Implementing Paperless Automation
Of Accounts Payable Invoices
For Small Business Accounting Systems

Athabasca University MBA
Applied Project
By Stephen Heath
Abstract

Technology becomes more powerful at exponential rates, but it often seems we are unable to harness the full benefits, and one of the primary areas where technological capabilities and dreams of what we would do with them are out of sync is the vast amounts of paper that we use in what was supposed to be the era of the paperless office. Reducing paper usage, when done effectively, has significant financial and environmental benefits, and yet business paper continues to be a growth industry.

In “The Myth of the Paperless Office”, researchers Abigail Sellen and Richard Harper (2002) have determined that the most successful paper reduction projects are those that have improving business processes as a goal rather than simply eliminating paper. The author of this project has decades of experience in information technology, and has seen numerous multi-company information sharing projects such as EDI come into existence at great expense, be adopted by large companies and their suppliers, and then fail to gain any additional traction. These two observations led to this project, which focuses on providing the essence of a technology which can improve a universal, multi-company business process, using techniques that allow scalability for any size business. The invoicing/accounts payable process was chosen for it’s applicability to almost every business and the large amount of paper printed and stored worldwide as a result of it.

The first step was to analyze the invoicing/payables process as it is currently handled and to examine ways it could be made more efficient with cost effective available technology. Next, existing technology that exists for a similar purpose was examined to determine what roadblocks were preventing them from increased adoption. The third step in the research was to examine unrelated technologies that had improved efficiencies for users and to determine what factors were involved in making them successful. With the results of these three steps, a software protocol was developed that could allow companies to improve, and eventually automate, the invoicing/payables process.

Analyzing existing processes showed that electronic document adoption had stalled for two opposing reasons. In some cases, the formats used were too limiting and unable to transfer all necessary data, yet in others it was the large size of the formats which caused the costs for implementation and custom programming to be prohibitively expensive. These difficulties were addressed by adding rudimentary artificial intelligence techniques to the new process; only a small core of mandatory data would be defined, and any other data could also be included using any definition desired. The client side implementations would include rudimentary artificial intelligence techniques so that the accounts payable clerks could “teach” the system “rules” about which information was important and which wasn’t, as well as what to do with it. This would also allow custom functionality for any group of users who desired to implement it without breaking the format for others, and all that would be required would be common consensus amongst those who wished to use the new features.

Moving to a truly paperless business world will not be a simple process, but by designing scalable protocols to allow both power and flexibility, business processes can be converted one step at a time, and many small steps can combine to make a great distance.
Table of Contents

1.0 Introduction .........................................................................................................4
2.0 The Paperless Office ............................................................................................5
  2.1 Benefits of Going Paperless .............................................................................5
  2.2 History of the Paperless Office .........................................................................6
  2.3 Roadblocks to Going Paperless .......................................................................7
  2.4 Hardware Technology ....................................................................................8
3.0 Current Processes in Accounts Payable ...............................................................9
4.0 Current Paperless Technologies .........................................................................10
  4.1 EDI ..............................................................................................................10
    4.1.1 History ...................................................................................................10
    4.1.2 Technical Information ............................................................................12
    4.1.3 Current Status and Limitations ...............................................................13
  4.2 XML .............................................................................................................14
    4.2.1 History ...................................................................................................15
    4.2.2 Industry Specific XML Languages .........................................................18
    4.2.3 Current Status and Limitations ...............................................................18
5.0 Adoption of Successful Software Protocols ........................................................19
  5.1 HTML ...........................................................................................................19
  5.2 PDF .............................................................................................................20
  5.3 QIF/OFX/QFX .............................................................................................23
  5.4 Open Source Software ...................................................................................26
  5.5 Success Factors to Implement ........................................................................26
6.0 Development of Invoice Protocol .......................................................................28
  6.1 Key Information Required .............................................................................28
  6.2 Protocol Delivery .........................................................................................29
  6.3 Protocol Branding .........................................................................................31
  6.4 Protocol Schematics ......................................................................................33
7.0 Examples of Protocol Usage ..............................................................................34
  7.1 Sample Invoice Files ....................................................................................34
    7.1.1 Simple Service Invoice ...........................................................................34
    7.1.2 Complex Service Invoice ........................................................................34
    7.1.3 Combination Goods and Services Invoice .............................................35
  7.2 Possible User-side Implementations ...............................................................36
8.0 Conclusions .......................................................................................................39
9.0 References .........................................................................................................41
10.0 Appendices .......................................................................................................44
  Appendix A: Samples Of Industry Specific XML Variant Invoice Files .................44
1.0 Introduction

Improving productivity is a goal for every well run organization, enabling them to do more with less. For an individual firm, this may result in a competitive advantage, enabling the company to realize increased profits, or to charge lower prices and increase market share. For improvements shared by multiple companies, an industry, or even the entire marketplace, the result may be either higher profit for all or lower prices for customers which in turn increases the overall demand. There is a natural barrier to these productivity improvements, and that is the cost to implement the systems or equipment that facilitates the change.

In the case of physical equipment, it is sometimes possible to increase productivity in stages as finances allow, but when implementing software technology, it is often essential to incur the maximum costs to provide complete support for the entire specification as a partial implementation might mean regular occurrences of the technology failing to work properly. These costs can grow exponentially due to the fast pace of technological change... if an alternate specification grows in popularity, another full implementation needs to be developed and integrated.

The usefulness of software technology that allows inter-company functionality is directly proportional to the number of companies using the technology. This is essentially an interpretation of Metcalfe’s Law, which states that the value of a network is proportional to the square of the number of the users of the system \((n^2)\). While there has been some debate about the accuracy of the increase in value, it is commonly accepted that the more users there are, the more value or potential value is provided to the users. This leads to two conclusions:

1) Any software technology developed should be flexible enough to provide the broadest range of usefulness to allow for, and attract, the highest possible number of users, which results in the greatest value of the technology.

2) A critical mass exists where the value provided to the users in terms of increased productivity will be equal to or greater than the costs involved in setting up the software. Moving from zero users to this critical mass will be difficult due to the negative actual value of the software at that point.

This paper will show that these are the key reasons why many companies have failed to embrace existing technologies that allow automated electronic transactions, and provide an alternative method to full implementation of a protocol that would enable extremely cost effective implementation. Due to the large scope of business operations, this paper will focus on one specific business process, the movement of an invoice from a company’s billing department to their customer’s accounts payable department, and demonstrate how a protocol could be implemented that could improve the efficiency of this process. The principles used in this process to bridge the gap between minimal and full scale implementation should be equally transferable to other business processes.
2.0 The Paperless Office

2.1 Benefits of Going Paperless

Increased productivity is not the only reason to improve inter-business automation. According to John Richard of the Resource Conservation Alliance, as of 2002 on average each office employee in the United States generated approximately 10,000 sheets of paper, or 27 pounds per person. While accounts payable invoices are not specifically broken out, even if they were only a small fraction of that total, the elimination of paper invoicing worldwide could lead to a reduction in paper usage of billions of pounds per year. According to Sam Martin (2004)

“The paper making process is not a clean one. According to the U.S. Toxic Release Inventory report published by the U.S. Environmental Protection Agency (EPA), pulp and paper mills are among the worst polluters to air, water and land of any industry in the country. The Worldwatch Institute offers similar statistics for the rest of the world. Each year millions of pounds of highly toxic chemicals such as toluene, methanol, chlorine dioxide, hydrochloric acid, and formaldehyde are released into the air and water from paper making plants around the world.

Paper making also uses up vast quantities of trees [...] much of the wood used by paper companies in the U.S. comes from privately owned tree farms [...] Only 9% of the wood used to make paper is harvested from old growth forests, which are impossible to replace because of their maturity.

Yet, while tree farms or plantations help feed the demand for wood, they can't provide the plant and animal diversity found in natural forests. Plus, according to a 1996 report from the U.S. Forest Service, the rate of harvest for softwood trees in the southern United States outpaced growth for the first time since 1953.”

These additional benefits to the reduction of paper in the workplace help explain the allure of the concept of the paperless office, as there is both an economic and environmental benefit, and when two paperless offices combine the paperless benefits are magnified through the network effect. In addition, the environmental benefits also magnify as paper reduction takes place since forested ecosystems would be free to flourish.
2.2 History of the Paperless Office

The concept of the paperless office was first articulated in a 1975 article in Business Week magazine discussing the future possibilities of moving automation from the factory and data processing floors into the office. As one contributor, George E. Pake, the head of PARC, Xerox's Palo Alto Research Center, predicted (1975) “I’ll be able to call up documents from my files on the screen, or by pressing a button. I can get my mail or any messages. I don’t know how much hard copy I’ll want in this world.” It was not until the 1980’s when personal computers became commonplace and word processing software was a standard in offices that it appeared that the time of the paperless office was imminent. Unfortunately, most computers at that time were standalone computers. As Abigail Sellen (2002) wrote in The Myth of the Paperless Office, “This had an important consequence: paper became in effect the connection between the users of the freestanding machines.” (p.6)

Not only did paper usage increase through its use as a virtual network, but many other aspects of technology at the time conspired to increase the importance of paper. Expensive storage with low capacities meant software routinely required all of the detailed data that had been collected for a period of time, such as a month, to be printed out before the details were deleted to make room for the next month’s transactions, and for many companies, the standard procedure was to print every possible report, whether they ever actually used them or not. This was considered critical by companies due to the poor nature of backup systems. Tape backup was expensive and floppy disk backup was not only unreliable as the disks would degrade and develop erroneous sectors, but it would require many floppy disks to store the company’s data.

Solutions for these limitations have developed over time. The development and popularization of internal networks, such as Ethernet and Token Ring enabled some reduction in paper usage by allowing internal email and server systems, and the subsequent development of the internet allowed this level of connectivity to be extended between companies at minimal cost. The vast increases in storage capacity along with continuing price reductions has made it feasible to maintain all of a company’s records in a live state, and backups can now be made to many reliable technologies, such as high capacity optical drives like DVD or the relatively new Blu-Ray, flash drives such as USB “thumb” drives, backups over the internet to server farms, or even backups of entire hard drives, which is feasible due to the low cost and high reliability of hard drive technology.

The only significant remaining technical hurdle in moving towards the paperless office is the cost of implementation. Unfortunately, this cost remains a significant burden. Any protocol that is robust enough to provide all the needed functionality for a large enterprise size firm will likely include vast amounts of functionality that is not required for a small company, making a “one-size-fits-all” protocol almost impossible.
2.3 Roadblocks to Going Paperless

While the technological impediments have in essence been solved, there remain a few roadblocks to the business world going paperless. The first is the development and adoption of common data formats. As mentioned previously, data formats are subject to the network effect, where they grow in value the more users there are. While there are existing formats such as EDI, and developing formats such as industry specific XML variants, growth in the former has come to a standstill and growth in the latter does not help a company go fully paperless as only other companies in the same industry would use the same variant.

Another impediment remains the willingness to change. This is rarely a difficulty at the company level, as the financial savings are usually quite significant; rather it is employees who have used paper for many years that are resistant to change. There are many reasons for this resistance; concern of technology failure causing loss of information, resistance to learning new methods and procedures, improper or inadequate training on new procedures, or simply a habitual inertia. The largest cause for resistance, however, according to Abigail Sellen, appears to be procedures that are developed with paperless operations as a goal, rather than as a byproduct for improved procedures.

