic RFID system consists of an antenna for the reader, a reader, and one or more RFID tags.

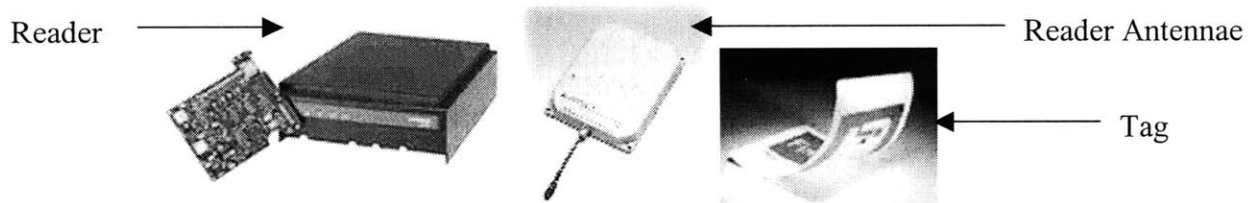


Figure 1: Basic RFID Components

The antenna emits radio waves to power the tag and modulates commands and data on top of these waves. The reader antenna also receives signals back from the tag. Common reader-tag operating frequencies are 13.56 MHz, 868-870 MHz (EU), 915 MHz (US), and 2.45 GHz. At 13.56 MHz, the reader uses inductive coupling to power the tag. At higher frequencies, passive and semi-passive tags use "back scattering" to communicate with the reader. In this scheme, the tag modulates its reflectance so the reader sees its own reflected signal varying in time so as to encode the tag's data [24].

When multiple tags are in a reader's field, they will each respond to the reader's signal, possibly interfering with each other. Consequently, the reader and tag system employs a

tag anti-collision algorithm. These algorithms are encoded in the reader-tag communication and are invisible to the outside world.

Anti-collision algorithms can be probabilistic or deterministic. One deterministic algorithm involves searching the "binary tree" of bits to determine which tags are in its field. For example, this algorithm might ask for all the tags with first bit 0 to respond. If multiple tags respond, then it might search deeper into the tree, asking for tags with initial bits 00 or 01. Depending on the responses, it might go further, until it eventually determines the ID of a single tag. At that point, the search backtracks and goes down another branch of the binary tree. Depending on the environment, operation frequency, and other considerations, a deterministic algorithm might function better than a probabilistic.

Upon completing a pass of the field, the reader generates events indicating the presence of each tag it found. Higher-level components pull this information from the reader to process the events, apply intelligence, and perform useful functions.

2.2 The Auto-ID Center

The Auto-ID Center is an industry-funded organization headquartered at M.I.T. whose goal is to develop the technology infrastructure for managing the next generation identification system to follow the Universal Product Code (UPC). It proposes the use of RFID tags as the enabling technology. The new "barcode" will be an RFID tag, most likely passive, which contains an Electronic Product Code (EPC). Today, the EPC is a 96-bit unique identifier, although built into its structure is the ability to support larger and smaller bit sizes. The EPC will be used to identify items at instance level. Unique identification is different from the current UPC barcode, which only identifies items at the class level. After a higher level processing system pulls the information from the reader, it will resolve the EPC into an Internet Protocol (IP) address through the use of Object Naming Service (ONS). ONS is similar to DNS, the system that resolves Internet domain names into IP addresses. After resolving an IP address, the system may then retrieve from that site instance level information about a particular item on the Internet. This information will be described using a new XML representation scheme called Physical Markup Language (PML).

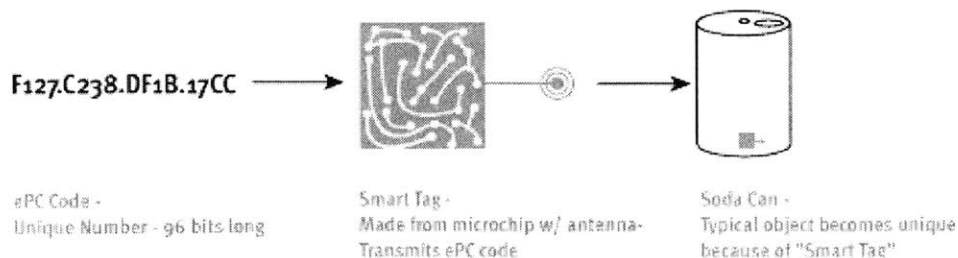


Figure 2: Auto-ID System

If each object currently using a barcode were to make use of the EPC, then the managing infrastructure would need to handle billions of events every second. To manage this

massive flow of events, the Auto-ID Center has proposed the use of modular components called Savants [23]. One way to construct a system would be to stack savants in a hierarchy, in which the lower level Savants would process, filter, and digest events, and the upper level Savants would perform aggregation to different levels of granularity. To reduce network traffic, a Savant may just forward events of interest or event summaries to higher level Savants.

The Auto-ID Center's I.T. infrastructure has been designed with the philosophy of minimalism. At its heart, the Auto-ID Center's IT infrastructure requires an object to identify itself with a unique number. All the information regarding that object is associated on the network with a direct link constructed from the unique number.

The only reason for this infrastructure to ever change would be if the fundamental principles upon which it was designed were to change.¹ If users wanted to change to a system without a network (or less reliance on the network), then more information than that single number may need to be stored on the object, or greater latency may need to be tolerated when the object crosses from one LAN to another. Users may eventually do this because the cost of maintaining a network might one day be greater than storing information directly on the item.

2.3 Auto-ID Center Technologies

2.3.1 The Savant

The Savant has 3 main components: a Real-time in-memory event database (RIED), a Task Management System (TMS), and an Event Management System (EMS). RIED is an optimized database that achieves some of its performance gains by supporting a limited subset of SQL. The TMS coordinates processes initiated by higher level Savants. The EMS connects readers to applications by managing the event flow generated by the reader. When discussing the Savant, the EMS is most often the subject. It is the most tangible part of the system, and it provides a platform on which users may compose interesting applications.

2.3.1.1 The Savant Event Management System

The diagram below illustrates how events are propagated through the Savant system. Everything in the box labeled "Savant" comprises the entity we refer to as the Savant Event Management System. As illustrated on the bottom of the diagram, many readers are capable of connecting to the Savant. A reader is a device that generates events and only communicates existence information. For example, it will state that a tag is present, but it will not state that the tag is absent. Absence is a higher-level notion that the Savant will have to determine on its own. Even if the tags in the reader's field do not change, the

¹ Note that this statement does not take into account security. The only additional functionality necessary for an object besides having a unique identity would be for it to have the intelligence of verifying the interrogator before revealing its own identity. This ability would require some security related information and processing power on the tag, but a deeper discussion of this topic is beyond the scope of this thesis [25].

reader continues to emit a steady event stream indicating all the tags it sees. In addition to announcing EPCs and the associated timestamp at which those EPCs were seen, a reader has the ability to emit non-EPC information (temperature, humidity, etc.) as well as status events. One status event might tell the Savant system a single pass of the reader's field has just been completed.

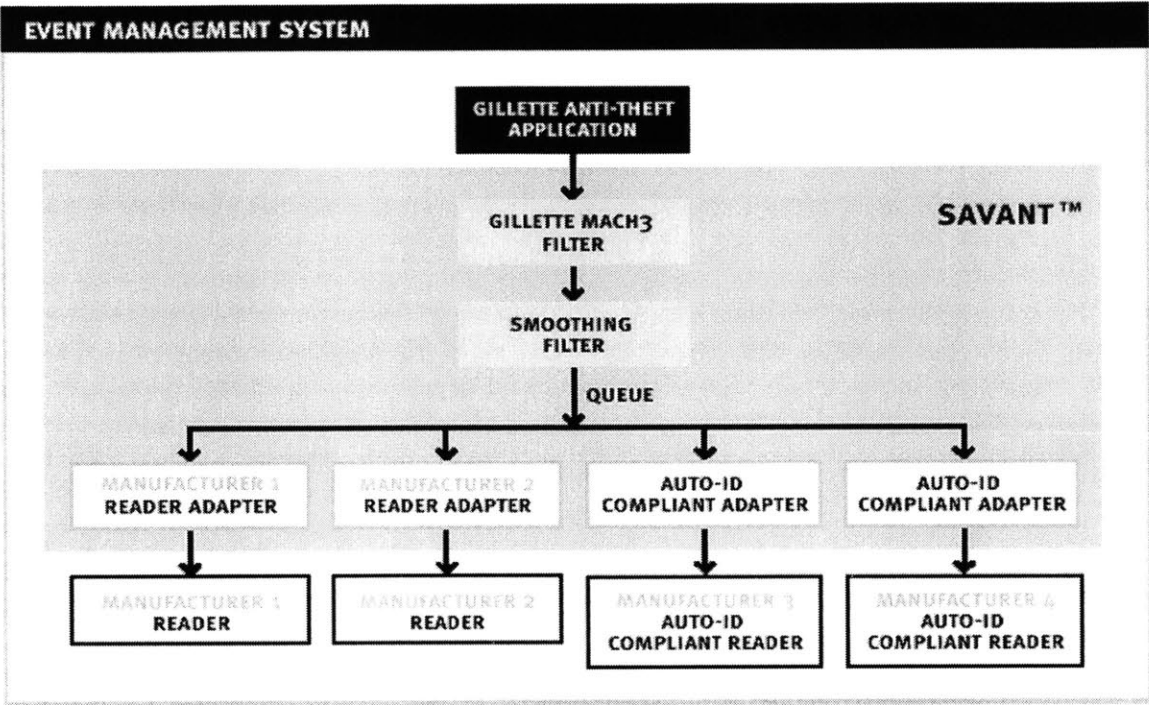


Figure 3: The Savant’s Event Management System [21]

These readers may connect to the computer world through a number of different physical means (serial, Ethernet, etc.), and they may support different, proprietary access protocols. To support readers of various types, the Savant has manufacturer-and-reader-specific *adapters* that handle the implementation details of connecting to different readers. If the reader is Auto-ID compliant, then only a single Auto-ID compliant adapter, independent of the manufacturer, is necessary.

