



MIGRATING EDI SYSTEMS TO WEBSERVICES

THE INDIRECT APPROACH

ABSTRACT

In this paper, I discuss the inherent limitations of the electronic data interchange (EDI) commerce systems and the inherent advantages of employing web services. I also discuss the application of the theory of integration using the indirect approach to migrate existing EDI systems to web services.

MIGRATING EDI SYSTEMS TO WEBSERVICES

THE INDIRECT APPROACH

INTRODUCTION

Electronic commerce is not new. It first appeared during the Berlin Airlift of 1949. When you consider that the first true digital computer was built in 1945, it is easy to say that e-commerce is as old as the computer itself. The long evolution of e-commerce is easy to trace. The US Department of Defense still uses exchange technology called MILSPEC for electronic data integration (EDI). The maximum record width for MILSPEC today just happens to be the maximum record width for a punch card.

You should be aware that EDI can have two meanings, one more specific than the other. In general, EDI applies to any exchange of electronic information between two organizations. EDI in the general sense encompasses a wide range of standards and technology. A more specific meaning of EDI applies only to the exchange of information between two organizations using the ANSI X12 standard or the UN EDIFACT standard. When we refer to MILSPEC as EDI, we are using the more general meaning.

STANDARD LEGACY MODEL

EDI in the general sense follows a predictable pattern of integration as shown in Figure 1. It begins with two companies that want to share data directly between their internal applications. The companies usually install a middleware server that translates that data and moves the data across a proprietary network. This model applies not only to EDI but also financial market data (Reuters, Bloomberg), banking (FIX, SWIFT), automated clearinghouses, etc.

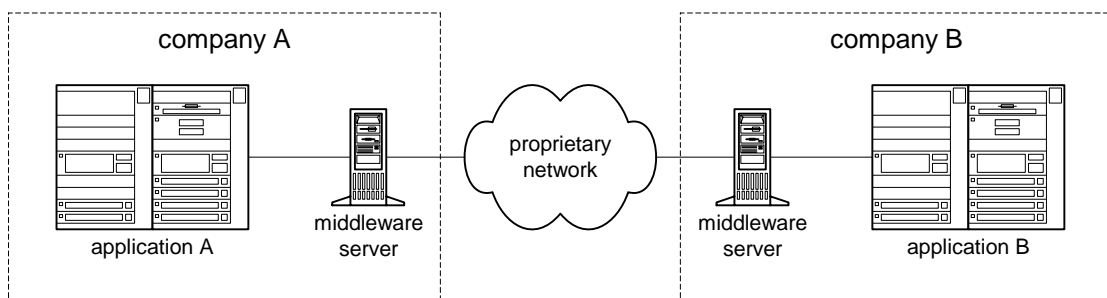


Figure 1 Standard model of legacy EDI

OUTBOUND MESSAGES

EDI is rather simple. We will walk through step by step how an application sends an EDI message to another company. These steps are shown as circled numbers in Figure 2.

STEP 1. It begins with a company's internal application such as enterprise resource planning (ERP), order fulfillment, inventory or even accounts payable. The application creates an export file in plain text that contains the information needed for exchange. The export file might be simple query results from a relational database. The resulting data is in an arbitrary text file format determined by the limits or capabilities of the application itself. For instance, SAP produces a file format called IDOC for EDI purposes. Peachtree Accounting has its own export file format. The export may occur through a scheduled task (a *cron* job in UNIX parlance) or it may be event based. For instance, the application user may give the instruction to export the information of a purchase order. This text file is placed in a designated directory or file that the EDI translator will search.