Sellen and Harper (2002), for their book “The Myth of the Paperless Office” investigated the continued growth in the use of paper by analyzing how people used paper in their daily activities; the processes and procedures that surround paper and why those processes exist. During their investigations they made a number of important discoveries, the most relevant of which for this paper is that business succeeds best in going paperless when their goal is to use the best, most efficient processes possible as opposed to simply attempting to eliminate paper. In one case a company made changes to its budget and planning process, replacing the paper based process with electronic based forms. The problem was that under the paper system, the managers could shift funds from one code to another to deal with unpredicted costs, which they could not do in the new electronic system. Since managers who brought a project in under budget were seen in a better light than those who overspent irrespective of the actual profit made on the project, it had created a culture where managers tried to pad the expenses and come in under budget. Therefore, since this would be more difficult, if not impossible, to do under the new system, there was extremely high resistance, and it was not until the way managers were assessed and held accountable for their work was changed that the new paperless system was embraced.

While many businesses are different and have different processes, there are some that are in essence common to all, and this paper will choose one of those as a basis for developing a protocol that will address the process side of going paperless – the invoicing / accounts payable process.
2.4 Hardware Technology

Although the key barrier remains software technology and the size of it’s user base, another important barrier is the willingness to adapt to new technology by employees and management. If the resistance to change is from a desire to maintain the status quo, no matter what the status quo is or how effective it may or may not be, it is a human resources issue that must be dealt with by management, and is beyond the scope of this paper. If, however, the resistance to change is due to the additional costs incurred to adopt the new technology not providing enough cost savings to justify the expense, there are two solutions for this problem; lowering the expenses required to change, or providing additional functionality which can provide additional cost savings or efficiencies.

When it comes to hardware technology, both solutions have been consistently occurring, to the point that it is often cheaper and more beneficial to buy new equipment or components than to repair old ones, and the cheapest hardware has incredible capabilities. For example, the smallest hard drive available from a nearby computer store holds 40 gigabytes, which is enough to store 19.2 million pages of text, for $59. For $155, you can buy one that holds 500 gigabytes, or 240 million pages of text. Likewise, functionality continues to increase, for example the purchase of a $10 modem and a $20 piece of software gives your computer the ability to send and receive faxes and use optical character recognition to turn the image into an editable document, a $20/month internet connection allows direct data sharing via email attachments or virtual private networks, and the minimum resolution on modern monitors in the $150 range present text crisply and cleanly. When a company decides to add paperless functionality, hardware costs are generally no obstacle.

Unfortunately, a large number of purchases, because they are such low cost, also are not afforded the time and planning that would occur with higher value capital expenditures. If a laser printer ceases operation, a company is much more likely to simply run to the nearest office supply store and pick up a replacement for $99 rather than evaluate the benefits of replacing individual desktop printers with a single networked printer, and since office technology has generally evolved with a paper base, it is likely that the quick replacements will also be paper based.

For example, the fax machine grew ubiquitous in the 1980’s, but the process behind it was often a user typing a letter in a word processor, printing the result on paper, then using the fax to send a copy to another company. In the early days of the fax machine, it printed on thermal paper which faded quickly, so often the next step was to make a photocopy of the fax. Even when plain paper faxes became commonplace, the result was a document that was not necessarily usable. Information might need to be typed in to the recipient’s computer for processing. If, at the time that fax was becoming a standard, the internet and email were as widespread as they are today, it is likely that the fax machine as we know it would not exist and scanners and email would be the method of transmission generally used instead.
3.0 Current Processes in Accounts Payable

Technology has allowed a large degree of improvement in the way accounts payable has traditionally been processed, but the basic steps have not changed too dramatically.

1. The billing department of the vendor receives the information on the goods or services that have been, or will be, provided to the customer and creates an invoice. Depending on the size of the company, this may be as elaborate as collating and cross checking paperwork received from sales, purchasing, and shipping departments, or may be as simple as a one man operation writing or typing an invoice.

2. The invoice may be printed, with multiple copies being sent to different locations. A. Nash Travel Inc. began operations by printing four copies of every invoice. One copy was stored in numerical order for tax and audit purposes and another was given to the booking agent, who filed it by customer and departure date along with their handwritten notes and accumulated printouts relating to that particular trip. The last two copies were mailed to the customer, theoretically so that the traveler could have a copy for themselves and have a copy to provide for expense reimbursement. It was found that virtually none of the customers were using the second copy for its purpose, and were instead mailing it back as a remittance copy, which allowed the company to drop it down to three copies.

3. The invoice is sent to the customer. If it is a printed invoice, that generally involves either putting it into an envelope and mailing it, or perhaps faxing it to the customer to expedite payment. If the invoice is instead offered in electronic format, there are two possibilities. Either the customer is notified by some method, such as email, that a new invoice exists and the customer accesses the vendor’s systems to retrieve it, or the invoice itself is pushed to the customer, either as an email attachment or through an existing channel such as EDI.

4. The customer receives the invoice and routes it to the accounts payable department, along with any documentation regarding the receipt of the goods or completion of the services. The accounts payable clerk compares the documentation, perhaps also verifying the order by comparing it to a purchase order created previously. If the order has been sent electronically, it is likely that a copy is printed for the clerk to work with. This copy will likely be filed when the invoice has been processed.

5. The clerk either keys in the data directly into the customer’s accounting system, or examines and approves any automated entry.

This process leads to multiple potential problems. The most likely error is one of transcription, where a clerk enters information from the vendor incorrectly. While the vendor can most likely be counted on to point out the error if it has resulted in a short payment, unfortunately it is not a certainty that this will be the case when they have been overpaid.
4.0 Current Paperless Technologies

4.1 EDI

4.1.1 History

EDI stands for Electronic Data Interchange, and it is a group of detailed standards which allows companies to transmit information directly... or at least fairly directly, since the internet did not exist when the standards were developed.

EDI’s initial design was not necessarily to connect buyers and sellers, but instead to provide information electronically to the transportation companies charged with delivering the goods between them. This led to the formation of the United States Transportation Data Coordinating Committee, which was an attempt to integrate and translate four sets of competing standards developed by the rail and trucking industries. Subsequently the American National Standards Institute upgraded the standards created by the Transportation Data Coordinating Committee with what became known as the X12 standards.

It is at this point that a fork was reached in the road to standardization. As Roger Clarke (1998) explains, “At about the same time, the U.K. Department of Customs and Excise, with the assistance of SITPRO (the British Simplification of Trade Procedures Board), was developing its own standards for documents used in international trade, called Tradacoms. These were later extended by the United Nations Economic Commission for Europe (UNECE) into what became known as the GTDI (General-purpose Trade Data Interchange standards), and were gradually accepted by some 2,000 British exporting organizations.”

This was not the only deviation from universal standards that was to occur. Large companies like General Motors, Ford, and Chrysler that were pushing forward with efficiency projects such as Just-In-Time inventory were also interested in the increased speed of EDI, which would allow even faster ordering and tracking of supplies, but each company was interested in having suppliers implement EDI in ways that were best for them, which would cause significant problems and excessive costs for suppliers that were servicing multiple automobile companies. At this point, the trade group for the automakers, the Automotive Industry Action Group, stepped in and developed an EDI standard for the automotive industry. Other industries have followed suit, and what began as an attempt to provide universal electronic communications to foster efficiency and save money has fragmented significantly.
This push to EDI occurred before the widespread adoption of the internet, but after the time when even small companies were operating personal computers and modems were relatively affordable. However, EDI proved to be a relatively costly exercise for small business. To sell to Wal-Mart, a small company needed to purchase EDI translation software as well as contract with one of the private networks that operated as EDI mailboxes… sending messages on the company’s behalf, and holding messages received by large companies such as Wal-Mart until the small company dialed in. In addition to these costs, the software had to understand the document standards used by Wal-Mart, and this required the purchase of data dictionaries. In the early 90’s, this could quickly add up to thousands of dollars as each dictionary was hundreds of dollars, and to communicate with Wal-Mart you might need a minimum of four templates:

- The Purchase Order template, which is used to decode Wal-Mart’s automated purchase order that is sent to you.
- The Purchase Order Acknowledgement template, which you use to confirm to Wal-Mart that you have received and accept the purchase order and its terms.
- The Shipping Information template, which tells Wal-Mart what has been shipped and where, matching up with the purchase order.
- The Invoice template, which is used to decode Wal-Mart’s automated invoice that is sent to you.

For small companies, integrating these procedures with their internal accounting systems would be extremely cost prohibitive, which meant that the usual procedure was as follows:

1. Dial in to send/receive any documents. If a purchase order is retrieved, print a copy. The system would generally automatically send an acknowledgement of receipt.
2. Manually enter the information from the printout into the internal systems to provide it to the production/shipping departments.
3. When the goods have been shipped, manually enter the shipping information and the invoices into documents to be sent and dial in to send them.

This has the ultimate effect that the costs and time spent were higher for the small company than had been the case before this “efficient” method, and increased the possibilities of transcription error. When Wal-Mart, the world’s largest retailer, insisting that it’s suppliers use EDI most of them had little choice but to accept the cost burden. Even with this sudden penetration of EDI, EDI capability was not significantly increased because most companies were only willing to spend the money for Wal-Mart templates, which were not usable to conduct business electronically with any other company.
4.1.2 Technical Information

What is commonly thought of as EDI generally includes both the actual format of the data being exchanged as well as the formats used in communicating between EDI participants, but for the purposes of this paper we will be focusing solely on the message details. There are two reasons for ignoring the communications portions of EDI, the first being that this portion has been undergoing significant change as the common method of implementing EDI has moved away from using a third party network to direct connections, usually over the internet. The second reason is that we are comparing the format to other formats which are independent of the method of communication. There is an argument that the tighter integration between the message and the method of delivery in EDI has been a detriment for the format, as it was slow to adapt when superior communication methods like the internet appeared, and alternatives were able to gain in popularity.

A message in EDI format is highly structured, such as in this sample purchase order:

```
ISA*00*   *00*    *08*61112500TST    *01*DEMO
WU000003
*970911*1039*U00302000009561*0*P?
GS*PO*611125001I*WU0000003   *970911*1039*9784*X*003020
ST*850*397822
BEG*00*RE*194743**970911
REF*AH*M109
REF*DP*641
REF*IA*000100685
DTM*010*970918
N1*BY*92*1287
N1*ST*92*87447
N1*ZZ*992*1287
PO1*1*1*EA*13.33**CB*80211*IZ*364*UP*718379271641
PO1*1*2*EA*13.33**CB*80211*IZ*382*UP*718379271573
PO1*1*3*EA*13.33**CB*80213*IZ*320*UP*718379271497
PO1*1*4*EA*13.33**CB*80215*IZ*360*UP*718379271848
PO1*1*5*EA*13.33**CB*80215*IZ*364*UP*718379271005
CTT*25
SE*36*397822
GE*1*9784
IEA*1*000009561
```

While some of the information can be divined without knowing the standards, such as the five items being ordered, the vast majority would need to be translated by computer software that is designed to interpret the codes. While the codes are defined by standards, the actual implementation of the codes can be manipulated by companies depending on their needs, which is what caused the need for data dictionaries, which are simply coded forms that convert information into and out of the necessary EDI formats.
4.1.3 Current Status and Limitations

EDI appears to be undergoing a change in that many companies are treating it no longer as a separate, standalone protocol, but rather as a subset of XML. This allows companies to use the internet as a delivery mechanism rather than proprietary networks, allowing more cost effective real time exchanges of information. Existing implementations are generally still being used, but newer implementations are being designed with the ability to decode both EDI and XML formats, allowing for greater versatility. Industries where EDI had not been prevalent, or where the EDI format was completely unsuitable for their purposes (such as encoding patient information for the health industry) are instead developing formats. It would appear that EDI will be relegated to a small pocket of legacy users who cannot afford to upgrade to newer formats that gain popularity.
4.2 XML

XML is not specifically a protocol so much as it is a meta language used to define other markup languages. It has defined methods of separating structure from information which can be used for any possible purpose, but those purposes need to be defined and codified in a set of standards that the users agree upon. These standards then become variants of XML and can be used by computers which are set up to understand these variants, as well as by human beings, since XML is also designed to be easily understood by users.