After events are produced by the readers and consumed by the reader adapters, they are forwarded to a queue. In this queue, events are automatically forwarded to filters. Depending on how the filter is defined, certain events may be "filtered" out. The filters forward the relevant events to applications for higher level processing.

2.3.2 Electronic Product Code (EPC)

The Auto-ID Center proposes EPCs as the new numbering scheme for identifying objects through the web [5]. EPCs are globally unique identifiers that enable tracking of goods down to the item level. The current UPC only identifies classes of items, such as Coke versus Pepsi. The EPC will not only differentiate between classes, it will also differentiate between members of the same class, such as two individual Coke cans.

Similar to an IP address, EPCs contain a built-in structure that facilitates data retrieval on a large-scale (refer to ONS section). EPCs may be of varying length. To minimize the tag costs initially, a 64-bit EPC is the low-end version. For other applications, a 96-bit EPC might be more appropriate. The structure of the EPC is such that the first few bits identify the version number of the EPC. Based on the version number, the system resolving the EPC can obtain the information necessary to process it.

2.3.3 Object Name Service (ONS)

The Auto-ID Center developed ONS as a mechanism to resolve EPCs into an Internet address [6]. ONS is similar to DNS in its function and its operation. ONS resolves the EPC into an EPC prefix and appends that to the EPC suffix to obtain a domain name. ONS was designed to work alongside DNS to utilize the existing Internet infrastructure.

2.3.4 Physical Markup Language (PML)

The Auto-ID Center envisions PML, an XML based language, as the lingua franca that will allow different entities to communicate [7]. For information to be understood among multiple industries and organizations, standards must be developed for describing that information. The Auto-ID Center has divided PML into two different logical spaces -- PML core and PML extensions [8]. PML-core represents that information which needs to be standardized among all entities. Examples of PML-core include time and physical measurements. PML extensions describe those pieces of information that are industry specific. For example, although the average consumer may want to refer to a car as a single entity, the automotive industry will likely view a car as an intricate aggregation of parts. To accommodate these different perspectives, the Auto-ID Center promotes industries to develop their own PML extension standards for their particular needs.

CHAPTER 3: Automating Business Processes

To automate business processes, industries can either build on top of their existing infrastructure or introduce an entirely new technology. In every scenario, there are two vital questions the industry have to address. First, in what format will information be communicated? Second, what is the channel for communicating this information? The first question pertains to the syntax and semantics of a message between two companies, and the second to the mechanism of messaging in a message.

Generally speaking, standards for syntax and semantics are important and facilitate dialogue with various players. The message will need to support necessary Auto-ID Center technology, such as the EPC and possibly some PML information. The next chapter discusses how Auto-ID technology affects specific existing message formats when trying to automate the shipping and receiving verification business process.

In terms of the channel, organizations must decide whether they will continue using the current enterprise backend systems or if they will incorporate the Auto-ID Center's vision of Savant-to-Savant communication.

3.1 Current Backend Systems

It is difficult to discuss with any detail the general design of industry's current backend systems. Each corporation has its own unique way of handling information. As such, members of the technical community have developed a vocabulary for dealing with these various systems.

Enterprise Terminology	
Legacy	Term describing applications and data " that have been inherited from languages, platforms, and techniques earlier than current technology."
ERP (Enterprise Resource Planning)	Term "for the broad set of activities supported by multi-module application software that helps a manufacturer or other business manage the important parts of its business, including product planning, parts purchasing, maintaining inventories, interacting with suppliers, providing customer service, and tracking orders."

Figure 4: Enterprise Terminology [9]

It is with ERP and legacy systems that industry will have to integrate Auto-ID technology if they want the functionality Auto-ID introduces. The obstacles of doing so are great. Many legacy applications and ERP systems are incapable of supporting the granularity of information Auto-ID makes possible. Also, the sheer volume of information suggested by an automatic tracking system like the Auto-ID Center's would require a remodeling of many current IT infrastructures. Perhaps most fundamentally, however, many legacy ERP systems simply do not provide the adequate hooks to enable any serious integration.

3.2 Savant-to-Savant Communication

It is the Auto-ID Center's vision that companies should be able to communicate through Savants. Doing so would give them visibility into each other's inventory through a standardized component. Today, many companies fail to share information because of the inherent difficulty in interfacing with various information systems. The links between immediate traders might be strong, but industry does not have a general infrastructure in place to leap-frog a member in the supply chain.

Currently, standards do not exist for Savants to communicate with one another. The Auto-ID Center's Software Action Group (SAG) is currently working to define these interfaces. The Center has expressed an interest in adopting existing and emerging standards as long as they have are open-interface and well designed. Even without these definitions, it is possible to discuss the general infrastructure and some of its implications.

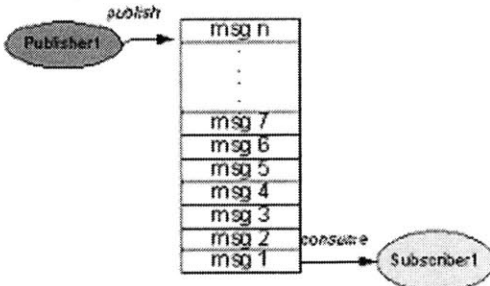
3.2.1 Messaging vs. Referencing

When communicating information, companies have the option of content messaging, i.e. sending the entire message, or referencing, i.e. sending a link that points to the information on the Internet. Messaging allows information "pre-positioning," while referencing requires real-time access over the network. "Pre-positioning" information means pushing the information to a certain location, anticipating its need, so when the system is ready to utilize the resource it does not have to spend time retrieving it. In a real-world environment where network access and servers often go down, the need for immediate access to information favors pre-positioning. Pre-positioning also allows a system to perform preliminary processing. Should the receiving system determine an error, it may be able to preemptively stop certain actions from occurring. As such, a messaging model will most often be used, but a referencing system could still be used as backup.

3.2.2 General Savant-to-Savant Infrastructure

When Savants communicate with one another, it is necessary there exist some intermediary to buffer the messages. For distinction, let us refer to the Savant sending the message as the producer, and the Savant receiving the message as the consumer. In this scenario, the producer may send a message while the consumer is down. Given that the consumer is down, the message would never be received and neither party would realize an error occurred until the message was needed.

To prevent this problem, message arrival may be guaranteed by having a reliable intermediary messaging service. A messaging service has the ability to act as an intelligent intermediary buffer. There are numerous messaging services available. Some of the important features that can also act as differentiators among messaging services are shown below.

Messaging Service Features	
Point-to-Point messaging model	The strict model enforces that a single publisher and consumer connect to a single queue. More general models allow multiple consumers and subscribers to attach themselves to the same queue. In this more general case, only a single subscriber need consume a message before that message gets removed from the queue. The process in which multiple subscribers consume from a queue is implementation specific and may or may not be deterministic.
Publish-subscribe messaging model	This model allows multiple publishers and subscribers to attach themselves to the same queue. Each registered subscriber receives each published message.
Non-persistent delivery	Guarantees the broker will only attempt to deliver the message once. If the consumer is unavailable or the broker is restarted, the message may never reach the intended recipient. The benefit of this quality of service is that messages are processed extremely quickly.
Persistent delivery	<p>Guarantees the broker will deliver the message exactly once to the intended recipient. The downside of this guarantee is that persistent messages are saved in some data repository (i.e. file, database), which incurs more processing overhead than their non-persistent counterparts. The obvious benefit is that persistent messages are guaranteed to eventually reach the intended destination.</p> <p>Relevant example: An ATM sends a deposit message to the IT system responsible for maintaining account balances. Persistent message delivery would be important here to maintain accurate account information, even if the update took longer than desired.</p>
Synchronous message consumption	<p>This feature enforces first-in, first-out (FIFO) message consumption. When the order in which messages are sent is important, this feature is useful. For example, online communication instant messengers like AIM (AOL Instant Messenger) need to perform FIFO consumption to ensure coherent dialogues.</p> <div style="text-align: center;">  <pre> graph LR P(Publisher1) -- publish --> Q[Queue] subgraph Queue direction TB Q1[msg 1] Q2[msg 2] Q3[msg 3] Q4[msg 4] Q5[msg 5] Q6[msg 6] Q7[msg 7] Qn[msg n] end Q -- consume --> S(Subscriber1) </pre> </div> <p style="text-align: center;">Synchronous message consumption off the queue.</p>
Asynchronous message consumption	Using this option, message consumption acts independently of message sent order. Messages may be sent with a priority level, allowing high priority messages to be consumed even in the presence of a long queue. Email is an example of when this might be useful. There may be a long queue of junk email with low priority when a high priority email arrives. Even though there are messages before it,