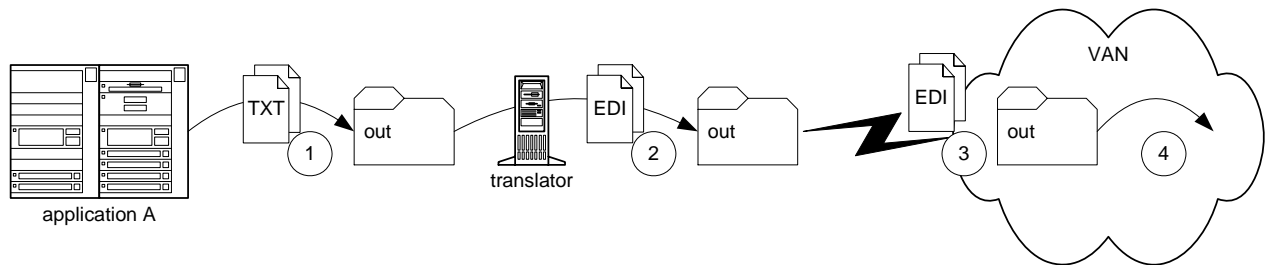


Figure 2 EDI outbound

STEP 2. The EDI translator pulls the text files out of the designated directories and translates them into EDI file format such as ANSI X12 or UN EDIFACT. The translator may constantly poll the directory for the arrival of new files or it may simply search for new files on a scheduled basis. The translator uses a set of instructions called a *map* to translate the arbitrary text format into the standard EDI format. The translator needs some sort of context to know which map to apply. The text file may have an encoded file name or it may have header information that states which map to apply. More than likely, however, the designated directory is directly associated with a particular map. At the end of step 2, the translator places all translated files in a specific directory designated for outbound messages.

STEP 3. The EDI translator usually comes with communication software. This software uses scripts to connect to the VANs. Different VANs use different means of communication. Some use X-modem or Y-modem, others may use FTP. The communication software dials up the VAN, makes the connection, and posts all the EDI files to the VAN. For those of you familiar with the Internet, this process is directly equivalent to making an FTP connection and executing a PUT command into the home directory.

STEP 4. The VAN takes the EDI files it has received and reads the header information. The header contains the routing information, that is, the sender and recipient. The VAN places those files into the in box or home directory of the designated recipient.

INBOUND MESSAGES

Receiving documents from the VAN is very much similar to sending documents. We will walk through step by step how an application receives an EDI message to another company. These steps are shown as circled numbers in Figure 3.

STEP 1. As we showed earlier, it is VAN's responsibility to route documents into the proper corporate inbox or home directory.

STEP 2. A company's EDI communication software periodically dials up and connects to the VAN. The software moves all files located in the inbox or home directory on the VAN to a local directory.

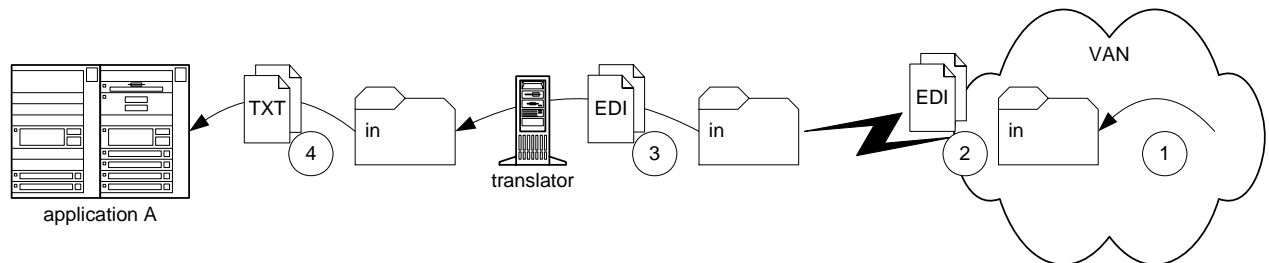


Figure 3 EDI inbound

STEP 3. The EDI translator now maps or translates the EDI files it finds in the local inbox into the proper text format. If the target application is SAP, for instance, then the proper text format is IDOC. Unlike outbound documents, the translator can determine which map or translation to apply by the EDI document type because EDI documents are standardized.

STEP 4. The internal application imports the text files. The import can be scheduled by a script call to an application program interface (API). Import can also be scheduled or executed manually from the application itself.

MAPS

The key technology to EDI is not the communication. As you can see from the previous walkthrough, the communication process of EDI is really quite simple if not primitive. The key to EDI is the translator, and the key to the translator is the translation instruction sets, or *maps* as they are called. We stated earlier that the translator needs a context in order to determine which map to apply. The reason is that there are many maps for an EDI implementation. Usually there are two maps for every document type per trading partner. For instance, if a company has 10 EDI trading partners and they exchange purchase orders and invoices with every trading partner, then the company needs two maps times two document types times 10 partners, or 40 maps.