An XML document can be composed of any type of information. The following example shows what an XML file that contained information about this paper might look like.

```xml
<Paper>
  <Title>Implementing Paperless Automation Of Accounts Payable Invoices For Small Business Accounting Systems</Title>
  <Chapter number='1'>
    <Heading>Introduction</Heading>
    <Text>Improving productivity is a goal for every well run organization, enabling them to do more with less...</Text>
  </Chapter>
</Paper>
```

The key features of XML documents are that information content is both defined and encapsulated by identifiers within `<brackets>`. In this example, the entire file is within the Paper brackets, meaning it is entirely one paper. All of the text between `<Title>` and `</Title>` is defined as the Title. Inside the brackets, attributes can be defined, must as we have identified the chapter as chapter 1 inside the `<Chapter>` bracket. While there are other optional features, this is the essence of an XML document.

The choice of headers was made to make the document understandable by a human being, which is not necessary, but is recommended. The file would be equally valid as an XML document in the following example, but less useful to a human:

```xml
<X1>
  <T1>Implementing Paperless Automation Of Accounts Payable Invoices For Small Business Accounting Systems</T1>
  <X2 n='1'>
    <T2>Introduction</T2>
    <X3>Improving productivity is a goal for every well run organization, enabling them to do more with less...</X3>
  </X2>
</X1>
```
Since this would only save 46 characters, and electronic storage space grows cheaper with every year, the benefits from using smaller tags are definitely outweighed by the benefits of using descriptive tags.

This is only part of a solution, since a computer program or website that was designed to store papers would need to know how to deal with the tags that I had selected. If the website expected the document to be defined as <Thesis> instead of <Paper>, it would completely ignore everything in the document. Differences in other tags might cause partial or erroneous displays. Another difficulty arises from one of XML’s virtues, its universality. Without an operating system that is designed to interpret XML documents, or at least to properly route them to the appropriate application, an XML document would require even greater work as a human first opened it up to determine what content it held, and then loaded that file into the appropriate application. That said, XML based formats hold the promise for the greatest longevity and compatibility over long periods of time.

### 4.2.1 History

XML is the latest iteration of a series of languages that stem from the root idea that data files should be coded in a generic format rather than a specific layout. Traditionally, files would be a defined format, with a specification that would define each record, such as the following example:

<table>
<thead>
<tr>
<th>Character Length</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Day Of Month</td>
</tr>
<tr>
<td>2</td>
<td>Month</td>
</tr>
<tr>
<td>2</td>
<td>Year</td>
</tr>
<tr>
<td>6</td>
<td>Record Number</td>
</tr>
<tr>
<td>6</td>
<td>Account Number</td>
</tr>
<tr>
<td>20</td>
<td>Description</td>
</tr>
</tbody>
</table>

This would result in a file that might look like this:

```
010195000001100020CASH_RECEIVED_______020195000002100020INTEREST_ON_ACCOUNT_
```

Alternately, instead of defining the length of each field, a separation character such as `^` could be used, which would make the file look like this:

```
01^01^95^000001^100020^CASH_RECEIVED^02^01^95^000002^100020^INTEREST_ON_ACCOUNT^```
According to Charles Goldfarb (2003), one of the earliest pioneers of generic data files:

“Many credit the start of the generic coding movement to a presentation made by William Tunnicliffe, chairman of the Graphic Communications Association (GCA) Composition Committee, during a meeting at the Canadian Government Printing Office in September 1967: his topic -- the separation of the information content of documents from their format.

Also in the late 1960s, a New York book designer named Stanley Rice proposed the idea of a universal catalog of parameterized 'editorial structure' tags. Norman Scharpf, director of the GCA, recognized the significance of these trends, and established a generic coding project in the Composition Committee.

The committee developed the 'GenCode(R) concept', recognizing that different generic codes were needed for different kinds of documents, and that smaller documents could be incorporated as elements of larger ones. The project evolved into the GenCode Committee, which later played an instrumental role in the development of the SGML standard.”

Charles Goldfarb was working for IBM in 1969 with teammates Edward Mosher and Raymond Lorie when they expanded on the concepts of Rice and Tunnicliffe, adding in formal definitions of the document type and the nesting structure which has been a key structure in most subsequent languages. The team named this language GML, the initials of the three inventors, which according to Charles Goldfarb, was not a coincidence. To maintain GML as a reasonable acronym, they also named it the General Markup Language, and the term markup language would stick through all of the subsequent derivative languages.

An example of GML code is provided below to help illustrate the differences between the original language and the subsequent derivatives.