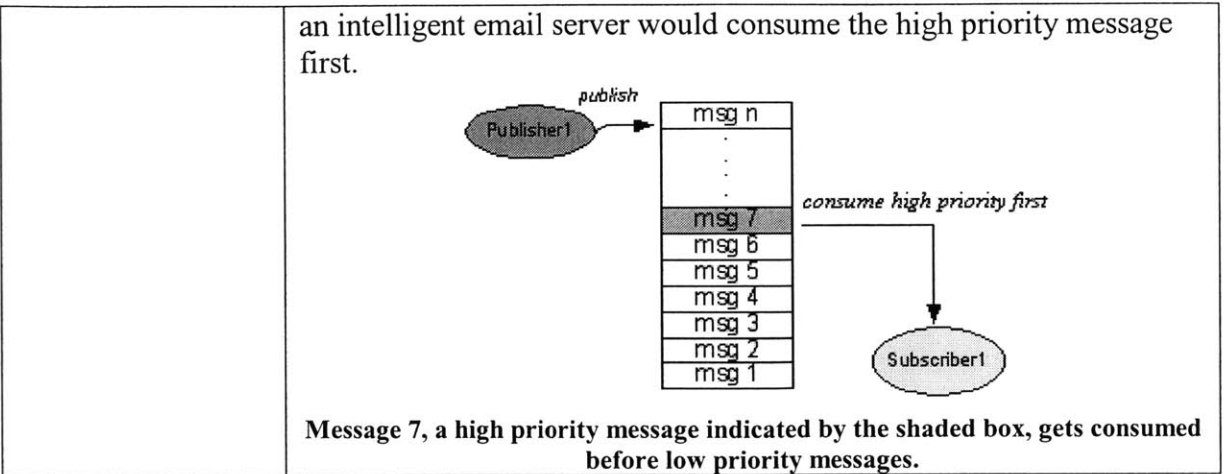


Figure 5: Messaging Service Features/Differentiators [10]

Depending on the implementation, the sender may actually have the ability to specify persistence within the message itself. This is a valuable feature because the message may be extremely time critical, and there may be no point delivering it after an elapsed time. Other features deal with defining the number of allowed producers and subscribers as well as how the subscribers consume the messages on the queue.

3.2.3 Secure Message Communication

Security can refer to any or all of the following capabilities: authentication, confidentiality, and integrity. Sometimes, authorization is also included in this list; but it is more a matter of maintaining access privileges and controlling access once authentication has been accomplished. Since sensitive message-communication is the subject of discussion, only the core three apply here.

There are various levels of security, and it is necessary while discussing security to clearly identify the current level being discussed.

Security Levels		
Level	Type	Description/Examples
1	Cryptographic algorithms	Mathematical functions generate public-private keys and symmetric keys
2	Key distribution	Diffie-Hellman key exchange, PKI, etc.
3	Implementations	Kerberos vs. digital signatures

Figure 6: Security Levels

The lowest security level comprises mathematical functions used to encrypt and decipher text. The term "key" often refers to a variable value used in conjunction with a mathematical function. Symmetric keys are generated at the same time and share a secret. If one of the keys becomes compromised, then it can be used to generate the other key, and messages signed by these keys can be deciphered. The alternative to symmetric key encryption is public-private key encryption. It is extremely difficult to

mathematically compute the private key from the public key. In this scenario, the sender keeps a key that he generates, termed the private key, and he distributes freely the public key.

The next level discusses how these various keys are distributed. Public key infrastructure (PKI) is one popular method of distributing keys. Using this system, a trusted third party distributes public keys [11]. Once an organization obtains another organization's public key, it can then authenticate the message and decrypt it. One alternative to PKI is Diffie-Hellman key exchange. Using this protocol, two users can exchange a secret key over an insecure medium without any prior secrets [12].

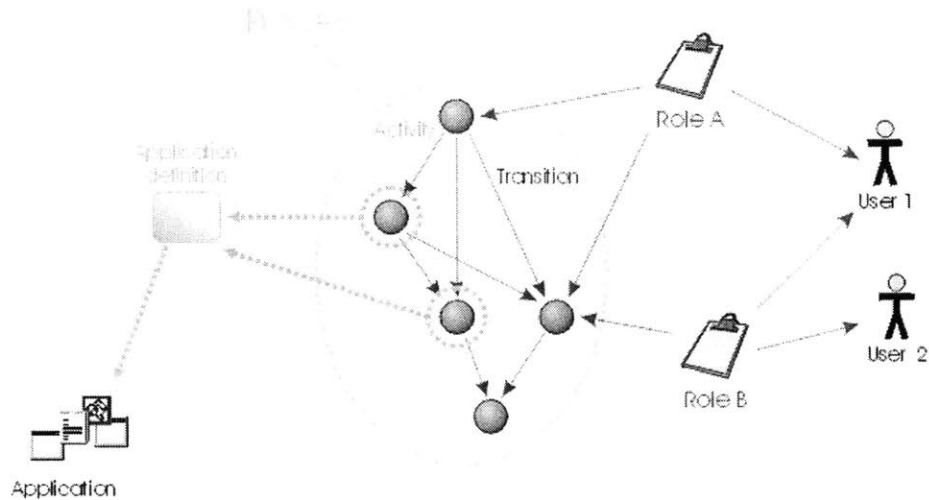
At the highest level are the various implementations to achieve secure communication. Two implementations that achieve varying degrees of secure communication are digital signatures and Kerberos. The digital signature technique achieves authentication and integrity, but not confidentiality [13]. It does this by having the sender create a hash of the message and then signing that hash with his private key. This hash is sometimes referred to as the message's "fingerprint," and one algorithm for obtaining such a fingerprint from a message is with the MD-5 algorithm [26]. Anyone wishing to confirm authenticity may decipher the hash with the sender's public key and confirm it by hashing the original message. As long as the two hashes match, the authenticator knows the message is unchanged.

Another security implementation commonly used within organizations is Kerberos. Kerberos achieves confidentiality, authentication, and integrity. Kerberos is an attractive option inside organizations because it can authenticate a user to a number of services without requiring the user to supply his password each time [14]. It does this by means of a trusted ticket-granting server that issues certificates to access services based on the presence of an initial valid certificate for which the user supplied a password.

Although Kerberos is a popular option inside organizations, for inter-organization communication it is less used. Inter-organization communication would require the constant use of a trusted server that is sitting outside both organizations' networks. In contrast, PKI infrastructures only use a trusted server initially to obtain the sender's public key. In addition, the issue of transitive trust poses a major problem for Kerberos. In transitive trust, after a single ticket-granting server gives a user access to resources, that user can then use the initial certificate to authenticate himself against another ticket-granting server that has access to extra resources. The system therefore can unintentionally give a user access to sensitive resources

3.3 Workflow Engines

In addition to the Savant architecture, enterprises will likely want to develop workflow engines to automate their business processes. Generally speaking, a workflow engine describes "who must do what, when, and how" [15]:



Who must do what when and how

Figure 7: Workflow Diagram [15]

A workflow engine has a graphical user interface that allows a user to visually define the information flow while incorporating external applications.

The screenshot shows the 'Manny Couceiro's Home Page' with the following sections:

- Main:** Home, Index, News, Tracker, Help, Logout
- Projects:** SEC
- Open Issue:** Open
- Anomalies:** New, Browse, Search, Reports
- Requests:** New, Browse
- Questions:** New, Browse
- Tips:** New, Browse
- Documentation:** E2Nc, Tech Notes, Press Releases

News section:

Date	Headline	Link
29 Feb 2001	Hello from the Konica beta	http://it.konica.com/news/egload.txt
28 Feb 2001	Please look at the press release from AHT and Konica	http://it.konica.com/news/egload.txt
9 May 2000	SEC is No Longer Accepting Escalations for IP Controller	

In Box section:

Date	Project	Issue	Description
25 Aug 2000	Marketing	Q-2056	Issue answered

Tracker section:

Changed	Project	Issue	Type	Status	Description
29 Nov 2000	Planning	F-2042	Feature Request	open	Feature Request submitted: Subset Stapling Customers have been asking for the ability to send a 1 set job (PostScript, Unix, A3H00) with subset stapling commands. Currently there is no good way to accomplish this for 2 reasons: software & engine limit
12 Oct 2000	Marketing	Q-2073	Marketing Q & A	open	Question submitted: How do you download the Postnet barcodes sent to the Force 60?
11 Oct 2000	Marketing	Q-2000	Marketing Q & A	open	Question submitted: When will Konica support Linux?