2 μαπς/δοχτυπε Ξ 2 δοχτυπες/τραδινη παρτνερ Ξ 10 τραδινη παρτνερσ = 40 μαπς

It may take a well-trained EDI technician four hours to create a map using a sophisticated EDI mapping tool. At \$100 per hour, the cost of creating the maps would be \$1,600 per trading partner.

COSTS OF LEGACY E-COMMERCE SYSTEMS

BASIC MODEL

Most every system of inter-company electronic data sharing follows a standard model. They have a middleware server at both ends of the connection and they connect across a proprietary network. As such, they have a standard costs associated with their implementation as shown in Figure 4.

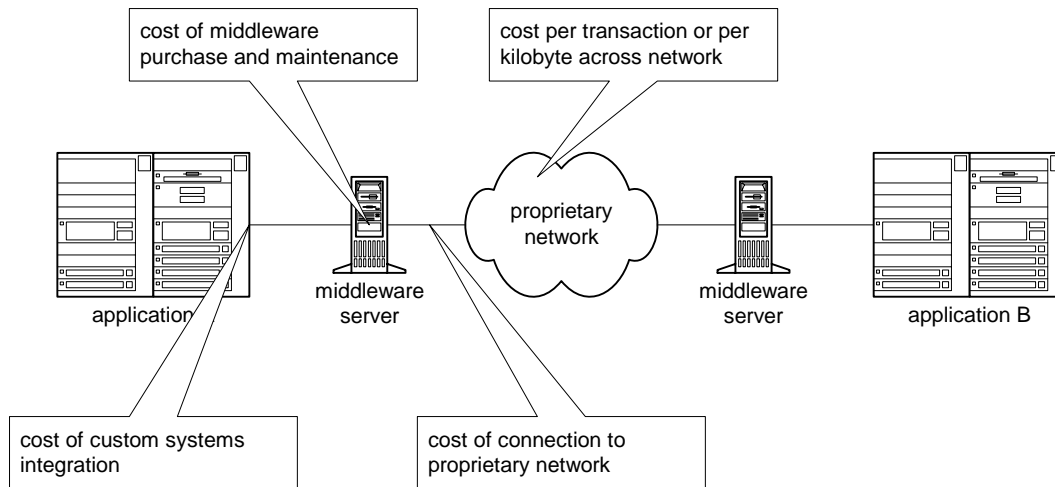


Figure 4 Standard costs of legacy e-commerce

NETWORK COSTS. The most conspicuous cost in a legacy e-commerce system is the network costs. Companies are usually charged for every kilobyte sent across these networks. These fees can become quite large. VAN charges can cost companies tens of thousands of dollars each month.

CONNECTION COSTS. The company must also pay for dedicated communication connections to the proprietary network. Usually this connection is modem equipment, dedicated phone lines and long distance toll charges. Sometimes, however, the connection entails a dedicated line such as frame relay or T-1, which can cost serious money.

MIDDLEWARE COSTS. A basic EDI translator costs at least \$10,000. Of course there is the annual maintenance licensing fees, but the price of the software is just the beginning. The company must pay to maintain someone on staff that is trained to operate the software. EDI experts don't grow on trees, you know.

INTEGRATION COSTS. The most significant cost to EDI is probably the most readily overlooked. Companies can and have spent millions of dollars on integrating EDI into their existing internal applications.

We keep harping on EDI as an example, but the cost model is pretty much the same for any legacy inter-company data exchange, to include market data feeds, electronic news feeds, and inter-bank communications like automated clearinghouses.

SELF-DEFEATING COST MODEL

This pricing model of charging per kilobyte causes the whole EDI system to be self-defeating. First, it drives companies to make their transaction documents as compact as possible. The

documents strip all metadata and use cryptic two or three-letter codes to represent values. These cryptic messages require special (read expensive) software on both ends. These cryptic messages are much more difficult to work with and thus harder to learn. Because the data format is hard to learn, few people do learn it. Basic microeconomics tells you that fewer EDI experts mean they are going to be relatively more expensive. Even if one has learned EDI, the cryptic messages make the data hard to work with, which means it takes more time to program integrations and even more time to debug integrations. Expensive programmers working longer hours drives implementation costs way up. Higher costs makes EDI unaffordable to many small and midsize companies.