```
:h1. Chapter 1: Introduction
:p. GML supported hierarchical containers, such as
:ol
:li. Ordered lists (like this one),
:li. Unordered lists, and
:li. Definition lists
:eol.
:as well as simple structures.
:p. Markup minimization (later generalized and formalized in SGML),
:allowed the end-tags to be omitted for the "h1" and "p" elements.
```
Years later, in 1978, the American National Standards Institute established a committee focusing on Computer Languages for the Processing of Text, and invited Goldfarb to join and lead a project to develop a text description language based on GML. Over the course of the next seven years SGML (Standard Generalized Markup Language) was defined and approved. SGML not only provides the basic structure for document storage, but also allows the creation of specific document structures using Document Type Definition functions. Thus, SGML, like XML, is a meta-format, which is not surprising since XML itself is a stripped down version of SGML, removing a number of the features of SGML that were unwieldy for internet use.

The first movement to a document format for the web, however, was HTML, which stands for Hypertext Markup Language, not XML. It began when Tim Berners-Lee began to develop proposals to combine hypertext capability (the ability to have links in documents to other documents) with a variant of SGML designed for the internet. It succeeded in becoming the standard, but it was not implemented as a true SGML language, which meant that as time went on and new features were added, it became more and more convoluted, which led to the need for an alternative. As Tim Anderson (2004) writes:

“In a nutshell, HTML is too limited and terminally polluted, while SGML itself is reckoned to be too complex for mortals to implement. In the late 1990s a group of people including Jon Bosak, Tim Bray, James Clark and others came up with XML, eXtensible Markup Language. Like SGML, XML is not itself a markup language, but a specification for defining markup languages. The W3C (World Wide Web Committee) immediately set about reshaping HTML as an XML application, with the result being XHTML. That is only one small part of what XML is all about. The key point is that using XML the industry can specify how to store almost any kind of data, in a form that applications running on any platform can easily import and process.”

XML Version 1.0 was initially defined in 1998, although those definitions were not locked in stone. Minor revisions have occurred since then without the specification receiving a new version number, although the specifications are on their fourth edition at this point. A second version, XML Version 1.1 was published in 2004, and like Version 1.0 has received some updates without receiving a new version number, it is currently on its second edition. Version 1.1 has some new features which are not commonly accepted, which has led to Version 1.0 being the more widespread version in use.
4.2.2 Industry Specific XML Languages

There are literally hundreds, if not thousands, of markup languages derived from XML, such as MathML, an XML variant used for describing mathematical notation, or SVG, short for Scalable Vector Graphics, used to produce vector graphics, with more being added all the time. A large number of these are industry specific, such as Chem eStandards, which are the standards for the chemical industry, PIDX for the petroleum industry, or adXML for the advertising industry. While there is no complete list of all the variants used, a list found at [http://xml.coverpages.org/xmlApplications.html](http://xml.coverpages.org/xmlApplications.html) shows over 600 XML sub languages, and that list hasn’t been updated since June of 2005.

A more generalized XML variant designed to serve business needs has been in development for a number of years, known as ebXML. Like EDI, however, ebXML seeks to fully design every aspect of it’s usage from communications through to business documents, and has grown into such a large behemoth that after 5 years of being an official standard, it’s market penetration is even less than EDI’s and early proponents such as Klaus Naujok feel that it has become merely a stepping stone rather than a solution for business. Another general format, UBL (Universal Business Language) has incorporated some of the work done on ebXML while having more of a focus on defining actual document formats, but it has failed to develop a critical mass, and also suffers from having documents that are too highly defined to allow for expansion without changes to the specification itself.

Samples of some invoice files for industry specific XML variants are provided in Appendix A.

4.2.3 Current Status and Limitations

XML is quickly being adopted as the basis for all kinds of electronic documents, largely due to it’s absorption and planned replacement of HTML as well as Microsoft’s recent decision to have OOXML (Office Open XML) be the default file format for Office 2007. In addition, a large number of free XML decoder programs are available; although admittedly decoding an XML document is the simplest part of the process; mapping the decoded content to its uses is the more difficult and time consuming part.

While there are a few known problems with XML, such as the increased size of the files resulting in an increased amount of storage and processing power needed, overall it is a superior method of inter-business communication, to a large degree because of the high level of adoption that currently exists.

Pure XML alone will not be adequate to use as a basis for the new protocol that we wish to create, rather, we will need to define an XML subset, except one that is generic enough to be used by any company, yet one that is simple enough to generate minimum additional costs where companies are already using an industry specific variant.
5.0 Adoption of Successful Software Protocols

5.1 HTML

HTML, when it was first introduced, was the evolution and combination of two trends, the move towards a natural language storage format for documents, separating the content from the format (as was pioneered in first GML and then SGML), and the ability to hyperlink documents. Hyperlinking itself was an evolution and combination of the academic citation and referencing procedures with electronic data storage. According to the Wikipedia entry on the history of the hyperlink (2007):

“The term "hyperlink" was coined in 1965 (or possibly 1964) by Ted Nelson at the start of Project Xanadu. Nelson had been inspired by "As We May Think," a popular essay by Vannevar Bush. In the essay, Bush described a microfilm-based machine (the Memex) in which one could link any two pages of information into a "trail" of related information, and then scroll back and forth among pages in a trail as if they were on a single microfilm reel. The closest contemporary analogy would be to build a list of bookmarks to topically related Web pages and then allow the user to scroll forward and backward through the list.

In a series of books and articles published from 1964 through 1980, Nelson transposed Bush's concept of automated cross-referencing into the computer context, made it applicable to specific text strings rather than whole pages, generalized it from a local desk-sized machine to a theoretical worldwide computer network, and advocated the creation of such a network.”

HTML was the creation of Tim Berners-Lee, and as a result he is widely considered to be the inventor of the World Wide Web. Additionally, Berners-Lee developed a prototype web browser for the NeXT computer in 1990. According to Dave Raggett, “The fact that the Web was invented in the early 1990s was no coincidence. Developments in communications technology during that time meant that, sooner or later, something like the Web was bound to happen. For a start, hypertext was coming into vogue and being used on computers. Also, Internet users were gaining in the number of users on the system: there was an increasing audience for distributed information. Last, but not least, the new domain name system had made it much easier to address a machine on the Internet.”

As an academic, it was natural for Lee to disseminate his work, and he used one of the new communication tools, an internet mailing list dedicated to the WWW. This expanded the development of the protocol to include other interested users, and attracted the attention of NCSA, the National Center for Supercomputer Applications, who decided to write their own browser called Mosaic. (One of the programmers on the team was Marc Andreessen, who would later go on to develop the Netscape Navigator browser, which would later fork to become the Firefox browser).
By 1994, HTML had grown in popularity to the point that a conference was held, both to celebrate what had been accomplished as well as to plan further evolutions of the language. This was also the year that the World Wide Web Consortium was founded to decide on official HTML standards. At this point, a number of browsers had “embraced and extended” the language, which caused incompatibilities, and to deal with this problem, Dan Connolly and some of his colleagues collected all of these changes and developed HTML version 2. Dave Raggett then expanded on that work, collecting a very large number of new features, including the introduction of tables, forms, and style sheets. While browser programmers adopted what would become known as HTML 3, the standards settings bodies that had assumed responsibility for development did not ratify it until version 3.2, and that ratification came in 1997, long after the browsers had implemented the features.

The work of the World Wide Web Consortium grew increasingly bogged down as more and more people wished to contribute to the development of the language, so Dave Raggett attempted to speed up the process by inviting the actual developers of the various browsers to work as a small group to standardize HTML. This proved to not only be successful, but to set a pattern of browsers implementing features long before the standards were actually set in stone. HTML has since gone on to not only develop multiple versions, but to be incorporated into XML as XHTTPML.

Therefore, the success factors of HTML were:
- The natural evolution and combination of existing technologies.
- The fulfillment of a need that had not been satisfied yet.
- The technology was free, not just of cost, but of licensing restrictions.
- Support from respected third parties capable of ensuring implementations.
- A somewhat open nature allowing expansion over time of the core language, however, the length of time it took to make changes in HTML did cause additional features to not be incorporated via HTML but instead via plugins, such as Java, Javascript, and Flash.

5.2 PDF

Like HTML, PDF has its roots in much older technologies designed to deal with documents. In the early days of computing, printers tended to have different methods of control, so any special formatting beyond straight text required the program to know what those codes were. Since the program rather than the operating system controlled the printing in those days, enabling different printer types was duplicate work for every programmer, which caused the market to tend to standardize on only a few control sets, such as the Epson printer control codes.
The problem with these control codes are the same as those faced by other closed protocols, it was not possible to expand the features to deal with changes such as laser printing, higher resolutions, or the move to color. John Warnock and Charles Geschke, two employees of Xerox felt the solution was to develop a language to control printers, so that all programs could output in that language, and all printers could interpret the language using whatever methods were the best for that model. When Xerox was unwilling to commercialize it, the pair jointly founded Adobe and developed PostScript, which, along with their other programs such as Adobe Type Manager and Adobe Illustrator, helped usher in the age of desktop publishing.

While PostScript output was uniform and could be printed essentially identically on any PostScript printer (depending on the print capabilities and resolution), and PostScript files could be saved and used independently of the program that created them, unlike faxes received on a computer modem, they could not be easily displayed on most small computers which were, at the time, limited to 640K of memory and displays that were lower resolution than many printers. While Adobe had a product called Display PostScript that would theoretically solve this problem, as John Warnock (1991) said in his Camelot Paper (which was the first plan for what would eventually become PDF):

“The reason the Display PostScript and PostScript solutions are not a total solution in today’s world is that this solution requires powerful desktop machines and PostScript printers. The Display PostScript and PostScript solutions are the correct long-term solution as the power of machines increases over time, but this solution offers little help for the vast majority of today’s users with today’s machines.”

As PostScript was in essence a fully featured language, it could easily be modified to operate in a different manner. Instead of saving files using high level methods that made for a smaller file but required higher printer processing power, it could save files that used lower level methods, for example, drawing four individual lines rather than providing starting coordinates, width and length for a square.

Warnock’s original plan encompassed most of the features that have come to be known in the PDF format; the ability to view documents independent of platform, the ability to use documents across a network or via email to allow print on demand, eliminating paper waste, providing visual data such as maps and instruction manuals, and text searching. Originally named IPS for “Interchange PostScript”, it was renamed to PDF when it was announced at Comdex in 1992, but when it was launched in 1993 it did not make a significant impact. As colors were restricted to RGB it was not much use to the printing industry, and Adobe was charging $695 USD for the personal Acrobat creator or $2495 USD for the network version. It even charged $50 USD for the reader software.
In late 1994 they released PDF version 1.1 which added support for external links similar to HTML links, article threads and document security features, and most importantly, began distributing the reader software for free. It wasn’t until version 1.2 in 1996 that it became useful for prepress work as it added support for OPI 1.3 specifications, CMYK colors, and halftone and overprint instructions. With each subsequent upgrade to the PDF standards, Adobe Acrobat and the Adobe Acrobat Reader gained additional features and additional markets, not just from office users and the publishing industry, but also home users, electronic books, and web scripting features. As of November 2006, the PDF specification was at version 1.7, with a few offshoots such as PDF/X for printing and graphic arts, PDF/A for corporate/government/library environments, and PDF/E for engineering drawings. Adobe is also reportedly exploring an XML base for the next generation of PDF, codenamed Mars.

As of version 1.7, Adobe made the PDF format an open standard, allowing other companies to use the PDF format without paying Adobe a licensing fee, and is in the process of making it an ISO standard.

The success factors for PDF were:

- The natural evolution and combination of existing technologies (postscript, publishing technologies, networking, digital imaging, and more powerful computer hardware).
- The fulfillment of a need that had not been satisfied yet. (While there were competitors at the time, they had not yet reached a critical mass to become useful).
- The technology did not become common until the reader was free, and while Acrobat Distiller did cost, the ability for developers to add plugins allowed for cost effective power increases where needed.
- A somewhat open nature that allowed new versions of the PDF standard to be developed while still allowing backwards compatibility, due to its status as a “language” like PostScript rather than a closed file format.
5.3 QIF/OFX/QFX

Similar to the goal of this project in eliminating the time consuming and error prone work of printing a paperless invoice only to turn around and enter it into the accounting system, the makers of Quicken personal financial software saw that users of it's software were viewing their transactions online and then re-entering them in exactly the same manner. To eliminate this duplication, the makers of Quicken first developed QIF, the Quicken Interchange Format, which was provided to banks and other potential users for free. Because Quicken was the dominant player in the personal software market, and had made their programs compatible with QIF, it was a minimal amount of work for banks to add this feature, which was greatly appreciated by customers. This had most of the same success factors as HTML, except for one significant difference - QIF was not easily upgradeable, and because the format was controlled by Quicken, it required the support of Intuit, the owners of Quicken, to add features. A sample of a QIF file follows:

!Type:Bank
D06/01/2007
T-50.00
MINTERNET BILL PAYMENT;VISA, SCOTIABANK-SCOTIALI;
^D06/01/2007
T-6.45
MPOS MERCHANDISE;MCDONALD'S #828;
^D06/01/2007
T-20.44
MINTERNET BILL PAYMENT;ENBRIDGE;
^D06/01/2007
T-10.00
MPOS MERCHANDISE;A & P #332;
^

As XML based formats increased in popularity and the range of financial products increased, the limitations of QIF quickly became apparent. Intuit itself claims on its QIF Data Import Users Resource Center (2007) that “QIF technology is over 10 years old and was designed for technical support purposes, it was not for transaction download. QIF Data Import requires many steps to download, is a poor customer experience and can lead to duplicate transactions and errors. Intuit wants to provide customers with the best possible download experience available as well as support the ongoing industry transition to OFX (Open Financial Exchange).” In 1997, Quicken, along with Checkfree (a partner company) and Microsoft (the largest competitor to Quicken, makers of Microsoft Money personal financial software) agreed to a more complete, open XML based standard called OFX. The same transactions shown above are now shown using OFX:
Like QIF, OFX would be offered free to banks to implement, and it is assumed that the benefit to the makers of the personal financial software would be increased sales from customers wishing to upgrade to take advantage of the superior benefits. Unfortunately, Intuit decided that those benefits would not be adequate, and decided to modify the OFX format into a custom format called QFX, and then charged the banks a license fee if they wished to offer the option their customers. QFX is almost identical to OFX, in most cases there is only a single additional field relating to the financial institution that is providing the download.
Quicken software prior to Quicken 2005 allowed users to import in the .QIF format, but that feature was removed for bank accounts as of that version, and .QIF importing for credit cards was removed in Quicken 2006. The code to import .QIF remains in the program, and is still used for some investment and asset accounts. In addition, Intuit has not provided the ability to import the free OFX format into its software, it only supports the QFX format. Because of the loss in functionality, a number of customers indicated that they would not upgrade, but would stay with earlier versions, and in response, Intuit chose to “sunset” (remove online support from) earlier versions at a faster rate to force customers to move to versions with .QIF support removed. As Ed Foster (2005) notes, “Figuring from when each product was released until its official sunset, Quicken 98 users got to use their product more than six years. Quicken 99 got five-plus years, Quicken 2000 four years and eight months, Quicken 2001 four years and seven months, and Quicken 2002 three years and seven months.” Even more unnerving from an ethics point of view, Quicken has chosen to require the financial institutions to pay for an additional license if they want to offer .QFX downloads to Mac users, however, the .QFX file downloaded is absolutely identical, it is just that Intuit has designed Quicken for the Mac to first check with Intuit that the bank has paid for a Mac license, and if not, it won’t process the file. Some financial institutions have chosen to risk not offering .QFX files, some only offer .OFX and .QIF files, and some offer .QFX files, but charge a fee to their customers for the service. As could be expected, since Quicken no longer holds a monopoly on personal financial software, adoption has been drastically lower for .QFX than .QIF and .OFX, and customers of Quicken are finding newer versions often provide less functionality rather than more.

This case helps highlight the success factors for QIF as contrasted by the lack of success of QFX. They are:

- The natural evolution and combination of existing technologies.
- The fulfillment of a need that had not been satisfied yet.
- The technology was free, not just of cost, but of licensing restrictions.
- Support from respected third parties capable of ensuring implementations.

QIF failed to maintain its success due to:

- The failure of the protocol to allow for change.
- The desire of a third party to ensure its failure for monetary gain.

OFX has succeeded for the same reasons as QIF, but not to the same degree because only Microsoft is presently supporting the free OFX format, which limits the possible penetration to Microsoft’s market share. QFX has partially succeeded, but that is entirely due to Intuit’s abuse of it’s market position to try and pit it’s customers, who are forced to upgrade and can now only use QFX, against banks, who do not want to pay the additional fees to Intuit for which Intuit is providing no service, support, or technology.
5.4 Open Source Software

Open source software is a concept way too broad to address in total, but there are a few concepts that are relevant to the success of the project based on the success factors identified in the previous sections. Open source software has often been confused with free software, largely because the advocates for open source are often the same advocates for free software (“information wants to be free”). There is a common expression regarding the difference, “free as in speech, not free as in beer”, which refers to the fact that open source software may cost money, but the technical details of its code are open, not hidden. As we have seen in the previous sections, using an open format is one of the common success factors for all of the successful adoptions as it helps spur the adoption of a critical mass. Building upon an existing open source format such as XML also means that there is a large body of free software and code in existence to decode XML formats that can be integrated into accounting software.

5.5 Success Factors to Implement

By modifying the success factors of the successful formats we can determine some of the features that should be incorporated in the new protocol.

Each of the successes we have looked at have been extrapolations of existing technologies put to new use. In this case, we will be continuing the trend by combining electronic delivery of business documents with rudimentary artificial intelligence rules based techniques to allow us to define an absolute minimum amount of the invoice to maximize its flexibility.

Another key feature has been that each success has fulfilled a need that had not yet been met. This is not to say that there were not competitive products or plans, but rather that they did not develop the critical mass necessary. In this case, EDI and ebXML have attempted to approach the need from a universal business fashion, but have failed to achieve critical mass because of the high level of definition required, which in turn require high levels of programming to use. Individual industry XML formats also fail to meet this need due to the high level of specialization involved; it would not be practical for other industries to implement the formats just to allow automated invoicing.

A third key factor was the formats being free to use and work with. While Adobe did charge for the PDF creation software, as they provided the reader for free and made PDF an open standard free software providers were able to provide PDF creation functionality, such as FreePDF. As there is no need to profit from the implementation, since it will lower costs (increasing profits) for all companies if it achieves critical mass, providing the format for free is viable.
A fourth key factor was the support of a third party who can provide support for the format. There is a large number of accounting software companies that provide mass market accounting software which could provide support for this protocol, such as Microsoft, who bought Great Plains Accounting software, Sage Software, makers of Simply Accounting, or MYOB Inc, makers of Mind Your Own Business. Obtaining the support of multiple companies would increase the chances of success, but doing so will not occur during the timeframe of this paper.

The final common success factor was the ability to expand the format to deal with new eventualities. This protocol will incorporate this success factor by being based on the open ended XML format and by including only a minimum amount of definitions, leaving the remainder of the file to contain whatever additional information that is desired and leaving it to the user to teach the accounting system which additional information is relevant and how it should act upon it.
6.0 Development of Invoice Protocol

6.1 Key Information Required

While the key to creating a universal protocol is to reduce it to the lowest common denominator, it is critical that certain minimum information be provided. There are three categories of information that we need to make this protocol successful. The first is structural information relating to the protocol itself. The second is universal data which applies to every single invoice, without which there is not enough information to enter the invoice and which therefore is more efficient to code into the system, and the last category covers all the remaining information that is part of an invoice but which may vary from company to company.

Category One: Structural Information

- We must indicate that the information is part of an invoice relating to this protocol, most likely by encapsulating the data.
- We must have a consistent way of indicating which pieces of information are fields and which are data elements.
- We must have a consistent way of indicating which data elements are dollar amounts, such as currency.
- We must have a consistent way to deal with totals and subtotals; either the individual dollar amounts are provided but not totaled as part of the protocol, or the totals are provided, in which case an indication that the data element is a total is needed.
- We must have a method of grouping data fields from a single line item, most likely by encapsulating each individual line.
- We may want a method of flagging items which are unusual or changed so that the receiving system knows to double check rules relating to that information item, such as when a supplier is selling goods with different payment terms than specified by the agreement. On the other hand, we may simply want to provide a structure which allows an alert to be provided to the clerk requiring their manual attention for the processing.
Category Two: Universal Data

- The supplier who is generating the invoice must be indicated in some manner which can be differentiated between similar companies.
- The date of the invoice and the invoice number must be provided by the supplier.
- The construction of each individual line or line item may be flexible (for example, a $1000.00 purchase might show $60.00 GST as part of the same line, might show the $60.00 GST on a separate line, or might show the GST for the entire invoice on a separate line), however, all of the individual currency elements must total up to the invoice total.
- Any information provided on other documents (for example, on packing slips sent with the shipment) must be consistent with the data provided in the invoice.

6.2 Protocol Delivery

Although this protocol is being designed to be a “universal” protocol, there are cases where more specific protocols will provide superior cost savings to companies, especially if systems have already been developed to take advantage of them. While this can be achieved by providing the invoice in both formats to the customer, it might be advantageous to integrate it with existing file formats. This is not possible with any of the EDI formats, but it would be possible using XML formats. The following code shows a sample of the structure that an invoice created with the VISA XML protocol would use:

```xml
<!DOCTYPE Invoice SYSTEM "invoice.1.0.dtd">
<Invoice sectorUsageVersion="1">
  <InvoiceHeader>
    :
  </InvoiceHeader>
  <InvoiceDetails>
    :
  </InvoiceDetails>
  <InvoiceSummary>
    :
  </InvoiceSummary>
</Invoice>
```
If the new protocol were appended to the bottom of this file, and used similar XML specifications, it might look like this:

```xml
<!DOCTYPE Invoice SYSTEM "invoice.1.0.dtd">
<Invoice sectorUsageVersion="1">
  <InvoiceHeader>
    ...
  </InvoiceHeader>
  <InvoiceDetails>
    ...
  </InvoiceDetails>
  <InvoiceSummary>
    ...
  </InvoiceSummary>
</Invoice>

<!DOCTYPE NewInvoice Protocol>
<NewInvoiceProtocol>
  <NewInvoice General Items/>
  <NewInvoice Line>
    <NewInvoice Line Items/>
  </NewInvoice Line>
</NewInvoiceProtocol>
```

Theoretically, this would allow an accounting system that was programmed to recognize both protocols to use the more detailed Visa protocol first, whereas accounting systems that were only programmed for the new invoice protocol would ignore everything between the `<Invoice>` and `</Invoice>` and begin with the `<NewInvoiceProtocol>` header. There are positives and negatives for this approach, as detailed below:

**Positives:**

- This would enable customers to have a single file to import no matter which of the two files were used.
- Costs to modify the vendor’s invoicing systems may be reduced (for example, if the vendor’s website allows downloading of previous invoices, the modifications to change the downloadable files may be less than the modifications required to allow the customer to download the invoice in multiple formats.)
Negatives:

- This would impossible to implement with any protocol other than an XML format, including EDI, which would mean a separate file for a significant amount of invoices, if not a majority of them. This could lead to confusion as to when an invoice has been combined as opposed to “standalone”.
- Combining with another file type might mean that features that are activated by recognizing it as a file in this protocol would not work consistently, such as double clicking the file.
- The size of the digital “file” would increase, taking up storage space for information that is worthless to the company, unless extra resources are spent by the customer to strip out the unwanted data.
- Merging with another protocol’s delivery mechanisms will make the new protocol less visible, which could be a negative during the buildup to a critical mass of users.
- Accounting systems that use these specialized formats would require additional programming to separate out the secondary format from the first.

In addition to the positives and negatives from the customer’s side, we can look at the cost of implementation from the vendor’s side. Since the only difference between those two options for the vendor would be the additional code to deal with the merged files, it appears that the ideal way to move forward is to keep the files completely separate.

6.3 Protocol Branding

At this point in the development process we should begin considering a name for the protocol. Not only will a proper nomenclature make subsequent references to this protocol more convenient, but it may also help encourage word-of-mouth adoption. While it is common to just use an acronym that doubles as a file extension (PDF is short for Portable Document Format and HTML is short for Hypertext Markup Language), one that also relates to the purpose would be ideal. As we are not attempting to build any sort of secrecy around the protocol, a fake or “working” name is not necessary, however, if another “branding” is desired by adoption partners we should remain prepared to change it before public release. Marketing will be an important part of the protocol in large part because of the critical mass required before the protocol’s value outweighs its costs; it will require marketing to convince potential users that they should embrace the protocol prior to critical mass being reached to help propel it towards that point.
In his book, “Marketing Management”, Philip Kotler (2003) describes how marketers need to think through the five levels of the product, from core product through to potential product, and using this technique to analyze our protocol should help with our branding. The first level is the core benefit, which in this case is a method for inter-company automation. It is a good thing we do not intend to charge for this product, as it is useless unless we reach a critical mass of users, and for that to occur, these users need software that can take advantage of the protocol. This brings us to the next level, the basic product. There are actually two basic products that are necessary; software to generate outbound invoices in the proper format, and software to process inbound invoices, although both may be combined in a single accounting system, as virtually every business both sends and receives invoices. As we will be relying on third parties to provide this basic product as part of their existing accounting packages to help develop our critical mass of users, it is essential that we also target our marketing at these partners.

The next level of Kotler’s is the expected product, and this will largely be shaped by the quality of software that the user is provided. A very simple rule based program will “get the job done”, but a rule based program that borrows from artificial intelligence techniques and combines it with an intuitive and highly productive user interface will raise the bar. It is unlikely, however, that products will reach this level in the short term, as partner companies will instead provide the minimal amount of functionality until such time as critical mass is reached. In a similar vein, the augmented product level will also not be “in play” until users begin demanding additional features related to the product.

If the techniques used in this protocol prove successful and are embraced by users, they can be applied to many other similar administrative tasks, and to the degree that these additional features are considered related to the new protocol developed in this paper, it may be that the fifth level, the potential product, may be reached very shortly after the other levels are reached.

Therefore, we have determined that the marketing should
- Be directed both at users and at potential partners
- Help convince users that early adoption is worthwhile
- Be integrated into the naming of the protocol as well as the filename convention.
- Be able to be changed by partners to help promote their needs as well
- Show that the protocol can work independent of any particular computer type or operating system.

Based on those criteria, this paper shall henceforth refer to this as the “Universal Invoice” format, and the filename extension shall be .Uni. This can be branded at the implementation level to account for the significant differences that may exist between user experiences, such as “Quickbooks featuring the Intuit Universal Invoice Engine” or “Mind Your Own Business’s new feature: Mind Your Own Invoices”. As long as partners agree with the .Uni file format, the branding of their interpreters can only serve to promote awareness of the core format.
6.4 Protocol Schematics

- The entirety of the information that is being provided should be provided inside the `<Invoice>` tag and the file will end with the `</Invoice>` tag.

- All line items should start with `<Line>` and end with `</Line>`.

- All items between `<Invoice>` and `<Line>` are considered global invoice items, all items inside the `<Line>` are line items. Although `<Line>` items could theoretically be nested, it is unlikely that any users will program in that functionality since there is little, if any, practical purpose for it.

- Text inside an `<Alert></Alert>` are used to pass information to the person “entering” the invoice into their accounting system.

- The `<Vendor></Vendor>` tag is mandatory as a global identifier of the company supplying the invoice. It can be any text that the Vendor chooses.

- The `<Invoice#></Invoice#>` tag is mandatory and must contain the vendor’s invoice number.

- The `<Invoice_Date></Invoice_Date>` tag is mandatory and must contain the invoice date in YYYY/MM/DD format.

- Any dollar amounts should include the attribute Type=”$”, for example `<GST Type=”$”>60.00</GST>`.

- If totals or subtotals are included the attribute Type should be set to “+”, for example `<Grand_Total Type=”+”>5362.00</Grand_Total>`. Totals may be ignored by some accounting systems which choose to calculate the totals by summing line items.

- Comments should use the standard XML format of `<!-- Comment Text Goes Here -->`. These will only be seen if the file is examined directly using a text editor.

- Any elements should be a single word, for example “Grand_Total” rather than “Grand Total”.

Beyond these restrictions and requirements, any properly formatted information is possible and permissible.
7.0 Examples of Protocol Usage

7.1 Sample Invoice Files

7.1.1 Simple Service Invoice

This is a sample of what a single service invoice could look like for a small operator, such as a one man janitorial operation that bills monthly.