Figure 8: Sample Workflow GUI [27]

It should have some built-in functionality for monitoring the workflow's status, measuring the workflow, and simulating the workflow. Only recently has the first open source workflow engine started being developed [15]. For industry to incorporate workflow engines requires a considerable amount of work, especially when considering the existing legacy systems already in place. Whether the engines used in the future are generic, specific, proprietary, or open source, only time can tell. Workflow engines will

inevitably become components of an automated system, as only they can fully utilize the various resources and reformat that information in a form understandable to the user.

CHAPTER 4: Case Study: Shipping and Receiving Verification (SRV)

There was a time when a business owner would oversee all the details of his or her organization. When purchasing goods and selling them, both the seller and the buyer would carefully inspect each item before going forward with the transaction. After the two individuals shook hands, all sales were final and each went his way.

In its essence, the same steps are repeated today: businesses exchange goods and money. As businesses have grown though, a number of factors have changed the topology of this basic transaction. Organizations' sheer size has forced them to separate the person who places the order from the receiver. Specialization has pushed traders into using third party distributors which leverage economies of scale to run a shipping business more efficiently. And finally, electronic communication has enabled companies to eliminate costly human intermediaries.

Although these optimizations have succeeded on some level, they have introduced their own unique problems. Outsourcing shipping, for instance, complicates inventory reconciliation because it is difficult to identify the owner when inventory shrinkage and damage occur. Electronic communication requires communication standards to be agreed upon. Despite using electronic communication, inaccurate, incomplete, or simply late data can enter into the system and introduce a slew of new problems.

Businesses today deal with these numerous problems by defaulting back to human support. For example, large retailers have divisions assigned to order reconciliation. The Auto-ID Center will optimize and automate many of these business processes through mass serialization and inventory tracking. Imagine a fully automated store in which a monitoring application confirms the number of goods arrived before any shrinkage occurs. To achieve this holy grail of a fully automated store backend, the Auto-ID Center must determine a system to automate information exchange alongside material good exchange.

4.1 Overview

The basic process of exchanging goods between companies can be highlighted as follows:

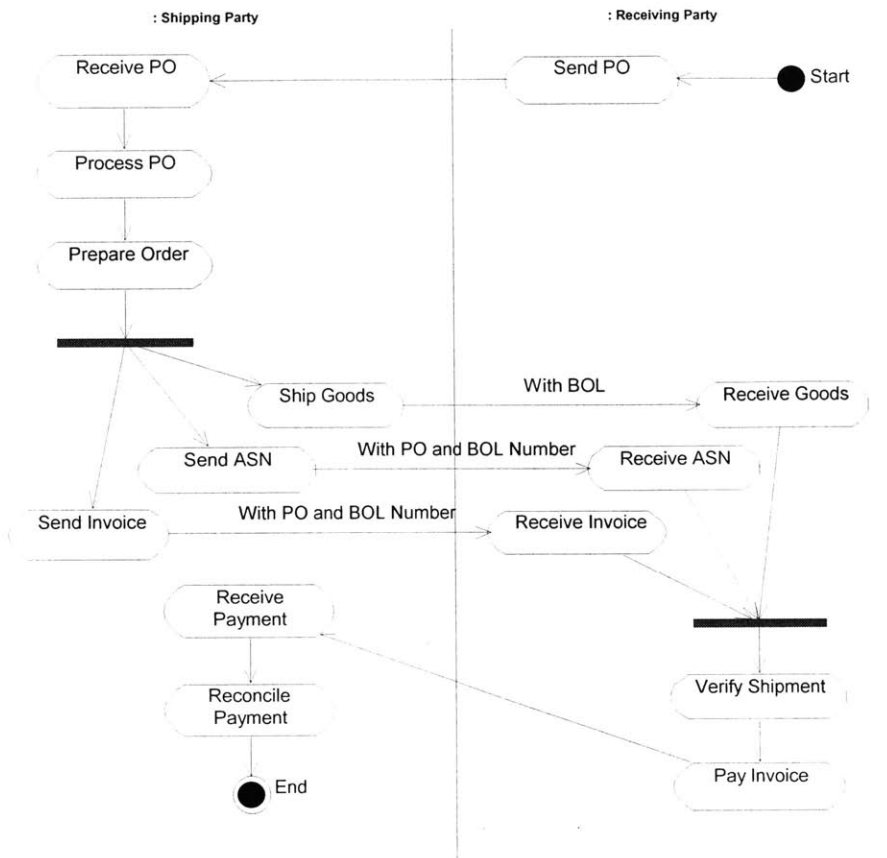


Figure 9: Basic Shipping and Receiving Verification Process [16]. Much gratitude goes to Target Corporation and P&G for insight into these business processes.

Here is the associated, detailed event flow:

- M.1.** Receiving Party:
 - M.1.1.** Sends Purchase Order (PO) to Shipping Party
 - M.1.1.1.** PO specifies carton codes
 - M.1.1.2.** Includes requested arrival date/time and other Terms of Sale
- M.2.** Shipping Party:
 - M.2.1.** Calculates a ship-date based on the requested arrival date from Shipping Party
 - M.2.2.** Processes order
 - M.2.3.** Picks the goods
 - M.2.3.1.** Scans the carton codes on the physical goods
 - M.2.3.2.** Visually confirms carton codes specified in the PO
 - M.2.4.** Sends an Advanced Shipping Notification (ASN) to the Receiving Party
 - M.2.4.1.** Includes the carton codes and BOL number

- M.2.5.** Ships the physical goods to the Receiving Party
 - M.2.5.1.** Goods accompanied by a Bill of Lading (BOL)
- M.2.6.** Sends an invoice to the Receiving Party
 - M.2.6.1.** Includes the carton codes and BOL number
- M.3.** Receiving Party:
 - M.3.1.** Receives the ASN with the carton codes and BOL number
 - M.3.2.** Receives the physical goods with BOL
 - M.3.2.1.** Scans the carton codes on the physical goods with Savant and Reader
 - M.3.2.2.** Manually reconciles the received goods with the PO, ASN, invoice and BOL data
 - M.3.2.3.** Sends a confirmation receipt to the Shipping Party (optional)
 - M.3.3.** Receives the invoice with the carton codes and BOL number
 - M.3.4.** Sends payment to the Shipping Party
 - M.3.4.1.1.** Adjusted with chargebacks or deductions (i.e. over/under/damaged)

There are several complications which have been omitted in this diagram and event flow for the sake of simplicity. In particular, omitting intermediary transporting parties eliminates a number of issues.

Transporting parties have various levels of knowledge regarding their shipments. When transporters perform a Shipper Load and Carry (SLAC), they have minimal knowledge about the goods. The sending party seals the trailer and the transporting party is responsible for simply transporting the trailer to the point of destination.

In another scenario known as Live Truck Load (LTL), transporters take inventory as goods are loaded into their trailer. They have intimate knowledge of the goods they are carrying, and they also verify the goods as they are unloaded at the receiving end. This latter scenario has proved inaccurate and inefficient because it requires considerable manual work, and the overseer more often than not fails to take inventory accurately. Omitting transporting parties eliminates the need to take these various scenarios into account.

The SRV process has another system based on sending "Shipment Status" messages between the transporting party and the Receiving party for aligning appointment times. Here is one example of how this system might work. The Shipping Party initially sets an appointment time with the Receiving Party. The transporting party then sends a message indicating when the trailer is ready. When the trailer has been picked up, the transporting party sends another message indicating the fact. It then sends its last message indicating the expected arrival time of the trailer. Eliminating transporting parties also eliminates the extra coordination involved in transporting goods.

These simplifications highlight the ASN and its role in the shipping and receiving process. This event flow will provide a base case reference from which to develop a simple workflow illustrating Savant-to-Savant communication.

4.2 ASN Challenges

The ASN informs the recipient of exactly what to expect. Ideally, this would match the product order placed by the recipient. Rarely though is this ever the case. There are many problems that arise in a real world setting when using ASNs.

Often ASNs contain incorrect or incomplete data because of human error. A system using the Auto-ID Center's IT infrastructure would drastically reduce these problems by eliminating manual data entry. Rather than having a worker visually identify or manually scan each item received, the RFID reader would quickly scan the items and provide a count, reducing check-in time 60 to 90% [17]. If the ASN contained improper packaging information, such as 6 packs of 2 items each versus 4 packs of 3 items each, then the corporation's IT infrastructure could obtain that information from a PML server and automatically reconcile the discrepancy. Current barcodes fail to identify items uniquely, so such automation is difficult. Mass serialization, achieved through the Auto-ID Center's EPC, ONS, and PML systems, makes such functionality possible. In deployments, business logic would be coupled with this functionality to determine if the cost associated with making the change outweighs the cost of feeding the products through the system.