Meanwhile, companies that have decided to implement EDI do so based on a logical return on investment (ROI) model. If it costs \$100 to process a manual purchase order and \$10 to process an electronic purchase order, then the EDI installation pays for itself. The math works out to be true if, and only if, all the trading partners use EDI. An ROI can even be achieved when less than all of the trading partners use EDI. The actual number of trading partners that must use EDI in order to achieve an ROI is called breakeven or, more appropriately, *critical mass*. However, most companies that have implemented EDI have less than 20% of their trading partners using EDI. In fact, 20% is considered to be good.

Meanwhile, the VAN has built a proprietary network that it must maintain. The VAN must recoup its costs per customer. With more customers, the fixed costs are spread over more customers and the price of service should go down or profitability should go up. However, the number of customers does not grow as quickly as one might originally anticipate. General Motors has 60,000 trading partners, but less than 20% connect to the VAN. Fewer customers means that the fixed costs per customer is higher. The price per kilobyte goes up, which drives companies to make their messages even smaller and more cryptic, which in turn raises the cost on implementation and pushes EDI out of reach of even more potential customers. It is a vicious circle, a self-defeating cost model.

COST ADVANTAGES OF THE INTERNET

Standard EDI such as X12 and EDIFACT has been around since the early 1980s and yet it has not changed the way we do business. The World Wide Web has been around since 1993, and by 1999 it had already significantly changed the way we do business. Over the past 20 years, about 300,000 companies have adopted EDI worldwide. Between 1992 and 1993, the number of Internet hosts doubled from 1 million to 2 million. By April 2000, there were 10 million registered domain names.

In chemistry terms, we would say that EDI is endothermic, it consumes energy, while the Web is exothermic, it gives off energy. We have seen that the EDI pricing model of charging per kilobyte is self-defeating. What is it about the World Wide Web that makes it self-propagating?

The basic components of the Web are free. The Apache web server commands more than half of the market, that is, more than half of all web servers connected to the Internet are Apache. Apache is free open source software. The browser is also free. Microsoft bundles Internet Explorer with most personal computers sold in the US. The Netscape browser became open source in 1998.

The connection to the Internet is relatively inexpensive. [more]

The Web is very easy to use. The Internet had been around long before the Web, but its command line interface was too difficult to learn or use for the average consumer. The graphical user interface (GUI) of the web browser made the Internet accessible to the consumer.