```xml
<Vendor>Joe’s Cleaning</Vendor>
<Invoice#>126</Invoice#>
<Invoice_Date>2007/05/31</Invoice_Date>
<Line>
<Description>Janitorial Services For May</Description>
<Price Type=""$">250.00</Price>
<GST Type="$">15.00</GST>
</Line>
</Invoice>
```

7.1.2 Complex Service Invoice

This is a sample of what a more complicated invoice could look like, and is modeled from the invoice information that is provided by our travel agency.

```xml
<Vendor>A. Nash Travel Inc.</Vendor>
<Invoice#>125143</Invoice#>
<Invoice_Date>2007-06-04</Invoice_Date>
<Agent>Larry Boldt</Agent>
<Phone>(905) 755-0102</Phone>
<Fax>(905) 755-0729</Fax>
<800#>800-263-0869</800#>
<Passengers>Smith/John Mr.</Passengers>
<Currency>CDN</Currency>
<GST#>123-231-987</GST#>
<TICO#>3012696</TICO#>
<PNR>JOTXLY</PNR>
<Line>
<Ticket#>2146665093</Ticket#>
<Supplier>Air Canada</Supplier>
<Ticket_Type>Electronic Ticket</Ticket_Type>
```
<Itinerary>Air Canada Flight AC400 (Seat 20C) Departing Toronto Pearson Airport Terminal 1 On June 21, 2007 at 7:00AM, Arriving at Montreal Trudeau at 8:12AM</Itinerary>

<Itinerary>Air Canada Flight AC425 (Seat 25C) Departing Montreal Trudeau Airport On June 21, 2007 at 7:00PM, Arriving at Toronto Pearson Airport at 8:18PM</Itinerary>

<Fare Type="$">354.00</Fare>
<GST Type="$">23.90</GST>
<QST Type="$">1.19</QST>
<Other_Tax Type="$">44.34</Other_Tax>
<FoP>AX3735***1011</FoP>

<Invoice>
<Supplier>A. Nash Travel Inc.</Supplier>
<Ticket_Type>Transaction Fee</Ticket_Type>
<Fare Type="$">30.00</Fare>
<GST Type="$">1.80</GST>
<FoP>AX3735***4002</FoP>
</Invoice>

7.1.3 Combination Goods and Services Invoice

This is a sample of what an invoice that was for the actual shipment of goods might look like.

<Invoice>
<Vendor>Sample Parts Inc.</Vendor>
<Invoice#>8622452</Invoice#>
<Invoice_Date>2007-06-04</Invoice_Date>
<Customer_PO#>AB2324234</Customer_PO#>
<Waybill#>F2343223</Waybill#>
<Customer_Ref#>Department 200</Customer_Ref#>
<Currency>CDN</Currency>
<Line>
<Part#>2342336</Part#>
<Quantity>23</Quantity>
<Unit>Boxes</Unit>
<Description>Toilet Seats For Government Use</Description>
<Item_Total Type="$">2300.00</Item_Total>
<Tax Type="$">0.00</Tax>
<!-- Note: Government is tax exempt. -->
</Line>
<Line>
<Part#>55288323</Part#>
<Quantity>6</Quantity>
<Unit>Each</Unit>
</Invoice>
7.2 Possible User-side Implementations

The key to this protocol ultimately fulfilling its mandate of saving time and paper is for the accounting system to have built up enough knowledge about how vendors are using the format that it is able to mimic the functionality of a fully defined protocol. The first step to achieving that is for the system to map the information found in the <Vendor> tag to the vendor number assigned by the purchasing company. The specific manner of doing so is dependent on how the programmer wants to design the interface, but in essence, the first time a Vendor is encountered, it should ask the user for the vendor number, and then for subsequent invoices from that Vendor, that information should be the default. The rule could be expressed as:

If <Vendor>=Joe’s Cleaning Then Vendor Number = 2042

The remaining header functions can be displayed along with the option to set rules based upon them. For example, you might see:

Customer_PO#: AB2324234
Waybill#: F2343223
Customer_Ref#: Department 200
Currency: CDN

The first time being used, the user could “teach” the program that the <Customer_PO#> field maps to the purchase order #, <Customer_Ref#> would map to an internal reference number, and the <Currency> would mean it is a Canadian dollar invoice. Alternately, fields could simply be ignored if the information is not relevant to the accounting system, such as the <Waybill#>.
Similarly, line items could be used or ignored depending on how the purchaser uses that information. A company that purchases goods for resale might map this line item as follows:

<table>
<thead>
<tr>
<th>Part#:</th>
<th>55288323</th>
<th>Maps to Customer Inventory #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity:</td>
<td>6</td>
<td>Is added to quantity on hand and confirmed against shipping receipt.</td>
</tr>
<tr>
<td>Unit:</td>
<td>Each</td>
<td>Already stored in the item description so it is ignored.</td>
</tr>
<tr>
<td>Description:</td>
<td>Government Hammer</td>
<td>Already stored in the item description so it is ignored.</td>
</tr>
<tr>
<td>Item_Total:</td>
<td>1200.00</td>
<td>Is compared to purchase order for price verification then mapped to item’s inventory cost.</td>
</tr>
<tr>
<td>Tax:</td>
<td>0.00</td>
<td>Is compared to purchase order for price verification then mapped to item’s inventory cost.</td>
</tr>
</tbody>
</table>

Alternately, a company that is just purchasing some replacement hammers for their maintenance department might instead teach the system that all items from the vendor Sample Parts Inc. should always default to the account for “Repairs Expense”.

The more invoices that are processed by the system, the more the system will learn, and the more time that can be saved by the employees. The first time an invoice is processed by a vendor that must be matched against a purchase order which contains a large number of items that must be matched to internal item numbers there may be negative time savings. However, a subsequent invoice being received may mean that the entire process is fully automated for that invoice.

Depending on the level of sophistication that companies seek to integrate into their accounting systems, the rules database can also provide increased functionality. By adding rules in directly, management could instantly block the entry of payables from certain suppliers, ensure that negotiated prices were properly charged, or require management approval for certain purchases. By changing the delivery system from email or website delivery to a direct connection between supplier and customer, the invoices could be entered free of tampering, preventing entry errors or fraud.
Another advantage of allowing undefined features in the protocol is that custom features can be easily added in. If a parent company of a conglomerate wants to add additional coding to automate inter-company transactions, they can simply decide on whatever custom elements they wish to use and teach their systems the meanings without breaking any of the core functionality of the system. If the user base wishes to add a feature through common consent, such as a <Logo> element that would contain the graphics from the invoice, they could do it merely by adding it in and providing the details on what format the graphics were in to other users by defining a new element type. It would merely be necessary for the users who wished to implement it to add some programming for a new element type, or merely add a rule for a new type of data.

Alternately, a company that can not afford regular custom programming changes will always be able to use any .Uni file ever created, and their systems will simply ignore all the new elements it does not know how to handle.
8.0 Conclusions

If Sellen is correct, then it is not enough for companies to want to go paperless, they must adjust their business processes to make them more efficient with paperless operations, and because invoicing is an inter-company process, to maximize efficiency will require an effort that involves as many companies as possible. Even with this understanding, the barrier to true paperless invoicing at the current time is the barrier to entry of initial cost for smaller companies. A minimal protocol combined with software that can learn from the user will enable that barrier to be lowered significantly for general use, while co-existing with more sophisticated protocols for specific industries. This protocol can easily be incorporated into the mass market accounting system packages for very little cost, and that market penetration could provide a large enough critical mass of users to spur universal adoption of the protocol.

As has been shown from the success of other protocols such as HTML, PDF, and QIF there are certain common factors that help lead to successful mass adoptions. These formats were all existing technologies that were combined and put to new use, and we are adopting this factor by combining rudimentary artificial intelligence techniques, XML formatting, and electronic delivery of business documents. Another reason for their success was that they fulfilled a need that had not yet been met. EDI and ebXML have attempted to meet the need for paperless document processing, but have failed to succeed due to the enormous costs and limitations, keeping them as niche implementations rather than universally adopted ones. By making our protocol scalable by using a very small amount of mandatory definitions, we can eliminate the cost difficulty that hampers the alternatives.

Another success factor was that the formats were free to use… companies could implement the standards in any manner they desired, and basic implementations were often provided free. This enabled more companies to take a chance and experiment with the functionality, as they were able to send data in that format, and if the recipient was not currently prepared to read that format, they could be directed to a free reader. By following this trend and providing the protocol to everyone for free, we can mimic this effect, especially if the makers of “off the shelf” accounting systems choose to integrate the protocol with their products, providing a huge initial user base, which was the fourth success factor… third party acceptance. The final success factor was the ability to expand the format to deal with new technologies or features, and it is this aspect where the protocol shines. Any new features can be implemented or ignored by users in a common manner, which means adding functionality is as simple as announcing it to others who want to use the new features so they can add the functionality into their systems.

Even if a small percentage of companies take advantage of the protocol to it’s full potential, this can still result in enormous benefits throughout the entire economy. Productivity can be significantly increased, and since the system “learns” over time and common conventions will probably evolve to reduce the learning necessary by the systems, this should lead to continuous productivity improvements, albeit on a smaller scale than the initial jump.
In addition to the productivity improvements, there will be significant paper usage reductions as companies gradually become comfortable with eliminating paper backups which can lead to extremely significant savings, both economically and environmentally. There are literally billions of invoices generated worldwide every year, and many of those also use envelopes for delivery. If even half a billion invoices were eliminated each year, that would save 17,333 tons of wood, 191.8 billion BTU’s of energy, 28.4 million pounds of greenhouse gases, 95.4 million gallons of wastewater, and 11.4 million pounds of solid waste. (Environmental impact estimates were made using the Environmental Defense Paper Calculator. For more information visit http://www.papercalculator.org.) At an average cost per printed invoice of 35 cents, that would be 175 million dollars in savings, and these are conservative numbers, the true potential savings could be magnitudes higher.

Just as email has now replaced a significant portion of business correspondence and web publication has in many places replaced catalogs and brochures, automated invoicing has the potential to be another major step on the road to the truly paperless office.
9.0 References


10.0 Appendices

Appendix A: Samples Of Industry Specific XML Variant Invoice Files

Chem eStandards Invoice