ASNs typically arrive late because large IT networks have human intermediaries and contain numerous checkpoints that can delay ASN transmission. Should the goods simply be shipped next door or down the road, then with the existing infrastructure the likelihood increases of goods arriving before their corresponding ASN. This delay poses a major problem for cross-docked goods.

Cross-docked pallets are prearranged by the shipper and specify the particular store where those goods should arrive. This differs from traditional operation where the distribution center rearranges the goods for its stores. Cross-docked goods are always supposed to be shipped immediately. Goods stored in the distribution center are set aside as reserve inventory. If the cross-docked goods arrive before the associated ASN, then the distribution center cannot verify the goods and forward them to the appropriate store. Instead, the distribution center has to store it in its own reserve inventory, and then await for the arrival of the ASN before moving it forward. Because of the extra manpower associated with moving goods into and out of inventory as well as the delay in shipping, the failure of an ASN to arrive can be quite costly.

In addition to all these problems with ASN's, the goods themselves may be damaged. An Auto-ID monitoring system may also have temperature and pressure sensors that could identify damaged goods with high probability. Such a system would assist assigning blame before the damaged goods traveled further into the supply chain.

4.3 Automated Shipping and Receiving Verification

4.3.1 The Message Format

Traders will need to decide on a single messaging format standard to encapsulate the new EPC information the Auto-ID Center infrastructure requires them to trade. At this time in the U.S., EDI appears to be the most used standard. If companies want to continue using EDI, they will have to introduce EPCs into the standard and have the EDI regulating authority approve it. Alternatively, the industry could also introduce a new messaging standard, more capable and better equipped to handle granular information at the item level. The following analyzes EDI and one emerging standard, Simpl-EB, as an exercise to incorporate EPCs.

4.3.1.1 EDI and the EPC

EDI documents are broken into *transaction sets*. A transaction set is a single EDI unit of transmission and is the equivalent of a message. Transaction sets include one or many data segments framed by a header and trailer. Each data segment consists of multiple data elements, units representing single facts [9]. Focusing on the specific transactions pertinent to shipping and receiving reduces the scope to two transaction sets: the EDI 856 document known as "Ship Notice/Manifest" and the EDI 214 document known as "Transportation Carrier Shipment Status Message." For this business case, these two transactions will need to support only EPC information.

The EDI "Ship Notice/Manifest" document, also known as an "Advanced Shipment Notification," has two data segments, LIN (line item) segment and MAN (marks and numbers) segment, which could support the EPC. LIN describes a specific product and currently has a location where a sender can specify the various associated identifiers. As a representative example, consider the following Target Corporation specific EDI specification:

Data Element Summary			
Ref.	Data	Name	Attributes
<u>Des.</u>	<u>Element</u>		
LIN01	350	Assigned Identification	O AN 1/20
		Alphanumeric characters assigned for differentiation within a transaction set	
LIN02	235	Product/Service ID Qualifier	M ID 2/2
		Code identifying the type/source of the descriptive number used in Product/Service ID (234)	
		CB	Buyer's Catalog Number
			Target Corporation's 9 digit DPCI number
		EN	European Article Number (EAN) (2-5-5-1)
			Not accepted by Target Stores.
		UP	U.P.C. Consumer Package Code (1-5-5-1)

Figure 10: Excerpt from LIN Segment Definition [18]

This specification defines the LIN data element and various ways in which it can be used. Depending on the attribute, specified on the excerpt's far right, the data element can have different meanings. The LIN data element with specified attribute "M" supports three identifiers for distinguishing items: CB, EN, and UP. None of these uniquely identifies items. The EPC could be included here as an additional identifier to augment this transaction and enable unique identification.

The MAN has similar descriptors for containers. Containers are collections of items. Senders often aggregate items and associate the collection with a single identifier to simplify handling.

Data Element Summary			
<u>Ref.</u>	<u>Data</u>	<u>Name</u>	<u>Attributes</u>
<u>Des.</u>	<u>Element</u>		
MAN01	88	Marks and Numbers Qualifier	M ID I/2
		Code specifying the application or source of Marks and Numbers (87)	
		AI	UCC/EAN-128 Application Identifier (AI) and Data
		GM	SSCC-18 and Application Identifier

Figure 11: Excerpt from MAN Segment Definition [18]

As was done with the previous data element, this definition could also be expanded to include EPC information alongside the currently supported identifiers. In this situation, the EPC would identify cases uniquely.

The 214 document, referred to as "Transportation Carrier Shipment Status Message," transports higher level information such as the product order number associated with the goods being transferred. As the OID (Order Identification Detail) data segment example from a Target Corporation specific 214 document indicates below, the PO size is normally 17 characters.

Seg. ID	Example 1: Vendor to DC pickup status message	Seg/ Ele.ID	Target Corporation Notes and Comments
OID	**0639-0699487-0986**CTN*130*L*239	OID	PO number is usually 17 characters.

Figure 12: Excerpt from a Sample 214 Document [19]

Although specific to Target Corporation, this information illustrates common industry practices. If this PO were based on an EPC, then the field holding it would have to be enlarged. The EPC will require a minimum of 23 fields using Hexadecimal representation without dot notation. Should other standard forms be used, the size would increase (refer to Appendix).

During this discussion of augmenting EDI with EPC information, it is important to highlight reservations companies may have for not wanting to do so. Many companies outsource their EDI efforts to third party value added networks (VANs), and sometimes these VANs charge for EDI on a per kilobyte basis. Because the size of these documents,

once augmented with item level EPC information, would grow drastically, this solution might not appeal to some.

Other reasons companies may refrain from modifying the EDI standard include the upgrades necessary to their existing IT infrastructures to support such detailed information. The Auto-ID Center proposition truly enables a massive amount of data to be communicated between companies. This is both the challenge and the virtue of the system. Although EDI can be forced to support EPC information, it may grow unnaturally large and become difficult to manage. While a solution like this may suffice temporarily, it is doubtful that such a solution will suffice for the long-term.

4.3.1.2 *Simpl-eb and the EPC*

Because organizations need to massively remodel IT infrastructures to fully utilize Auto-ID Center information, it may make sense for enterprises to adopt one of the newer XML-based emerging standards for inter-organization communication. This section analyzes Simpl-eb, one promising standard, to determine its ability to handle EPCs.

Simpl-eb's despatch advice document definition, known in XML jargon as a schema, references other schemas. One schema it references is Identification.xsd, a schema that defines the types of identifiers used to identify parties and items.

```
<!-- Party Identification -->
<xsd:complexType name="PartyIdentificationType">
  <xsd:sequence>
    <xsd:choice>
      <xsd:element name="gln"
        type="GlobalLocationNumberType" />
      <xsd:element name="alternatePartyIdentification"
        type="AlternatePartyIdentificationType" />
    </xsd:choice>
    <xsd:element name="additionalPartyIdentification"
      type="AlternatePartyIdentificationType"
      minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
<!-- Item Identification -->
<xsd:complexType name="ItemIdentificationType">
  <xsd:sequence>
    <xsd:choice>
      <xsd:element name="gtin"
        type="GlobalTradeItemNumberType" />
      <xsd:element name="alternateItemIdentification"
        type="AlternateItemIdentificationType" />
    </xsd:choice>
    <xsd:element name="additionalItemIdentification"
      type="AlternateItemIdentificationType"
      minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
```

Figure 13: Excerpt from Simpl-eb Identification.xsd [22]

As illustrated in this excerpt, the Identification.xsd schema identifies items and parties with the GTIN and the GLN, respectively. The Uniform Code Council (UCC) developed these identifiers to support unique identification. There also exists a placeholder for an alternative identifier, but the definition does not define any qualities of the value. As a result, the EPC could be used, but XML editors would not be able to type-check the value to ensure it is properly formatted. Because XML editors support type checking, using the EPC without that ability would increase the likelihood of errors. The only way for XML editors to type check the EPC value would be if the Identification.xsd schema specifically supported EPCs as an additional identifier. The UCC, the authority regulating the Simpl-eb specification, would have to approve the EPC for it to be included in the Identification.xsd schema. Negotiations with the UCC are ongoing on this front.

4.3.2 Communication Channel

The previous discussion on possible communication channels highlights the Auto-ID Center's Savant-to-Savant intercommunication as an alternative to integrating with existing legacy systems. This discussion will focus only on Savant-to-Savant workflows, as the alternative would be implementation specific.

4.3.3 Savant-to-Savant Sample Workflow

This workflow incorporates the Savant architecture into the previous workflow describing the shipping and receiving business process.