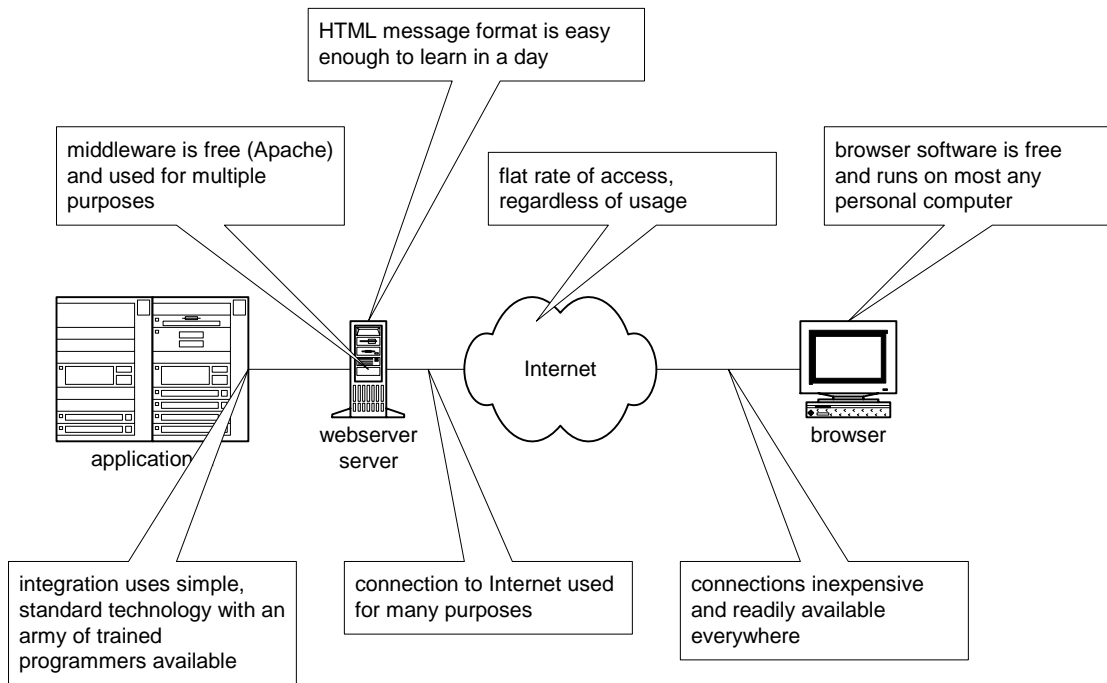


Figure 5 Cost factors in implementing the Web

The real energy source of the Web is the simplicity of hypertext markup language (HTML) and the common gateway interface (CGI). HTML is so simple that most people can learn the basics and publish their first web page in a single day. The genius of the Web, however, is CGI.

HTML began with university researchers publishing scientific papers over the Internet. The previous command-line or EMACS-like text format used for previous Internet applications was not suitable for communicating scientific information. Visuals such as diagrams and graphs are indispensable. But HTML had only been about the publishing of documents, it would never have changed our lives. It was the ability to handle interactive and dynamically generated information in a simple fashion that made HTML explode. The common gateway interface (CGI) is what made the interactive and dynamic generation possible.

The next stage of growth was the introduction of application servers. This stage was empowered by the extension of CGI with the servlet standard, only made possible by the coincidental development of Java. Now we are undergoing a third phase of web services. This phase is built upon expanding servlets with simple object access protocol (SOAP), made possible again by the coincidental development of XML.

IMPLEMENTING E-COMMERCE

Conducting e-commerce over the Internet can take advantage of some of the cost advantages of the Internet. The company can get rid of per-kilobyte charges of proprietary networks such as VANs. They can use their existing Internet connection and drop the expense of dedicated connections. More importantly, the advantages of the Internet can break the self-defeating cost structure of EDI. With the Internet, companies are less conscious of document sizes. They can begin sending less cryptic, more self-describing transaction documents. These message types are easier to understand, easier to learn and thus easier implement and debug. The ease of implementation drives the cost of

implementation down. Lower costs make the system more accessible to small and midsize companies. More companies mean the fixed costs are more distributed, which drives the cost of implementation even lower. Metcalfe effects kick in and the system transitions from endothermic to exothermic.

As straightforward as this transition might seem, companies and organizations can still screw it up. In fact, at the time of writing, most are screwing it up. The easiest way to screw it up is to make the new message structures for e-commerce complicated. Complexity drives cost. If you let a bunch of engineers sit in a room and define a new message structure for a purchase order, you are in for a disappointment. They will inevitably over-engineer the document. Engineers are trained to optimize a system. The key is to properly define the boundary of the system. The document structure itself is a very poor system boundary. The system must include the applications that will process the documents, the programmers that will make the integration, and the business people who generate and use the documents.

If engineers optimize the message structure, the system as a whole will be compromised and sub optimal, possibly even unworkable.

DIRECT APPROACH

The instinctive thing to do is to replicate the existing legacy e-commerce infrastructure using XML-based middleware. The objective is sound, replace proprietary message structures and networks, but the approach is flawed. The approach requires the implementer to incur the largest cost e-commerce, systems integration, up front without receiving any of the benefits of e-commerce until that cost is completely swallowed. The critics can easily claim that the vendor is fixing something that is not broken. The fate of the e-commerce initiative hangs by a thread, and we have witnessed during the recession of 2001 many of those threads snap.