```xml
<?xml version="1.0" encoding="UTF-8" ?>
-
-<Invoice>
-  -<Header>
-   -<ThisDocumentIdentifier>
-     -<DocumentIdentifier>abc12345</DocumentIdentifier>
-   </ThisDocumentIdentifier>
-  -<ThisDocumentDateTime>
-     -<DateTime DateTimeQualifier="On">2001-12-17T09:30:47-05:00</DateTime>
-   </ThisDocumentDateTime>
-  -<RequestingDocumentIdentifier>
-     -<DocumentIdentifier>45600345</DocumentIdentifier>
-   </RequestingDocumentIdentifier>
-  -<RequestingDocumentDateTime>
-     -<DateTime DateTimeQualifier="On">2001-12-17T09:30:47-05:00</DateTime>
-   </RequestingDocumentDateTime>
-  -<From>
-     -<PartnerInformation>
-       -<PartnerName>Rexanol plc</PartnerName>
-       -<PartnerIdentifier Agency="D-U-N-S">293032454</PartnerIdentifier>
-     </PartnerInformation>
-     -<ContactInformation>
-       -<ContactName>Narinder Singh</ContactName>
-       -<ContactNumber>+44 23 8062 9429</ContactNumber>
-     </ContactInformation>
-     -<AddressInformation>
-       -<AddressLine>Chesil House, Shakespeare Road</AddressLine>
-       -<CityName>Eastleigh</CityName>
-       -<StateOrProvince>Hampshire</StateOrProvince>
-       -<PostalCode>SO50 4SY</PostalCode>
-       -<PostalCountry>UK</PostalCountry>
-     </AddressInformation>
-     -<URL>www.rexprocess.com</URL>
-   </From>
-  -<To>
-     -<PartnerInformation>
-       -<PartnerName>Stretch-SAP GmbH</PartnerName>
-       -<PartnerIdentifier Agency="D-U-N-S">293032457</PartnerIdentifier>
-     </PartnerInformation>
-   </To>
- </Invoice>
```
Jane Jones

AP Manager

+49 62 27 83 16 17

Neurottstrasse 17
Walldorf
69190
Germany

www.stretchsap.de

Debit

2001-12-17T09:30:47-05:00

5200.00

5200.00
<CurrencyCode Domain="ISO-4217">EUR</CurrencyCode>

</MonetaryAmount>

</InvoiceTotal>

</InvoiceTotals>

</PaymentTerms>

</PaymentTermsOfSale>

<TermsOfSaleDescription>Payment 30 days after invoice</TermsOfSaleDescription>

<NetDaysDue>30</NetDaysDue>

</PaymentTermsOfSale>

</PaymentTerms>

</InvoiceProperties>

</InvoicePartners>

</Buyer>

</PartnerInformation>

<PartnerName>Stretch-SAP GmbH</PartnerName>

<PartnerIdentifier Agency="D-U-N-S">293032457</PartnerIdentifier>

</ContactInformation>

<ContactName>Jessica Brown</ContactName>

<ContactNumber>+49 62 27 83 16 02</ContactNumber>

</ContactInformation>

</AddressInformation>

<AddressLine>Neurottstrasse 17</AddressLine>

<CityName>Walldorf</CityName>

<PostalCode>69190</PostalCode>

<PostalCountry>Germany</PostalCountry>

</AddressInformation>

</PartnerInformation>

</Seller>

</PartnerInformation>

<PartnerName>Rexanol plc</PartnerName>

<PartnerIdentifier Agency="D-U-N-S">293032454</PartnerIdentifier>

</ContactInformation>

<ContactName>Narinder Singh</ContactName>

<ContactNumber>+44 23 8062 9429</ContactNumber>

</ContactInformation>

</AddressInformation>

<AddressLine>Chesil House, Shakespeare Road</AddressLine>

<CityName>Eastleigh</CityName>

<StateOrProvince>Hampshire</StateOrProvince>

<PostalCode>SO50 4SY</PostalCode>

<PostalCountry>UK</PostalCountry>

</AddressInformation>

</PartnerInformation>

</Seller>

</InvoicePartners>

</InvoiceDetails>

</InvoiceLineItem>
UBL Invoice

<?xml version="1.0" encoding="UTF-8" ?>

<Invoice xmlns:qdt="urn:oasis:names:specification:ubl:schema:xsd:QualifiedDataTypes-2"
         xmlns:udt="urn:un:unece:uncefact:data:draft:UnqualifiedDataTypesSchemaModule:2"

   <LineNumber>1</LineNumber>

   <ProductIdentification>
      <ProductIdentifier>CH3OH</ProductIdentifier>
      <ProductName>Rexanol 1</ProductName>
   </ProductIdentification>

   <ProductQuantity>
      <Measurement>
         <MeasurementValue>26000</MeasurementValue>
         <UnitOfMeasureCode Domain="UN-Rec-20">L</UnitOfMeasureCode>
      </Measurement>
   </ProductQuantity>

   <Pricing PriceType="GrossPrice">
      <PricingLumpSum>
         <MonetaryAmount>
            <MonetaryValue>5200.00</MonetaryValue>
            <CurrencyCode Domain="ISO-4217">EUR</CurrencyCode>
         </MonetaryAmount>
      </PricingLumpSum>
   </Pricing>

   <ReferenceInformation ReferenceType="PurchaseOrderNumber">
      <DocumentReference>
         <DocumentIdentifier>987544</DocumentIdentifier>
         <DateTime DateTimeQualifier="On">2001-12-17T09:30:47-05:00</DateTime>
      </DocumentReference>
   </ReferenceInformation>

   <LineItemTotal>
      <MonetaryAmount>
         <MonetaryValue>5200.00</MonetaryValue>
         <CurrencyCode Domain="ISO-4217">EUR</CurrencyCode>
      </MonetaryAmount>
   </LineItemTotal>

</Invoice>
<cbc:UBLVersionID>2.0</cbc:UBLVersionID>
<cbc:ID>A00095678</cbc:ID>
<cbc:CopyIndicator>false</cbc:CopyIndicator>
<cbc:UUID>849FBBCE-E081-40B4-906C-94C5FF9D1AC3</cbc:UUID>
<cbc:IssueDate>2005-06-21</cbc:IssueDate>
<cbc:InvoiceTypeCode>SalesInvoice</cbc:InvoiceTypeCode>
<cbc:Note>sample</cbc:Note>
<cbc:TaxPointDate>2005-06-21</cbc:TaxPointDate>
= <cac:OrderReference>
  <cbc:ID>AEG012345</cbc:ID>
  <cbc:SalesOrderID>CON0095678</cbc:SalesOrderID>
  <cbc:UUID>6E09886B-DC6E-439F-82D1-7CCAC7F4E3B1</cbc:UUID>
  <cbc:IssueDate>2005-06-20</cbc:IssueDate>
= <cac:AccountingSupplierParty>
  <cbc:CustomerAssignedAccountID>CO001</cbc:CustomerAssignedAccountID>
  = <cac:Party>
    = <cac:PartyName>
      <cbc:Name>Consortial</cbc:Name>
    </cac:PartyName>
    <cbc:PostalAddress>
      <cbc:StreetName>Busy Street</cbc:StreetName>
      <cbc:BuildingName>Thereabouts</cbc:BuildingName>
      <cbc:BuildingNumber>56A</cbc:BuildingNumber>
      <cbc:CityName>Farthing</cbc:CityName>
      <cbc:PostalZone>AA99 1BB</cbc:PostalZone>
      <cbc:CountrySubentity>Heremouthshire</cbc:CountrySubentity>
      <cac:AddressLine>
        <cbc:Line>The Roundabout</cbc:Line>
      </cac:AddressLine>
    </cbc:PostalAddress>
    = <cac:Country>
    </cac:Country>
    = <cac:PartyTaxScheme>
      <cbc:RegistrationName>Farthing Purchasing Consortia</cbc:RegistrationName>
      <cbc:CompanyID>175 269 2355</cbc:CompanyID>
      <cbc:ExemptionReason>N/A</cbc:ExemptionReason>
    </cac:PartyTaxScheme>
    = <cac:TaxScheme>
      <cbc:ID>VAT</cbc:ID>
      <cbc:TaxTypeCode>VAT</cbc:TaxTypeCode>
    </cac:TaxScheme>
  </cac:Party>
  = <cac:Contact>
    <cbc:Name>Mrs Bouquet</cbc:Name>
    <cbc:Telephone>0158 1233714</cbc:Telephone>
    <cbc:Telefax>0158 1233856</cbc:Telefax>
    <cbc:ElectronicMail>bouquet@fpconsortial.co.uk</cbc:ElectronicMail>
  </cac:Contact>
</cac:AccountingSupplierParty>
= <cac:AccountingCustomerParty>
  <cbc:CustomerAssignedAccountID>XFB01</cbc:CustomerAssignedAccountID>
  <cbc:SupplierAssignedAccountID>GT00978567</cbc:SupplierAssignedAccountID>
  = <cac:Party>
    = <cac:PartyName>
      <cbc:Name>IYT Corporation</cbc:Name>
    </cac:PartyName>
<cac:PartyName/>
  <cac:PostalAddress>
    <cbc:StreetName>Avon Way</cbc:StreetName>
    <cbc:BuildingName>Thereabouts</cbc:BuildingName>
    <cbc:BuildingNumber>56A</cbc:BuildingNumber>
    <cbc:CityName>Bridgtow</cbc:CityName>
    <cbc:PostalZone>ZZ99 1ZZ</cbc:PostalZone>
    <cbc:CountrySubentity>Avon</cbc:CountrySubentity>
  </cac:PostalAddress>
  <cac:AddressLine>
    <cbc:Line>3rd Floor, Room 5</cbc:Line>
  </cac:AddressLine>
  <cac:Country>
  </cac:Country>
  <cac:PartyTaxScheme>
    <cbc:RegistrationName>Bridgtow District Council</cbc:RegistrationName>
    <cbc:CompanyID>12356478</cbc:CompanyID>
    <cbc:ExemptionReason>Local Authority</cbc:ExemptionReason>
    <cac:TaxScheme>
      <cbc:ID>UK VAT</cbc:ID>
      <cbc:TaxTypeCode>VAT</cbc:TaxTypeCode>
    </cac:TaxScheme>
  </cac:PartyTaxScheme>
  <cac:Contact>
    <cbc:Name>Mr Fred Churchill</cbc:Name>
    <cbc:Telephone>0127 2653214</cbc:Telephone>
    <cbc:Telefax>0127 2653215</cbc:Telefax>
    <cbc:ElectronicMail>fred@iytcorporation.gov.uk</cbc:ElectronicMail>
  </cac:Contact>
</cac:Party>

<cac:AccountingCustomerParty/>

<cac:Delivery>
  <cbc:ActualDeliveryDate>2005-06-20</cbc:ActualDeliveryDate>
  <cbc:ActualDeliveryTime>11:30:00.0Z</cbc:ActualDeliveryTime>
</cac:Delivery>

<cac:DeliveryAddress>
  <cbc:StreetName>Avon Way</cbc:StreetName>
  <cbc:BuildingName>Thereabouts</cbc:BuildingName>
  <cbc:BuildingNumber>56A</cbc:BuildingNumber>
  <cbc:CityName>Bridgtow</cbc:CityName>
  <cbc:PostalZone>ZZ99 1ZZ</cbc:PostalZone>
  <cbc:CountrySubentity>Avon</cbc:CountrySubentity>
</cac:DeliveryAddress>

<cac:AddressLine>
  <cbc:Line>3rd Floor, Room 5</cbc:Line>
</cac:AddressLine>

<cac:Country>
</cac:Country>

<cac:DeliveryAddress/>

<cac:PaymentMeans>
  <cbc:PaymentMeansCode>20</cbc:PaymentMeansCode>
  <cbc:PaymentDueDate>2005-07-21</cbc:PaymentDueDate>
</cac:PaymentMeans>

<cac:PayeeFinancialAccount>