- M.1.** Receiving Party:
 - M.1.1.** Savant sends Purchase Order (PO) to Shipping Party
 - M.1.1.1.** PO specifies EPCs of products
 - M.1.1.2.** Includes requested arrival date/time and other Terms of Sale
- M.2.** Shipping Party:
 - M.2.1.** Savant calculates a ship date based on the requested arrival date from Shipping Party
 - M.2.2.** Savant processes order
 - M.2.3.** Picks the goods
 - M.2.3.1.** Scans the EPCs on the physical goods with Savant and Reader
 - M.2.3.2.** Savant confirms scanned EPCs against those specified in the PO
 - M.2.4.** Savant sends an Advanced Shipping Notification (ASN) to the Receiving Party's Savant
 - M.2.4.1.** Includes EPCs and BOL number
 - M.2.5.** Ships the physical goods to the Receiving Party
 - M.2.5.1.** Goods accompanied by a Bill of Lading (BOL)
 - M.2.6.** Savant sends an invoice to the Receiving Party's Savant
 - M.2.6.1.** Includes the EPCs and BOL number
- M.3.** Receiving Party:

- M.3.1. Savant receives the ASN with EPCs and BOL number
- M.3.2. Receives the physical goods with BOL
 - M.3.2.1. Scans the EPCs on the physical goods with Savant and Reader
 - M.3.2.2. Savant reconciles the received goods with the PO, ASN, invoice and BOL data
 - M.3.2.3. Sends a confirmation receipt to the Shipping Party (optional)
- M.3.3. Savant receives the invoice with the carton codes and BOL number
- M.3.4. Savant sends payment to the Shipping Party
 - M.3.4.1. Adjusted with chargebacks or deductions (i.e. over/under/damaged)

Here is the related activity diagram:

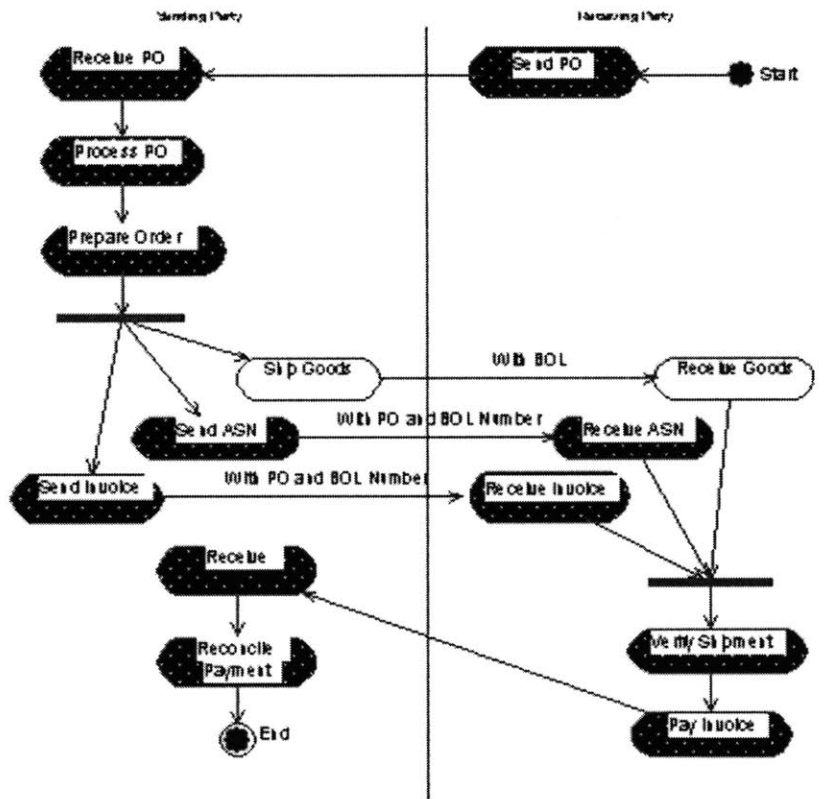


Figure 14: Savant-to-Savant Automated SRV

The filled-in states in this diagram represent Savant automated functions. The Savant automates most of the information flow in the shipping and receiving verification process. For the sake of simplicity, the previous workflow assumes a single trailer represents a single product order (product orders are often spread among multiple trailers). Making this assumption eliminates the need for the Savant to cache partial shipments in case the remaining shipment arrives in the following trailer or over multiple weeks.

4.3.4 Workflow Exceptions

4.3.4.1 Case 1: Receiving Party's Savant Malfunctions

If the Receiving Party's Savant stops functioning, the automated system described above will cease to function. Here is a reworking of the previous workflow to illustrate how such a problem could be handled automatically:

- M.1.** Receiving Party:
 - M.1.1.** Secondary System sends Purchase Order (PO) to Shipping Party
 - M.1.1.1.** PO specifies EPCs of products
 - M.1.1.2.** Includes requested arrival date/time and other Terms of Sale
- M.2.** Shipping Party:
 - M.2.1.** Savant calculates a ship date based on the requested arrival date from Shipping Party
 - M.2.2.** Savant processes order
 - M.2.3.** Picks the goods
 - M.2.3.1.** Scans the EPCs on the physical goods with Savant and Reader
 - M.2.3.2.** Savant confirms scanned EPCs against those specified in the PO
 - M.2.4.** Savant sends an Advanced Shipping Notification (ASN) to the Receiving Party's Savant
 - M.2.4.1.** Includes EPCs and BOL number
 - M.2.5.** Ships the physical goods to the Receiving Party
 - M.2.5.1.** Goods accompanied by a Bill of Lading (BOL)
 - M.2.6.** Savant sends an invoice to the Receiving Party's Savant
 - M.2.6.1.** Includes the EPCs and BOL number
 - M.2.7.** Savant stores ASN, BOL, and Invoice in PML server for later retrieval
- M.3.** Receiving Party:
 - M.3.1.** Receives the physical goods with BOL
 - M.3.1.1.** Scans the EPCs on the physical goods with hand Reader
 - M.3.1.2.** Secondary system retrieves PO, ASN, invoice and BOL data based on PML associated with received EPCs
 - M.3.1.3.** Secondary systems reconciles the received goods with the PO, ASN, invoice and BOL data
 - M.3.1.4.** Secondary systems send a confirmation receipt to the Shipping Party (optional)
 - M.3.2.** Secondary systems send payment to the Shipping Party
 - M.3.2.1.** Adjusted with chargebacks or deductions (i.e. over/under/damaged)

Receiving Party's Savant
fails to receive ASN. →

Receiving Party's Savant
fails to receive invoice. →

In this scenario, the Receiving Party will need secondary systems to process data. Secondary systems might be redundant Savants that the system uses for "fail-over" or load-sharing. They may also be other electronic systems or human support. For

example, the Receiving Party could verify their goods visually or use hand readers. These hand readers could retrieve the proper electronic documents by referencing them from the Shipping Party's system using the EPC of the received good as a key. The system still provides value over the current barcode system, although it would require considerably more human involvement.

This example illustrates the need for both messaging and referencing. When a message fails to arrive for automated SRV, the system may default to referencing based on the EPC of a received good. Here is a clear example of how messaging is the preferred communication channel, but referencing still provides a valuable backup system. The Receiving Party would use the PML system to obtain the Purchase Order number and other important information associated with the good.

This scenario also highlights the need for secondary systems to support existing communication channels. The Receiving Party's original PO was originally sent by the Savant. To minimize disruption in the Shipping Party's system, the Receiving Party must use the existing communication channel. Alternatively, both parties could establish a backup communication channel. Sending secondary system composed documents electronically via email or fax are examples of possible alternative communication channels.

4.3.4.2 Case 2: Shipping Party's Savant Malfunctions

In this scenario, the Shipping Party's Savant goes down. Consider the following reworking of the original Savant-to-Savant SRV workflow.

- M.1.** Receiving Party:
 - M.1.1.** Savant sends Purchase Order (PO) to Shipping Party
 - M.1.1.1.** PO specifies EPCs of products
 - M.1.1.2.** Includes requested arrival date/time and other Terms of Sale
 - M.2.** Shipping Party:
 - M.2.1.** Secondary system calculates a ship date based on the requested arrival date from Shipping Party
 - M.2.2.** Secondary system processes order
 - M.2.3.** Picks the goods
 - M.2.3.1.** Scans the EPCs on the physical goods with manual Reader
 - M.2.3.2.** Secondary systems confirm scanned EPCs against those specified in the PO
 - M.2.4.** Secondary systems send an Advanced Shipping Notification (ASN) to the Receiving Party's Savant or coordinates with Receiving Party to use alternate system
 - M.2.4.1.** Includes EPCs and BOL number
 - M.2.5.** Ships the physical goods to the Receiving Party
 - M.2.5.1.** Goods accompanied by a Bill of Lading (BOL)
- Secondary system replaces Savant's role. →

- M.2.6.** Secondary systems send an invoice to the Receiving Party's Savant or coordinates with Receiving Party to use alternate system
 - M.2.6.1.** Includes the EPCs and BOL number
- M.3.** Receiving Party:
 - M.3.1.** Savant or secondary system receives the ASN with EPCs and BOL number
 - M.3.2.** Receives the physical goods with BOL
 - M.3.2.1.** Scans the EPCs on the physical goods with Savant and Reader
 - M.3.2.2.** Savant reconciles the received goods with the PO, ASN, invoice and BOL data
 - M.3.2.3.** Sends a confirmation receipt to the Shipping Party (optional)
 - M.3.3.** Savant or secondary system receives the invoice with the carton codes and BOL number
 - M.3.4.** Savant or secondary system sends payment to the Shipping Party
 - M.3.4.1.** Adjusted with chargebacks or deductions (i.e. over/under/damaged)

The automated functions of the Shipping Party include constructing and handling the PO, ASN, and invoice. A secondary system would have to do handle these if their Savant malfunctioned. It is important that there exist some indicator to alert the secondary system the Savant is down. If the secondary system is human support, then without a system failure indicator, it might take a complaint phone call from the receiving end to inform the Shipping Party its Savant is down. By that time though, considerable disruption to the system will already have been done.