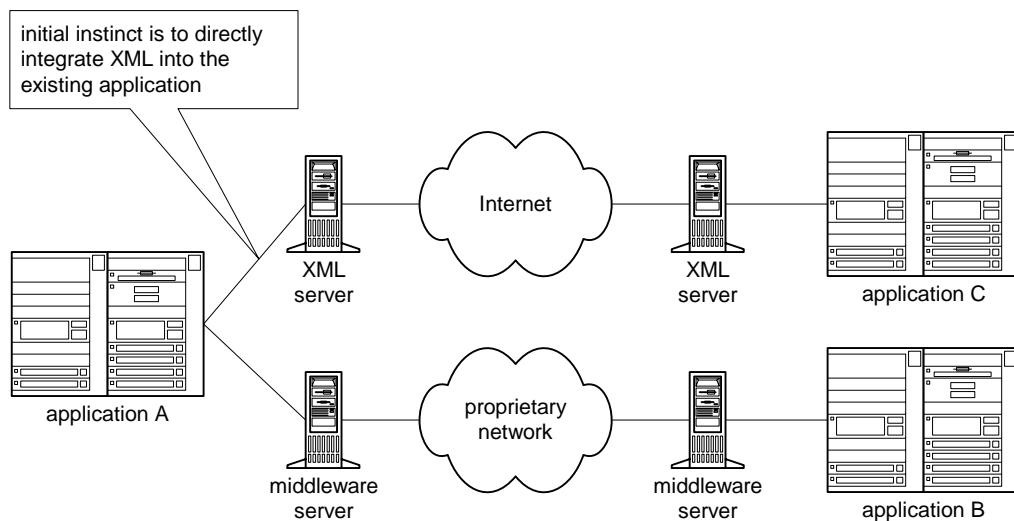


Figure 6 Direct approach

If few people know the proprietary middleware being replaced, even fewer are going to know the legacy system it is connected to. The direct approach requires the combination of people who know legacy systems and people who know the very latest Internet technology. These two skills are almost mutually exclusive. Hats off to anyone who has both skills because they are going to be worth their weight in gold. And gold is just what we want to spare in a systems integration effort.

INDIRECT APPROACH

An indirect approach has immediate incremental success and leverages existing investments in legacy e-commerce. It delivers lower risk and faster speed to market.

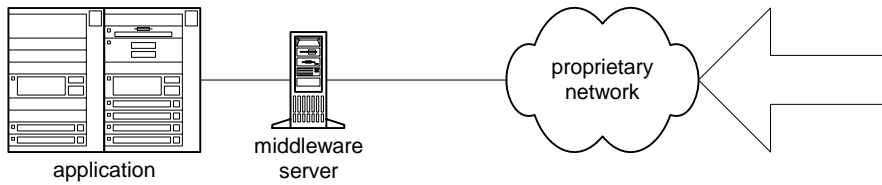


Figure 7 Legacy e-commerce

PHASE 1

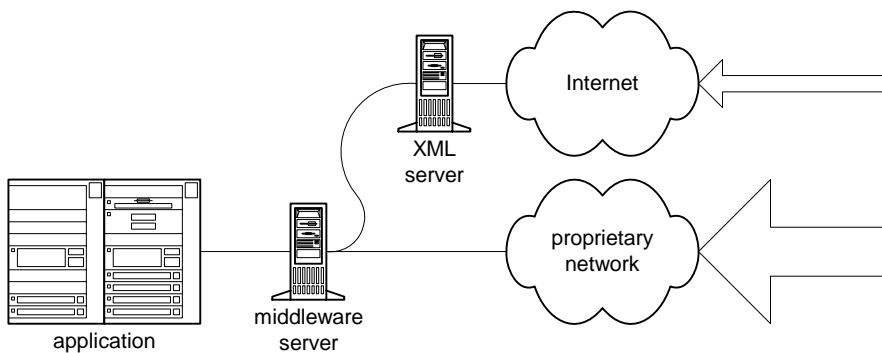


Figure 8 Phase 1, bolt on XML interface

The indirect approach begins by bolting XML middleware directly on to the existing legacy middleware server. The middleware is, after all, producing a published data format which is going to make translation or mapping process a lot simpler. Plus the XML vendor gets a lot more software reuse early on because the existing legacy middleware software is a lot smaller set of applications than all of the legacy mainframe systems they connect to. This “bolt-on” approach is also less intrusive to a company's IT department. Few CIOs want a bunch of Internet kids banging on their mainframe system. The existing middleware system acts as a buffer zone as the deployment gains its sea legs and the IT department begins to gain trust in the new XML middleware.

Thus, through the indirect bolt-on approach, we avoid the cost of mainframe integration and get to market immediately. Another advantage is that we maintain a single point of clearance and auditing because all messages are still passing through the legacy system.