  <cbc:ID>12345678</cbc:ID>
  <cbc:Name>Farthing Purchasing Consortia</cbc:Name>
  <cbc:AccountTypeCode>Current</cbc:AccountTypeCode>
  <cbc:CurrencyCode>GBP</cbc:CurrencyCode>
</cac:PayeeFinancialAccount>

<cac:FinancialInstitutionBranch>
  <cbc:ID>10-26-58</cbc:ID>
  <cbc:Name>Open Bank Ltd, Bridgstow Branch</cbc:Name>
</cac:FinancialInstitutionBranch>
= <cac:FinancialInstitution>
  <cbc:ID>10-26-58</cbc:ID>
  <cbc:Name>Open Bank Ltd</cbc:Name>
= <cac:Address>
  <cbc:StreetName>City Road</cbc:StreetName>
  <cbc:BuildingName>Banking House</cbc:BuildingName>
  <cbc:BuildingNumber>12</cbc:BuildingNumber>
  <cbc:CityName>London</cbc:CityName>
  <cbc:PostalZone>AQ1 6TH</cbc:PostalZone>
  <cbc:CountrySubentity>London</cbc:CountrySubentity>
= <cac:AddressLine>
  <cbc:Line>5th Floor</cbc:Line>
</cac:AddressLine>
= <cac:Country>
</cac:Country>
= <cac:FinancialInstitutionBranch>
= <cac:Country>
</cac:Country>
= <cac:Address>
  <cbc:StreetName>Busy Street</cbc:StreetName>
  <cbc:BuildingName>The Mall</cbc:BuildingName>
  <cbc:BuildingNumber>152</cbc:BuildingNumber>
  <cbc:CityName>Farthing</cbc:CityName>
  <cbc:PostalZone>AA99 1BB</cbc:PostalZone>
  <cbc:CountrySubentity>Heremouthshire</cbc:CountrySubentity>
= <cac:AddressLine>
  <cbc:Line>West Wing</cbc:Line>
</cac:AddressLine>
= <cac:Country>
</cac:Country>
= <cac:Address>
  <cbc:StreetName>City Road</cbc:StreetName>
  <cbc:BuildingName>Banking House</cbc:BuildingName>
  <cbc:BuildingNumber>12</cbc:BuildingNumber>
  <cbc:CityName>London</cbc:CityName>
  <cbc:PostalZone>AQ1 6TH</cbc:PostalZone>
  <cbc:CountrySubentity>London</cbc:CountrySubentity>
= <cac:AddressLine>
  <cbc:Line>5th Floor</cbc:Line>
</cac:AddressLine>
= <cac:Country>
</cac:Country>
= <cac:Address>
  <cbc:StreetName>Busy Street</cbc:StreetName>
  <cbc:BuildingName>The Mall</cbc:BuildingName>
  <cbc:BuildingNumber>152</cbc:BuildingNumber>
  <cbc:CityName>Farthing</cbc:CityName>
  <cbc:PostalZone>AA99 1BB</cbc:PostalZone>
  <cbc:CountrySubentity>Heremouthshire</cbc:CountrySubentity>
= <cac:AddressLine>
  <cbc:Line>West Wing</cbc:Line>
</cac:AddressLine>
= <cac:Country>
</cac:Country>
= <cac:Address>
  <cbc:StreetName>City Road</cbc:StreetName>
  <cbc:BuildingName>Banking House</cbc:BuildingName>
  <cbc:BuildingNumber>12</cbc:BuildingNumber>
  <cbc:CityName>London</cbc:CityName>
  <cbc:PostalZone>AQ1 6TH</cbc:PostalZone>
  <cbc:CountrySubentity>London</cbc:CountrySubentity>
= <cac:AddressLine>
  <cbc:Line>5th Floor</cbc:Line>
</cac:AddressLine>
= <cac:Country>
</cac:Country>
= <cac:Address>
  <cbc:StreetName>Busy Street</cbc:StreetName>
  <cbc:BuildingName>The Mall</cbc:BuildingName>
  <cbc:BuildingNumber>152</cbc:BuildingNumber>
  <cbc:CityName>Farthing</cbc:CityName>
  <cbc:PostalZone>AA99 1BB</cbc:PostalZone>
  <cbc:CountrySubentity>Heremouthshire</cbc:CountrySubentity>
= <cac:AddressLine>
  <cbc:Line>West Wing</cbc:Line>
</cac:AddressLine>
= <cac:Country>
</cac:Country>
= <cac:Address>
  <cbc:StreetName>City Road</cbc:StreetName>
  <cbc:BuildingName>Banking House</cbc:BuildingName>
  <cbc:BuildingNumber>12</cbc:BuildingNumber>
  <cbc:CityName>London</cbc:CityName>
  <cbc:PostalZone>AQ1 6TH</cbc:PostalZone>
  <cbc:CountrySubentity>London</cbc:CountrySubentity>
= <cac:AddressLine>
  <cbc:Line>5th Floor</cbc:Line>
</cac:AddressLine>
= <cac:Country>
</cac:Country>
= <cac:Address>
  <cbc:StreetName>Busy Street</cbc:StreetName>
  <cbc:BuildingName>The Mall</cbc:BuildingName>
  <cbc:BuildingNumber>152</cbc:BuildingNumber>
  <cbc:CityName>Farthing</cbc:CityName>
  <cbc:PostalZone>AA99 1BB</cbc:PostalZone>
  <cbc:CountrySubentity>Heremouthshire</cbc:CountrySubentity>
= <cac:AddressLine>
  <cbc:Line>West Wing</cbc:Line>
</cac:AddressLine>
= <cac:Country>
</cac:Country>
= <cac:Address>
  <cbc:StreetName>City Road</cbc:StreetName>
  <cbc:BuildingName>Banking House</cbc:BuildingName>
  <cbc:BuildingNumber>12</cbc:BuildingNumber>
  <cbc:CityName>London</cbc:CityName>
  <cbc:PostalZone>AQ1 6TH</cbc:PostalZone>
  <cbc:CountrySubentity>London</cbc:CountrySubentity>
= <cac:AddressLine>
  <cbc:Line>5th Floor</cbc:Line>
</cac:AddressLine>
= <cac:Country>
</cac:Country>
= <cac:Address>
  <cbc:StreetName>Busy Street</cbc:StreetName>
  <cbc:BuildingName>The Mall</cbc:BuildingName>
  <cbc:BuildingNumber>152</cbc:BuildingNumber>
  <cbc:CityName>Farthing</cbc:CityName>
  <cbc:PostalZone>AA99 1BB</cbc:PostalZone>
  <cbc:CountrySubentity>Heremouthshire</cbc:CountrySubentity>
= <cac:AddressLine>
  <cbc:Line>West Wing</cbc:Line>
</cac:AddressLine>
= <cac:Country>
</cac:Country>
= <cac:PaymentTerms>
  <cbc:Note>Payable within 1 calendar month from the invoice date</cbc:Note>
</cac:PaymentTerms>
= <cac:Allowance Charge>
  <cbc:ChargeIndicator>false</cbc:ChargeIndicator>
  <cbc:MultiplierFactorNumeric>0.10</cbc:MultiplierFactorNumeric>
  <cbc:Amount currencyID="GBP">10.00</cbc:Amount>
</cac:Allowance Charge>
= <cac:TaxTotal>
  <cbc:TaxAmount currencyID="GBP">17.50</cbc:TaxAmount>
</cac:TaxTotal>
= <cac:TaxSubtotal>
  <cbc:Taxable Amount currencyID="GBP">100.00</cbc:Taxable Amount>
  <cbc:TaxAmount currencyID="GBP">17.50</cbc:TaxAmount>
</cac:TaxSubtotal>
= <cac:TaxCategory>
  <cbc:ID>A</cbc:ID>
</cac:TaxCategory>
= <cac:TaxScheme>
  <cbc:IdentificationCode>UK VAT</cbc:IdentificationCode>
  <cbc:TaxTypeCode>VAT</cbc:TaxTypeCode>
</cac:TaxScheme>
= <cac:TaxCategory>
</cac:TaxSubtotal>
<cac:TaxTotal>
    <cbc:LineExtensionAmount currencyID="GBP">100.00</cbc:LineExtensionAmount>
    <cbc:TaxExclusiveAmount currencyID="GBP">90.00</cbc:TaxExclusiveAmount>
    <cbc:AllowanceTotalAmount currencyID="GBP">10.00</cbc:AllowanceTotalAmount>
    <cbc:PayableAmount currencyID="GBP">107.50</cbc:PayableAmount>
</cac:TaxTotal>

<cac:InvoiceLine>
    <cbc:ID>A</cbc:ID>
    <cbc:InvoicedQuantity unitCode="KG">100</cbc:InvoicedQuantity>
    <cbc:LineExtensionAmount currencyID="GBP">100.00</cbc:LineExtensionAmount>
</cac:InvoiceLine>

<cac:OrderReference>
    <cbc:ID>AEG012345</cbc:ID>
    <cbc:SalesOrderID>CON0095678</cbc:SalesOrderID>
    <cbc:UUID>6E09886B-DC6E-439F-82D1-7CCAC7F4E3B1</cbc:UUID>
    <cbc:IssueDate>2005-06-20</cbc:IssueDate>
</cac:OrderReference>

<cac:TaxTotal>
    <cbc:TaxAmount currencyID="GBP">17.50</cbc:TaxAmount>
</cac:TaxTotal>

<cac:Item>
    <cbc:Description>Acme beeswax</cbc:Description>
    <cbc:Name>beeswax</cbc:Name>
    <cbc:BuyersItemIdentification>
        <cbc:ID>6578489</cbc:ID>
    </cbc:BuyersItemIdentification>
    <cbc:SellersItemIdentification>
        <cbc:ID>17589683</cbc:ID>
    </cbc:SellersItemIdentification>
    <cbc:LotIdentification>
        <cbc:LotNumberID>546378239</cbc:LotNumberID>
    </cbc:LotIdentification>
    <cbc:Price>
        <cbc:PriceAmount currencyID="GBP">1.00</cbc:PriceAmount>
    </cbc:Price>
</cac:Item>
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="invoice.1.0.xsl"?>
<!DOCTYPE Invoice SYSTEM "invoice.1.0.dtd">
<Invoice sectorUsageVersion="1">
  <InvoiceHeader>
    <InvoicedType stdValue="380"/>
    <Invoicestatus stdValue="53"/>
    <TaxTreatment stdValue="NIL"/>
    <InvoiceTreatment stdValue="P"/>
    <InvoiceNumber>ISSUER041</InvoiceNumber>
    <InvoiceDate>2001-01-01</InvoiceDate>
    <Currency stdValue="USD"/>
    <Party stdValue="BY">
      <PartyID>IOV1954</PartyID>
      <Name>
        <Name1>Office Depot</Name1>
      </Name>
      <Street>
        <Street1>Any Street</Street1>
      </Street>
      <PostalInfo>
        <City>Any City</City>
        <CountrySubEntity>AB</CountrySubEntity>
        <PostalCode>XXXXX</PostalCode>
        <Country>AB</Country>
      </PostalInfo>
      <Contact>
        <Name1>Administrator</Name1>
        <TelNum>XXXXXXXXXX</TelNum>
      </Contact>
    </Party>
    <Party stdValue="SU">
      <Name>
        <Name1>Any Company</Name1>
      </Name>
      <Street>
        <Street1>Any Street</Street1>
      </Street>
      <PostalInfo>
        <City>Any City</City>
        <CountrySubEntity>AB</CountrySubEntity>
        <PostalCode>XXXXX</PostalCode>
        <Country>AB</Country>
      </PostalInfo>
      <Ref>PA9847562341</Ref>
      <Ref stdValue="XA">US123456789</Ref>
    </Party>
    <Party stdValue="PI">
      <Name>
        <Name1>Office Depot 001CA</Name1>
      </Name>
      <PostalInfo>
        <City>Any City</City>
        <CountrySubEntity>AB</CountrySubEntity>
        <Country>AB</Country>
      </PostalInfo>
      <Ref stdValue="ABC">1234</Ref>
      <Ref stdValue="ADQ">5111</Ref>
    </Party>
    <Ref stdValue="ADQ">MM</Ref>
    <Ref stdValue="AWE">GR123</Ref>
  </InvoiceHeader>
</InvoiceDetails>