When secondary systems are composing the various documents, they should employ a "best-effort" policy to provide as much information in these important transaction documents as the primary system. Typically, the Shipping Party's Savant would communicate the documents. Once composed by secondary systems, the documents could be fed into the existing communication channel to keep other automated systems further down the supply chain running. Alternatively, there could exist a pre-established secondary communication channel to send messages.

Although the systems in Case 1 were able to continue functioning in a mostly automated manner, Case 2 systems experienced significant disruption. Such system failure would disturb the supply chain and have costly repercussions. It is important that these systems and their secondary, backup systems be designed with these consequences in mind.

4.3.4.3 Case 3: Shipping Party Does Not Support Auto-ID Center Technology

In the final case, the Shipping Party does not support any Auto-ID Center technology. In this event, the same system processes in place today will exist. The Receiving Party will have to manually tag the goods and enter that information into its system if it wants to make use of its tracking system internally and automate shipping further down the supply chain.

4.4 Implementation

I implemented a system that allows two ASN servers written with the Java 1.3.x SDK to communicate with each other over a Java Messaging Service. The two ASN servers could easily be connected to Savants to illustrate Savant-to-Savant communication. Currently, keystrokes mimic actions representing product shipment and arrival. If the ASN applications were connected to Savants, then items could actually be tagged and a desktop demo with physical items could be demonstrated.

The two servers communicate via a Java Messaging Service. The JMS provider I chose was OpenJMS, an open source distribution of the Sun JMS specification. It included a number of important features, including the publish/subscribe topics that would allow multiple users to connect to the same queue. Messages were secured using public-private keys. These keys were generated by the Java Cryptography Extension (JCE), an optional package I downloaded and installed. After generating my own public-private key pair, I distributed the public key manually, and then signed all my messages with the Shipping Party's private key and the Receiving Party's public key. Signing the message in this manner achieved authentication, confidentiality, and integrity, the main three components of secure communication discussed earlier.

CHAPTER 5: The Aggregation Database - an IT Infrastructure Implication

Shippers and receivers often aggregate items and cases on a palette and refer to the aggregation as a single entity. This basic process has implications throughout the Auto-ID infrastructure. As the goods are tracked, the tracking system must create links between the individual items and the aggregation, most likely referenced in the system by an EPC. Data structures that capture physical realities such as breaking down aggregations will also need to be developed. Considering the vast number of EPCs flowing through a retailer's system, concern was indicated regarding the system requirements necessary for supporting such a system. Although relational databases could be used, they are considerably more heavyweight than the solution need be. The following is a discussion of the aggregation database, a physical database optimized specifically for handling EPCs and aggregation information.

5.1 Design Considerations

Database architects must take into consideration physical constraints when trying to develop a useable database. Primary memory consists of silicon memory chips. Secondary storage utilizes magnetic disks, a significantly cheaper alternative to silicon. The price paid for the low cost of magnetic storage techniques is the relatively long time it takes to access data. Often it takes more time to access a page of information and read it from a disk than it takes for the computer to examine all the information read. Because databases require tremendous memory storage capacity, database design caters to magnetic disks. As a result, database design takes into consideration two main factors: the number of disk accesses and computing time [11].

The number of disk accesses depends on hardware size. If a system has extraordinarily large primary memory, then it would be possible to load the entire database into main memory. Realistically though, primary memory is smaller than secondary memory by at least an order of magnitude. Primary memory normally limits the table size in a database. Smaller primary memory therefore implies an increase in the levels of hierarchy.

Computation time depends on the algorithm chosen and the data set. For a given data set, a particular algorithm may be faster. The most common physical database designs are based on either hash tables, B-trees, or some hybrid.

Algorithm	Search Time (avg.)	Search Time (worst-case)	Ideal Data-set	Extra Notes
Hash tables	$O(1)$ w/reasonable assumptions	$\theta(n)$	few keys relative to size of key universe	Hashes must have collision reconciliation
B-trees	$O(\log h)$ $h = \log_{\text{branching}} n$	$O(\log h)$ $h = \log_{\text{branching}} n$	many keys relative to size	

			of key universe	
--	--	--	-----------------	--

5.2 DB Design

One design for the aggregation database would encode EPCs into a tree comprising hash tables. This tree would exist for the sole purpose of finding an EPC. Alongside each EPC in the tree would sit satellite information that would encode the aggregation tree. The satellite information would include the children list (zero or many) and the immediate ancestor. To optimize the DB for performing ancestor lookups, the satellite information might also include a preprocessed list of the *entire* ancestor history, but for simplicity this fact will be ignored in the upcoming calculations. With this setup, the general algorithm for a query requesting the *immediate* children or parent would be:

1. Find EPC, report immediate parent or children information

For finding ancestor and descendant *lists*:

1. Find EPC, then perform ancestor search on parent (report ancestor list if already preprocessed) or perform descendent search on children
2. Repeat until search complete

There are many optimizations that could be done on this algorithm. However, as the following calculations reveal, the aggregation database is not actually a problem that requires significant attention at this time.

5.3 DB Analysis

Overview

The size of the aggregation database composed of hash tables can be computed using the following formulas:

$$\begin{aligned} \text{Total size of DB} &= (\text{size of inner tables}) + (\text{size of leaf tables}) \\ \text{Size of inner tables} &= (\text{memory pointer} + \text{key}) * (\# \text{ of entries}) * (\# \text{ of tables}) \\ \text{Size of leaf tables} &= (\text{size of single leaf table}) * (\# \text{ of tables}) \\ \text{Size of single leaf table} &= (\text{memory pointer} + \text{key} + \text{avg. satellite info size}) * (\# \text{ of entries}) \end{aligned}$$

Please note that these are rough calculations highlighting the main memory factors and ignoring various details, such as the memory needed to encode the hash tables themselves.

A relatively low-end storage solution might require 100 GB of secondary memory and 1 GB of primary memory. With this type of storage solution, the largest single table an aggregation database would support would be approximately 1 GB. As large a table size as possible should be used to reduce the number of levels in the tree hierarchy. The more tables that need to be loaded into primary memory, the greater the number of disk accesses. The goal is to minimize this number so that EPC aggregation information can be retrieved quickly.

The EPC supports the following quantities for each of its segments:

96 Bit Type I EPC

	Version	Domain Manager	Object Class	Serial Number
Bits	8	28	24 bits	36 bits
Unique #s	256 (~3 hundred)	268,435,456 (~300 million)	16,777,216 (~20 million)	68,719,476,736 (~70 billion)
Logical Example	Version 1, 96-bit	Gillette	Mach-3	Unique Item #

Assumptions:

A large retailer might necessitate storage capacity for the following figures (rough estimates):

- 2 EPC version numbers
- 1,000 domain managers
- 350,000 different object classes
- 10,000,000 individual items

Memory pointers are generally 32 bits.

Average satellite info (assume 1 parent, 20 children) = 21 * 96 bits = 2016 bits

Calculations:

Given the previous assumptions, we can fill in the calculations and determine the number of entries the proposed hardware solution would support. There will be a single hash table assigned to each piece of the EPC segment, where the key is actually the EPC segment itself.

General equation:

$$\text{Size of inner table} = (\text{memory pointer} + \text{key}) * (\# \text{ of entries})$$

For EPC version number hash table:

$$\text{Size of inner table} = (32 \text{ bits} + 8 \text{ bits}) * (2) = 10 \text{ Bytes}$$

For EPC domain managers hash table:

$$\text{Size of inner table} = (32 \text{ bits} + 28 \text{ bits}) * (1000) = 7,500 \text{ Bytes}$$

For EPC object class hash table:

$$\text{Size of inner table} = (32 \text{ bits} + 24 \text{ bits}) * (350,000) = 2.45 \text{ MB}$$

The total size for all these tables is approximately 2.5 MB. All these tables can be loaded into the primary memory, delivering almost instantaneous search capability. The remaining primary memory, approximately 1 GB, can be used to load hash tables from disk that represent the serialized information.