PHASE 2

The companies migrating to XML want to get rid of their legacy middleware, not add a level of complexity on top of it. Eventually the XML middleware must integrate directly with the mainframe. The indirect approach allows companies to do so in a gradual manner. It provides an environment that enables testing and verification of implementation without an abrupt cut-over.

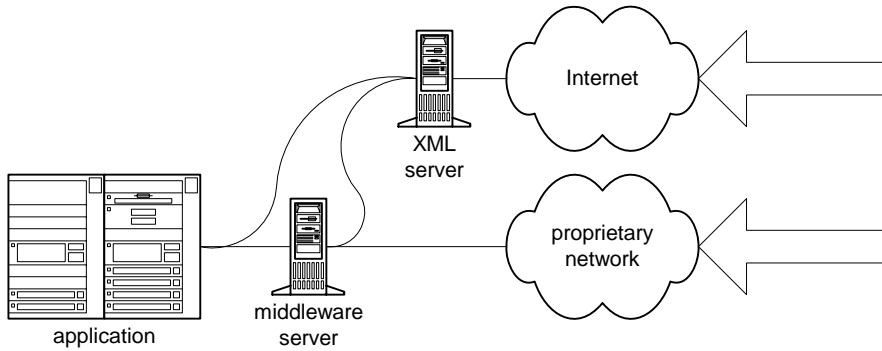


Figure 9 Phase 2, Build and test direct interface

Think of it this way: when they are replacing a highway bridge, they don't blow up the old bridge before the new bridge is built. The traffic must continue flowing. They build a new bridge in parallel with the old bridge, then shift traffic from the old to the new. Why should a company manage its e-commerce any differently?

PHASE 3

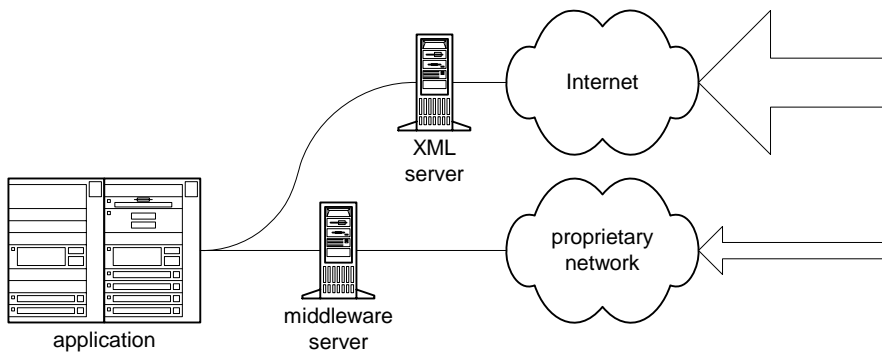


Figure 10 Phase 3, Migrate traffic from legacy to Internet

We must keep in mind that it is not just the company making the transition to XML but its trading partners as well. On the other end of the wire, the trading partners are undergoing a similar experience. Traffic must migrate from the legacy network to the to the Internet. In the mean time, by employing an indirect approach, the company is able to validate its new XML-integration and shut off the original bolt-on connection.

The key to switch over is not necessarily the stability of the maps. Rather, it is the role of prime logging and auditing. At switch over, the XML server must have the role as prime auditor.

PHASE 4

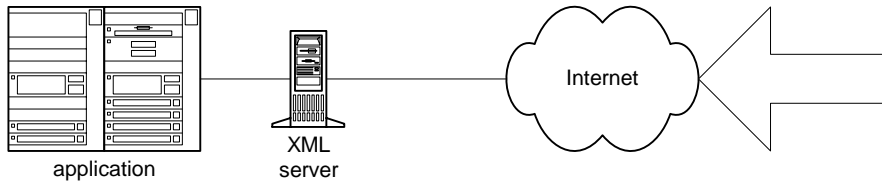


Figure 11 Phase 3, Turn off the legacy connection

Eventually the company is able to switch over to the XML server entirely. That does not necessarily mean that all of its trading partners are able to switch over. Some companies are quite abstinent in their use of legacy technology. Just look at the number of DOS applications that are still out there in 2001. What is possible is for the company to employ a third party to make these abstinent trading partners appear to be XML./internet-based. It will depend on the company's relative position of strength as to whether the company passes that cost of the third-party on to the trading partner or incurs the cost itself.