Given there exists 100 GB of secondary memory, this storage solution can support approximately 100 1 GB size leaf tables. The key in this instance is the remaining serialization EPC fragment of 36 bits. Each leaf table yields the following number of entries:

$$\text{Size of single leaf table} = (\text{memory pointer} + \text{key} + \text{avg. satellite info size}) * (\# \text{ of entries})$$

$$1 \text{ GB} = (32 \text{ bits} + 36 \text{ bits} + 2016 \text{ bits}) * (\# \text{ of entries})$$

$$\# \text{ of entries} \approx 3.8 \text{ million}$$

This solution implies that the total number of leaf tables would support:

$$(100 \text{ leaf tables}) * (3.8 \text{ million entries per tables}) \approx 380 \text{ million entries}$$

5.4 Aggregation DB Conclusion

The storage solution initially suggested for a single retail outlet was a relatively low end solution, costing in today's marketplace around \$2,500 [20]. Because a typical retail outlet typically only handles 10 million individual items per year, this solution goes well beyond satisfying the basic requirements. Supporting 380 million entries indicates this solution would sustain a large retail store aggregation information for 38 yrs. In addition, the solution was able to find a single EPC within 1-2 disk accesses. These figures reveal that the aggregation database is a component of a corporation's technology infrastructure that does not require significant optimization.

It is important to note that satellite information dominated the calculations. That information alone accounted for 96% of the overall storage solution:

$$(380 \text{ million entries}) * (2016 \text{ bits avg. satellite info per entry}) \approx 960 \text{ MB}$$

This observation suggests all other factors (memory pointers, EPC size, etc.) can be ignored when they are relatively small *to estimate* storage requirements.

CHAPTER 6: Conclusion

Automating business processes, like shipping and receiving, require a number of components to function together. Companies must choose a communication standard, a communication platform, and develop the IT infrastructure to support the information transfer. In addition to system design, companies will have to modify their day-to-day processes to gain proposed benefits.

It is likely that companies will go through a long transitional period given the extent of changes implicated in adopting the Auto-ID Center's IT infrastructure. This thesis has analyzed EDI, one existing communication standard, to determine how it would support Auto-ID Center technology. It also analyzed another emerging communication standard for the same purpose. This thesis recommends a phased adoption to facilitate transition. In analyzing these various standards, this thesis also suggests how such adoption might be accomplished.

Even with an automated system in place, to be effective the system must have a significant level of tolerance to withstand core component failure. As was illustrated in two workflow complications, secondary systems were necessary. If these secondary systems require human support, then they may prove quite costly and considerably decrease efficiency. In the scenario where the Receiving Party's Savant fails, this thesis illustrates how an automated system can still function despite some of its vital components malfunctioning. An attempt must be made to design systems resiliently so that in the event parts of a system fail to operate, the system can still function by defaulting to alternative processes without human intervention. The workflow complication including the Shipping Party's Savant malfunctioning illustrates disproportionate system disruption. The impact is significantly greater in this scenario than when the Receiving Party's Savant goes down because there does not exist a clear alternative default process.

To support the added information necessary to automate business processes, this thesis discussed the aggregation database. Although it was discovered that the aggregation database would support hierarchy information comfortably with today's hardware technology, other information may overload systems. This thesis acknowledges the need to analyze business processes in their entirety and understand their infrastructure requirements. Innovation in the aggregation database was unnecessary, but other components may require more reworking. System designers will need to choose selectively only the most vital information to keep systems manageable. With the benefit of more information access comes the responsibility of managing it.

6.1 Future Work

A more thorough application integrated with all of the Auto-ID Center's infrastructure components could be developed. In the end though, these are merely proof-of-concepts, and the true test of this system will be achieved when actual physical deployments are put in place alongside the information systems.

To simulate this research in a real-world environment, an ongoing Auto-ID Business Use-Case (A-biz) pilot inside in the Auto-ID Center is working with Target Corporation, Proctor & Gamble, and Sun Microsystems to automate the exchange of goods from P&G to Target. This pilot will need to address many of the business decisions alluded to in this thesis to achieve completion.

As a case study, this thesis focuses on the SRV process. There are many other business processes that could also be analyzed for additional learning. These processes include but are not limited to: continuous replenishment, theft detection, and inventory tracking.

This thesis analyzed two communication standards to understand how they would support AIDC technology. There are many other standards available, some possibly better suited than others for supporting this technology. An analysis could be done to determine how well these other standards would support AIDC technology.

REFERENCES

- [1] Sarma, S. *Towards the 5 cent tag*. Technical Report MIT-AUTOID-WH-006. The AutoID Center, MIT. Cambridge, Massachusetts. Published Nov 1, 2001.
<http://www.autoidcenter.org/research.asp>
- [2] Overby C.S., Charron C., and Chaskey K. *RFID: The Smart Product [R]evolution*. Aug 2002. Forrester Research, Inc. Cambridge, Massachusetts.
- [3] *Creating the Business Case for Global Data Synchronisation in Your Company*. Oct 2002. Cap Gemini Ernst & Young/Global Commerce Initiative.
- [4] Joshi, Y. *Information Visibility and its Effect On Supply Chain Dynamics*. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published June 1, 2000.
<http://www.autoidcenter.org/research.asp>
- [5] Brock, D.L. *The Electronic Product Code (EPC) – A Naming Scheme For Physical Objects*. Technical Report MIT-AUTOID-WH-002. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Jan 1, 2001.
<http://www.autoidcenter.org/research.asp>
- [6] Oat Systems and MIT Auto-ID Center. *The Object Name Service - Version 0.5 (Beta)*. Technical Report MIT-AUTOID-TM-004. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Feb 1, 2002.
<http://www.autoidcenter.org/research.asp>
- [7] Brock D.L., T. P. Milne, Y.Y. Kang and B. Lewis. *The Physical Markup Language*. Technical Report MIT-AUTOID-WH-005. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Jun 1, 2001.
<http://www.autoidcenter.org/research.asp>
- [8] Floerkemeier, Christian and Robin Koh. *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>
- [9] <http://whatis.com>. July 31, 2001.
- [10] Alateras, Jim. *OpenJMS White Paper*. Feb 15, 2001.
<http://openjms.sourceforge.net>
- [11] Cormen, T.H., Leiserson, C.E., And Rivest, R.L. (1990). *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts.
- [12] Diffie, W. and Hellman, M.E. *New Directions in Cryptography*. IEEE Trans. Inform. Theory IT.22 (Nov. 1976), 644-654.

- [13] Bloch, Josh and John Papageorge. *How Do Digital Signatures Work?* July 1997.
<http://developer.java.sun.com/developer/technicalArticles/Interviews/DigitalSigs/>
- [14] J. G. Steiner, B. Clifford Neuman, and J.I. Schiller. *Kerberos: An Authentication Service for Open Network Systems*. In *Proceedings of the Winter 1988 Usenix Conference*. Feb 1988.
<http://web.mit.edu/kerberos/www/>
- [15] <http://Openflow.it>, OpenFlow Introduction. July 24, 2002.
- [16] 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 Feb 3, 2003.
<http://www.autoidcenter.org/research.asp>
- [17] Alexander K., T. Gilliam, K. Gramling, M. Kindy, D. Moogimane, M. Schultz, and M. Woods. *Focus on the Supply Chain: Applying Auto-ID within the Distribution Center*. IBM-AUTOID-BC-002. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Sept 1, 2002. <http://www.autoidcenter.org/research.asp>
- [18] Compliments of Target Corporation
- [19] <http://www.partnersonline.com> , May, 2003.
- [20] <http://www.dell.com> , May, 2003.
- [21] Goyal, A. *Savant Guide*. MIT-AUTOID-TR015. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published April, 2003.
- [22] Compliments of Uniform Code Council
[http://www.uc-council.org/smp/schemas/eanucc Identification.xsd](http://www.uc-council.org/smp/schemas/eanucc%20Identification.xsd)
- [23] Oat Systems, The MIT Auto-ID Center. *The Savant - Version 0.1 (Alpha) Oat Systems & MIT Auto-ID Center*. The Auto-ID Center, MIT. Cambridge, Massachusetts. Published Feb 1, 2002.
- [24] Scharfield, T. *Abstract 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.
- [25] Sarma S., Weis S.A., Engels D.W. *RFID Systems, Security, & Privacy Implications*. The Auto-ID Center. Cambridge, Massachusetts. Published Nov 1, 2002.
- [26] <http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html>
Rivest's MD-5 algorithm MD5 Homepage (unofficial)

[27] <http://www.sec.konicabt.com/flowshots.html>
Konica workflow screenshot.

APPENDIX

EPC Dot Notation

by Tim Milne

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. An example is the nesting of an EPC number in an Advanced Shipping Notification for transmission between companies.

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

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

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

Maximum Domain Manager Number		
Binary	Hexadecimal	Decimal
11111111111111111111111111111111	FFFFFFFF	268435455

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

Maximum Object Class Number		
Binary	Hexadecimal	Decimal
11111111111111111111111111111111	FFFFFF	16777215

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

